Tuning Hyperparameters without Grad Students: Scaling up Bandit Optimisation

Kirthevasan Kandasamy

October 2018
CMU-ML-18-110

Machine Learning Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA

Thesis Committee:
Aarti Singh
Barnabás Póczos (Co-chair)
Jeff Schneider (Co-chair)
Zoubin Ghahramani

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Copyright © 2018 Kirthevasan Kandasamy

This research was sponsored by the Air Force Research Laboratory award numbers FA87501220324 and FA87501720130, the Department of Energy award number DESC0011114, a grant from Foxconn Technology group, and fellowships from Facebook Inc., the Siebel Scholars Foundation, and the CMU Presidential Fellowship program.

Keywords: Decision making under uncertainty, Bandits, Bandit Optimisation, Bayesian Optimisation

To Amma, Appa,
and Thangachchi

Abstract

This thesis explores scalable methods for adaptive decision making under uncertainty in stateless environments, where the goal of an agent is to design an experiment, observe the outcome, and plan subsequent experiments so as to achieve a desired goal. Typically, each experiment incurs a large computational or economic cost, and we need to keep the number of experiments to a minimum. Many of such problems fall under the *bandit* framework, where the outcome of each experiment can be viewed as a reward signal, and the goal is to optimise for this reward, i.e. find the design that maximises this reward. A common use case for bandits, pervasive in many industrial and scientific applications, is *hyperparameter tuning*, where we need to find the optimal configuration of a black box system by tuning the several knobs which affect the performance of the system. Some applications include statistical model selection, materials design, optimal policy selection in robotics, and maximum likelihood inference in simulation based scientific models. More generally, bandits are but one class of problems studied under the umbrella of adaptive decision making under uncertainty in stateless environments. Problems such as active learning and design of experiments are other examples of adaptive decision making, but unlike bandits, progress towards a desired goal is not made known to the agent via a reward signal.

With increasingly expensive experiments and demands to optimise over complex input spaces, bandit optimisation tasks face new challenges today. At the same time, there are new opportunities that have not been exploited previously. We study the following questions in this thesis so as to enable the application of bandit and more broadly adaptive decision making methods to modern real world applications.

- Conventional bandit methods work reliably in low dimensional settings, but scale poorly with input dimensionality. Scaling such methods to high dimensional domains requires addressing several computational and statistical challenges.
- In many applications, an expensive experiment can be cheaply approximated. We study techniques that can use information from these cheap lower fidelity approximations to speed up the overall optimisation process.
- Conventional bandit methods are inherently sequential. We study parallelisation techniques so as to deploy several experiments at the same time.
- Typical methods assume that a design can be characterised by a Euclidean vector. We study bandit methods on graph-structured spaces. As a specific application, we study neural architecture search, which optimises for the structure of the neural network by viewing it as a directed graph with node labels and node weights.
- Current methods for adaptive data collection are designed for specific tasks, and have limited applicability in problems with complex and application specific goals. We study a general framework for sequential design of experiments which allows one to specify their goal and incorporate other domain expertise.

We first delve into the above topics in the bandit framework and then study how they can be extended to broader decision making problems. We develop methods with theoretical guarantees which simultaneously enjoy good empirical performance. As part of this thesis, we also develop an open source Python framework for scalable and robust bandit optimisation.

Acknowledgments

First and foremost, I would like to thank my advisors Jeff Schneider and Barnabás Póczos who have been instrumental in fashioning my research interests and skills. Both Jeff and Barnabás have guided me in picking interesting problems to work on, developing high level intuitions about the solution and figuring out the details in its execution. I owe them both for the many things I have learned in machine learning over the last five years and my personal development as a researcher.

I am also thankful to Aarti Singh and Zoubin Ghahramani who are serving on my thesis committee. I have enjoyed discussing topics on information theory, active learning, and adaptive decision making with Aarti. Zoubin's experience and expertise in machine learning is inspirational. In my few conversations with him, he has motivated me to think about big picture questions in the realm of adaptive decision making under uncertainty.

I have had some incredible friends and collaborators over my PhD, including Akshay Krishnamurthy, who helped me get started with theoretical research, Willie Neiswanger, for listening to many of my ideas and critiquing them, and Gautam Dasarathy, who taught me that solutions to difficult theoretical questions can come from simple intuitions. You have all become wonderful friends and I look forward to continue collaborating with you in the future.

I have had the opportunity to interact with several amazing faculty during my PhD, both at CMU and otherwise. Larry Wasserman taught me many of the fundamentals of nonparametric statistics, both in class and via research. I also enjoyed collaborating with other faculty, such as Eric Xing, James Robins, Jay Whitacre, Newell Washburn, Sanjay Shakkottai, and Venkat Viswanathan. I also learned a lot from my interactions with other CMU faculty such as Artur Dubrawski, Alessandro Rinaldo, Andrew Moore, Andy Pavlo, Roy Maxion, and Ryan Tibshirani. I am thankful to many of my teachers from the University of Moratuwa, including Chulantha Kulasekere, Ranga Rodrigo, and Rohan Munasinghe, who were instrumental in helping me get started with research. I am also thankful to Yoram Bachrach who was my mentor during my internship at Microsoft Research, along with Daniel Tarlow, David Carter, and Ryota Tomioka from whom I learned about deep learning and reinforcement learning. For all individuals above, in addition to their wisdom on deeply technical matters, I also appreciated their advise on personal and career related matters.

I have had the opportunity to work with with several fantastic collaborators which include Adarsh Dave, Biswajit Paria, Chris Collins, Chun-liang Li, Hai Pham, Jared Mitchell, Junier Oliva, Karun Raju Vysyaraju, Ksenia Korovina, Maruan Al-Shedivat, Rajat Sen, Reed Zhang, Shalom Yiblet, Shuli Jiang, Yaoliang Yu, and Yusha Liu. I also cherish the many memories with other friends at grad school, including Aaditya Ramdas, Andrew Wilson, Anthony Platanios, Ben Boecking, Calvin McCarter, Chris Dann, Dana Van Aken, Dougal Sutherland, Guru Guruganesh, Jing Xiang, Kirstin Early, Kumar Avinava Dubey, Leila Wehbe, Manzil Zaheer, Maria De-Arteaga, Mariya Toneva, Martin Azizyan, Matt Barnes, Mrinmaya Sachan,

Nika Haghtalab, Renato Negrinho, Veeru Sadhanala, Yifei Ma, Yining Wang, and Yuxiang Wang. I have cherished the many conversations with all of you, both related to research and otherwise. Thank you for the amazing memories!

This PhD would not have been possible if not for many of the awesome staff members in the Machine Learning Department. In particular, our PhD Program Administrator, Diane Stidle was instrumental in handling many of the administrative tasks for a smooth PhD. Many staff members of the Auton lab, such as Jarod Wang, Predrag Punosevac, Saswati Ray, and Simon Heath were also very supportive in various forms during my PhD.

I would also like to thank Facebook Inc., the Siebel Scholars Foundation, and the CMU Presidential Fellowship program for generously supporting my PhD with fellowships. I am honoured to have been recognised with such prestigious awards among many other competitive candidates.

Finally, I would like to thank my parents and my sister for their support throughout my PhD and patience at times when research consumed all of my time. I owe many of my successes, both in this PhD and otherwise, to their unconditional love, guidance, and support.

Contents

1 Introduction **1**
- 1.1 Stochastic Optimisation under Bandit Feedback . . . 3
- 1.2 Other Examples of Stateless Decision Making Under Uncertainty . . . 4
- 1.3 Thesis Outline & Summary of Contributions . . . 5
- 1.4 Thesis Organisation . . . 9
- 1.5 Other Remarks . . . 10

2 Background Material **11**
- 2.1 A Review of Gaussian Processes (GPs) . . . 11
- 2.2 A Review of Gaussian Process Bandit (Bayesian) Optimisation . . . 12
- 2.3 A Review of K-armed Bandits . . . 15
- 2.4 Some Useful Theoretical Results . . . 16

3 High Dimensional Bandits **19**
- 3.1 Additive Models for High Dimensional Bandits . . . 20
  - 3.1.1 Algorithms . . . 21
  - 3.1.2 Practical Considerations . . . 21
- 3.2 Experiments . . . 23
- 3.3 Proofs of Theoretical Results . . . 29

4 Multi-fidelity Bandits **33**
- 4.1 Multi-fidelity K-armed Bandits . . . 35
  - 4.1.1 Problem Formalism . . . 36
  - 4.1.2 The Multi-Fidelity Upper Confidence Bound (MF-UCB) Algorithm . . . 38
  - 4.1.3 Theoretical Analysis . . . 39
  - 4.1.4 Experiments . . . 42
- 4.2 Multi-fidelity GP Bandits with a Finite Number of Approximations . . . 44
  - 4.2.1 Problem Formalism & Challenges . . . 45
  - 4.2.2 The Multi-fidelity Gaussian Process Upper Confidence Bound (MF-GP-UCB) Algorithm . . . 48
  - 4.2.3 Theoretical Results . . . 50
  - 4.2.4 Implementation Details . . . 57
  - 4.2.5 Experiments . . . 58

4.3 Multi-fidelity GP Bandits with Continuous Approximations . . . 66
4.3.1 Problem Formalism . . . 67
4.3.2 Bayesian Optimisation with Continuous Approximations (BOCA) . . . 69
4.3.3 Theoretical Analysis . . . 71
4.3.4 Experiments . . . 72
4.4 Proofs for Theoretical Results in Chapter 4.1 . . . 77
4.4.1 Proof Outline . . . 77
4.4.2 Proof of Upper Bound . . . 79
4.4.3 Proof of Lower Bound . . . 85
4.5 Proofs of Theoretical Results in Chapter 4.2 . . . 88
4.5.1 Set Up & Notation . . . 88
4.5.2 Discrete $\mathcal{X}$ . . . 90
4.5.3 Compact and Convex $\mathcal{X}$ . . . 96
4.6 Proofs of Theoretical Results in Chapter 4.3 . . . 103
4.6.1 Set Up & Notation . . . 103
4.6.2 Some Technical Lemmas . . . 105
4.6.3 Proof of Main Result . . . 106

5 Parallel Bandits **111**
5.1 Preliminaries . . . 112
5.2 Thompson Sampling for Parallel Bayesian Optimisation . . . 114
5.3 Theoretical Results . . . 116
5.4 Experiments . . . 119
5.5 Proofs of Theoretical Results . . . 124
5.5.1 Notation & Set up . . . 124
5.5.2 Bounding the regret in terms of the number of evaluations . . . 125
5.5.3 Proofs for Parallel TS with Random Evaluation Times . . . 130

6 Bandits on Graph-structured Domains: An Example in Neural Architecture Search **141**
6.1 Problem Set Up . . . 143
6.1.1 A Graph-theoretic Formalism for Neural Networks . . . 144
6.2 OTMANN: Optimal Transport Metrics for Architectures of Neural Networks . . . 145
6.2.1 Preliminaries . . . 145
6.2.2 Description of OTMANN . . . 146
6.2.3 Optimal Transport Reformulation . . . 149
6.2.4 Some Illustrations of the OTMANN distance . . . 150
6.3 NASBOT: Neural Architecture Search with Bayesian Optimisation & Optimal Transport . . . 150
6.3.1 The Kernel . . . 153
6.3.2 Optimising the Acquisition . . . 153
6.3.3 Implementation Details . . . 154
6.4 Experiments . . . 161
6.5 Proofs of Theoretical Results . . . 166

7 Dragonfly: An Open Source Library for Robust & Scalable Bandit Optimisation 179
7.1 Robust Bayesian Optimisation in Dragonfly . . . 180
7.1.1 Choice and Optimisation of Acquisition . . . 180
7.1.2 GP Hyperparameters . . . 182
7.2 Scalable Bayesian Optimisation in Dragonfly . . . 184
7.3 Bayesian Optimisation Implementation in Dragonfly . . . 187
7.3.1 Domains and Fidelity Spaces . . . 188
7.3.2 Kernels . . . 191
7.3.3 Optimising the Acquisition . . . 192
7.3.4 Initialisation . . . 192
7.3.5 Multi-objective Optimisation . . . 193
7.4 Experiments . . . 195
7.4.1 Experiments on Synthetic Benchmarks . . . 195
7.4.2 Experiments on Astrophysical Maximum Likelihood Problems . . . 204
7.4.3 Experiments on Model Selection Problems . . . 205
7.4.4 Dragonfly in Use Elsewhere . . . 209
7.5 User Interface and APIs . . . 212

8 Beyond Bandits: Adaptive Decision Making in Stateless Environments 215
8.1 A General Framework for Adaptive Goal Oriented Design of Experiments . . . 215
8.1.1 Formalism . . . 218
8.1.2 Myopic Posterior Sampling (MPS) for Design of Experiments . . . 219
8.1.3 Theoretical Analysis . . . 220
8.1.4 Examples & Experiments . . . 223
8.2 Active Posterior Estimation . . . 229
8.2.1 Bayesian Active Posterior Estimation (BAPE) . . . 231
8.2.2 Experiments . . . 237
8.3 Proofs of Theoretical Results in Chapter 8.1 . . . 243
8.3.1 Notation and Set up . . . 244
8.3.2 Comparing $\pi_{\mathrm{M}}^{\mathrm{PS}}$ against $\pi_{\mathrm{M}}^{\star}$ and $\pi_{\mathrm{G}}^{\star}$ . . . 245
8.3.3 On Conditions 70 and 72 . . . 248

9 Conclusion 253
9.1 Summary . . . 253
9.2 Future Work on Individual Chapters . . . 254
9.3 Impactful Application Domains for Bandits & Stateless Decision Making . . . 256
9.4 Open Problems & Future Research Directions . . . 259
9.5 Final Thoughts . . . 262

A Notation 263

B Abbreviations 275

C Open Source Software Released with this Thesis 277

Bibliography **279**

List of Figures

2.1 An illustration of GPs and BO. The first figure shows the function of interest f (black line) before any observations and illustrates a GP that represents the prior uncertainty. The shaded region represents a 99% confidence region for f and the coloured lines are samples from the GP. The second figure shows some noisy observations (black $\times$'s) of f and the posterior GP conditioned on the observations. The confidence region has shrunk around the observations. In the third figure, we illustrate GP-UCB when we have to pick the next point x_t given observations as shown in the second figure. The GP-UCB acquisition φ_t upper bounds f. At time t we choose the maximiser of φ_t for evaluation, i.e $x_t = \text{argmax}_x \varphi_t(x)$. In the last figure, we illustrate Thompson sampling, where we first sample a function g from the posterior GP and choose its maximiser for evaluation, i.e $x_t = \text{argmax}_x g(x)$. 12

3.1 An illustration of Add-GP-UCB on a two dimensional function $f^{(2)} = f^{(1)}(x^{(1)}) + f^{(2)}(x^{(2)})$ where $x^{(1)} = \{x_1\}$ and $x^{(2)} = \{x_2\}$. Suppose we have already evaluated at the points shown via $\times$. We construct posteriors for $f^{(1)}$ and $f^{(2)}$ separately (illustrated via the coloured samples). Then we maximise the individual upper confidence bounds on each $f^{(j)}$ and combine them to obtain the next point x_t. 23

3.2 Results on the synthetic datasets for high dimensional Bandits. In all figures the x-axis is the number of queries and the y-axis is the regret in $\log$ scale. We have indexed each experiment by their (p, p', M') values. The first column is the simple regret S_n for the experiments with (p, p', M') set to $(10, 3, 3)$, $(24, 6, 4)$, and $(24, 11, 2)$. The second column is R_T/T for the same experiments. In some figures, the error bars are not visible since they are small and hidden by the bullets. All figures were produced by averaging over 20 independent runs. 25

3.3 Results on the synthetic datasets for high dimensional Bandits. See caption in Figure 3.2 for details. The first column is the simple regret S_n for the experiments with (p, p', M') set to $(40, 5, 8)$, $(40, 18, 2)$, and $(40, 35, 1)$. The second column is R_T/T for the same experiments. 26

3.4 Results on the synthetic datasets for high dimensional Bandits. See caption in Figure 3.2 for details. The first column is the simple regret S_n for the experiments with (p, p', M') set to $(96, 5, 19)$, $(96, 29, 3)$, and $(120, 55, 2)$. The second column is R_T/T for the same experiments. 27

3.5 Results on the Astrophysical experiment (a) and the Viola and Jones dataset (b). The x-axis is the number of queries and the y-axis is the maximum value. All figures were produced by averaging over atleast 15 runs. 28

4.1 Average cross validation log likelihood on datasets of size 300 and 3000 on a synthetic kernel density estimation task. The crosses are the maxima. The maximisers are different since optimal hyperparameters depend on the training set size. That said, the curve for $n = 300$ approximates the $n = 3000$ curve quite well. 33

4.2 Illustration of the partition $\mathcal{X}^{(m)}$'s for a $M = 4$ fidelity problem. The sets $\mathcal{J}^{(m)}_{\zeta^{(m)}+2\gamma^{(m)}}$ are indicated next to their boundaries. $\mathcal{X}^{(1)}, \mathcal{X}^{(2)}, \mathcal{X}^{(3)}, \mathcal{X}^{(4)}$ are shown in yellow, green, red and purple respectively. The optimal arms $\mathcal{X}_\star$ are shown as a black circle. 40

4.3 An illustration of the means of the arms used the simulation problems. The top row are the Gaussian reward problems with (K, M) equal to $(500, 3)$, $(500, 4)$ while the second row are the Bernoulli rewards with $(200, 2)$, $(1000, 5)$ respectively. 42

4.4 Simulations results on the synthetic problems. The first four figures compares UCB against MF-UCB on four synthetic problems. The title states K, M and the costs $\lambda^{(1)}$, $\ldots, \lambda^{(M)}$. The first two used Gaussian rewards and the last two used Bernoulli rewards. The last two figures show the number of plays by UCB and MF-UCB on a $K = 500, M = 3$ problem with Gaussian observations (corresponding to the first figure). 43

4.5 An illustration of the challenges in multi-fidelty optimisation. See main text for explanations. 47

4.6 The 6 panels illustrate an execution of MF-GP-UCB in 2 fidelities at times $t = 6, 8, 10, 11, 14, 50$. In each panel, the top figure illustrates the upper bounds and selection of x_t while the bottom figure illustrates the selection of m_t. We have initialised MF-GP-UCB with 5 random points at the first fidelity. In the top figures, the solid lines in brown and blue are $f^{(1)}, f^{(2)}$ respectively, and the dashed lines are $\varphi^{(1)}_t, \varphi^{(2)}_t$. The solid green line is $\varphi_t = \min(\varphi^{(1)}_t, \varphi^{(2)}_t)$. The small crosses are queries from 1 to $t-1$ and the red star is the maximiser of φ_t, i.e. the next query x_t. $x_\star$, the optimum of $f^{(2)}$ is shown in magenta. In the bottom figures, the solid orange line is $\beta_t^{1/2}\sigma^{(1)}_{t-1}$ and the dashed black line is $\gamma^{(1)}$. When $\beta_t^{1/2}\sigma^{(1)}_{t-1}(x_t) \leq \gamma^{(1)}$ we play at fidelity $m_t = 2$ and otherwise at $m_t = 1$. The cyan region in the last panel is the good set $\mathcal{X}_g$ described in Chapter 4.2.3. 49

4.7 Illustration of the partition $\mathcal{H}^{(m)}$'s for a $M = 4$ fidelity problem. The sets $\mathcal{J}^{(m)}_{\zeta^{(m)}}$ are indicated next to their boundaries. The sets $\mathcal{H}^{(1)}, \mathcal{H}^{(2)}, \mathcal{H}^{(3)}, \mathcal{H}^{(4)}$ are shown in green, blue, yellow and red respectively. Most of the capital invested at points in $\mathcal{H}^{(m)}$ will be due to queries to the m^{th} fidelity function $f^{(m)}$. 53

4.8 Empirically computed values for the ratio $\text{vol}(\mathcal{H}^{(2)})/\text{vol}(\mathcal{X})$ for a one dimensional (left) and two dimensional (right) 2-fidelity problem. For this, the samples $f^{(1)}, f^{(2)}$ were generated using the generative mechanism of Chapter 4.2.1, under the stipulated value for $\zeta^{(1)}$. In both cases, we used an SE kernel with bandwidth 1 and scale parameter 1. The y-axis is the mean value for the ratio over several samples and the x-axis is $\zeta^{(1)}$. In both cases, we used $\gamma^{(1)} = \zeta^{(1)}/3$, and approximated the continuous domain with a uniform grid of size 10^4. The figure indicates that as the approximation improves, i.e. $\zeta^{(1)}$ decreases, the ratio decreases and consequently, we get better bounds. 56

4.9 The simple regret $S(\Lambda)$ (4.9) against the spent capital Λ on the synthetic functions. The title states the function, its dimensionality, the number of fidelities and the costs we used for each fidelity in the experiment; for example, in the fourth panel, we used $M = 3$ fidelities, with costs $\lambda^{(1)} = 1, \lambda^{(2)} = 10, \lambda^{(3)} = 100$ on the 3 dimensional Hartmann function. All curves barring DiRect (which is a deterministic), were produced by averaging over 20 experiments. The error bars indicate one standard error. All figures follow the legend in the first figure for the Currin exponential function. The last panel shows the number of queries at different function values at each fidelity for the Hartmann-3D example. 59

4.10 The performance of our implementation of MF-GP-UCB for different values of $\lambda^{(1)}$ in the 2 fidelity Borehole experiment. Our implementation uses the techniques and heuristics described in Chapter 4.2.4. In all experiments we used $\lambda^{(2)} = 1$. We have also shown the curve for GP-UCB for reference. 60

4.11 (a): the functions used in the Bad Currin Exponential experiment where $f^{(1)} = -f^{(2)}$. (b): the simple regret for this experiment. See caption under Figure 4.9 for more details. 61

4.12 The performance of MF-GP-UCB for different choices of *fixed* threshold values $\gamma^{(1)}$. The curves were averaged over 20 independent runs and in this figure, they start when at least 10 of the 20 runs have queried at least once at the top (second) fidelity. This experiment was run on the $3-$dimensional Hartmann function in the two fidelity set up where $\zeta^{(1)} \approx 0.112$. The true $\zeta^{(1)}$ value was made known to MF-GP-UCB. 62

4.13 Results on the real experiments. The first three figures are hyperparameter tuning tasks while the last is an astrophysical maximum likelihood problem. The title states the experiment, dimensionality (number of hyperparameters or cosmological parameters) and the number of fidelities. For the three hyperparameter tuning tasks we plot the best cross validation error (lower is better) and for the astrophysics task we plot the highest log likelihood (higher is better). For the hyperparameter tuning tasks we obtained the lower fidelities by using smaller training sets, indicated by n_{tr} in the figures and for the astrophysical problem we used a coarser grid for numerical integration, indicated by "Grid". MF-NAIVE is not visible in the last experiment because it performed very poorly. All curves were produced by averaging over 10 experiments. The error bars indicate one standard error. The lengths of the curves are different in time as we ran each method for a pre-specified number of iterations and they concluded at different times. 63

4.14 Samples drawn from a GP with 0 mean and SE kernel with bandwidths $h = 0.01, h = 0.15, 0.5$. Samples tend to be smoother across the domain for large bandwidths. 67

4.15 $g : \mathcal{Z} \times \mathcal{X} \to \mathbb{R}$ is a function defined on the product space of the fidelity space $\mathcal{Z}$ and domain $\mathcal{X}$. The purple line is $f(x) = g(z_\bullet, x)$. We wish to find the maximiser $x_\star \in \text{argmax}_{x\in\mathcal{X}} f(x)$. The multi-fidelity framework is attractive when g is smooth across $\mathcal{Z}$ as illustrated in the figure. 68

4.16 Results on 6 synthetic problems where we plot the simple regret $S(\Lambda)$ (lower is better) against the capital Λ. The title states the function used, and the fidelity and domain dimesions. For the first two figures we used capital $30\lambda(z_\bullet)$, therefore a method which only queries at $g(z_\bullet, \cdot)$ can make at most 30 evaluations. For the third figure we used $50\lambda(z_\bullet)$, for the fourth $100\lambda(z_\bullet)$ and for the last $50\lambda(z_\bullet)$ to reflect the dimensionality d of $\mathcal{X}$. The curves for the multi-fidelity methods start mid-way since they have not queried at $z_\bullet$ up until that point. All curves were produced by averaging over 20 experiments and the error bars indicate one standard error. 74

4.17 Results on the supernova (a) and news group experiments (b). We have plotted the maximum value (higher is better) against wall clock time. Both curves were produced by averaging over 10 xperiments each. The error bars indicate one standard error. . . . 75

4.18 Illustration of the sets $\{\mathcal{F}_n^{(\ell)}\}_{\ell=1}^{m-1}$ with respect to $\mathcal{H}_\tau^{(m)}$. The grid represents a $r\sqrt{d}/n^{1/(2d)}$ covering of $\mathcal{X}$. The yellow region is $\widehat{\mathcal{H}}_\tau^{(m)}$. The area enclosed by the solid red line (excluding $\widehat{\mathcal{H}}_\tau^{(m)}$) is $\mathcal{H}_\tau^{(m)}$. $\mathcal{H}_{\tau,n}^{(m)}$, shown by a dashed red line, is obtained by dilating $\mathcal{H}_\tau^{(m)}$ by $r\sqrt{d}/n^{\alpha/2d}$. The grey shaded region represents $\bigcup_{\ell=1}^{m-1}\mathcal{F}_n^{(\ell)}$. By our definition, $\bigcup_{\ell=1}^{m-1}\mathcal{F}_n^{(\ell)}$ contains the cells which are entirely outside $\mathcal{H}_\tau^{(m)}$. However, the inflation $\mathcal{H}_{\tau,n}^{(m)}$ is such that $\widehat{\mathcal{H}}_\tau^{(m)} \cup \mathcal{H}_{\tau,n}^{(m)} \cup \bigcup_{\ell=1}^{m-1}\mathcal{F}_n^{(\ell)} = \mathcal{X}$. We further note that as $n \to \infty$, $\mathcal{H}_{\tau,n}^{(m)} \to \mathcal{H}_\tau^{(m)}$. 97

5.1 An illustration of the synchronous (left) and asynchronous (right) settings using $M = 3$ workers. The short vertical lines indicate when a worker finished its last evaluation. In the synchronous setting the grey shaded regions indicate idle time after a worker finishes its job. The horizontal location of a number indicates when the worker started its next evaluation while the number itself denotes the order in which the evaluation was dispatched by the algorithm. 113

5.2 Results on the synthetic experiments. The title states the function used, its dimensionality d, the number of workers M and the distribution used for the time. All distributions were constructed so that the expected time for one evaluation was one time unit (for e.g., in the half normal $\mathcal{HN}(\zeta^2)$ in Table 5.1, we used $\zeta = \sqrt{\pi/2}$). The dotted lines depict synchronous methods while the solid lines are for asynchronous methods. The error bars indicate one standard error. All figures were averaged over at least 15 experiments. . . . 121

5.3 The first five panels are results on synthetic experiments. See caption under Figure 5.2 for more details. The last panel compares seqTS, synTS, and asyTS against the number of evaluations n. 122

5.4 Cross validation results on the Cifar-10 experiment. The figure plots the best validation error (lower is better) vs time for each method. We have excluded some methods which performed poorly in the synthetic experiments due to the expensive nature of this experimental set up. The results presented are averaged over 9 experiments. Error bars indicate one standard error. 123

6.1 An illustration of some CNN architectures. In each layer, i: indexes the layer, followed by the label (e.g `conv3`), and then the number of units (e.g. number of filters). The input and output layers are pink while the decision (`softmax`) layers are green.
From Chapter 6.2: The layer mass is denoted in parentheses. The following are the normalised and unnormalised OTMANN distances $d, \bar{d}$. All self distances are 0, i.e. $d(\mathcal{G},\mathcal{G}) = \bar{d}(\mathcal{G},\mathcal{G}) = 0$. The unnormalised distances are, $d(\text{a},\text{b}) = 175.1$, $d(\text{a},\text{c}) = 1479.3$, $d(\text{b},\text{c}) = 1621.4$. The normalised distances are, $\bar{d}(\text{a},\text{b}) = 0.0286$, $\bar{d}(\text{a},\text{c}) = 0.2395$, $\bar{d}(\text{b},\text{c}) = 0.2625$. 144
6.2 An example of 2 CNNs which have $d = \bar{d} = 0$ distance. The OT solution matches the mass in each layer in the network on the left to the layer horizontally opposite to it on the right with 0 cost. For layer 2 on the left, its mass is mapped to layers 2 and 3 on the left. However, while the descriptor of these networks is different, their functional behaviour is the same. 148
6.3 Two dimensional t-SNE embeddings of 100 randomly generated CNN architectures based on the OTMANN distance (top) and its normalised version (bottom). Some networks have been indexed a-n in the figures; these network architectures are illustrated in Figure 6.4. Networks that are similar are embedded close to each other indicating that the OTMANN induces a meaningful topology among neural network architectures. . . . 151
6.4 Illustrations of the nextworks indexed a-n in Figure 6.3. 152
6.5 Each point in the scatter plot indicates the log distance between two architectures (x axis) and the difference in the validation error (y axis), on the Indoor, Naval and Slice datasets. We used 300 networks, giving rise to $\sim 45K$ pairwise points. On all datasets, when the distance is small, so is the difference in the validation error. As the distance increases, there is more variance in the validation error difference. Intuitively, one should expect that while networks that are far apart could perform similarly or differently, networks with small distance should perform similarly. 153
6.6 Initial pool of CNN network architectures. The first 3 networks have structure similar to the VGG nets [228] and the remaining have blocked feed forward structures as in He et al. [91]. 159
6.7 Initial pool of MLP network architectures. 160
6.8 Cross validation results for neural architecture search. In all figures, the x axis is time. The y axis is the mean squared error (MSE) in the first 6 figures and the classification error in the last. Lower is better in all cases. The title of each figure states the dataset and the number of parallel workers (GPUs). All figures were averaged over at least 5 independent runs of each method. Error bars indicate one standard error. 162
6.9 We compare NASBOT for different design choices in our framework. (a): Comparison of NASBOT using only the normalised distance $e^{-\beta\bar{d}}$, only the unnormalised distance $d^{-\beta d}$, and the combination $e^{-\beta d} + e^{-\bar{\beta}\bar{d}}$. (b): Comparison of NASBOT using only the EA modifiers which change the computational units (top 4 in Table 6.2), modifiers which only change the structure of the networks (bottom 5 in Table 6.2), and all 9 modifiers. (c): Comparison of NASBOT with different choices for p and $\bar{p}$. In all figures, the x axis is the number of evaluations and the y axis is the negative maximum value (lower is better). All figures were produced by averaging over at least 10 runs. 164
6.10 Optimal network architectures found with NASBOT on the Cifar10 dataset. 169

6.11 Optimal network architectures found with EA on the Cifar10 dataset. 170
6.12 Optimal network architectures found with RAND on the Cifar10 dataset. 171
6.13 Optimal network architectures found with TreeBO on the Cifar10 dataset. 172
6.14 Optimal network architectures found with NASBOTon the indoor location dataset. . . . 173
6.15 Optimal network architectures found with EA on the indoor location dataset. 173
6.16 Optimal network architectures found with RAND on the indoor location dataset. 174
6.17 Optimal network architectures found with TreeBO on the indoor location dataset. . . . 174
6.18 Optimal network architectures found with NASBOTon the slice localisation dataset. . . 175
6.19 Optimal network architectures found with EA on the slice localisation dataset. 175
6.20 Optimal network architectures found with RAND on the slice localisation dataset. . . . 176
6.21 Optimal network architectures found with TreeBO on the slice localisation dataset. . . . 176
6.22 Optimal network architectures found with NASBOTon the naval propulsion dataset. . . 177
6.23 Optimal network architectures found with NASBOTon the news dataset. 177
6.24 Optimal network architectures found with NASBOTon the protein structure prediction dataset. 178
6.25 Optimal network architectures found with NASBOTon the blog dataset. 178

7.1 A screenshot of the Dragonfly repository on Github, availabe at `dragonfly.github.io`. 180
7.2 Comparison of using individual acquisitions such as GP-UCB, GP-EI, TTEI, TS, PI, and Add-GP-UCB versus the ensemble method as described in Chapter 7.1.1. We have also shown random sampling (RAND) for comparison. The ensemble approach is typically able to perform almost as well as the single best acquisition on each individual problem. We plot the simple regret (1.1), so lower is better. Error bars indicate one standard error. All curves were produced by averaging over 10 independent runs. 181
7.3 Comparison of using only maximum likelihood (ML), only posterior sampling (PS) and the ensemble method (ML+PS) as described in Chapter 7.1.2. We have also shown random sampling (RAND) for comparison. The combined ensemble approach is able to perform as well as or better than the best choice for the given problem. We plot the simple regret (1.1), so lower is better. Error bars indicate one standard error. All curves were produced by averaging over 10 independent runs. 183
7.4 An illustration of GP sample paths drawn from the exponential decay kernel [242] conditioned on being positive. They are suitable for representing the validation accuracy along a fidelity dimension in machine learning applications where, for e.g. validation accuracy tends to increase as we use more data and/or train for more iterations. 186
7.5 An example domain specification in Dragonfly in JSON format. The domain includes an integral variable, a discrete variable, and a discrete numeric variable. 189
7.6 An example domain specification in Dragonfly in JSON format. The domain includes several discrete and discrete numeric variables while the fidelity space consists of a Euclidean variable. 190

7.7 Comparison of Dragonfly with other algorithms and BO packages on functions with *noiseless* evaluations defined on Euclidean domains. We plot the simple regret (1.1) so lower is better. The title states the name of the function, and its dimensionality. All curves were produced by averaging over 20 independent runs. Error bars indicate one standard error. The legend for all curves is available in the first figure. 196

7.8 Comparison of Dragonfly with other algorithms and BO packages on functions with *noiseless* evaluations defined on high dimensional Euclidean domains. We plot the simple regret (1.1) so lower is better. SMAC's initialisation procedure did not work in dimensions larger than 40 so it is not shown in the respective figures. Spearmint is not shown on all figures since it was too slow to run on high dimensional problems. See caption under Figure 7.7 for more details. 197

7.9 Comparison of Dragonfly with other algorithms and BO packages on functions with *noisy* evaluations defined on Euclidean domains. We plot the simple regret (1.1) so lower is better. The title states the name of the function, and its dimensionality. See caption under Figure 7.7 for more details. 198

7.10 Comparison of Dragonfly with other algorithms and BO packages on functions with *noisy* evaluations defined on Euclidean domains. We plot the simple regret (1.1) so lower is better. SMAC's initialisation procedure did not work in dimensions larger than 40 so it is not shown in the respective figures. Spearmint is not shown on all figures since it was too slow to run on high dimensional problems. See caption under Figure 7.7 for more details. 199

7.11 Comparison of Dragonfly with other algorithms and BO packages on synthetic functions with *noiseless* evaluations defined on non-Euclidean domains. We plot the maximum value, so higher is better. The x-axis shows the expended capital, which was chosen so that a single fidelity method would perform exactly 200 evaluations. The title states the name of the function, and its dimensionality (number of variables). We do not state the dimensionality for the synthetic CNN function since the dimensionality of a space of CNN architectures is not defined. See `dragonfly.github.io` for a description of these functions and the approximations for the multi-fidelity curves. All curves were produced by averaging over 20 independent runs. Error bars indicate one standard error. The legend for all curves is available in the first figure. We do not compare Spearmint, HyperOpt, SMAC, and GPyOpt on the synthetic CNN functions since they do not support optimising over neural architectures. 201

7.12 Comparison of Dragonfly with other algorithms and BO packages on synthetic functions with *noisy* evaluations defined on non-Euclidean domains. We plot the maximum value, so higher is better. See caption under Figure 7.13 for more information on the figures. . 202

7.13 Comparison of Dragonfly with RAND and EA on synthetic functions with constraints on the domain. We plot the maximum value, so higher is better. We perform experiments on three different synthetic functions where the left column is when the function evaluations are noiseless, and the right column is when noise is added to the evaluations. The title states the name of the function, and its dimensionality (number of variables). See `github.com/dragonfly/dragonfly/tree/master/demos_synthetic` for a description of these functions and the approximations for the multi-fidelity curves. All curves were produced by averaging over 20 independent runs. Error bars indicate one standard error. 203
7.14 Results on the maximum likelihood estimation problem on the luminous red galaxies dataset [244]. The x-axis is the number of evaluations and the y-axis is the highest likelihood found so far (higher is better). All curves were produced by averaging over 10 independent runs. Error bars indicate one standard error. 204
7.15 Results on the maximum likelihood estimation problem on the Type Ia supernova dataset [50]. The x-axis is time and the y-axis is the highest likelihood found so far (higher is better). We do not compare PDOO, HyperOpt and Spearmint because they do not provide an API for optimising over time. All curves were produced by averaging over 10 independent runs. Error bars indicate one standard error. 205
7.16 Results on the SALSA model selection problem comparing Dragonfly to other packages.The x-axis is wall clock time and the y-axis is the regression error (lower is better). We do not compare HyperOpt and Spearmint because they do not provide an API for optimising over time. All curves were produced by averaging over 10 independent runs. Error bars indicate one standard error. 206
7.17 Results on the random forest regression and gradient boosted regression problems comparing Dragonfly to other packages.The title states the method, the data set used and the dimensionality of the problem. The x-axis is wall clock time and the y-axis is the regression error (lower is better). We do not compare HyperOpt and Spearmint because they do not provide an API for optimising over time. All curves were produced by averaging over 10 independent runs. Error bars indicate one standard error. 207
7.18 Results on the neural architecture search experiments. In all figures, the x-axis is time. The y axis is the mean squared validation error (lower is better). The title of each figure states the dataset. In all cases, we used a parallel set up of two asynchronous workers, where each worker is a single GPU training a single model. We used a one dimensional fidelity space where we chose the number of batch iterations from 4000 to 20,000 ($z_\bullet = 20,000$). All figures were averaged over 5 independent runs. Error bars indicate one standard error. 208
7.19 Visuals of the apparatus used in the electrolyte design task. (a): A picture of the experimental set up. (b): Dragonfly is interfaced with the LabVIEW software to communicate experimental configurations and measurements. (c): Ingredients are chosen according to specifications from Dragonfly. (d)- (f): Various stages of taking a measurement. (g): The conductivity (and other measurements) are measured and fed back to Dragonfly. The entire video can be viewed at `https://youtu.be/XUPWv1J_DX4`. These experiments were conducted by researchers at the Scott Institutute for Energy at CMU. Willie Neiswanger put together the video. 210

7.20 Experimental results comparing Dragonfly to EA and RAND for optimising a real time streaming system where data was streamed through a Spark/Kafka pipeline and written to a Redis database. The left figure shows when the best latency values found by each method (lower is better) when the throughput was fixed at 20K while the right figure shows the same when the throughput was fixed to 10K. Hai Pham was primarily responsible for these experiments. 210

7.21 Comparison of multi-objective optimisation methods implemented in Dragonfly (UCB Tch and TS Tch) with other methods for MOO on the Locality Sensitive Hashing and Viola and Jones problems described in Chapter 7.4. The y-axis is the Tsebychev simple regret for MOO (see (7.1) and Paria et al. [190]) and the x-axis is the number of evaluations. Biswajit Paria was responsible for these experiments and these results are taken directly from Paria et al. [190]. 211

7.22 An example options file in Dragonfly. In this file, `--acq` specifies the list of acquisitions to use in the ensemble method, and `--budget` indicates the budget of evaluations. . . . 213

8.1 Results on the synthetic active learning experiments in Chapter 8.1.4 comparing all methods on the squared error reward. The title states the model and the dimensionality. In all figures, the x axis is the number of experiments n. In the left figures, the y axis is the final negative reward $-\lambda(\theta_\star, n)$ at the n^{th} iteration. In the right figures, it is the corresponding negative cumulative reward $-\Lambda(\theta_\star, n)$. Lower is better in both cases. The legend is given in the left figures. All curves were averaged over 20 runs, and error bars indicate one standard error. 224

8.2 Results on the synthetic active learning experiments in Chapter 8.1.4 comparing all methods on the log likelihood reward. In all figures, the x axis is the number of experiments n. In the left figures, the y axis is the final negative reward $-\lambda(\theta_\star, n)$ at the n^{th} iteration. In the right figures, it is the corresponding negative cumulative reward $-\Lambda(\theta_\star, n)$. See caption under Figure 8.1 for more details. 225

8.3 Results on the real experiments. The first row is for the posterior estimation problem, the second row is for the level set estimation problem, and the third row is for the combined objective problem, all of which are described in Chapter 8.1.4. In the left figures, the y axis is the negative reward $-\lambda(\theta_\star, D_n)$ and in the right figures, it is the negative cumulative reward $-\Lambda(\theta_\star, D_n)$ for the corresponding experiment. The legend is given in the left figures. See caption under Figure 8.1 for more details. 227

8.4 (a) depicts the uncertainty for the log joint probability via samples g drawn from the GP. (b) illustrates the induced uncertainty model $F_{\theta|\mathbf{X_{obs}}}$ for the posterior via the exponentiated and normalised samples $f = \exp g / \int \exp g$. 231

8.5 An illustration of the NED utility. θ_+ is a candidate for the next evaluation. p_B, p_R, p_G denote values for $p_+ = \log P(\theta_+, \mathbf{X_{obs}})$ sampled from the GP. We add them as hallucinated points and rebuild our GP and generate samples (second step). These samples are exponentiated and normalised (third step) and then its KL divergence with the estimate is computed. 233

8.6 (a): Samples drawn from the GP in the log joint probability space. (b): The same samples after exponentiation. High variance in the low likelihood regions are squashed and low variances in the high likelihood regions are blown up. This is the key insight that inspires our methods and the EV utility in particular. 234
8.7 The first column shows the log joint probability and the corresponding posterior. In the second column we have estimates of the log joint and the posterior for uniformly spaced points. In the third column we have the same except that more points were chosen in high likelihood regions. 235
8.8 (a) and (b) are the true log joint probability and joint probability in blue. Assume that we have already queried at the brown crosses and let the red circles (x) and (y) be candidates for the next query. In BAPE we would be interested in querying (y) but not (x). In AGPR we would be interested in both (x) and (y) whereas in BO we would be keen in neither. . 236
8.9 (a), (b): Comparison of NED/EV against MCMC-DE, ABC, MCMC-R and RAND for the 1D and 2D synthetic experiments respectively. The x-axis is the number of queries and the y-axis is the KL divergence between the truth and the estimate. All figures were obtained by averaging over 60 trials. 238
8.10 The 100 points chosen in order by NED for the 2D experiment. The green contours are the true posterior. Initially the algorithm explores the space before focusing on high probability regions. 239
8.11 The first row is for the functionals T_1, T_2 in $d = 5$ dimensions and the second for is for the functionals T_3, T_4. The last twoape rows are the same four functionals for $d = 15$. The x-axis is the number of queries and the y-axis is $|\widehat{T}_i - T_i|/|T_i|$. We use 500 queries for $d = 5$ and 3200 queries for $d = 15$. All figures were obtained by averaging over 30 trials. 241
8.12 (a): Comparison of NED/EV against MCMC-DE, ABC, Emcee, MCMC-R and RAND on the Type Ia Supernovae dataset. For all regression methods we show results for up to 1600 queries and up to 4 times as many for MCMC and ABC. For evaluation, KL was approximated via numeric integration on a $(100)^3$ grid. Note that MCMC and ABC require several queries before a nontrivial KL with the truth is obtained. All curves were obtained by averaging over 30 runs. (b): Projections of the points selected by EV (bottom row) and the marginal distributions (top row). 242
8.13 The projections of the first 6000 points queried by RAND MCMC, and EV respectively on to the first 2 dimensions in cyan. The points shown in red are queries at high likelihood ($\log P > -50$) points. 243
8.14 Comparison of EV against MCMC-R and RAND. We use up to 12000 queries for all methods. The y-axis is the mean squared reconstruction error. The curves were obtained by averaging over 16 runs. 243

List of Tables

1.1 A summary of various paradigms for adaptive decision making under uncertainty. In this thesis, we focus exclusively on stateless systems/environments. Chapters 3-7 study scalability topics in stateless explicit reward (i.e. bandit) settings and Chapter 8 is on stateless implicit reward settings. 2

4.1 Bounds on the expected number of plays for each $k \in \mathcal{X}^{(m)}$ (columns) at each fidelity (rows) after n time steps (i.e. n plays at any fidelity) in MF-UCB. 79

4.2 Bounds on the number of queries for each $x \in \mathcal{H}^{(m)}$ (columns) at each fidelity (rows). The bound for $T_n^{(M)}(x)$ in $\mathcal{H}^{(M)}$ holds for all arms except the optimal arm $x_\star$ (note $\Delta^{(M)}(x_\star) = 0$). 92

5.1 Descriptions of the random delay models analysed for the synchronous and asynchronous parallel set ups. The second column shows the probability density functions $p(x)$ for the uniform Unif(a, b), half-normal $\mathcal{HN}(\zeta^2)$, and exponential Exp(λ) distributions. The subsequent columns show the expected number of evaluations $n_{\text{seq}}, n_{\text{syn}}, n_{\text{asy}}$ for seqTS, synTS, and asyTS respectively with M workers. synTS always completes fewer evaluations than asyTS. For example, in the exponential case, the difference could be a $\log(M)$ factor. 118

5.2 Test sesults on the Cifar-10 experiment. The table gives the test error (lower is better) on an independent test set of $10K$ images after training the best model chosen by each method for 80 epochs. The results presented are averaged over 9 experiments. 123

6.1 An example label mismatch cost matrix M. There is zero cost for matching identical layers, < 1 cost for similar layers, and infinite cost for disparate layers. 147

6.2 Descriptions of modifiers to transform one network to another. The first four change the number of units in the layers but do not change the architecture, while the last five change the architecture. 155

6.3 The label mismatch cost matrix M we used in our CNN experiments. $M(\texttt{x}, \texttt{y})$ denotes the penalty for transporting a unit mass from a layer with label `x` to a layer with label `y`. The labels abbreviated are `conv3`, `conv5`, `conv7`, `max-pool`, `avg-pool`, `fc`, and `softmax` in order. A blank indicates ∞ cost. We have not shown the `ip` and `op` layers, but they are similar to the `fc` column, 0 in the diagonal and ∞ elsewhere. 156

6.4 The label mismatch cost matrix M we used in our MLP experiments. The labels abbreviated are `relu`, `crelu`, `<rec>`, `logistic`, `tanh`, and `linear` in order. `<rec>` is place-holder for any other rectifier such as `leaky-relu`, `softplus`, `elu`. A blank indicates ∞ cost. The design here was simple. Each label gets 0 cost with itself. A rectifier gets 0.1 cost with another rectifier and 0.25 with a sigmoid; vice versa for all sigmoids. The rest of the costs are infinity. We have not shown the `ip` and `op`, but they are similar to the `lin` column, 0 in the diagonal and ∞ elsewhere. 157

6.5 The first row gives the number of samples N and the dimensionality D of each dataset in the form (N, D). The subsequent rows show the regression MSE or classification error (lower is better) on the *test set* for each method. The last column is for Cifar10 where we took the best models found by each method in 24K iterations and trained it for $120K$ iterations. When we trained the VGG-19 architecture using our training procedure, we got test errors 0.1718 (60K iterations) and 0.1018 (150K iterations). 163

7.1 Final least squared errors in the regression problems of Chapter 7.4.3. In addition to the methods in Figure 7.17, we compare Dragonfly and random search at the lowest fidelity, as well as Hyperband. 207

List of Algorithms

1 GP-UCB from Srinivas et al. [235] . . . 13
2 Add-GP-UCB from Kandasamy et al. [120] . . . 22
3 MF-UCB from Kandasamy et al. [125] . . . 38
4 MF-GP-UCB from Kandasamy et al. [123, 124] . . . 50
5 BOCA from Kandasamy et al. [127] . . . 70
6 seqTS from Thompson [246] . . . 115
7 asyTS from Kandasamy et al. [129] . . . 115
8 synTS from Kandasamy et al. [129] . . . 116
9 Compute $\delta_{\text{op}}^{\text{rw}}(u)$ for all $u \in \mathcal{L}$ in OTMANN, from Kandasamy et al. [130] . . . 156
10 BO in Dragonfly with M asynchronous workers, from Kandasamy et al. [133] . . . 193
11 MPS ($\pi_{\text{M}}^{\text{PS}}$) from Kandasamy et al. [131, 132] . . . 219
12 BAPE from Kandasamy et al. [121, 128] . . . 232

Chapter 1

Introduction

Artificial intelligence and machine learning have made great strides in the recent past, achieving human or super-human level performance in tasks requiring prediction from data. However, an agent becomes intelligent only when it goes beyond supervised learning; it should be able to assimilate information from past experience and plan actions so as to acquire new information and achieve a desired goal. For instance, when presented with data on several electrolytes tested on a battery design problem, a learning agent can only predict the properties of a new electrolyte. However, an intelligent agent will be able to use this data and suggest new electrolytes to test so as to discover a previously unknown high performing electrolyte for a given application.

Such problems are typically studied under the umbrella of decision making under uncertainty, where, an agent, tasked with achieving a certain goal, interacts with a system by taking actions and observing the result of said action. Once it has this observation, the agent is able to learn something about the system and make a more informed choice the next time it takes an action. A fundamental tension that arises in such problems is that of learning system characteristics vs achieving the desired goal, commonly known in the literature as the *exploration-exploitation trade-off*. An extensive exploration of system characteristics can be inefficient as it might require collecting an unnecessarily large amount of data; however, developing some understanding of the system is paramount to achieving the desired goal. For instance, in electrolyte discovery, the agent's goal is to find an electrolyte with desirable conductivity, viscosity and solubility properties. However, it will not be able to find such an electrolyte without developing some understanding of the relationships between these properties and design parameters such as salt concentrations, solvent fractions, and process conditions. Hence, it will need to conduct some exploratory experiments with the intention of learning such system characteristics.

Paradigms for Decision Making Under Uncertainty

To contextualise the work in this thesis, we identify two dichotomies in adaptive decision making problems. We draw these distinctions primarily to identify the settings for this thesis, and note

	Explicit Reward	Implicit Reward	
Stateless Environment	Bandits	Active learning, Adaptive design of experiments	This thesis.
Stateful Environment	Reinforcement learning	System identification, Dynamic model learning	

Table 1.1: A summary of various paradigms for adaptive decision making under uncertainty. In this thesis, we focus exclusively on stateless systems/environments. Chapters 3-7 study scalability topics in stateless explicit reward (i.e. bandit) settings and Chapter 8 is on stateless implicit reward settings.

that different authors may draw these delineations differently. We will use the terms system and environment interchangeably to denote the external environment an agent is interacting with.

Stateless vs Stateful Environments: The first is on *stateless* vs *stateful* environments/systems. In the former, the available set of actions do not change over time, and an agent's actions do not change how the system responds to future actions. In stateful systems, either of these may be true. Electrolyte design, where each action is a specific design of an electrolyte that the agent chooses to test, is an example of a stateless system. The set of available designs do not change throughout the entire experimentation process and the properties of a given design do not depend on the order in which it was tested. An example of the latter is a game of chess, where, between actions (moves) by an agent, the available actions can change and the opponent's moves can change depending on the order of the agent's moves.

Explicit vs Implicit Reward: The second distinction is *explicit reward* vs *implicit reward* problems. In the former, progress towards the desired goal is made explicitly known to the agent via a, possibly noisy, reward signal. In the latter, the agent receives feedback from the system, but this feedback does not directly inform the agent of its progress; instead, the agent needs to learn an implicitly defined reward function from the action-observation pairs it has collected. For instance, assume that our goal in the above electrolyte design example is to find an electrolyte with high conductivity, and that the experiment for each design measures the conductivity. This is an explicit reward problem as the agent can use the feedback (measured conductivity) as a reward that needs to be maximised. A game of chess is a stateful explicit reward problem, since, at the end of the game, the agent receives a reward signal indicating whether it won (1), drew $(1/2)$ or lost (0). An example of implicit reward problems is active learning, where, say, an agent needs to choose which data points to label so as to learn a regression function g. The agent's goal is to minimise the error $\|g - \hat{g}\|$ between the true regression function and an estimate $\hat{g}$. However, this error is not made known explicitly nor can it be computed from the available data since g is unknown. An example of a stateful implicit reward task is system identification, where we wish to learn the control dynamics of a statful system, such as a robot or an industrial apparatus. As we will see in Chapter 8 of this thesis, implicit reward systems are, in general, more difficult than explicit reward systems as the agent needs to learn the reward function, while simultaneously working to maximise it.

We have summarised the above paradigms in Table 1.1. The majority of the decision making

literature focuses on explicit reward settings, where the stateless versions of this paradigm are studied in the *bandit* framework, and stateful versions in the *reinforcement learning* framework.

This thesis exclusively focuses on *stateless* systems, and in particular, developing scalable methods for such decision making problems in real world applications. In Chapters 3 to 7 of this thesis, we will delve deep into such scalability topics in the bandit setting. In Chapter 8, we develop methods for stateless implicit reward systems. In Chapter 9, we discuss how many of the scalability ideas from bandits can be extended to other stateless decision making settings. We then present some thoughts on extending these techniques to stateful environments. Next, we present an overview of these stateless paradigms in the next two sections.

1.1 Stochastic Optimisation under Bandit Feedback

Many problems in stateless adaptive decision making can be cast as the optimisation of a black-box function $f : \mathcal{X} \to \mathbb{R}$ over a discrete or continuous domain $\mathcal{X}$, where each experiment is an evaluation of f. In *bandit optimisation*, f is accessible only through possibly noisy point evaluations, is potentially non-convex and has no gradient information. In typical applications, each evaluation is expensive, incurring a large computational or economic cost. Hence, the goal is to maximise f using as few evaluations as possible. A common use case for such noisy zeroth order optimisation problems is *hyperparameter tuning*, where we need to tune the several knobs of a black-box system which affect its performance for a given problem. Some applications for hyperparameter tuning include statistical model selection, materials design, configuring industrial systems, scientific studies, optimal policy selection in robotics, and maximum likelihood inference in simulation based scientific models [77, 99, 159, 170, 191].

Historically, the bandit framework was studied in settings where we maximise the cumulative sum of all evaluations to f as opposed to just finding the maximum. Let $x_\star = \operatorname{argmax}_{x\in\mathcal{X}} f(x)$ be a maximiser of f. Suppose we evaluate f at $x_1, \ldots, x_n$ in n time steps. The goal of a bandit algorithm is typically to achieve small value for either the simple regret S_n or the cumulative regret R_n defined below.

$$S_n = f(x_\star) - \max_{t=1,\ldots,n} f(x_t), \qquad R_n = \sum_{t=1}^{n} \Big(f(x_\star) - f(x_t) \Big). \tag{1.1}$$

Both notions of regret are closely related since any algorithm with sub-linear cumulative regret, i.e. $R_n/n \to 0$, also has vanishing simple regret, i.e. $S_n \to 0$. In an adaptive sequential strategy, at time t, we will have queried f at $t-1$ points $x_1, \ldots, x_{t-1}$ and observed $y_1, \ldots, y_{t-1}$, where $\mathbb{E}[y_i] = f(x_i)$ for $i = 1, \ldots, t-1$. A bandit method is a strategy to determine the next point x_t for evaluation using knowledge of f acquired via previous query-observation pairs $\{(x_i, y_i)\}_{i=1}^{t-1}$. Below, we list two applications each for simple and cumulative regret.

Model selection in statistics/machine learning [103, 232]: Statistical models require tuning hyperparameters for good empirical performance, e.g. kernel parameters in support vector machines. Typically, these hyperparameters are chosen by maximising a validation set accuracy.

We can cast this as a bandit optimisation problem where each evaluation requires training and validating the algorithm for given values x of the hyperparameters in a space of values $\mathcal{X}$. A pertinent example in today's context is deep learning which has recently been successful in many applications. Most deep learning models have a number of hyperparameters for the architecture and regularisation. Currently, they are manually tuned by domain experts which is cumbersome as training a deep model can be time consuming. Hence, smart techniques for model selection have become particularly relevant today.

Computational astrophysics [50, 121]: Given astronomical observations Z_{obs}, we wish to infer the cosmological parameters (e.g. Hubble constant, Baryonic density). For a given choice of these parameters x, cosmological simulators can produce simulations of a cosmological process (e.g. galaxy formation or supernova) which can be compared to the observations. Physicists wish to find the maximum likelihood estimate of Z_{obs}. The task at hand is to maximise the likelihood $f(x) = P(Z_{\text{obs}}|x)$, whose computation for a given set of values x for the cosmological parameters can be noisy and requires an expensive simulation of a cosmological process.

Online advertising [34]: A popular use case for cumulative regret is online advertising. At each time step, we need to choose an ad from a large pool to display on a web page; the goal is to maximise the cumulative number of clicks over all ads displayed within a specified period. This translates to identifying ads with the highest click through rate and displaying them most of the time. Here, $\mathcal{X}$ is a finite discrete set of ads and f is the expected number of clicks for each ad within a specified period. When we display an ad x, we incur the opportunity cost for our choice and observe a noisy estimate of $f(x)$.

Realtime tuning of industrial/technological systems [83, 251]: In many real world applications, we wish to find the optimal configuration of a system by experimenting with it when the system is running. For example, one wishes to find the optimal configuration of a stream processing system while it is serving traffic. Similarly, one wishes to find the optimal parameters of an industrial system, such as a tokamak nuclear reactor, while making sure that the system is stable and that the output is not adversely affected. As in online advertising, in these settings, an agent tries a new configuration and is rewarded with the cumulation of how well the system performed over time, as opposed to finding the single best configuration. Here, $\mathcal{X}$ is the space of possible configurations of the system, and f is a performance criterion that we are interested in.

1.2 Other Examples of Stateless Decision Making Under Uncertainty

As we will see in this thesis, there exist a plethora of stateless decision making settings outside of the bandit framework, each of which will require its own formalism. Instead of attempting a formal description of each individual setting here, we will provide some examples to demonstrate that the bandit formalism alone would not suffice for many problems.

Active Learning: Consider applications where we wish to perform usual machine learning tasks,

such as regression, classification or posterior distribution estimation, when data is expensive. The goal is to design actions, i.e. obtain labels, at regions which are most critical to the desired learning objective. For instance, in the computational astrophysics application described above, sometimes astrophysicists are interested in estimating the entire posterior distribution of the cosmological parameters, instead of just finding the maximum likelihood estimate [121].

Materials Design: In the electrolyte design example above, a chemist wishes to choose actions, i.e. test electrolyte designs, so that she can optimise for conductivity while simultaneously learning the relationship between the design parameters and electrolyte properties such as viscosity and solubility [71]. In this example, we wish to do both optimisation and active learning on different properties of the electrolyte. Similarly, in alloy design, one wishes to conduct experiments to identify phase transitions in the crystal structure of an alloy as the composition of the individual metals changes [28].

In the examples above, there is no explicit reward feedback to the decision maker on the progress it is making towards the goal. Chapter 8 describes a general framework for stateless decision making from a design of experiments standpoint, which includes the settings above. Precisely, we demonstrate that when the implicit reward is a function of the *unknown* system characteristics and the data collected, the decision making problem is still tractable and practical.

1.3 Thesis Outline & Summary of Contributions

In this thesis, we develop techniques for scaling up bandit and other stateless settings for adaptive decision making under uncertainty. We will study different notions of scalability, many of them motivated by practical challenges and opportunities arising in modern applications. On each topic, we strive to achieve the following two goals.

1. Theoretically quantify the difference between our methods and vanilla bandit methods via an appropriate notion of regret for the setting.

2. Empirically demonstrate that our algorithms outperform such unadorned versions on practical applications.

To that end, we believe that this thesis supports the following claim.

Thesis Statement: *Adaptive decision making under uncertainty in stateless paradigms can be made scalable and practical for modern applications, both in theory and practice.*

Next, we summarise the main contributions of this thesis. Chapters 3-6 describe specific scalability paradigms for bandit methods, chapter 7 describes an open source software platform for scalable bandit optimisation that integrates the above (and other) techniques, and Chapter 8 describes a general and flexible framework for adaptive decision making under uncertainty that subsumes the bandit setting among many others.

Chapter 3: High Dimensional Bandits

Problem & Motivation: We study techniques for bandit optimisation on Euclidean domains $\mathcal{X} \subset [0,1]^d$ when d is large. High dimensional bandit problems occur in several applications where we need to tune several hyperparameters, such as statistical model selection, astrophysics, and tuning industrial systems. While there have been many successes for bandits in low dimensions, scaling it to high dimensions has been notoriously difficult. Prior work on this topic were under very restrictive assumptions. Making progress in this front requires making structural assumptions on the problem which hold water in practice.

Solution: In Kandasamy et al. [120], we overcome the statistical and computational challenges in high dimensional bandits by assuming an additive structure for f. This setting is substantially more expressive and contains a richer class of functions than previous work. We prove that, for additive functions the regret of classical bandit techniques (e.g. GP-UCB) has only linear dependence on d even though f depends on all d dimensions. Moreover, we propose Add-GP-UCB, which can leverage the additive structure in a computationally efficient manner. Via synthetic examples, an astrophysical maximum likelihood problem and a statistical model selection problem, we demonstrate that our method outperforms naive methods on additive functions and on several examples where the function is not additive.

Chapter 4: Multi-fidelity Bandits

Problem & Motivation: Traditionally, the bandit literature assumes a single source of function evaluations or experimental outcomes, where querying this source can be expensive. However, in many cases, cheap approximations to this source may be available. For instance, the cross validation curve of an expensive machine learning algorithm can be approximated by cheaper training routines using less data and/or fewer iterations. Similarly, the expensive real world behaviour of a robot can be approximated by a cheap computer simulation. In such settings, it is natural to ask if one could leverage these cheap sources to learn about the expensive function and speed up the optimisation process.

Solution: We formalise this task as a *multi-fidelity* bandit problem. We study variants of the classical K-armed bandit and the GP bandit where a decision-maker can query either a desired experiment, or an available cheap approximation. In all settings, we theoretically prove and empirically demonstrate that our methods use the cheap approximations to eliminate bad regions in $\mathcal{X}$ and deploy the expensive evaluations in a small promising region to speedily identify the optimum. Consequently, they achieve better regret than naive strategies which ignore the approximations.

In Kandasamy et al. [125], we first analyse the K-armed bandit setting, where, at each time step the decision-maker may choose to play one of K arms at any one of M fidelities. The highest fidelity (desired outcome) expends cost $\lambda^{(M)}$. The m^{th} fidelity (an approximation) expends $\lambda^{(m)} < \lambda^{(M)}$ and returns a biased estimate of the highest fidelity. We develop MF-UCB, a novel upper confidence bound procedure for this setting and prove that it naturally adapts to the se-

quence of available approximations and costs, thus attaining better regret than naive strategies which ignore the approximations. We complement this result with a lower bound and show that MF-UCB is nearly minimax optimal under standard regularity conditions.

We then extend this algorithm and analysis to the GP setting, where we study multi-fidelity Bayesian optimisation when we have access to a finite number of approximations in Kandasamy et al. [123, 124], and a continuous spectrum of approximations in Kandasamy et al. [127]. We develop MF-GP-UCB for the former setting and BOCA for the latter, both based on upper confidence bound techniques. In our theoretical analysis, we demonstrate that they achieve better simple regret than strategies which ignore multi-fidelity information. Empirically, MF-GP-UCB and BOCA outperform such naive strategies and other multi-fidelity methods on several simulations and real world tasks in model selection and astrophysics.

Chapter 5: Parallel Bandits

Problem & Motivation: The bandit literature has predominantly focused on the sequential setting where the algorithm has to wait for an evaluation to the function to complete before proceeding to the next. However, in many applications, we are able to execute multiple evaluations in parallel. For example, in model selection, we may have the computing infrastructure to train multiple models in parallel with different hyperparameter configurations. In materials design and drug discovery, we now have high throughput screening equipment that can test several hundred materials/drugs at the same time.

Solution: In Kandasamy et al. [129], we design and analyse variations of the classical Thompson sampling (TS) procedure for bandits where function evaluations are expensive but can be performed in parallel. Our theoretical analysis shows that a direct application of the sequential Thompson sampling algorithm in either synchronous or asynchronous parallel settings yields a surprisingly powerful result: making n evaluations distributed among M workers is essentially equivalent to performing n evaluations in sequence. Further, by modelling the time taken to complete a function evaluation, we show that, under a time constraint, asynchronous parallel TS achieves asymptotically lower regret than both the synchronous and sequential versions. These results are complemented by an experimental analysis, showing that asynchronous TS outperforms a suite of existing parallel bandit algorithms in simulations and in a hyperparameter tuning application. In addition to these, the proposed procedure is computationally cheaper than existing work for parallel bandit optimisation, scaling only linearly in the number of workers.

Chapter 6: Bandits on Graph-structured Domains

Problem & Motivation: Traditionally, bandit methods have been studied on "simple" domains, such as a set of K discrete arms, or discrete/compact Euclidean spaces. In the context of model selection in statistics and machine learning, this only permits tuning scalar hyperparameters of machine learning algorithms. However, with the surge of interest in deep learning, there is an

increasing demand to tune neural network *architectures*. This motivates studying bandit methods in more complex and structured graphical spaces.

Solution: In Kandasamy et al. [130], we develop NASBOT, a Gaussian process based bandit framework for neural architecture search. To accomplish this, we develop OTMANN, a distance metric in the space of neural network architectures which can be computed efficiently via an optimal transport program. We demonstrate that NASBOT outperforms other alternatives for architecture search in several cross validation based model selection tasks on multi-layer perceptrons and convolutional neural networks. The OTMANN distance might be of independent interest to the deep learning community as it may find applications outside of architecture search. Moreover, we believe that the techniques used in NASBOT can be extended to handle the optimisation of other graph structured objects such as chemical molecules and crystal structures.

Chapter 7: An Open Source Library for Scalable Bandit Optimisation

Problem & Motivation: One of the main challenges for the bandit and black-box optimisation research is the large disconnect between theoretical and methodological developments and the availability of practical tools that can be deployed in real world problems. Existing software are usually not able to handle complex problems well. For example, they suffer in high dimensions, cannot optimise over complex domains, and do not have functionality to handle variants of the bandit framework such as multi-fidelity optimisation or multi-objective optimisation. Moreover, most model based bandit methods tend to be quite sensitive to the choice of the model which can severely limit their usefulness in practice.

Solution: In Kandasamy et al. [133], we develop Dragonfly, a Python library for scalable bandit optimisation which integrates the scalability ideas in the previous chapters into one software framework. Dragonfly uses ensemble strategies which combine multiple models to decide the next recommendation which are more robust than existing model-based techniques which rely on a single model throughout the entire optimisation process. In addition, we have implemented functionality for multi-objective optimisation from some of our other work [190]. Dragonfly is implemented in a modular and extensible fashion which allows users to seamlessly integrate new algorithms and bandit formalisms. Empirically, we demonstrate that Dragonfly either outperforms or is competitive with several other tools for bandits in a variety of synthetic and real world tasks. We have released Dragonfly open source under the MIT license which we hope will be a useful tool for practitioners and foster future research in scalable bandit methods.

Chapter 8: Beyond Bandits, Adaptive Decision Making in Stateless Environments

Problem & Motivation: In this chapter, we take a step back from bandits and analyse more general settings for sequential decision making under certainty in stateless environments. In these problems, as in the bandit setting, an agent takes an action, observes the result of the action, and proceeds sequentially to fulfil a certain goal. This setting subsumes the bandit framework where the agent directly observes a reward for the associated action. However, in general, such a reward

may not be available and the agent needs to learn this reward signal from past observations while simultaneously working to maximise it. Such problems are pervasive in many industrial and scientific applications where we need to carry out very application-specific goals such as those described in Chapter 1.2.

Solution: In Kandasamy et al. [131, 132], we design a new Bayesian myopic strategy for a wide class of sequential design of experiment (DOE) problems, where the goal is to collect data in order to fulfil a certain problem specific goal. Our approach, Myopic Posterior Sampling (MPS), is inspired by the Thompson sampling algorithm for multi-armed bandits and leverages the flexibility of probabilistic programming and approximate Bayesian inference to address a broad set of problems. The MPS framework is general enough to incorporate domain expertise when available, and flexible enough to encompass a variety of goals that typically arise in DOE tasks. On the theoretical side, we leverage ideas from adaptive submodularity and reinforcement learning to derive conditions under which MPS achieves sublinear regret against myopic and globally optimal oracle policies which know the characteristics of the environment. Empirically, this general-purpose strategy is competitive with more specialised methods in a wide array of DOE tasks in astrophysics and materials science. More importantly, it enables addressing complex DOE goals where no existing method seems applicable.

In Kandasamy et al. [121, 128], we delve deeper into one such DOE task, active posterior estimation, developing specialised algorithms for this setting. This problem finds applications in settings where the likelihood of a Bayesian model is expensive to evaluate, such as in astrophysics where computation of the likelihood might involve an expensive cosmological simulation. Existing techniques for posterior estimation are based on generating samples representative of the posterior, which do not consider efficiency in terms of likelihood evaluations. In order to be query efficient, we treat posterior estimation in an active regression framework and propose two myopic query strategies to choose where to evaluate the likelihood. Via experiments on a series of synthetic and real examples we demonstrate that our approach is significantly more query efficient than existing techniques and other heuristics for posterior estimation.

Finally, we mention that many of the scalability ideas in chapters 3-7 can be extended to the general decision making framework of chapter 8. We discuss this in more detail and provide concrete suggestions in our conclusion in Chapter 9.

1.4 Thesis Organisation

In Chapter 2, we briefly review some relevant background material on Gaussian processes and bandits. Chapters 3–8 present the main technical contributions on individual topics. In each chapter or sub-chapter, we usually begin with an introduction to the topic and discuss related work. We then formalise the problem set up. Next, we present the main methodological contributions and relevant theoretical results. Finally, we conclude with experimental results. The proofs of theoretical results for each chapter are typically found at the end of the chapter. Chap-

ter 9 concludes with some high level thoughts and directions for future work. Appendices A and B respectively summarise the notation and abbreviations used throughout this thesis. Appendix C lists and provides links to software released with this thesis.

1.5 Other Remarks

Collaborators

This thesis is based on work with multiple collaborators. My advisors Jeff Schneider and Barnabás Póczos were collaborators on all chapters. In addition to this, Chapter 4 was joint work with Gautam Dasarathy and Junier Oliva. Chapter 5 was joint work with with Akshay Krishnamurthy. Chapter 6 was joint work with with Willie Neiswanger and Eric Xing. Chapter 7 was joint work with with Karun Raju Vysyaraju, Willie Neiswanger, Biswajit Paria, Christopher Collins, and Eric Xing. Chapter 8 was joint work with with Willie Neiswanger, Akshay Krishnamurthy, and Reed Zhang.

This manuscript only describes the work where I was primarily responsible for the conception of the idea, derivation of theorems, and practical implementation. However, in the interest of completeness, I make the following three exceptions.

- Akshay Krishnamurthy derived the theoretical results for exponential delays in Chapter 5.5.3, specifically Lemma 60, Theorem 62, Theorem 63, and Theorem 64.
- The methods for multi-objective optimisation in Dragonfly is based on joint work with Biswajit Paria who was the lead author of that paper [190]. These methods were incorporated into Dragonfly and are described briefly in Chapter 7.3.5.
- Dragonfly has been used in a variety of applications by some research groups. Some of them are highlighted at the end of Chapter 7.4, in order to demonstrate real world applications for methods developed in this thesis. While I assisted said groups in setting up Dragonfly, I was not responsible for planning and executing those experiments.

Excluded Research

The following are some of my lead authored PhD research that have been excluded from this manuscript as they do not fall within the scope of this thesis.

- Nonparametric estimation of information theoretic functionals [118, 119].
- Kernel methods for high dimensional nonparametric regression [117].
- Estimation of hidden Markov models with nonparametric emission probabilities [122].
- Improving neural conversation models using deep reinforcement learning [126].

Chapter 2

Background Material

To facilitate the discussion in the forthcoming chapters, we briefly review some background material including Gaussian processes, Bayesian optimisation, and K-armed bandits. The last sub-chapter presents some theoretical results used frequently in our proofs.

2.1 A Review of Gaussian Processes (GPs)

In this thesis, we will mostly, albeit not entirely, rely on Gaussian processes (GP) [203] to model the pay-off function f in the bandit framework. They are popularly used in practice and known to perform well empirically [232]. Moreover, their theoretical properties are well understood [235]. We begin with a brief review of GPs. However, it is important to keep in mind, that most of the underlying ideas in this thesis are orthogonal to GPs and apply straightforwardly to other models.

A Gaussian Process (GP) over a space $\mathcal{X}$ is a random process from $\mathcal{X}$ to $\mathbb{R}$. GPs are typically used as a prior for functions in Bayesian nonparametrics. A GP is characterised by a mean function $\mu : \mathcal{X} \to \mathbb{R}$ and a covariance function (or kernel) $\kappa : \mathcal{X}^2 \to \mathbb{R}$. If $f \sim \mathcal{GP}(\mu, \kappa)$, then $f(x)$ is distributed normally $\mathcal{N}(\mu(x), \kappa(x, x))$ for all $x \in \mathcal{X}$. Two popular kernels of choice are the squared exponential (SE) kernel $\kappa_{\sigma,h}$ and the Matérn kernel $\kappa_{\nu,h}$. Writing $z = \|x - x'\|_2$, they are defined as

$$\kappa_{\sigma,h}(x, x') = \sigma \exp\left(-\frac{z^2}{2h^2}\right), \qquad \kappa_{\nu,\rho}(x, x') = \frac{2^{1-\nu}}{\Gamma(\nu)}\left(\frac{\sqrt{2\nu}z}{\rho}\right)^{\nu} B_\nu\left(\frac{\sqrt{2\nu}z}{\rho}\right),$$

respectively. Here $\sigma, h, \nu, \rho > 0$ are parameters of the kernels and Γ, B_ν are the Gamma and modified Bessel functions. A convenience the GP framework offers is that posterior distributions are analytically tractable.

Suppose that we are given n observations $\mathcal{D}_n = \{(x_i, y_i)\}_{i=1}^n$ from this GP, where $x_i \in \mathcal{X}$, $y_i = f(x_i) + \epsilon_i \in \mathbb{R}$ and $\epsilon_i \sim \mathcal{N}(0, \eta^2)$. Then the posterior process $f|\mathcal{D}_n$ is also a GP with mean

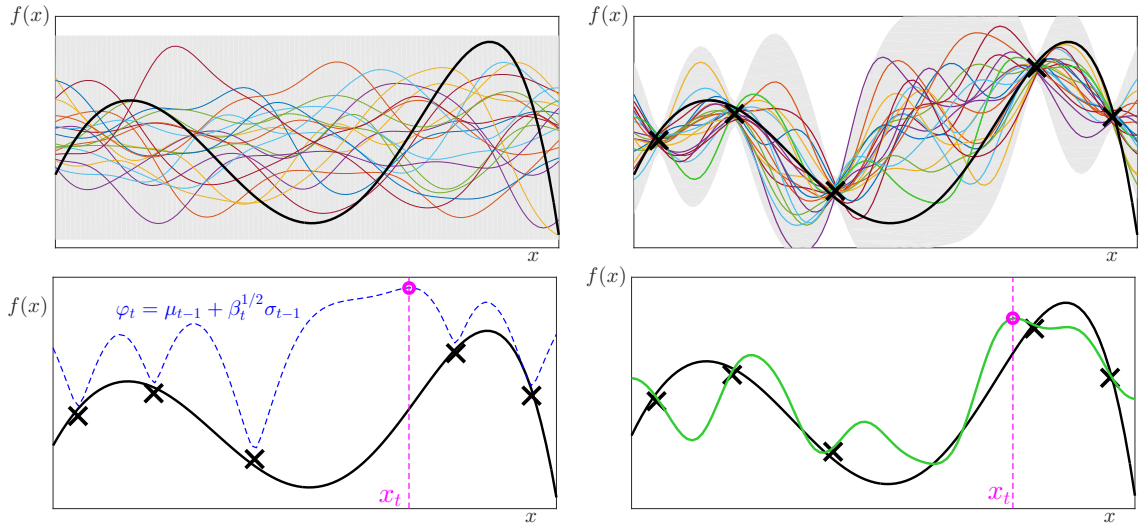


Figure 2.1: An illustration of GPs and BO. The first figure shows the function of interest f (black line) before any observations and illustrates a GP that represents the prior uncertainty. The shaded region represents a 99% confidence region for f and the coloured lines are samples from the GP. The second figure shows some noisy observations (black $\times$'s) of f and the posterior GP conditioned on the observations. The confidence region has shrunk around the observations. In the third figure, we illustrate GP-UCB when we have to pick the next point x_t given observations as shown in the second figure. The GP-UCB acquisition φ_t upper bounds f. At time t we choose the maximiser of φ_t for evaluation, i.e $x_t = \text{argmax}_x \varphi_t(x)$. In the last figure, we illustrate Thompson sampling, where we first sample a function g from the posterior GP and choose its maximiser for evaluation, i.e $x_t = \text{argmax}_x g(x)$.

μ_n and covariance κ_n given by,

$$\mu_n(x) = k^\top (K + \eta^2 I_n)^{-1} Y, \qquad \kappa_n(x, x') = \kappa(x, x') - k^\top (K + \eta^2 I_n)^{-1} k'. \tag{2.1}$$

Here $Y \in \mathbb{R}^n$ is a vector with $Y_i = y_i$, and $k, k' \in \mathbb{R}^n$ are such that $k_i = \kappa(x, x_i), k'_i = \kappa(x', x_i)$. I_n is the $n \times n$ identity matrix. The Gram matrix $K \in \mathbb{R}^{n \times n}$ is given by $K_{i,j} = \kappa(x_i, x_j)$. We have illustrated the prior and posterior GPs in Figure 2.1. We refer the reader to Chapter 2 of Rasmussen and Williams [203] for more on the basics of GPs and their use in regression.

2.2 A Review of Gaussian Process Bandit (Bayesian) Optimisation

Bayesian Optimisation (BO) refers to a suite of methods for bandit optimisation in the Bayesian paradigm which use a prior belief distribution for f. At time t, BO methods use the posterior for f conditioned on the previous observations $\{(x_i, y_i)\}_{i=1}^{t-1}$ to determine where to evaluate f next. Typically, this is done by first constructing an acquisition function $\varphi_t : \mathcal{X} \to \mathbb{R}$ which captures the value of performing an experiment at any given $x \in \mathcal{X}$; then it maximises the acquisition to determine the next point, $x_t = \text{argmax}_{x \in \mathcal{X}} \varphi_t(x)$. In many cases, the ancillary optimisation procedure for the acquisition φ_t can be nontrivial. However, since φ_t is analytically available, it

is usually assumed that the effort for optimising φ_t is negligible when compared to an evaluation of f which requires executing an expensive black box experiment.

While there are several choices for the prior, the most popular option is to use a Gaussian Process. One of the common choices for such acquisition based BO methods is the Gaussian process upper confidence bound (GP-UCB) method of Srinivas et al. [235], where φ_t is defined as,

$$\varphi_t(x) \ = \mu_{t-1}(x) + \beta_t^{1/2}\sigma_{t-1}(x). \tag{2.2}$$

φ_t forms an upper confidence bound for f. Here μ_{t-1} is the posterior mean of the GP after $t-1$ observations and is our current estimate of f. The posterior standard deviation, σ_{t-1}, is the uncertainty associated with this estimate. The μ_{t-1} term encourages an *exploitative* strategy – in that we want to query regions where we already believe f is high – and σ_{t-1} encourages an *exploratory* strategy – in that we want to query where we are uncertain about f lest we miss high valued regions which have not been queried yet. β_t, which is typically increasing with t, controls the trade-off between exploration and exploitation. Algorithm 1 below, summarises GP-UCB.

Algorithm 1 GP-UCB from Srinivas et al. [235]

Input: kernel κ.

- $\mathcal{D}_0 \leftarrow \varnothing$, $(\mu_0, \sigma_0) \leftarrow (\mathbf{0}, \kappa^{1/2})$.
- **for** $t = 1, 2, \ldots$
  1. $x_t \leftarrow \operatorname{argmax}_{x\in\mathcal{X}} \varphi_t(x)$
  2. $y_t \leftarrow$ Query f at x_t.
  3. Perform Bayesian posterior updates to obtain μ_t, σ_t. See (2.1).

Another common BO strategy is Thompson sampling (TS) [246], which stipulates that we sample x_t according to the posterior probability that it is the optimum. That is, x_t is drawn from the posterior density $p_{x_\star}(\cdot|\mathcal{D}_t)$ where $\mathcal{D}_t = \{(x_i, y_i)\}_{i=1}^{t-1}$ is the history of query-observation pairs up to step t. For GPs, this allows for a very simple and elegant algorithm. Observe that we can write $p_{x_\star}(x|\mathcal{D}_t) = \int p_{x_\star}(x|g)\, p(g|\mathcal{D}_t)\mathrm{d}g$, and that $p_{x_\star}(\cdot|g)$ puts all its mass at the maximiser $\operatorname{argmax}_x g(x)$ of g. Therefore, at step t, we draw a sample g from the posterior for f conditioned on our past observations $\{(x_i, y_i)\}_{i=1}^{t-1}$, and set $x_t = \operatorname{argmax}_x g(x)$ to be the maximiser of g. We then evaluate f at x_t. Here, the drawing of the sample encourages exploration and the maximisation incentivises exploitation. Algorithmically, this is equivalent to replacing step 1 in Algorithm 1 with $x_t \leftarrow \operatorname{argmax}_{x\in\mathcal{X}} g(x)$ where g is drawn from the posterior GP. TS has been previously explored for BO [15, 222], and some recent theoretical advances have characterised the performance of TS [6, 42, 135, 211, 213]. We have illustrated GP-UCB and TS in Figure 2.1.

Summary of Theoretical Results for BO

For what follows, we summarise some theoretical results for BO when $\mathcal{X}$ is a Euclidean space. We begin by defining the *Maximum Information Gain* (MIG) which characterises the statistical difficulty of GP bandits.

Definition 1. *(Maximum Information Gain [235]) Let $f \sim \mathcal{GP}(\mathbf{0}, \kappa)$. Consider any $A \subset \mathbb{R}^d$ and let $A' = \{x_1, \ldots, x_n\} \subset A$ be a finite subset. Let $f_{A'}, \epsilon_{A'} \in \mathbb{R}^n$ such that $(f_{A'})_i = f(x_i)$ for $i = 1, \ldots, n$ and $(\epsilon_{A'})_i \sim \mathcal{N}(0, \eta^2)$. Let $y_{A'} = f_{A'} + \epsilon_{A'}$. Denote the Shannon Mutual Information by I. The Maximum Information Gain of the set A is*

$$\Psi_n(A) = \max_{A' \subset A, |A'| = n} I(y_{A'}; f_{A'}).$$

The MIG depends on the kernel and the set A. It is known that for the SE kernel, $\Psi_n([0,1]^d) \in \mathcal{O}(d^d(\log(n))^{d+1})$ and for the Matérn kernel with parameter ν [203], $\Psi_n([0,1]^d) \in \mathcal{O}(n^{\frac{d(d+1)}{2\nu + d(d+1)}} \log(n))$ [216, 235]. For a given kernel it typically scales with the volume of A, i.e. $\Psi_n(A) \propto \Psi_n([0,1]^d)\mathrm{vol}(A)$. Next, when $\mathcal{X}$ is a Euclidean subset, we will need the following regularity condition on the kernel. It is satisfied for four times differentiable kernels such as the SE kernel and Matérn kernel when $\nu > 2$ [72].

Assumption 1. *Let $f \sim \mathcal{GP}(\mathbf{0}, \kappa)$, where $\kappa : [0,r]^d \times [0,r]^d \to \mathbb{R}$ is a stationary kernel [203]. The partial derivatives of f satisfy the following condition. There exist constants $a, b > 0$ s.t.*

$$\text{for all } J > 0, \text{ and for all } i \in \{1, \ldots, d\}, \quad \mathbb{P}\left(\sup_x \left|\frac{\partial f(x)}{\partial x_i}\right| > J\right) \leq a e^{-(J/b)^2}.$$

For simplicity, our presentation will mostly focus on the simple regret S_n (1.1) in this manuscript. In *all* cases, this bound is obtained by first bounding the cumulative regret R_n and then using the bound $S_n \leq \frac{1}{n} R_n$, which follows from the fact that the minimum is smaller than the average. Theorem 1 below states a high probability regret bound for GP-UCB. Theorem 2 states an expected regret bound for GP-UCB and TS.

Theorem 1. *(Theorems 1 and 2 in [235]) Let $f \sim \mathcal{GP}(\mathbf{0}, \kappa)$, $f : \mathcal{X} \to \mathbb{R}$ and the kernel κ satisfies Assumption 1). At each query, we have noisy observations $y = f(x) + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \eta^2)$. Denote $C_1 = 8/\log(1 + \eta^{-2})$. Pick a failure probability $\delta \in (0,1)$. The following bounds on the simple regret S_n hold with probability $> 1 - \delta$ for all $n \geq 1$.*

- *If $\mathcal{X}$ is a finite discrete set, run* GP-UCB *with $\beta_t = 2\log(|\mathcal{X}| t^2 \pi^2 / 6\delta)$. Then,*

$$\text{for all } n \geq 1, \qquad S_n \leq \sqrt{\frac{C_1 \beta_n \Psi_n(\mathcal{X})}{n}}$$

- *If $\mathcal{X} \subset [0,1]^d$, run* GP-UCB *with $\beta_t = 2\log\left(\frac{2\pi^2 t^2}{3\delta}\right) + 2d\log\left(t^2 bdr\sqrt{\frac{4ad}{\delta}}\right)$. Then,*

$$\text{for all } n \geq 1, \qquad S_n \leq \sqrt{\frac{C_1 \beta_n \Psi_n(\mathcal{X})}{n}} + \frac{2}{n}$$

Theorem 2. *(Proposition 5 in [211], Theorem 11 in [129]) Assume the same conditions and quantities as in Theorem 1. Assume β_t is set for* GP-UCB *for the discrete and continuous settings as described below. If we run either* GP-UCB *or* TS *we have the following bounds on the simple regret S_n which holds in expectation for all $n \geq 1$.*

- *If $\mathcal{X}$ is a finite discrete set, let $\beta_t = \log(t^2|\mathcal{X}|/\sqrt{2\pi})$. Then,*

$$\text{for all } n \geq 1, \qquad \mathbb{E}[S_n] \leq \sqrt{\frac{C_1 \beta_n \Psi_n(\mathcal{X})}{n}}$$

- *Let $\mathcal{X} \subset [0,1]^d$ be compact and let $\beta_t = 4(d+1)\log(n) + 2d\log(dab\sqrt{\pi})$. Then*

$$\text{for all } n \geq 1, \qquad \mathbb{E}[S_n] \leq \sqrt{\frac{C_1 \beta_n \Psi_n(\mathcal{X})}{n}} + \frac{4}{n}$$

It is worth noting that while the regret can be bound for GP-UCB both in high probability and in expectation, for TS only bounds in expectation are possible. This is typical of bandit analyses in the Bayesian setting. However, note that GP-UCB requires setting the parameter β_t which might depend on unknown problem-dependent constants. In contrast, the bound in Theorem 2 holds for TS for any sequence $\{\beta_t\}_{t\geq 0}$ such that $|f - \mu_{t-1}| \leq \beta_t^{1/2}\sigma_{t-1}$ holds for all $t \geq 1$.

We finally note that Bayesian optimisation has also been studied using a variety of other techniques such as expected improvement (GP-EI), probability of improvement, and entropy search [27, 94, 113, 176].

2.3 A Review of K-armed Bandits

Since the seminal work by Robbins [207], the multi-armed bandit problem has been studied extensively in the K-armed setting. We briefly review the problem set up and describe the UCB algorithm, which has inspired the plurality of the bandit literature, including GP-UCB.

In the K-armed bandit, we have a set $\mathcal{X} = \{1, \ldots, K\}$ of K arms where each $k \in \mathcal{K}$ corresponds to a probability distribution θ_k with $\mathbb{E}_{\theta_k}[X] = \mu_k$. At time step t, the decision-maker selects an arm $x_t \in \mathcal{X}$ and observes $y_t \sim \theta_k$. This model is consistent with our formalism above where $f(k) = \mu_k$ and upon playing arm x_t, the decision maker observes $f(x_t) + \epsilon$ where $\mathbb{E}[\epsilon] = \mathbb{E}[y_t - f(x_t)] = 0$. Let $x_\star \in \operatorname{argmax}_{k\in\mathcal{X}} \mu_k$ be an optimal arm and $\mu_\star = \max_{k\in\mathcal{X}} \mu_k$ be the optimal value. In typical applications, the goal of the decision-maker is to devise an adaptive strategy for playing each arm at each time step so as to minimise the cumulative regret $R_n = \sum_{t=1}^n \mu_\star - \mu_k$.

Assume that the distributions of each arm satisfied a sub-Gaussian condition of the form $\mathbb{P}(|\mu_k - Y| > \epsilon) \leq \nu e^{-\sigma\epsilon^2}$. In the UCB algorithm, at each time step we maintain an upper confidence bound $\mathcal{B}_{k,t-1}$ for arm k which is defined as follows.

$$\hat{\mu}_{k,t-1} = \frac{1}{T_k(t-1)} \sum_{j:x_j=k} y_j, \qquad \mathcal{B}_{k,t-1} = \hat{\mu}_{k,t-1} + \sqrt{\frac{\rho \log(t)}{T_k(t-1)}}.$$

Here $T_k(t-1)$ denotes the number of times we have already played arm k in the first $t-1$ steps, $\hat{\mu}_{k,t-1}$ is an estimate of the sample mean of arm k at time t, and $\mathcal{B}_{k,t-1}$ is the upper confidence

bound. ρ is a parameter of the algorithm. As in GP-UCB, the UCB algorithm for K-armed bandits suggests that we play the arm with the highest upper confidence bound at time t, i.e. $x_t = \text{argmax}_{k \in \mathcal{K}} \mathcal{B}_{k,t-1}$.

The following theorem upper bounds the cumulative regret for UCB in expectation. We will denote $\Delta_k = \mu_\star - \mu_k$.

Theorem 3 (Regret of UCB [10, 25]). *Assume that the reward distributions are σ-sub-Gaussian and that $\rho > 2$. Then, the regret for* UCB *satisfies,*

$$\mathbb{E}[R_n] \leq \sum_{k:\Delta_k > 0} \Big(\rho \frac{4\sigma^2}{\Delta_k} + \frac{\rho}{\rho - 2} \Big).$$

Note that unlike Theorems 1 and 2, this is not a Bayesian bound and the regret depends on parameters of the problem, such as Δ_k.

2.4 Some Useful Theoretical Results

We state some common theoretical results that are used repeatedly in our proofs.

Results on Gaussian Processes

The first is a standard result on Gaussian concentration.

Lemma 4 (Gaussian Concentration). *Let $Z \sim \mathcal{N}(0,1)$. Then $\mathbb{P}(Z > \epsilon) \leq \frac{1}{2}\exp(-\epsilon^2/2)$.*

The next result is an expression for the Information Gain in a GP from Srinivas et al. [235].

Lemma 5 (Mutual Information in GPs, Lemma 5.3 [235]). *Let $f \sim \mathcal{GP}(\mathbf{0}, \kappa)$, $f : \mathcal{X} \to \mathbb{R}$ and we observe $y = f(x) + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \eta^2)$. Let A be a finite subset of $\mathcal{X}$ and f_A, y_A be the function values and observations on this set respectively. Using the basic Gaussian properties it can be shown that the mutual information $I(y_A; f_A)$ is,*

$$I(y_A; f_A) = \frac{1}{2} \sum_{t=1}^{n} \log(1 + \eta^{-2} \sigma_{t-1}^2(x_t)).$$

where σ_{t-1}^2 is the posterior GP variance after observing the first $t-1$ points.

The following two results are on the supremum of a Gaussian process when $f \sim \mathcal{GP}(\mathbf{0}, \kappa)$ and κ is well behaved. The first states that the supremum of a GP is bounded in expectation; it is satisfied when κ is twice differentiable. The second assigns positive probability to the event that the supremum of a GP in a bounded domain is smaller than any given $\epsilon > 0$.

Lemma 6 (Adler [4]). *Let $f \sim \mathcal{GP}(0, \kappa)$ have continuous sample paths. Then, $\mathbb{E}\|f\|_\infty = \Xi < \infty$.*

Lemma 7 (Theorem 4 in [72]). *Let $\mathcal{X} = [0, r]^d$ and $f \sim \mathcal{GP}(\mathbf{0}, \kappa)$. Let $\kappa : \mathcal{X} \times \mathcal{X}^d \to \mathbb{R}$ be continuous. Then, for all $\epsilon > 0$, there exists $Q(\epsilon) > 0$ such that,*

$$\mathbb{P}\Big(\sup_{x \in \mathcal{X}} |f(x)| < \epsilon\Big) > Q(\epsilon).$$

Some Information Theoretic Results

The following result is a version of Pinsker's inequality.

Lemma 8 (Pinsker's inequality). *Let $X, Z \in \mathcal{X}$ be random quantities and $f : \mathcal{X} \to [0, B]$. Then, $\big|\mathbb{E}[f(X)] - \mathbb{E}[f(Z)]\big| \leq B\sqrt{\frac{1}{2}\mathrm{KL}(P(X)\|P(Z))}$.*

The next, taken from Russo and Van Roy [213], relates the KL divergence to the mutual information for two random quantities X, Y.

Lemma 9 (Russo and Van Roy [213], Fact 6). *For random quantities $X, Z \in \mathcal{X}$, $I(X; Z) = \mathbb{E}_X[\mathrm{KL}(P(Y|X)\|P(Y))]$.*

The next result is a property of the Shannon mutual information.

Lemma 10. *Let X, Y, Z be random quantities such that Y is a deterministic function of X. Then, $I(Y; Z) \leq I(X; Z)$.*

Proof. Let Y' capture the remaining randomness in X so that $X = Y \cup Y'$. Then, since conditioning reduces entropy, $I(Y; Z) = H(Z) - H(Z|Y) \leq H(Z) - H(Z|Y \cup Y') = I(X; Z)$. □

Sub-Gaussian and Sub-exponential Random Variables

We first introduce the notion of sub-Gaussianity, which characterises one of the stronger types of tail behaviour for random variables.

Definition 2 (Sub-Gaussian Random Variables). *A zero mean random variable is said to be τ* sub-Gaussian *if it satisfies, $\mathbb{E}[e^{\lambda X}] \leq e^{\frac{\tau^2\lambda^2}{2}}$ for all $\lambda \in \mathbb{R}$.*

It is well known that Normal $\mathcal{N}(0, \zeta^2)$ variables are ζ sub-Gaussian and bounded random variables with support in $[a, b]$ are $(b - a)/2$ sub-Gaussian. For sub-Gaussian random variables, we have the following important and well known result.

Lemma 11 (Sub-Gaussian Tail Bound). *Let $X_1, \ldots, X_n$ be zero mean independent random variables such that X_i is σ_i sub-Gaussian. Denote $S_n = \sum_{i=1}^n X_i$ and $\sigma^2 = \sum_{i=1}^n \sigma_i^2$. Then, for all*

$\epsilon > 0$,

$$\mathbb{P}\left(S_n \geq \epsilon\right) \;\leq\; \exp\left(\frac{-\epsilon^2}{2\sigma^2}\right), \quad \mathbb{P}\left(S_n \leq \epsilon\right) \;\leq\; \exp\left(\frac{-\epsilon^2}{2\sigma^2}\right).$$

The following result shows that Lipschitz functions of Gaussian random variables are sub-Gaussian, see Theorem 5.6 in Boucheron et al. [19].

Lemma 12 (Gaussian Lipschitz Concentration [19])**.** *Let $X \in \mathbb{R}^n$ such that $X_i \sim \mathcal{N}(0, \zeta^2)$ iid for $i = 1, \ldots, n$. Let $F : \mathbb{R}^n \to \mathbb{R}$ be an L-Lipschitz function, i.e. $|F(x) - F(y)| \leq L\|x - y\|_2$ for all $x, y \in \mathbb{R}^n$. Then, for all $\lambda > 0$, $\mathbb{E}[\exp^{\lambda F(X)}] \leq \exp\left(\frac{\pi^2 L^2 \zeta^2}{8}\lambda^2\right)$. That is, $F(X)$ is $\frac{\pi L \zeta}{2}$ sub-Gaussian.*

We next introduce Sub-exponential random variables, which have a different tail behavior.

Definition 3 (Sub-exponential Random Variables)**.** *A zero mean random variable is said to be* sub-exponential *with parameters (τ^2, b) if it satisfies, $\mathbb{E}[e^{\lambda X}] \;\leq\; e^{\frac{\tau^2\lambda^2}{2}}$ for all λ with $|\lambda| \leq 1/b$.*

Sub-exponential random variables are a special case of Sub-Gamma random variables (See Chapter 2.4 in Boucheron et al. [19]) and allow for a Bernstein-type inequality.

Proposition 13 (Sub-exponential tail bound [19])**.** *Let $X_1, \ldots, X_n$ be independent sub-exponential random variables with parameters (σ_i^2, b_i). Denote $S_n = \sum_{i=1}^n X_i$ and $\sigma^2 = \sum_{i=1}^n \sigma_i^2$ and $b = \max_i b_i$. Then, for all $\epsilon > 0$,*

$$\mathbb{P}\left(\left|S_n - \sum_{i=1}^n \mu_i\right| \geq \sqrt{2\sigma^2 t} + bt\right) \leq 2\exp(-t).$$

Chapter 3

High Dimensional Bandits

In this chapter we consider settings where $\mathcal{X}$ is a compact subset of $\mathbb{R}^d$. While bandit techniques such as Bayesian optimisation have been successful in many applications, they have been used mostly in low dimensional (typically $d < 10$) settings [264]. Expensive high dimensional functions occur in several fields such as computer vision [277], antenna design [99], computational astrophysics [191] and biology [77]. Scaling BO methods to high dimensions for practical problems has been challenging. Even current theoretical results suggest that BO is exponentially difficult in high dimensions without further assumptions [27, 235]. To the best of our knowledge, the only approach to date has been to perform regular BO on a low dimensional subspace. This works only under strong assumptions.

In Kandasamy et al. [120], we identify two key challenges in scaling BO to high dimensions. The first is the *statistical challenge* in estimating the function. Nonparametric regression is inherently difficult in high dimensions with known lower bounds depending exponentially in d [87]. This sample complexity for regression is invariably reflected in the regret bounds for BO. The second is the *computational challenge* in maximising φ_t. Commonly used global optimisation heuristics used to maximise φ_t themselves require computation exponential in dimension. We show that we can overcome both challenges by modeling f as an additive function. The contributions of this chapter are as follows.

1. We propose using additive models for f to combat the curse of dimensionality in high dimensional BO. Theoretically, we show that the simple regret for GP-UCB has only linear dependence on the dimension d when f is additive.

2. We propose the Add-GP-UCB algorithm for Bayesian optimisation when f is additive; the Add-GP-UCB acquisition function is easy to optimise in high dimensions. Empirically we demonstrate that Add-GP-UCB outperforms naive BO on synthetic experiments, an astrophysical simulator and the Viola and Jones face detection problem. Critically, Add-GP-UCB does well on several examples when the function is not additive.

Related Work

Prior to our work in Kandasamy et al. [120], most literature for BO in high dimensions are in the setting where the function varies only along a very low dimensional subspace [37, 53, 264]. In these works, the authors do not encounter either the statistical or computational challenge as they perform BO in either a random or carefully selected lower dimensional subspace. However, assuming that the problem is an easy (low dimensional) one hiding in a high dimensional space is often too restrictive. Indeed, our experimental results confirm that such methods perform poorly on real applications when the assumptions are not met. While our additive assumption is strong in its own right, it is considerably more expressive. It is more general than the setting in Chen et al. [37]. Even though it does not contain the settings in Djolonga et al. [53], Wang et al. [264], unlike them, we still allow the function to vary along the entire domain.

Using additive structure is standard in high dimensional regression[54, 88, 117, 204]. Using an additive model has several advantages even if f is *not* additive. It is well understood that when we only have a few samples, using a simpler model to fit our data may give us a better trade off for estimation error against approximation error. This observation is *crucial*: in many applications for bandit optimisation we are forced to work in the low sample regime since calls to the black box are expensive. Though the additive assumption is biased for non-additive functions, it enables us to do well with only a few samples. Finally, we mention that several follow up work on high dimensional BO (e.g. [69, 153, 209, 257]) has built on our ideas here.

3.1 Additive Models for High Dimensional Bandits

Key structural assumption: In order to make progress in high dimensions, we will assume that f decomposes into the following additive form,

$$f(x) = f^{(1)}(x^{(1)}) + f^{(2)}(x^{(2)}) + \cdots + f^{(M)}(x^{(M)}). \tag{3.1}$$

Here each $x^{(j)} \in \mathcal{X}^{(j)} = [0,1]^{p_j}$ are lower dimensional components. We will refer to the $\mathcal{X}^{(j)}$'s as *groups* and the grouping of different dimensions into these groups $\{\mathcal{X}^{(j)}\}_{j=1}^{M}$ as the *decomposition*. The groups are *disjoint* – i.e. if we treat the coordinates as a set, $x^{(i)} \cap x^{(j)} = \varnothing$. We are primarily interested in the case when d is very large and the group dimensionality is bounded: $p_j \leq p \ll d$. Paranthesised superscripts index the groups and a union over the groups denotes the reconstruction of the whole from the groups (e.g. $x = \bigcup_j x^{(j)}$ and $\mathcal{X} = \bigcup_j \mathcal{X}^{(j)}$).

In keeping with the BO literature, we will assume that each $f^{(j)}$ is sampled from a GP, $\mathcal{GP}(\mathbf{0}, \kappa^{(j)})$ where the $f^{(j)}$'s are independent. Here, $\kappa^{(j)} : \mathcal{X}^{(j)} \times \mathcal{X}^{(j)} \to \mathbb{R}$ is the covariance for $f^{(j)}$. This implies that f itself is sampled from a GP with an additive kernel $\kappa(x, x') = \sum_j \kappa^{(j)}(x^{(j)}, x^{(j)\prime})$. We will call a kernel such as $\kappa^{(j)}$ which acts only on p variables a p^{th} order kernel. A kernel which acts on all the variables is a d^{th} order kernel.

3.1.1 Algorithms

A natural first inclination given (3.1) is to try GP-UCB with an additive kernel. Since an additive kernel is simpler than a d^{th} order kernel, we can expect statistical gains. We formalise this via the theorem below which provides a better bound for the MIG for additive kernels.

Theorem 14 (Kandasamy et al. [120]: MIG for additive kernels)**.** *Let f satisfy (3.1).*

1. *If each $\kappa^{(j)}$ is an at most p^{th} order SE kernel, then $\Psi_n([0,1]^d) \in \mathcal{O}(dp^p(\log n)^{p+1})$.*
2. *If each $\kappa^{(j)}$ is an at most p^{th} order Matérn kernel, then $\Psi_n([0,1]^d) \in \mathcal{O}(d2^p n^{\frac{p(p+1)}{2\nu+p(p+1)}}\log(n))$.*

However, the main challenge in directly using GP-UCB is that optimising φ_t in high dimensions can be computationally prohibitive in practice. For example, using any grid search or branch and bound method, maximising a function to within ζ accuracy, requires $\mathcal{O}(\zeta^{-D})$ calls to φ_t. To circumvent this, in Kandasamy et al. [120] we proposed Add-GP-UCB which exploits the additive structure in f to construct an alternative acquisition function. For this we first describe inferring the individual $f^{(j)}$'s using observations from f.

Inference in Additive GPs: Suppose we are given observations $Y = \{y_1, \ldots, y_n\}$ at $X = \{x_1, \ldots, x_n\}$, where $y_i = f(x_i) + \epsilon$ and $\epsilon \sim \mathcal{N}(0, \eta^2)$. For Add-GP-UCB, we will need the distribution of $f^{(j)}(x_*^{(j)})$ conditioned on X, Y, which can be shown to be the following Gaussian.

$$f^{(j)}(x_*^{(j)})|x_*, X, Y \sim \mathcal{N}\big({k^{(j)}}^\top (K+\eta^2 I_n)^{-1} Y \qquad \kappa^{(j)}(x_*^{(j)}, x_*^{(j)}) - {k^{(j)}}^\top (K+\eta^2 I_n)^{-1} k^{(j)}\big) \tag{3.2}$$

where $k^{(j)} \in \mathbb{R}^n$ are such that $k_i^{(j)} = \kappa^{(j)}(x, x_i)$. I_n is the $n \times n$ identity matrix. The Gram matrix $K \in \mathbb{R}^{n\times n}$ is given by $K_{i,j} = \kappa(x_i, x_j) = \sum_j \kappa^{(j)}(x_i, x_j)$.

Add-GP-UCB[1]: We define the Add-GP-UCB acquisition φ_t as,

$$\varphi_t(x) = \sum_{j=1}^{M} \mu_{t-1}^{(j)}(x^{(j)}) + \beta_t^{1/2}\sigma_{t-1}^{(j)}(x^{(j)}). \tag{3.3}$$

φ_t can be maximised by maximising $\mu_{t-1}^{(j)} + \beta_t^{1/2}\sigma_{t-1}^{(j)}$ separately on $\mathcal{X}^{(j)}$. As we need to solve M at most p dimensional optimisation problems, it requires only $\mathcal{O}(M^{p+1}\zeta^{-p})$ calls in total to optimise within ζ accuracy – far more favourable than maximising φ_t. We summarise the resulting procedure below in Algorithm 2.

3.1.2 Practical Considerations

Our practical implementation differs from the above description in the following aspects.

[1]Theorem 5 in Kandasamy et al. [120] states that Add-GP-UCB achieves essentially the same regret as GP-UCB with an additive kernel. However, post publication we discovered a mistake in our analysis.

Algorithm 2 Add-GP-UCB from Kandasamy et al. [120]

Input: Kernels $\kappa^{(1)}, \ldots, \kappa^{(M)}$, Decomposition $(\mathcal{X}^{(j)})_{j=1}^{M}$

- $\mathcal{D}_0 \leftarrow \varnothing$,
- **for** $j = 1, \ldots, M$, $\quad (\mu_0^{(j)}, \kappa_0^{(j)}) \leftarrow (\mathbf{0}, \kappa^{(j)})$.
- **for** $t = 1, 2, \ldots$
  1. **for** $j = 1, \ldots, M$, $\quad \mathbf{x}_t^{(j)} \leftarrow \text{argmax}_{z \in \mathcal{X}^{(j)}} \mu_{t-1}^{(j)}(z) + \beta_t^{1/2} \sigma_{t-1}^{(j)}(z)$.
  2. $\mathbf{x}_t \leftarrow \bigcup_{j=1}^{M} \mathbf{x}_t^{(j)}$.
  3. $\mathbf{y}_t \leftarrow$ Query f at $\mathbf{x}_t$.
  4. $\mathcal{D}_t = \mathcal{D}_{t-1} \cup \{(\mathbf{x}_t, \mathbf{y}_t)\}$.
  5. Obtain posterior for $\mu_t^{(j)}, \sigma_t^{(j)}$ for $j = 1, \ldots, M$ conditioned on $\mathcal{D}_t$. See (3.2)

Choice of β_t: β_t as specified by Theorem 1, usually tends to be conservative in practice [235]. For good empirical performance a more aggressive strategy is required. In our experiments, we set $\beta_t = 0.2p \log(2t)$ which offered a good tradeoff between exploration and exploitation.

Data dependent prior: Our analysis assumes that we know the GP kernel of the prior. In reality this is rarely the case. In our experiments, we choose the hyperparameters of the kernel by maximising the GP marginal likelihood [203] periodically.

Initialisation: Marginal likelihood based kernel tuning can be unreliable with few data points. This is a problem in the first few iterations. Following the recommendations in Bull [27] we initialise Add-GP-UCB (and GP-UCB) using 50 points selected uniformly at random.

Decomposition and non-additive functions: If f is additive and the decomposition is known, we use it directly. But it may not always be known or f may not be additive. Then, we could treat the decomposition as a hyperparameter of the additive kernel and maximise the marginal likelihood w.r.t the decomposition. However, given that there are $d!/p!^M M!$ possible decompositions, computing the marginal likelihood for all of them is infeasible. We circumvent this issue by randomly selecting a few ($\mathcal{O}(d)$) decompositions and choosing the one with the largest marginal likelihood. Intuitively, if the function is not additive, with such a "partial maximisation" we can hope to capture some existing marginal structure in f. At the same time, even an exhaustive maximisation will not do much better than a partial maximisation if there is no additive structure. Empirically, we found that partially optimising for the decomposition performed slightly better than using a fixed decomposition or a random decomposition at each step. We incorporate this procedure for finding an appropriate decomposition as part of the kernel hyperparameter learning procedure.

In all the experiments in this chapter, we keep d fixed and demonstrate Add-GP-UCB for different values of d. In Chapter 7, we describe more robust choices for tuning the additive decomposition, along with othe hyperparameters of the kernel which we used in Dragonfly.

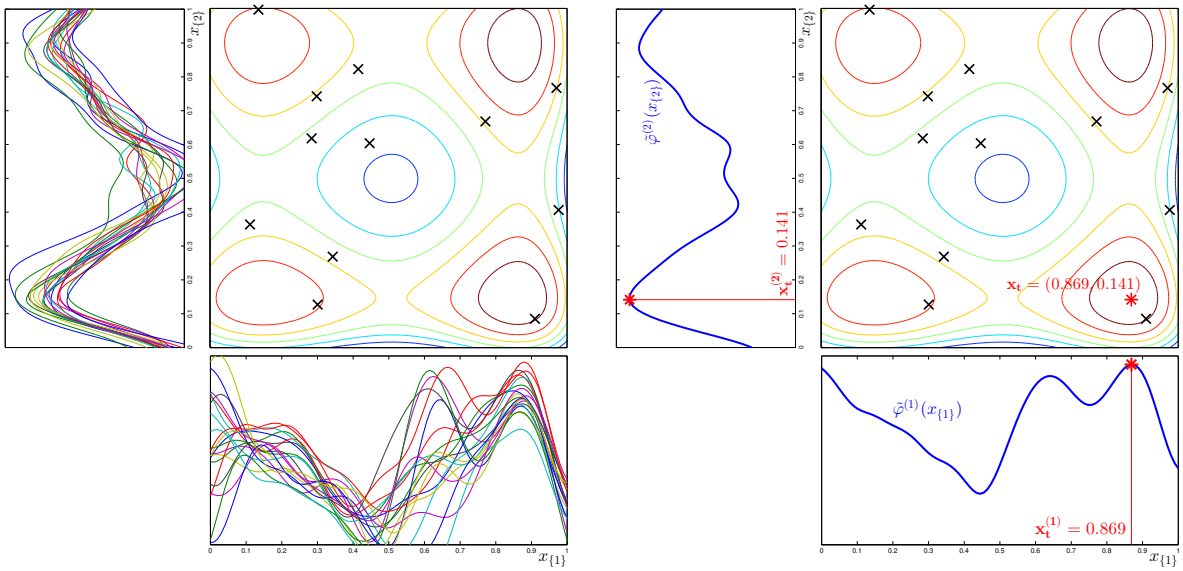


Figure 3.1: An illustration of Add-GP-UCB on a two dimensional function $f^{(2)} = f^{(1)}(x^{(1)}) + f^{(2)}(x^{(2)})$ where $x^{(1)} = \{x_1\}$ and $x^{(2)} = \{x_2\}$. Suppose we have already evaluated at the points shown via $\times$. We construct posteriors for $f^{(1)}$ and $f^{(2)}$ separately (illustrated via the coloured samples). Then we maximise the individual upper confidence bounds on each $f^{(j)}$ and combine them to obtain the next point x_t.

3.2 Experiments

To demonstrate the efficacy of Add-GP-UCB over BO we optimise the acquisition function under a constrained budget. Following, Brochu et al. [23] we use DiRect to maximise $\varphi_t, \widetilde{\varphi}_t$. We compare Add-GP-UCB against GP-UCB, random querying (RAND) and DiRect[2]. On the real datasets we also compare it to the Expected Improvement (GP-EI) acquisition function which is popular in BO applications and the method of Wang et al. [264] which uses a random projection before applying BO (REMBO). We have multiple instantiations of Add-GP-UCB for different values for (p, M). We provide results for both the simple regret S_T and the time averaged cumulative regret R_T/T. We use SE kernels for each additive kernels and use the same scale σ and bandwidth h hyperparameters for all the kernels. Every 25 iterations we maximise the marginal likelihood with respect to these 2 hyperparameters in addition to the decomposition.

Synthetic High Dimensional Experiments

First we demonstrate our technique on a series of synthetic examples. For this we construct additive functions for different values for the maximum group size p' and the number of groups M'. We use the prime to distinguish it from Add-GP-UCB instantiations with different combinations

[2]There are several optimisation methods based on simulated annealing, cross entropy and genetic algorithms. We use DiRect since its easy to configure and known to work well in practice.

of (d, M) values. The p' dimensional function $f_{p'}$ is,

$$f_d(x) = \log\left(0.1\frac{1}{h_{p'}^{p'}}\exp\left(\frac{\|x-v_1\|^2}{2h_{p'}^2}\right) + \quad 0.1\frac{1}{h_{p'}^{p'}}\exp\left(\frac{\|x-v_2\|^2}{2h_{p'}^2}\right) + 0.8\frac{1}{h_{p'}^{p'}}\exp\left(\frac{\|x-v_3\|^2}{2h_{p'}^2}\right)\right) \tag{3.4}$$

where v_1, v_2, v_3 are fixed p' dimensional vectors and $h_{p'} = 0.01{p'}^{0.1}$. Then we create M' groups of coordinates by randomly adding p' coordinates into each group. On each such group we use $f_{p'}$ and then add them up to obtain the composite function f. Precisely, $f(x) = f_{p'}(x^{(1)}) + \cdots + f_{p'}(x^{(M)})$. The remaining $D - p'M'$ coordinates do not contribute to the function. Since $f_{p'}$ has 3 modes, f will have $3^{M'}$ modes.

In the synthetic experiments we use an instantiation of Add-GP-UCB that knows the decomposition – i.e. $(p, M) = (p', M')$ and the grouping of coordinates. We refer to this as Add-$\star$. For the rest we use a (p, M) decomposition by creating M groups of size at most p and find a good grouping by partially maximising the marginal likelihood as described previously. We refer to them as Add-p/M. For non-additive BO methods, we allocate a budget of $b = \min(5000, 100D)$ DiRect function evaluations to optimise the acquisition function. For all Add-p/M methods, we maximise each group under a budget b/M so as to allow a fair comparison.

The results are given in Figures 3.2-3.4. We refer to each example by the configuration of the additive function–its (D, p', M') values. In the $(10, 3, 3)$ example Add-$\star$ does best since it knows the correct model and the acquisition function can be maximised within the budget. However Add-$3/4$ and Add-$5/2$ models do well too and outperform GP-UCB. Add-$1/10$ performs poorly since it is statistically not expressive enough to capture the true function. In the $(24, 11, 2)$, $(40, 18, 2)$, $(40, 35, 1)$, $(96, 29, 3)$ and $(120, 55, 2)$ examples Add-$\star$ outperforms GP-UCB. However, it is not competitive with the Add-p/M for small p. Even though Add-$\star$ knew the correct decomposition, there are two possible failure modes since p' is large. The kernel is complex and the estimation error is very high in the absence of sufficient data points. In addition, optimising the acquisition is also difficult. This illustrates our previous argument that using an additive kernel can be advantageous even if the function is not additive or the decomposition is not known. In the $(24, 6, 4)$, $(40, 5, 8)$ and $(96, 5, 19)$ examples Add-$\star$ performs best as p' is small enough. But again, almost all Add-p/M instantiations outperform GP-UCB. In contrast to the small D examples, for large D, GP-UCB and Add-p/M with large d perform worse than DiRect. This is probably because our budget for maximising φ_t is inadequate to optimise the acquisition function to sufficient accuracy. For some of the large D examples the cumulative regret is low for Add-GP-UCB and Add-p/M with large d. This is probably since they have already started exploiting where as the Add-p/M with small d methods are still exploring. We posit that if we run for more iterations we will be able to see the improvements.

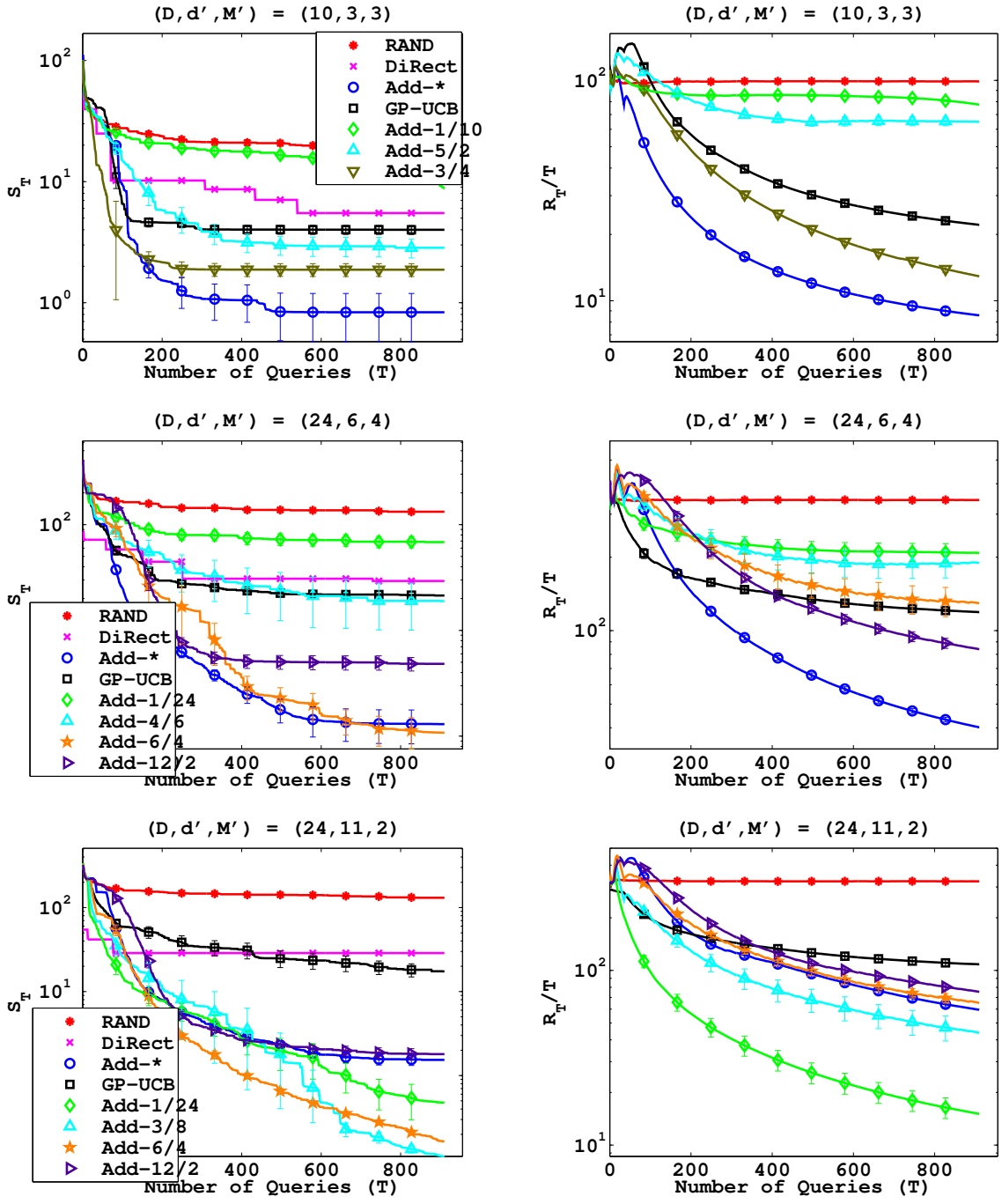


Figure 3.2: Results on the synthetic datasets for high dimensional Bandits. In all figures the x-axis is the number of queries and the y-axis is the regret in log scale. We have indexed each experiment by their (p, p', M') values. The first column is the simple regret S_n for the experiments with (p, p', M') set to $(10, 3, 3)$, $(24, 6, 4)$, and $(24, 11, 2)$. The second column is R_T/T for the same experiments. In some figures, the error bars are not visible since they are small and hidden by the bullets. All figures were produced by averaging over 20 independent runs.

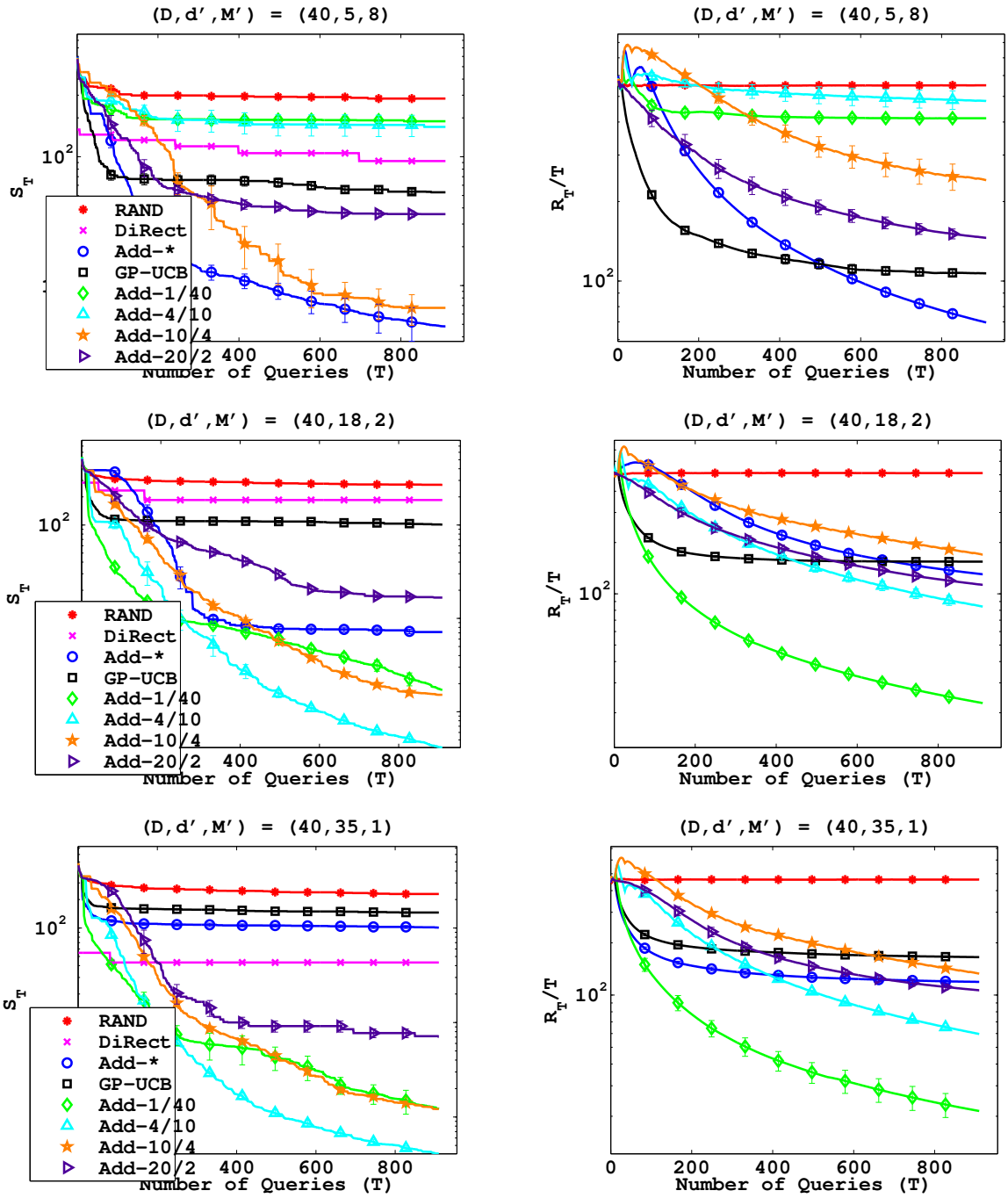


Figure 3.3: Results on the synthetic datasets for high dimensional Bandits. See caption in Figure 3.2 for details. The first column is the simple regret S_n for the experiments with (p, p', M') set to $(40, 5, 8)$, $(40, 18, 2)$, and $(40, 35, 1)$. The second column is R_T/T for the same experiments.

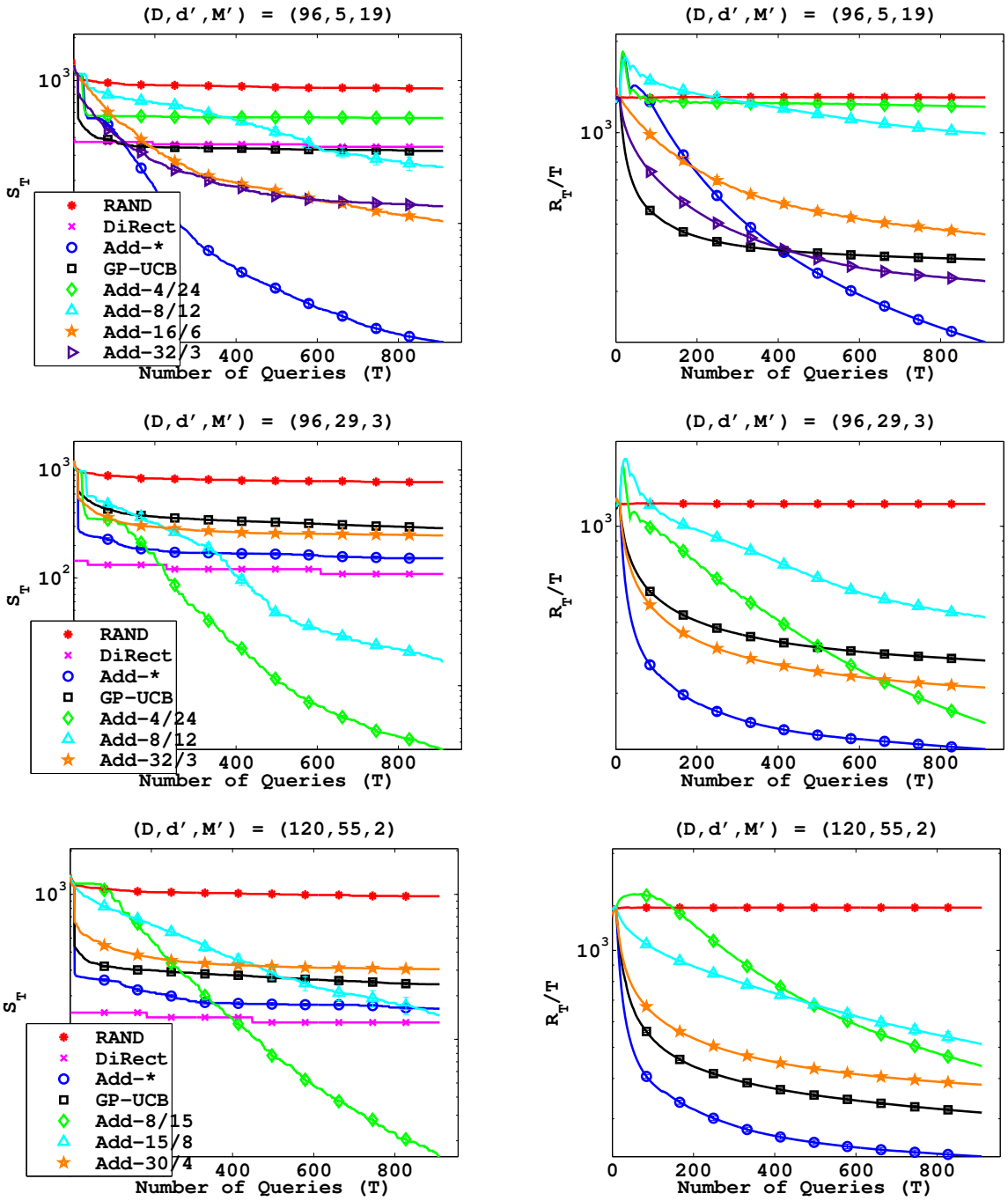
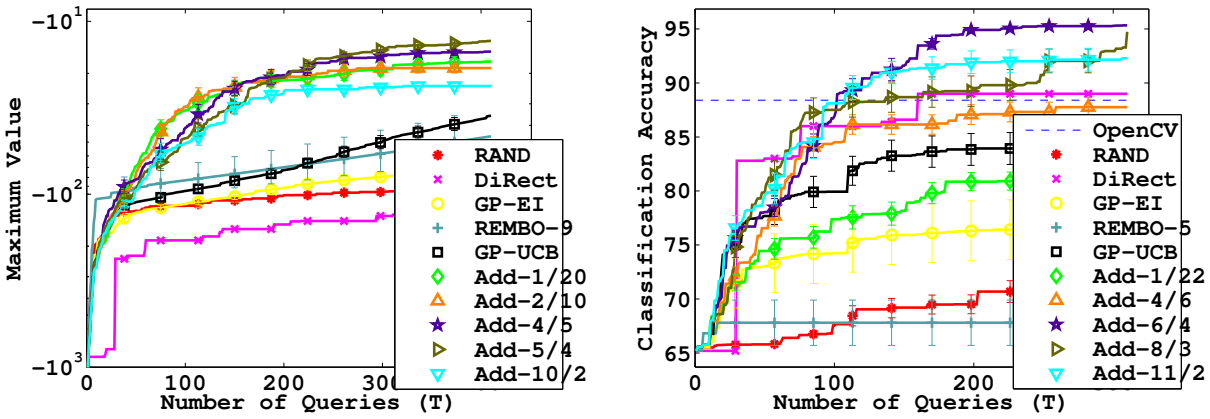


Figure 3.4: Results on the synthetic datasets for high dimensional Bandits. See caption in Figure 3.2 for details. The first column is the simple regret S_n for the experiments with (p, p', M') set to $(96, 5, 19)$, $(96, 29, 3)$, and $(120, 55, 2)$. The second column is R_T/T for the same experiments.



(a) (b)

Figure 3.5: Results on the Astrophysical experiment (a) and the Viola and Jones dataset (b). The x-axis is the number of queries and the y-axis is the maximum value. All figures were produced by averaging over atleast 15 runs.

SDSS Astrophysical Dataset

Here we used Galaxy data from the Sloan Digital Sky Survey (SDSS). The task is to find the maximum likelihood estimators for a simulation based astrophysical likelihood model. Data and software for computing the likelihood are taken from Tegmark et al [244]. The software itself takes in only 9 parameters but we augment this to 20 dimensions to emulate the fact that in practical astrophysical problems we may not know the true parameters on which the problem is dependent. This also allows us to effectively demonstrate the superiority of our methods over alternatives.

We have shown the Maximum value obtained over 400 iterations of each algorithm in Figure 3.5(a). Note that RAND outperforms DiRect here since a random query strategy is effectively searching in 9 dimensions. Despite this advantage to RAND all BO methods do better. Moreover, despite the fact that the function may not be additive, all Add-p/M methods outperform GP-UCB. Since the function only depends on 9 parameters we use REMBO with a 9 dimensional projection. Yet, it is not competitive with the Add-p/M methods. Possible reasons for this may include the scaling of the parameter space by $\sqrt{d}$ in REMBO and the imperfect optimisation of the acquisition function. Here Add-$5/4$ performs slightly better than the rest since it seems to have the best tradeoff between being statistically expressive enough to capture the function while at the same time be easy enough to optimise the acquisition function within the allocated budget.

Viola & Jones Face Detection

The Viola & Jones (VJ) Cascade Classifier [254] is a popular method for face detection in computer vision based on the Adaboost algorithm. The K-cascade has K weak classifiers which outputs a score for any given image. When we wish to classify an image we pass that image

through each classifier. If at any point the score falls below a certain threshold the image is classified as negative. If the image passes through all classifiers then it is classified as positive. The threshold values at each stage are usually pre-set based on prior knowledge. There is no reason to believe that these threshold values are optimal. In this experiment we wish to find an optimal set of values for these thresholds by optimising the classification accuracy over a training set.

For this task, we use 1000 images from the Viola & Jones face dataset containing both face and non-face images. We use the implementation of the VJ classifier that comes with OpenCV [22] which uses a 22-stage cascade and modify it to take in the threshold values as a parameter. As our domain $\mathcal{X}$ we choose a neighbourhood around the configuration given in OpenCV. Each function call takes about 30-40 seconds and is the the dominant cost in this experiment. We use 1000 DiRect evaluations to optimise the acquisition function for GP-UCB, GP-EI and REMBO and 900 for the Add-p/M instantiations. Since we do not know the structure of the function we use REMBO with a 5 dimensional projection. The results are given in Figure 3.5(b). Not surprisingly, REMBO performs worst as it is only searching on a 5 dimensional space. Barring Add-$1/22$ all other instantiations perform better than GP-UCB and GP-EI with Add-$6/4$ performing the best. Interestingly, we also find a value for the thresholds that outperform the configuration used in OpenCV.

3.3 Proofs of Theoretical Results

We prove Theorem 14 from Chapter 3.1. For this we will use the following result taken from Seeger et al. [216].

Theorem 15. *(Bound on Information Gain, [216]) Suppose that $\mathcal{X}$ is compact and κ is a kernel on d dimensions satisfying Assumption 1. Choose any $\tau > 0$, and let $N_n = C_9 n^\tau \log(n)$ where $C_9 = 4p + 2$. For any $n_* \in \{1, \ldots, \min(n, N_n)\}$, let $B_\kappa(n_*) = \sum_{s>n_*} \lambda_s$. Here $(\lambda_i)_{i\in\mathbb{N}}$ are the eigenvalues of κ w.r.t the uniform distribution over $\mathcal{X}$. Then,*

$$\Psi_n \leq \inf_\tau \Big(\frac{1/2}{1-e^{-1}} \cdot \max_{r\in\{1,\ldots,n\}} \big(n_* \log(rN_n/\eta^2) + C_9\eta^2(1-r/n)(n^{\tau+1}B_\kappa(n_*)+1)\log n \big) + \mathcal{O}(n^{1-\tau/p}) \Big).$$

Proof of Theorem 14-1 (Additive SE Kernel)

Proof. We will use bounds on the eigenvalues for the simple squared exponential kernel given in [216]. It was shown that the eigenvalues $\{\lambda_s^{(i)}\}$ of $\kappa^{(i)}$ satisfied $\lambda_s^{(i)} \leq c^p B^{s^{1/p_i}}$ where $B < 1$ (See Remark 1). Since the kernel is additive, and $x^{(i)} \cap x^{(j)} = \varnothing$ the eigenfunctions corresponding to $\kappa^{(i)}$ and $\kappa^{(j)}$ will be orthogonal. Hence the eigenvalues of κ will just be the union of the

eigenvalues of the individual kernels, i.e. $\{\lambda_s\} = \bigcup_{j=1}^{M}\{\lambda_s^{(j)}\}$. As $B < 1$, $\lambda_s^{(i)} \leq c^p B^{s^{1/p}}$. Let $n_+ = \lfloor n_*/M \rfloor$ and $\alpha = -\log B$. Then,

$$\begin{aligned} B_\kappa(n_*) = \sum_{s>n_*} \lambda_s \leq Mc \sum_{s>n_+} B^{s^{1/p}} \leq c^p M \left(B^{n_+^{1/p}} + \int_{n_+}^{\infty} \exp(-\alpha x^{1/p}) \right) \mathrm{d}x \\ \leq c^p M \left(B^{n_+^{1/p}} + p\alpha^{-p}\Gamma(p, \alpha n_+^{1/p}) \right) \leq c^p M e^{-\alpha n_+^{1/p}} \left(1 + p!p\alpha^{-p}(\alpha n_+^{1/p})^{p-1} \right). \end{aligned}$$

The last step holds true whenever $\alpha n_+^{1/p} \geq 1$. Here in the second step we bound the series by an integral and in the third step we used the substitution $y = \alpha x^{1/p}$ to simplify the integral. Here $\Gamma(s,x) = \int_x^\infty t^{s-1}e^{-t}\mathrm{d}t$ is the (upper) incomplete Gamma function. In the last step we have used the following identity and the bound for integral s and $x \geq 1$

$$\Gamma(s,x) = (s-1)!e^{-x}\sum_{k=0}^{s-1}\frac{x^k}{k!} \leq s!e^{-x}x^{d-1}.$$

By using $\tau = p$ and by using $n_* \leq (M+1)n_+$, we use Theorem 15 to obtain the following bound on Ψ_n,

$$\begin{aligned} \Psi_n \leq \frac{1/2}{1-e^{-1}} \max_{r\in\{1,\ldots,T\}} \Bigg((M+1)n_+ \log(rN_n/\eta^2) + \\ C_9\eta^2(1-r/n)\log n \left(1 + c^p M e^{-\alpha n_+^{1/p}} n^{p+1} \left(1 + p!p\alpha^{-p}(\alpha n_+^{1/p})^{p-1} \right) \right) \Bigg). \end{aligned} \tag{3.5}$$

Now we need to pick n_+ so as to balance these two terms. We will choose $n_+ = \left(\frac{\log(nN_n)}{\alpha}\right)^p$ which is less than $\min(n, N_n)/M$ for sufficiently large n. Then $e^{-\alpha n_+^{1/p}} = 1/nN_n$. Then the first term S_1 inside the paranthesis is,

$$\begin{aligned} S_1 = (M+1)\log^p\left(\frac{nN_n}{\alpha}\right)\log\left(\frac{rN_n}{\eta^2}\right) &\in \mathcal{O}\left(M\left(\log(nN_n)\right)^p \log(rN_n)\right) \\ &\in \mathcal{O}\left(M\left(\log(n^{p+1}\log n)\right)^p \log(rn^p \log n)\right) \\ &\in \mathcal{O}\left(Mp^{p+1}(\log n)^{p+1} + Mp^p(\log n)^p \log(r)\right). \end{aligned}$$

Note that the constant in front has exponential dependence on d but we ignore it since we already have d^d, $(\log n)^d$ terms. The second term S_2 becomes,

$$\begin{aligned} S_2 &= C_9\eta^2(1-r/n)\log n \left(1 + \frac{c^p M}{nN_n} n^{p+1} \left(1 + p!p\alpha^{-p}(\log(nN_n)^{p-1}) \right) \right) \\ &\leq C_9\eta^2(1-r/n) \left(\log n + \frac{c^p M}{C_9} \left(1 + p!p\alpha^{-p}(\log(nN_n)^{p-1}) \right) \right) \\ &\leq C_9\eta^2(1-r/n) \left(\mathcal{O}(\log n) + \mathcal{O}(1) + \mathcal{O}(p!p^p(\log n)^{p-1}) \right) \Bigg) \\ &\in \mathcal{O}\left((1-r/n)p!p^p(\log n)^{p-1}\right). \end{aligned}$$

Since S_1 dominates S_2, we should choose $r = n$ to maximise the RHS in (3.5). This gives us,

$$\Psi_n \ \in \ \mathcal{O}\left(Mp^{p+1}(\log n)^{p+1}\right) \ \in \ \mathcal{O}\left(Dp^p(\log n)^{p+1}\right).$$

□

Proof of Theorem 14-2 (Additive Matérn Kernel)

Proof. Once again, we use bounds given in [216]. It was shown that the eigenvalues $\{\lambda_s^{(i)}\}$ for $\kappa^{(i)}$ satisfied $\lambda_s^{(i)} \leq c^p s^{-\frac{2\nu+p_j}{p_j}}$ (See Remark 1). By following a similar argument to above we have $\{\lambda_s\} = \bigcup_{j=1}^{M}\{\lambda_s^{(j)}\}$ and $\lambda_s^{(i)} \leq c^p s^{-\frac{2\nu+p}{p}}$. Let $n_+ = \lfloor n_*/M \rfloor$. Then,

$$\begin{aligned} B_\kappa(n_*) &= \sum_{s>n_*} \lambda_s \leq Mc^p \sum_{s>n_+} s^{-\frac{2\nu+p}{p}} \\ &\leq Mc^p \left(n_+^{-\frac{2\nu+p}{p}} + \int_{n_+}^{\infty} s^{-\frac{2\nu+p}{p}} \right) \leq C_8 2^p M n_+^{1-\frac{2\nu+p}{p}}. \end{aligned}$$

where C_8 is an appropriate constant. We set $n_+ = (nN_n)^{\frac{p}{2\nu+p}}(\log(TN_n))^{-\frac{p}{2\nu+p}}$ and accordingly we have the following bound on Ψ_n as a function of $n_+ \in \{1, \ldots, \min(n, N_n)/M\}$,

$$\Psi_n \leq \inf_\tau \left(\frac{1/2}{1-e^{-1}} \max_{r\in\{1,\ldots,n\}} \left((M+1)n_+ \log(rN_n/\eta^2) + \right. \right. \tag{3.6}$$
$$\left. \left. C_9\eta^2(1-r/n)\left(\log n + C_8 2^p M n_+ \log(nN_n)\right) \right) + \mathcal{O}(n^{1-\tau/p}) \right).$$

Since this is a concave function on r we can find the optimum by setting the derivative w.r.t r to be zero. We get $r \in \mathcal{O}(n/2^p \log(nN_n))$ and hence,

$$\begin{aligned} \Psi_n &\in \inf_\tau \left(\mathcal{O}\left(Mn_+ \log\left(\frac{nN_n}{2^p \log(nN_n)} \right) \right) + \mathcal{O}\left(M2^p n_+ \log(nN_n)\right) + \mathcal{O}(n^{1-\tau/p}) \right) \\ &\in \inf_\tau \left(\mathcal{O}\left(M2^p \log(nN_n) \left(\frac{n^{\tau+1}\log(n)}{(\tau+1)\log(n) + \log\log n} \right)^{\frac{p}{2\nu+p}} \right) + \mathcal{O}(n^{1-\tau/p}) \right) \\ &\in \inf_\tau \left(\mathcal{O}\left(M2^p \log(nN_n) n^{\frac{(\tau+1)p}{2\nu+p}} \right) + \mathcal{O}(n^{1-\tau/p}) \right) \\ &\in \mathcal{O}\left(M2^p n^{\frac{p(p+1)}{2\nu+p(p+1)}} \log(n) \right). \end{aligned}$$

Here in the second step we have substituted the values for n_+ first and then N_n. In the last step we have balanced the polynomial dependence on n in both terms by setting $\tau = \frac{2\nu p}{2\nu+p(p+1)}$.

□

Remark 1. *The eigenvalues and eigenfunctions for the kernel are defined with respect to a base distribution on $\mathcal{X}$. In the development of Theorem 15, Srinivas et al. [235] draw N_n samples from the uniform distribution on $\mathcal{X}$. Hence, the eigenvalues/eigenfunctions should be w.r.t the uniform distribution. The bounds given in Seeger et al. [216] are for the uniform distribution for the Matérn kernel and a Gaussian Distribution for the Squared Exponential Kernel. For the latter case, Srinivas et al. [235] argue that the uniform distribution still satisfies the required tail constraints and therefore the bounds would only differ up to constants.*

Chapter 4

Multi-fidelity Bandits

Traditionally, the bandit framework and Bayesian optimisation methods have only been studied at "single fidelity" settings. It is assumed that the decision maker has access to just a single expensive function f which needs to be maximised by repeatedly obtaining noisy evaluations. In many practical problems however, cheap approximations to f might be available. These "lower fidelity" approximations can be used to discard regions in $\mathcal{X}$ with low function value. We can then reserve the expensive evaluations for a small promising region.

For instance, when tuning hyperparameters of learning algorithms, the goal is to maximise a cross validation score on a training set, which can be expensive if the training set is large. However validation curves tend to vary smoothly with training set size; therefore, we can train and cross validate on small subsets to approximate the validation accuracies of the entire dataset. For a concrete example, consider kernel density estimation (KDE), where we need to tune the bandwidth h of a kernel when using a dataset of size 3000. Figure 4.1 shows the average cross validation likelihood against h for a dataset of size $n = 3000$ and a smaller subset of size $n = 300$. Since the cross validation performance of a hyperparameter depends on the training set size [252], we can obtain only a biased estimate of the cross validation performance with 3000 points using a subset of size 300. Consequently, the two maximisers are also different. That said, the curve for $n = 300$ approximates the $n = 3000$ curve quite well. Since training and cross validation on small n is cheap, we can use it to eliminate bad values of the hyperparameters and reserve the expensive experiments with the entire dataset for the promising hyperparameter values (for example, boxed region in Figure 4.1).

In the conventional treatment for online advertising, each query to f is, say, the public display

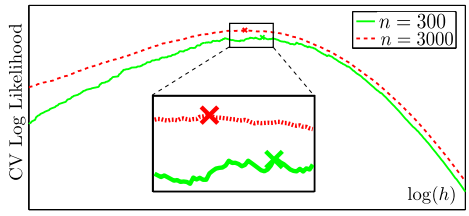


Figure 4.1: Average cross validation log likelihood on datasets of size 300 and 3000 on a synthetic kernel density estimation task. The crosses are the maxima. The maximisers are different since optimal hyperparameters depend on the training set size. That said, the curve for $n = 300$ approximates the $n = 3000$ curve quite well.

of an ad on the internet for a certain time period. However, we could also choose smaller experiments by, say, confining the display to a small geographic region and/or for shorter periods. The estimate is biased, since users in different geographies are likely to have different preferences, but will nonetheless be useful in gauging the all round performance of an ad. In optimal policy search in robotics and autonomous driving, vastly cheaper computer simulations are used to approximate the expensive real world performance of the system [48, 250]. Scientific experiments can be approximated to varying degrees using less expensive data collection, analysis, and computational techniques [191]. In this chapter, we study techniques which can leverage these cheap approximations

Related Work

There has been some interest in multi-fidelity methods for optimisation in many applied domains of research such as hyperparameter tuning and industrial design [60, 100, 139, 197, 241]. However, these works do not formalise or analyse notions of *regret* in the multi-fidelity setting. Multi-fidelity methods are used in the robotics community for reinforcement learning tasks by modeling each fidelity as a Markov decision process [48]. Zhang and Chaudhuri [279] study active learning with a cheap weak labeler and an expensive strong labeler. The objective of these papers however is not to handle the exploration-exploitation trade-off inherent to the bandit setting. Some work has studied multi-fidelity methods for hyperparameter tuning [5, 154, 214]. While some of the above tasks can be framed as optimisation problems, the methods themselves are specific to the problem considered. Our methods here are more general as they apply to any bandit optimisation task. Bogunovic et al. [17] study a version of BO where an algorithm might use cheap, noisy, yet *unbiased* approximations to a function f; but as we will see shortly, this is different to the multi-fidelity problem where the approximations can only provide biased approximations. A line of work on budgeted multi-armed bandits [249, 272] study a variant of the K-armed bandit where each arm has a random reward and cost and the goal is to play the arm with the highest reward/cost ratio as much as possible. This is different from our setting where each arm has multiple fidelities which serve as an approximation.

This chapter describes our line of work [123, 124, 125, 127] on this topic, where we studied multi-fidelity problems under various bandit settings. To the best of our knowledge, this is the first line of work that theoretically formalises and analyses this problem. Following this, some work has extended our ideas here to develop multi-fidelity methods in other settings, both Bayesian [234] and otherwise [217, 218]. This chapter is broken down into three main parts. In Chapter 4.1, we study multi-fidelity bandits with a finite number of approximations in the K-armed setting, in Chapter 4.2, we study Bayesian optimisation with a finite number of approximations, and in Chapter 4.3, we study Bayesian optimisation with a continuous spectrum of approximations. The proofs of the theoretical results in each sub-chapter above is given in Chapters 4.4, 4.5, and 4.6 respectively.

4.1 Multi-fidelity K-armed Bandits

Since the seminal work of Robbins [207], the multi-armed bandit has become an attractive framework for studying exploration-exploitation trade-offs inherent to tasks arising in online advertising, finance and other fields. As described in Chapter 2.3, in its most basic form we have a set $\mathcal{X} = \{1, \ldots, K\}$ of K arms (e.g. K ads in online advertising). At each time step $t = 1, 2, \ldots$, an arm is played and a corresponding reward is realised. The well known (UCB) algorithm [10], achieves regret $\mathcal{O}(K \log(n))$ after n plays (ignoring mean rewards) and is minimax optimal [151].

In this chapter, we study the multi-fidelity version on this problem, where we associate a cost to playing each arm. Moreover, we allow the decision maker to play any arm at a lower cost and obtain a cheap approximation of the desired outcome. For instance, in on-line advertising the goal is to maximise the cumulative number of clicks over a given time period. Conventionally, an arm pull maybe thought of as the display of an ad for a specific time, say one hour. However, we may approximate its hourly performance by displaying the ad for shorter periods. This estimate is *biased* (and possibly noisy), as displaying an ad for longer intervals changes user behaviour. It can nonetheless be useful in gauging the long run click through rate. We can also obtain biased estimates of an ad by displaying it only to certain geographic regions or age groups.

We will refer to such approximations as fidelities. Consider a 2-fidelity problem where the cost at the low fidelity is $\lambda^{(1)}$ and the cost at the high fidelity is $\lambda^{(2)}$. We will present a cost weighted notion of regret for this setting for a strategy that expends a capital of Λ units. A classical K-armed bandit strategy such as UCB, which only uses the highest fidelity, can obtain at best $\mathcal{O}(\lambda^{(2)} K \log(\Lambda/\lambda^{(2)}))$ regret [151]. In contrast, this paper will present multi-fidelity strategies that achieve $\mathcal{O}\left((\lambda^{(1)} K + \lambda^{(2)} |\mathcal{X}_g|) \log(\Lambda/\lambda^{(2)})\right)$ regret. Here $\mathcal{X}_g$ is a (typically) small subset of arms with high expected reward that can be identified using plays at the (cheaper) low fidelity. When $|\mathcal{X}_g| < K$ and $\lambda^{(1)} < \lambda^{(2)}$, such a strategy will outperform the more standard UCB algorithms. Intuitively, this is achieved by using the lower fidelities to eliminate several of "bad" arms and reserving expensive higher fidelity plays for a small subset of the most promising arms. We formalise the above intuitions in the sequel. Our main contributions in this section are,

1. A novel bandit **formalism** when one has access to multiple fidelities for each arm, with each successive fidelity providing a better approximation to the most expensive one.

2. A new **algorithm** that we call Multi-Fidelity Upper Confidence Bound (MF-UCB) that adapts the classical Upper Confidence Bound (UCB) strategies to our multi-fidelity setting. Empirically, we demonstrate that our algorithm outperforms naive UCB on simulations.

3. A **theoretical characterisation** of the performance of MF-UCB that shows that the algorithm (a) uses the lower fidelities to explore all arms and eliminates arms with low expected reward, and (b) reserves the higher fidelity plays for arms with rewards close to the optimal value. We derive a lower bound on the regret and demonstrate that MF-UCB is near-optimal on this problem.

4.1.1 Problem Formalism

We now present our formalism for multi-fidelity bandits in the K-armed setting. Recall that in the classical setting, each arm $k \in \mathcal{X} = \{1, \ldots, K\}$ is associated with a real valued distribution θ_k with mean μ_k. Let $\mathcal{X}_\star = \text{argmax}_{k \in \mathcal{X}} \mu_k$ be the set of optimal arms, $k_\star \in \mathcal{X}_\star$ be an optimal arm and $\mu_\star = \mu_{k_\star}$ denote the optimal mean value. A bandit strategy would *play* an arm $I_t \in \mathcal{X}$ at each time step t and observe a sample from θ_{I_t}. Its goal is to maximise the sum of expected rewards after n time steps $\sum_{t=1}^{n} \mu_{I_t}$, or equivalently minimise the cumulative pseudo-regret $\sum_{t=1}^{n} \mu_\star - \mu_{I_t}$ for *all values* of n. In other words, the objective is to be competitive, in expectation, against an oracle that plays an optimal arm all the time.

In the multi-fidelity set up, we differ from the usual bandit setting in the following aspect. For each arm k, we have access to $M-1$ successively approximate distributions $\theta_k^{(1)}, \theta_k^{(2)}, \ldots, \theta_k^{(M-1)}$ to the desired distribution $\theta_k^{(M)} = \theta_k$. We will refer to these approximations as fidelities. Clearly, these approximations are meaningful only if they give us some information about $\theta_k^{(M)}$. In what follows, we will assume that the m^{th} fidelity mean of an arm is within $\zeta^{(m)}$, a *known quantity*, of its highest fidelity mean, where $\zeta^{(m)}$, decreasing with m, characterise the successive approximations. That is, $|\mu_k^{(M)} - \mu_k^{(m)}| \leq \zeta^{(m)}$ for all $k \in \mathcal{X}$ and $m = 1, \ldots, M$, where $\zeta^{(1)} > \zeta^{(2)} > \cdots > \zeta^{(M)} = 0$ and the $\zeta^{(m)}$'s are known. It is possible for the lower fidelities to be misleading under this assumption: there could exist an arm k with $\mu_k^{(M)} < \mu_\star = \mu_{k_\star}^{(M)}$ but with $\mu_k^{(m)} > \mu_\star$ and/or $\mu_k^{(m)} > \mu_{k_\star}^{(m)}$ for any $m < M$. In other words, we wish to explicitly account for the *biases* introduced by the lower fidelities, and not treat them as just a higher variance observation of an expensive experiment. This problem of course becomes interesting only when lower fidelities are more attractive than higher fidelities in terms of some notion of cost. Towards this end, we will assign a cost $\lambda^{(m)}$ (such as advertising time, money etc.) to playing an arm at fidelity m where $\lambda^{(1)} < \lambda^{(2)} \cdots < \lambda^{(M)}$.

Notation: In this section and Chapter 4.4, $T_{k,t}^{(m)}$ denotes the number of plays at arm k, at fidelity m until t time steps. $T_{k,t}^{(>m)}$ is the number of plays at fidelities greater than m. $Q_t^{(m)} = \sum_{k \in \mathcal{X}} T_{k,t}^{(m)}$ is the number of fidelity m plays at all arms until time t. $\overline{X}_{k,s}^{(m)}$ denotes the mean of s samples drawn from $\theta_k^{(m)}$. Denote $\Delta_k^{(m)} = \mu_\star - \mu_k^{(m)} - \zeta^{(m)}$. When s refers to the number of plays of an arm, we will take $1/s = \infty$ if $s = 0$. $\overline{A}$ denotes the complement of a set $A \subset \mathcal{K}$. While discussing the intuitions in our proofs and theorems we will use $\asymp, \lesssim, \gtrsim$ to denote equality and inequalities ignoring constants.

Regret in the multi-fidelity setting: A strategy for a multi-fidelity bandit problem, at time t, produces an arm-fidelity pair (I_t, m_t), where $I_t \in \mathcal{K}$ and $m_t \in \{1, \ldots, M\}$, and observes a sample X_t drawn (independently of everything else) from the distribution $\theta_{I_t}^{(m_t)}$. The choice of (I_t, m_t) could depend on previous arm-observation-fidelity tuples $\{(I_i, X_i, m_i)\}_{i=1}^{t-1}$. The multi-fidelity setting calls for a new notion of regret. For any strategy $\mathcal{A}$ that expends Λ units of the resource, we will define the pseudo-regret $R(\Lambda, \mathcal{A})$ as follows. Let q_t denote the *instantaneous pseudo-reward* at time t and $r_t = \mu_\star - q_t$ denote the instantaneous pseudo-regret. We will discuss choices for q_t shortly. Any notion of regret in the multi-fidelity setting needs to account for this

instantaneous regret along with the cost of the fidelity at which we played at time t, i.e. $\lambda^{(m_t)}$. In particular, we will view the means of the arms $\mu_k^{(m)}$ and consequently the instantaneous reward q_t as being the reward per unit cost expended. Moreover, we should receive no reward (maximum regret) for any unused capital. These observations lead to the following definition,

$$R(\Lambda, \mathcal{A}) \;=\; \Lambda\mu_\star - \sum_{t=1}^{N} \lambda^{(m_t)} q_t \;=\; \underbrace{\left(\Lambda - \sum_{t=1}^{N} \lambda^{(m_t)}\right)\mu_\star}_{\tilde{r}(\Lambda,\mathcal{A})} \;+\; \underbrace{\sum_{t=1}^{N} \lambda^{(m_t)} r_t}_{\tilde{R}(\Lambda,\mathcal{A})}. \tag{4.1}$$

Above, N is the (random) number of plays within capital Λ by $\mathcal{A}$, i.e. the largest n such that $\sum_{t=1}^{n} \lambda^{(m_t)} \leq \Lambda$. To motivate our choice of q_t we consider an online advertising example where $\lambda^{(m)}$ is the advertising time at fidelity m and $\mu_k^{(m)}$ is the expected number of clicks per unit time. While we observe from $\theta_{I_t}^{(m_t)}$ at time t, we wish to reward the strategy according to its highest fidelity distribution $\theta_{I_t}^{(M)}$. Therefore regardless of which fidelity we play we set $q_t = \mu_{I_t}^{(M)}$. Here, we are competing against an oracle which plays an optimal arm at any fidelity all the time. Note that we might have chosen q_t to be $\mu_{I_t}^{(m_t)}$. However, this does not reflect the motivating applications for the multi-fidelity setting that we consider. For instance, a clickbait ad might receive a high number of clicks in the short run, but its long term performance might be poor. Furthermore, for such a choice, we may as well ignore the rich structure inherent to the multi-fidelity setting and simply play the arm $\operatorname{argmax}_{m,k} \mu_k^{(m)}$ at each time. There are of course other choices for q_t that result in very different notions of regret; we discuss this briefly towards the end of the chapter.

The distributions $\theta_k^{(m)}$ need to be well behaved for the problem to be tractable. We will assume that they satisfy concentration inequalities of the following form. For all $\epsilon > 0$,

$$\forall\, m,\, k, \qquad \mathbb{P}\big(\overline{X}_{k,s}^{(m)} - \mu_k^{(m)} > \epsilon\big) \;<\; \nu e^{-s\psi(\epsilon)}, \qquad \mathbb{P}\big(\overline{X}_{k,s}^{(m)} - \mu_k^{(m)} < -\epsilon\big) \;<\; \nu e^{-s\psi(\epsilon)}. \tag{4.2}$$

Here $\nu > 0$ and ψ is an increasing function with $\psi(0) = 0$ and is at least increasing linearly $\psi(x) \in \Omega(x)$. For example, if the distributions are sub-Gaussian, then $\psi(x) \in \Theta(x^2)$.

The performance of a multi-fidelity strategy which switches from low to high fidelities can be worsened by artificially inserting fidelities. Consider a scenario where $\lambda^{(m+1)}$ is only slightly larger than $\lambda^{(m)}$ and $\zeta^{(m+1)}$ is only slightly smaller than $\zeta^{(m)}$. This situation is unfavourable since there isn't much that can be inferred from the $(m+1)^{\text{th}}$ fidelity that cannot already be inferred from the m^{th} by expending the same cost. We impose the following regularity condition to avoid such situations.

Assumption 2. *The $\zeta^{(m)}$'s decay fast enough such that $\sum_{i=1}^{m} \frac{1}{\psi(\zeta^{(i)})} \leq \frac{1}{\psi(\zeta^{(m+1)})}$ for all $m < M$.*

Assumption 2 is not necessary to analyse our algorithm, however, the performance of MF-UCB when compared to UCB is most appealing when the above holds. In cases where M is small enough and can be treated as a constant, the assumption is not necessary. For sub-Gaussian distributions, the condition is satisfied for an exponentially decaying $(\zeta^{(1)}, \zeta^{(2)}, \dots)$.

Our goal is to design a strategy $\mathcal{A}_0$ that has low expected pseudo-regret $\mathbb{E}[R(\Lambda, \mathcal{A}_0)]$ for all values of (sufficiently large) Λ, i.e. the equivalent of an anytime strategy, as opposed to a fixed time horizon strategy, in the usual bandit setting. The expectation is over the observed rewards which also dictates the number of plays N. From now on, for simplicity we will write $R(\Lambda)$ when $\mathcal{A}$ is clear from context and refer to it just as regret.

4.1.2 The Multi-Fidelity Upper Confidence Bound (MF-UCB) Algorithm

As the name suggests, the MF-UCB algorithm maintains an upper confidence bound corresponding to $\mu_k^{(m)}$ for each $m \in \{1, \ldots, M\}$ and $k \in \mathcal{K}$ based on its previous plays. Following UCB strategies [9, 10], we define the following set of upper confidence bounds,

$$\begin{aligned} \mathcal{B}_{k,t}^{(m)}(s) &= \overline{X}_{k,s}^{(m)} + \psi^{-1}\Big(\frac{\rho \log t}{s}\Big) + \zeta^{(m)}, \quad \text{for all } m \in \{1, \ldots, M\}, \ k \in \mathcal{K} \\ \mathcal{B}_{k,t} &= \min_{m=1,\ldots,M} \mathcal{B}_{k,t}^{(m)}(T_{k,t-1}^{(m)}). \end{aligned} \tag{4.3}$$

Here ρ is a parameter in our algorithm and ψ is from (4.2). Each $\mathcal{B}_{k,t}^{(m)}(T_{k,t-1}^{(m)})$ provides a high probability upper bound on $\mu_k^{(M)}$ with their minimum $\mathcal{B}_{k,t}$ giving the tightest bound (see Chapter 4.4). Similar to UCB, at time t we play the arm I_t with the highest upper bound $I_t = \text{argmax}_{k \in \mathcal{X}} \mathcal{B}_{k,t}$.

Since our setup has multiple fidelities associated with each arm, the algorithm needs to determine at each time t which fidelity (m_t) to play the chosen arm (I_t). For this consider an arbitrary fidelity $m < M$. The $\zeta^{(m)}$ conditions on $\mu_k^{(m)}$ imply a constraint on the value of $\mu_k^{(M)}$. If, at fidelity m, the uncertainty interval $\psi^{-1}(\rho \log(t)/T_{I_t,t-1}^{(m)})$ is large, then we have not constrained $\mu_{I_t}^{(M)}$ sufficiently well yet. There is more information to be gleaned about $\mu_{I_t}^{(M)}$ from playing the arm I_t at fidelity m. On the other hand, playing at fidelity m indefinitely will not help us much since the $\zeta^{(m)}$ elongation of the confidence band caps off how much we can learn about $\mu_{I_t}^{(M)}$ from fidelity m; i.e. even if we knew $\mu_{I_t}^{(m)}$, we will have only constrained $\mu_{I_t}^{(M)}$ to within a $\pm\zeta^{(m)}$ interval. Our algorithm captures this natural intuition. Having selected I_t, we begin checking at the first fidelity. If $\psi^{-1}(\rho \log(t)/T_{I_t,t-1}^{(1)})$ is smaller than a threshold $\gamma^{(1)}$ we proceed to check the second fidelity, continuing in a similar fashion. If at any point $\psi^{-1}(\rho \log(t)/T_{I_t,t-1}^{(m)}) \geq \gamma^{(m)}$, we play I_t at fidelity $m_t = m$. If we go all the way to fidelity M, we play at $m_t = M$. The resulting procedure is summarised below in Algorithm 3.

Algorithm 3 MF-UCB from Kandasamy et al. [125]

- for $t = 1, 2, \ldots$
  1. Choose $I_t \in \text{argmax}_{k \in \mathcal{X}} \mathcal{B}_{k,t}$. (See equation (4.3).)
  2. $m_t = \min_m \{ m \mid \psi^{-1}(\rho \log t / T_{I_t,t-1}^{(m)}) \geq \gamma^{(m)} \ \vee \ m = M \}$ (See equation (4.4).)
  3. Play $X \sim \theta_{I_t}^{(m_t)}$.

Choice of $\gamma^{(m)}$: In our algorithm, we choose

$$\gamma^{(m)} = \psi^{-1}\left(\frac{\lambda^{(m)}}{\lambda^{(m+1)}}\psi\big(\zeta^{(m)}\big)\right) \tag{4.4}$$

To motivate this choice, note that if $\Delta_k^{(m)} = \mu_\star - \mu_k^{(m)} - \zeta^{(m)} > 0$ then we can conclude that arm k is not optimal. Step 2 of the algorithm attempts to eliminate arms for which $\Delta_k^{(m)} \gtrsim \gamma^{(m)}$ from plays above the m^{th} fidelity. If $\gamma^{(m)}$ is too large, then we would not eliminate a sufficient number of arms whereas if it was too small we could end up playing a suboptimal arm k (for which $\mu_k^{(m)} > \mu_\star$) too many times at fidelity m. As will be revealed by our analysis, the given choice represents an optimal tradeoff under the given assumptions.

4.1.3 Theoretical Analysis

We will be primarily concerned with the term $\tilde{R}(\Lambda, \mathcal{A}) = \tilde{R}(\Lambda)$ from (4.1). $\tilde{r}(\Lambda, \mathcal{A})$ is a residual term; it is an artefact of the fact that after the $N+1^{\text{th}}$ play, the spent capital would have exceeded Λ. For any algorithm that operates oblivious to a fixed capital, it can be bounded by $\lambda^{(M)}\mu_\star$ which is negligible compared to $\tilde{R}(\Lambda)$. Accordingly, we have the following expressions for $\tilde{R}(\Lambda)$:

$$\tilde{R}(\Lambda) \;=\; \sum_{k\in\mathcal{X}} \Delta_k^{(M)} \left(\sum_{m=1}^{M} \lambda^{(m)} T_{k,N}^{(m)}\right), \tag{4.5}$$

Central to our analysis will be the following partitioning of $\mathcal{X}$. First denote the set of arms whose fidelity m mean is within η of $\mu_\star$ to be $\mathcal{J}_\eta^{(m)} = \{k \in \mathcal{X};\, \mu_\star - \mu_k^{(m)} \le \eta\}$. Define $\mathcal{X}^{(1)} \triangleq \overline{\mathcal{J}}^{(1)}_{\zeta^{(1)}+2\gamma^{(1)}} = \{k \in \mathcal{X};\, \Delta_k^{(1)} > 2\gamma^{(1)}\}$ to be the arms whose first fidelity mean $\mu_k^{(1)}$ is at least $\zeta^{(1)} + 2\gamma^{(1)}$ below the optimum $\mu_\star$. Then we recursively define,

$$\begin{aligned}
\mathcal{X}^{(m)} &\triangleq \overline{\mathcal{J}}^{(m)}_{\zeta^{(m)}+2\gamma^{(m)}} \cap \left(\bigcap_{\ell=1}^{m-1} \mathcal{J}^{(\ell)}_{\zeta^{(\ell)}+2\gamma^{(\ell)}}\right), \; \forall m \le M-1, \\
\mathcal{X}^{(M)} &\triangleq \overline{\mathcal{X}_\star} \cap \left(\bigcap_{\ell=1}^{M-1} \mathcal{J}^{(\ell)}_{\zeta^{(\ell)}+2\gamma^{(\ell)}}\right).
\end{aligned} \tag{4.6}$$

Observe that for all $k \in \mathcal{X}^{(m)}$, $\Delta_k^{(m)} > 2\gamma^{(m)}$ and $\Delta_k^{(\ell)} \le 2\gamma^{(\ell)}$ for all $\ell < m$. For what follows, for any $k \in \mathcal{X}$, $[\![k]\!]$ will denote the partition k belongs to, i.e. $[\![k]\!] = m$ s.t. $k \in \mathcal{X}^{(m)}$. We will see that $\mathcal{X}^{(m)}$ are the arms that will be played at the m^{th} fidelity but can be excluded from fidelities higher than m using information at fidelity m. See Figure 4.2 for an illustration of these partitions.

Regret Bound for MF-UCB

Recall that $N = \sum_{m=1}^{M} Q_N^{(m)}$ is the total (random) number of plays by a multi-fidelity strategy within capital Λ. Let $n_\Lambda = \lfloor \Lambda/\lambda^{(M)} \rfloor$ be the (non-random) number of plays by any strategy that

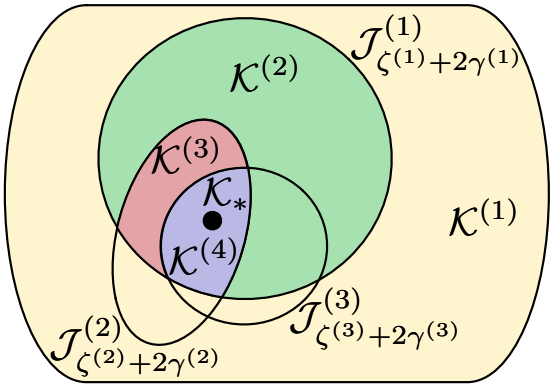


Figure 4.2: Illustration of the partition $\mathcal{X}^{(m)}$'s for a $M = 4$ fidelity problem. The sets $\mathcal{J}^{(m)}_{\zeta^{(m)}+2\gamma^{(m)}}$ are indicated next to their boundaries. $\mathcal{X}^{(1)}, \mathcal{X}^{(2)}, \mathcal{X}^{(3)}, \mathcal{X}^{(4)}$ are shown in yellow, green, red and purple respectively. The optimal arms $\mathcal{X}_\star$ are shown as a black circle.

operates only on the highest fidelity. Since $\lambda^{(m)} < \lambda^{(M)}$ for all $m < M$, N could be large for an arbitrary multi-fidelity method. However, our analysis reveals that for MF-UCB, $N \lesssim n_\Lambda$ with high probability. The following theorem bounds R for MF-UCB. The proof is given in Chapter 4.4.2. For clarity, we ignore the constants but they are fleshed out in the proofs.

Theorem 16 (Regret Bound for MF-UCB)**.** *Let $\rho > 4$. There exists Λ_0 depending on $\lambda^{(m)}$'s such that for all $\Lambda > \Lambda_0$,* MF-UCB *satisfies,*

$$\frac{\mathbb{E}[R(\Lambda)]}{\log(n_\Lambda)} \lesssim \sum_{k \notin \mathcal{X}_\star} \Delta_k^{(M)} \cdot \frac{\lambda^{(\llbracket k \rrbracket)}}{\psi(\Delta_k^{(\llbracket k \rrbracket)})} \asymp \sum_{m=1}^{M} \sum_{k \in \mathcal{X}^{(m)}} \Delta_k^{(M)} \frac{\lambda^{(m)}}{\psi(\Delta_k^{(m)})}$$

Let us compare the above bound to UCB whose regret is $\frac{\mathbb{E}[R(\Lambda)]}{\log(n_\Lambda)} \asymp \sum_{k \notin \mathcal{X}_\star} \Delta_k^{(M)} \frac{\lambda^{(M)}}{\psi(\Delta_k^{(M)})}$. We will first argue that MF-UCB does not do significantly worse than UCB in the worst case. Modulo the $\Delta_k^{(M)} \log(n_\Lambda)$ terms, regret for MF-UCB due to arm k is $R_{k,\text{MF-UCB}} \asymp \lambda^{(\llbracket k \rrbracket)}/\psi(\Delta_k^{(\llbracket k \rrbracket)})$. Consider any $k \in \mathcal{X}^{(m)}$, $m < M$ for which $\Delta_k^{(m)} > 2\gamma^{(m)}$. Since

$$\Delta_k^{(M)} \leq \Delta_k^{(\llbracket k \rrbracket)} + 2\zeta^{(\llbracket k \rrbracket)} \lesssim \psi^{-1}\Big(\frac{\lambda^{(\llbracket k \rrbracket+1)}}{\lambda^{(\llbracket k \rrbracket)}} \psi(\Delta_k^{(\llbracket k \rrbracket)})\Big),$$

a (loose) lower bound for UCB for the same quantity is $R_{k,\text{UCB}} \asymp \lambda^{(M)}/\psi(\Delta_k^{(M)}) \gtrsim \frac{\lambda^{(M)}}{\lambda^{(\llbracket k \rrbracket+1)}} R_{k,\text{MF-UCB}}$. Therefore for any $k \in \mathcal{X}^{(m)}, m < M$, MF-UCB is at most a constant times worse than UCB. However, whenever $\Delta_k^{(\llbracket k \rrbracket)}$ is comparable to or larger than $\Delta_k^{(M)}$, MF-UCB outperforms UCB by a factor of $\lambda^{(\llbracket k \rrbracket)}/\lambda^{(M)}$ on arm k. As can be inferred from the theorem, most of the cost invested by MF-UCB on arm k is at the $\llbracket k \rrbracket^{\text{th}}$ fidelity. For example, in Fig. 4.2, MF-UCB would not play the yellow arms $\mathcal{X}^{(1)}$ beyond the first fidelity (more than a constant number of times). Similarly all green and red arms are played mostly at the second and third fidelities respectively. Only the blue arms are played at the fourth (most expensive) fidelity. On the other hand UCB plays all arms at the fourth fidelity. Since lower fidelities are cheaper MF-UCB achieves better regret than UCB.

It is essential to note here that $\Delta_k^{(M)}$ is small for arms in in $\mathcal{X}^{(M)}$. These arms are close to the optimum and require more effort to distinguish than arms that are far away. MF-UCB, like UCB

, invests $\log(n_\Lambda)\lambda^{(M)}/\psi(\Delta_k^{(M)})$ capital in those arms. That is, the multi-fidelity setting does not help us significantly with the "hard-to-distinguish" arms. That said, in cases where K is very large and the sets $\mathcal{X}^{(M)}$ is small the bound for MF-UCB can be appreciably better than UCB.

Lower Bound

Since, $N \geq n_\Lambda = \lfloor \Lambda/\lambda^{(M)} \rfloor$, any multi-fidelity strategy which plays a suboptimal arm a polynomial number of times at any fidelity after n time steps, will have worse regret than MF-UCB (and UCB). Therefore, in our lower bound we will only consider strategies which satisfy the following condition.

Assumption 3. *Consider the strategy after n plays at any fidelity. For any arm with $\Delta_k^{(M)} > 0$, we have $\mathbb{E}[\sum_{m=1}^{M} T_{k,n}^{(m)}] \in o(n^a)$ for any $a > 0$.*

For our lower bound we will consider a set of Bernoulli distributions $\theta_k^{(m)}$ for each fidelity m and each arm k with mean $\mu_k^{(m)}$. It is known that for Bernoulli distributions $\psi(\epsilon) \in \Theta(\epsilon^2)$. To state our lower bound we will further partition the set $\mathcal{X}^{(m)}$ into two sets $\mathcal{X}_{\checkmark}^{(m)}, \mathcal{X}_{\times}^{(m)}$ as follows,

$$\mathcal{X}_{\checkmark}^{(m)} = \{k \in \mathcal{X}^{(m)} \ : \Delta_k^{(\ell)} \leq 0 \ \forall \ell < m\}, \qquad \mathcal{X}_{\times}^{(m)} = \{k \in \mathcal{X}^{(m)} \ : \exists\, \ell < m \ \text{s.t.}\ \Delta_k^{(\ell)} > 0\}. \tag{4.7}$$

For an arm $k \in \mathcal{X}^{(m)}$, our lower bound, given below, is different depending on whether k belongs to $\mathcal{X}_{\checkmark}^{(m)}$ or $\mathcal{X}_{\times}^{(m)}$.

Theorem 17 (Lower bound for $R(\Lambda)$)**.** *Consider any set of Bernoulli reward distributions with $\mu_\star \in (1/2, 1)$ and $\zeta^{(1)} < 1/2$. Then, for any strategy satisfying Assumption 3 the following holds.*

$$\liminf_{\Lambda\to\infty} \frac{\mathbb{E}[R(\Lambda)]}{\log(n_\Lambda)} \ \geq \ c \cdot \sum_{m=1}^{M} \left[\sum_{k\in\mathcal{X}_{\checkmark}^{(m)}} \Delta_k^{(M)} \frac{\lambda^{(m)}}{{\Delta_k^{(m)}}^2} \ + \ \sum_{k\in\mathcal{X}_{\times}^{(m)}} \Delta_k^{(M)} \min_{\ell\in\mathcal{L}_m(k)} \frac{\lambda^{(\ell)}}{{\Delta_k^{(\ell)}}^2} \right] \tag{4.8}$$

Here c is a problem dependent constant. $\mathcal{L}_m(k) = \{\ell < m : \Delta_k^{(\ell)} > 0\} \cup \{m\}$ is the union of the m^{th} fidelity and all fidelities smaller than m for which $\Delta_k^{(\ell)} > 0$.

Comparing this with Theorem 16 we find that MF-UCB meets the lower bound on all arms $k \in \mathcal{X}_{\checkmark}^{(m)}$, $\forall m$. However, it may be loose on any $k \in \mathcal{X}_{\times}^{(m)}$. The gap can be explained as follows. For $k \in \mathcal{X}_{\times}^{(m)}$, there exists some $\ell < m$ such that $0 < \Delta_k^{(\ell)} < 2\gamma^{(\ell)}$. As explained previously, the switching criterion of MF-UCB ensures that we do not invest too much effort trying to distinguish whether $\Delta_k^{(\ell)} < 0$ since $\Delta_k^{(\ell)}$ could be very small. That is, we proceed to the next fidelity only if we cannot conclude $\Delta_k^{(\ell)} \lesssim \gamma^{(\ell)}$. However, since $\lambda^{(m)} > \lambda^{(\ell)}$ it might be the case that $\lambda^{(\ell)}/{\Delta_k^{(\ell)}}^2 < \lambda^{(m)}/{\Delta_k^{(m)}}^2$ even though $\Delta_k^{(m)} > 2\gamma^{(m)}$. Consider for example a two fidelity problem where $\Delta = \Delta_k^{(1)} = \Delta_k^{(2)} < 2\sqrt{\lambda^{(1)}/\lambda^{(2)}}\zeta^{(1)}$. Here it makes sense to

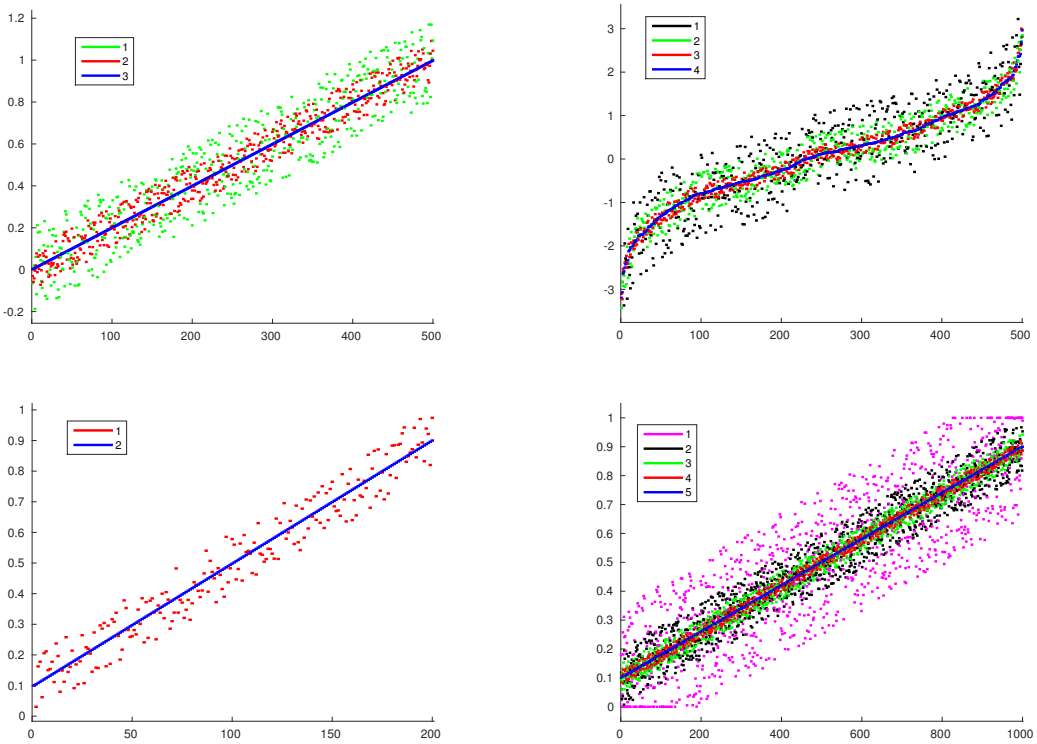


Figure 4.3: An illustration of the means of the arms used the simulation problems. The top row are the Gaussian reward problems with (K, M) equal to $(500, 3)$, $(500, 4)$ while the second row are the Bernoulli rewards with $(200, 2)$, $(1000, 5)$ respectively.

distinguish the arm as being suboptimal at the first fidelity with $\lambda^{(1)} \log(n_\Lambda)/\Delta^2$ capital instead of $\lambda^{(2)} \log(n_\Lambda)/\Delta^2$ at the second fidelity. However, MF-UCB distinguishes this arm at the higher fidelity as $\Delta < 2\gamma^{(m)}$ and therefore does not meet the lower bound on this arm. While it might seem tempting to switch based on estimates for $\Delta_k^{(1)}, \Delta_k^{(2)}$, this idea is not desirable as estimating $\Delta_k^{(2)}$ for an arm requires $\log(n_\Lambda)/\psi(\Delta_k^{(2)})$ samples at the second fidelity; this is is exactly what we are trying to avoid for the majority of the arms via the multi-fidelity setting. We leave it as an open problem to resolve this gap.

4.1.4 Experiments

We compare UCB against MF-UCB on a series of synthetic problems which were generated as follows. Denote $\vec{\zeta} = (\zeta^{(1)}, \zeta^{(2)}, \ldots, \zeta^{(M)})$ and $\vec{\lambda} = (\lambda^{(1)}, \lambda^{(2)}, \ldots, \lambda^{(M)})$.

1. Gaussian: $M = 500$, $M = 3$, $\vec{\zeta} = (0.2, 0.1, 0)$, $\vec{\lambda} = (1, 10, 1000)$.
   The high fidelity means were chosen to be a uniform grid in $(0, 1)$. The Gaussian distributions had standard deviation 0.2.
2. Gaussian: $M = 500$, $M = 4$, $\vec{\zeta} = (1, 0.5, 0.2, 0)$, $\vec{\lambda} = (1, 5, 20, 50)$.

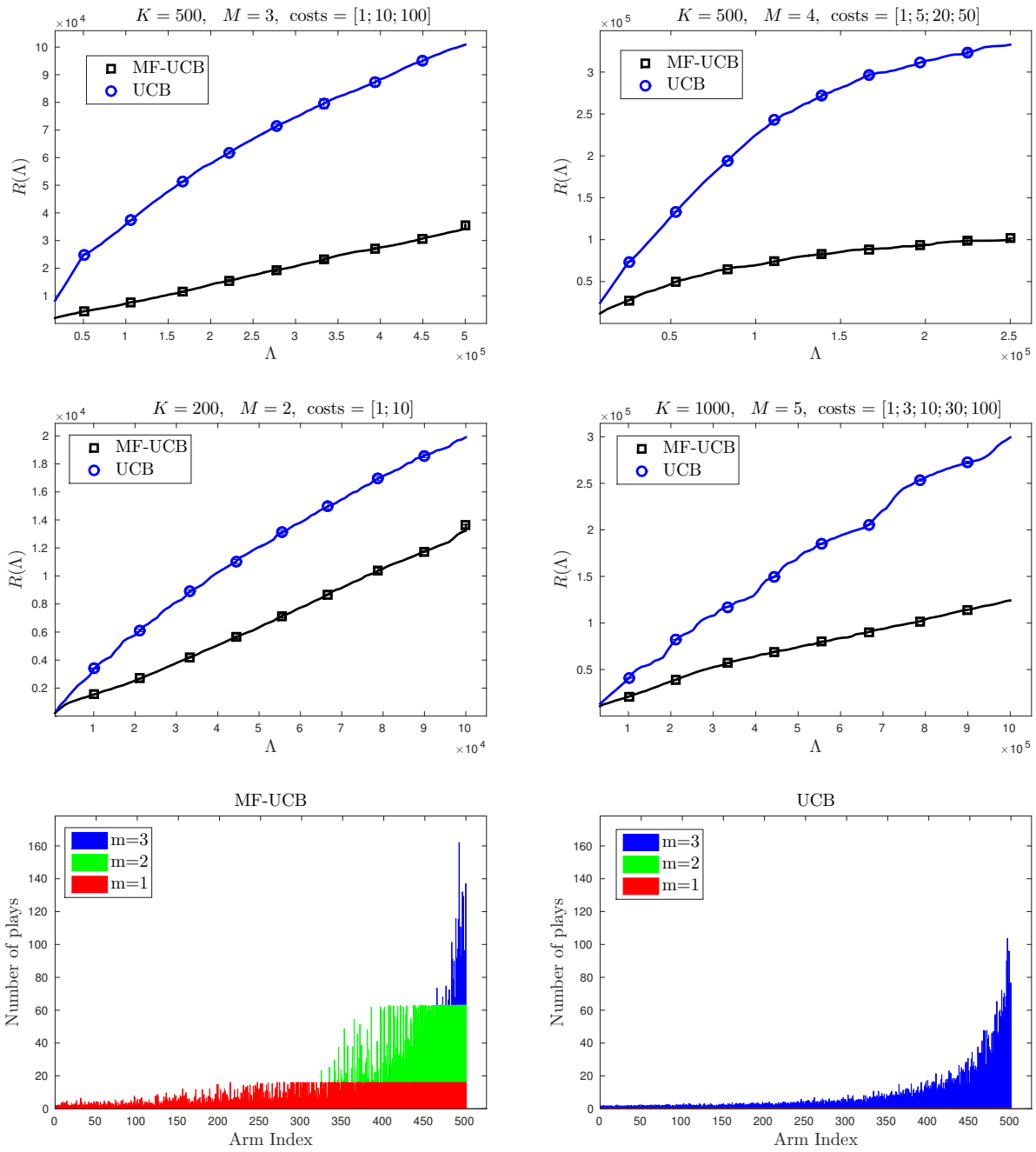


Figure 4.4: Simulations results on the synthetic problems. The first four figures compares UCB against MF-UCB on four synthetic problems. The title states K, M and the costs $\lambda^{(1)}, \ldots, \lambda^{(M)}$. The first two used Gaussian rewards and the last two used Bernoulli rewards. The last two figures show the number of plays by UCB and MF-UCB on a $K = 500, M = 3$ problem with Gaussian observations (corresponding to the first figure).

The high fidelity means were sampled from a $\mathcal{N}(0,1)$ distribution. The Gaussian distributions had standard deviation 1.

3. Bernoulli: $M = 200$, $M = 2$, $\vec{\zeta} = (0.2, 0)$, $\vec{\lambda} = (1, 10)$.
   The high fidelity means were chosen to be a uniform grid in $(0.1, 0.9)$. The Gaussian distributions had standard deviation 1.

4. Bernoulli: $M = 1000$, $M = 5$, $\vec{\zeta} = (0.5, 0.2, 0.1, 0.05, 0)$, $\vec{\lambda} = (1, 3, 10, 30, 100)$.
   The high fidelity means were chosen to be a uniform grid in $(0.1, 0.9)$. The Gaussian distributions had standard deviation 1.

Figure 4.3 illustrates the mean values of these arms. In all cases above, the lower fidelity means were sampled uniformly within a $\pm\zeta^{(m)}$ band around $\mu_k^{(M)}$. In addition, for the Gaussian distributions we modified the lower fidelity means of the optimal arm $\mu_{k_\star}^{(m)}, m < M$ to be lower than the corresponding mean of a suboptimal arm. For the Bernoulli rewards, if $\mu_k^{(m)}$ fell outside of $(0, 1)$ its value was truncated. Figure 4.3 illustrates the mean values of these arms.

The results are given in Figure 4.4. Note that MF-UCB outperforms UCB on all these problems. Critically, note that the gradient of the curve is also smaller than that for UCB – corroborating our theoretical insights. We have also illustrated the number of plays by MF-UCB and UCB at each fidelity for one of these problems in the last row. The arms are arranged in increasing order of $\mu_k^{(M)}$ values. As predicted by our analysis, most of the very suboptimal arms are only played at the lower fidelities. As lower fidelities are cheaper, MF-UCB is able to use more higher fidelity plays at arms close to the optimum than UCB.

4.2 Multi-fidelity GP Bandits with a Finite Number of Approximations

In this section, we study the Gaussian process (Bayesian Optimisation) version of the multi-fidelity bandit problem. Recall, that in BO settings, we wish to optimise a function $f : \mathcal{X} \to \mathbb{R}$ by sequentially querying it at some $x \in \mathcal{X}$ and obtaining a possibly noisy evaluation of $f(x)$. We will refer to conventional methods which assume access to only this single expensive function of interest as *single fidelity* methods. In contrast, in our setting, we will have access to cheap approximations to f which can be queried by the decision maker to speed up the optimisation process. Our contributions in this section are as follows.

1. We present a formalism for multi-fidelity bandit optimisation using Gaussian process (GP) assumptions on f and its approximations. We develop a novel algorithm, Multi-Fidelity Gaussian Process Upper Confidence Bound (MF-GP-UCB) for this setting.
2. Our theoretical analysis proves that MF-GP-UCB explores the space $\mathcal{X}$ at lower fidelities and uses the high fidelities in successively smaller regions to converge on the optimum. As lower fidelity queries are cheaper, MF-GP-UCB has better regret than single fidelity strategies which have to rely on the expensive function to explore the entire space.

3. We demonstrate that MF-GP-UCB outperforms single fidelity methods and other alternatives empirically, via a series of synthetic examples, three hyperparameter tuning tasks and one inference problem in astrophysics.

4.2.1 Problem Formalism & Challenges

Recall that we wish to maximise a function $f : \mathcal{X} \to \mathbb{R}$. For simplicity, we will assume that $\mathcal{X}$ is a finite discrete or compact subset of $[0, r]^d$. Here $r > 0$ and d is the dimension of $\mathcal{X}$. If $x_\star \in \text{argmax}_{x\in\mathcal{X}} f(x)$ is a maximiser of f, and $f(x_\star) = f(x_\star)$ is the maximum value, the goal in bandit optimisation is to achieve small *simple regret* $S_n = \min_{t=1,\ldots,n} f(x_\star) - f(x_t)$ (see (1.1)), after n queries to f. Here $x_t \in \mathcal{X}$ is the point queried at time t by a sequential procedure.

Our primary distinction from the usual setting is that we have access to $M - 1$ successively accurate approximations $f^{(1)}, f^{(2)}, \ldots, f^{(M-1)}$ to the function of interest $f = f^{(M)}$. We refer to these approximations as fidelities. The multi-fidelity framework is attractive when the following two conditions are true about the problem.

1. *The approximations $f^{(1)}, \ldots, f^{(M-1)}$ approximate $f^{(M)}$.* To this end, we will assume a uniform bound for the fidelities, $\|f^{(M)} - f^{(m)}\|_\infty \leq \zeta^{(m)}$ for $m = 1, \ldots, M$, where the bounds $\zeta^{(1)} > \zeta^{(2)} > \cdots > \zeta^{(M)} = 0$ are known.
2. *The approximations are cheaper than evaluating at $f^{(M)}$.* We will assume that a query at fidelity m expends a cost $\lambda^{(m)}$ of a resource, such as computational effort or money. The costs are known and satisfy $0 < \lambda^{(1)} < \lambda^{(2)} < \cdots < \lambda^{(M)}$.

Therefore, as the fidelity m increases, the approximations become better but are also more costly. An algorithm for multi-fidelity bandits is a sequence of query-fidelity pairs $\{(x_t, m_t)\}_{t\geq 0}$, where at time n, the algorithm chooses (x_n, m_n) using information from previous query-observation-fidelity triples $\{(x_t, y_t, m_t)\}_{t=1}^{n-1}$. Here $y_t = f^{(m_t)}(x_t) + \epsilon_t$ where, the ϵ_t values are independent at each time step t and $\mathbb{E}[\epsilon_t] = 0$.

The Generative Process for Multi-fidelity Optimisation

In keeping with the above framework, we assume the following generative model for the functions $f^{(1)}, \ldots, f^{(M)}$. A generative mechanism is given constants $\zeta^{(1)}, \ldots, \zeta^{(M-1)}$. It then generates the functions as follows.

Step 1. Sample $f^{(m)} \sim \mathcal{GP}(0, \kappa)$ for $m = 1, \ldots, M$. **(A1)**

Step 2. Check if $\|f^{(M)} - f^{(m)}\|_\infty \leq \zeta^{(m)}$ for all $m = 1, \ldots, M - 1$. If true, then deliver $f^{(1)}, \ldots, f^{(M)}$. If false, go back to Step 1. **(A2)**

In addition to this, we will also assume that upon querying $f^{(m)}$ at x_t we observe $f^{(m)}(x_t) + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \eta^2)$ is Gaussian noise with variance η^2.

Condition **A2** characterises the approximation conditions for the lower fidelities. Lemma 18 shows that **A2** is satisfied with positive probability when $f^{(1)}, \ldots, f^{(M)}$ are sampled from a GP.

Hence this is a valid generative process since **A2** will eventually be satisfied. Moreover, in Chapter 4.2.2 we argue that while **A2** renders the computation of the true posterior of all GPs inefficient via closed form equations such as in (2.1), it is still possible to derive an efficient algorithm that uses (2.1) to determine future points for evaluation.

We note that other natural approximation conditions can be used to characterise the cheaper fidelities. We choose a uniform bound condition because it provides a simple way to reason about one fidelity from the others, hence keeping the analysis tractable while ensuring the model is interesting enough so as to yield reasonable results in practice. That said, we believe that the intuitions in this work can be used to develop other upper confidence based multi-fidelity BO algorithms for other approximation conditions. In fact, the approximation conditions in our follow up work in Kandasamy et al. [127], are of a Bayesian flavour via a kernel on the fidelities. The algorithm, BOCA, builds on the key insights developed here.

It is worth mentioning that while our theoretical results are valid for arbitrary M and $\lambda^{(m)}$ values, it is instructive to think of M as being a small fixed value and of $\lambda^{(1)}$ as being comparable to $\lambda^{(M)}$. For instance, in many practical applications of multi-fidelity optimisation, while an approximation may be cheaper than the real experiment, it could itself be quite expensive and hence require an intelligence procedure, such as Bayesian optimisation, to choose the next point. This is the regime the current paper focuses on, as opposed to asymptotic regimes where $M \to \infty$ and/or $\lambda^{(1)} \to 0$. Moreover, very large values of M are better handled by the formalism in our follow up work in Kandasamy et al. [127].

Finally, we note that Assumption **A1** can be relaxed to hold for different kernels and noise variances for each fidelity, i.e. different $\kappa^{(m)}, \eta^{(m)}$ for $m = 1, \ldots, M$, with minimal modifications to our analysis but we use the above form to simplify the presentation of the results. In fact, our practical implementation uses different kernels.

Simple Regret for Multi-fidelity Optimisation

Our goal is to achieve small simple regret $S(\Lambda)$ after spending capital Λ of a resource. We will aim to provide *any-capital* bounds, meaning that we will assume that the game is played indefinitely and will try to bound the regret for all (sufficiently large) values of Λ. This is similar in spirit to any-time analyses in single fidelity bandit methods as opposed to fixed time horizon analyses. Let $\{m_t\}_{t\geq 0}$ be the fidelities queried by a multi-fidelity method at each time step. Let N be the *random* quantity such that $N = \max\{n \geq 1 : \sum_{t=1}^{n} \lambda^{(m_t)} \leq \Lambda\}$, i.e. it is the number of queries the strategy makes across all fidelities until capital Λ. Only the optimum of $f = f^{(M)}$ is of interest to us. The lower fidelities are useful to the extent that they help us optimise $f^{(M)}$ with less cost, but there is no reward for optimising a cheaper approximation. Accordingly, we set the instantaneous reward q_t at time t to be $-\infty$ if $m_t \neq M$ and $f^{(M)}(x_t)$ if $m_t = M$. If we let $r_t = f(x_\star) - q_t$ denote the instantaneous regret, we have $r_t = +\infty$ if $m_t \neq M$ and $f(x_\star) - f^{(M)}(x_t)$ if $m_t = M$. For optimisation, the simple regret is simply the best instantaneous

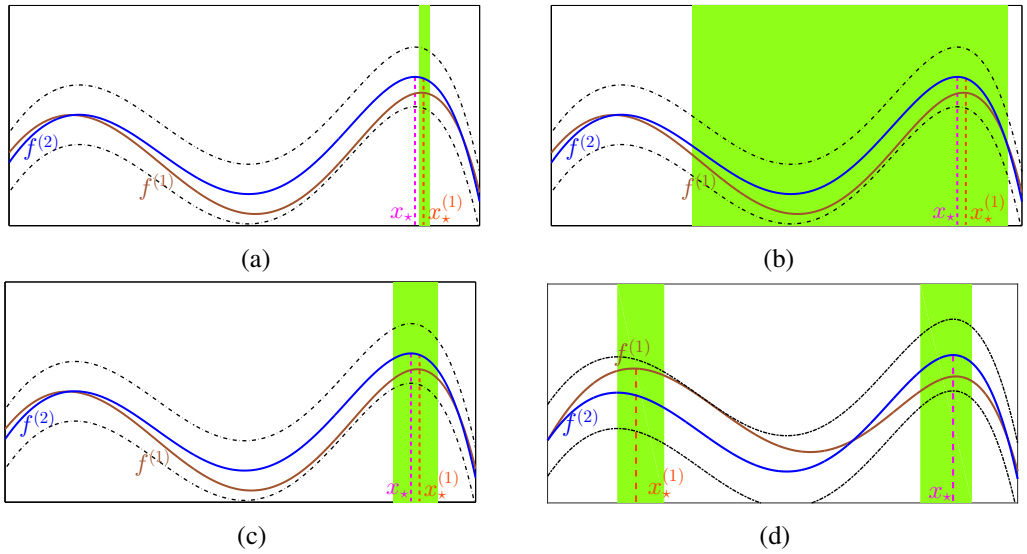


Figure 4.5: An illustration of the challenges in multi-fidelty optimisation. See main text for explanations.

regret, $S(\Lambda) = \min_{t=1,\ldots,N} r_t$. Equivalently,

$$S(\Lambda) = \min_{t=1,\ldots,N} r_t = \begin{cases} \min\limits_{\substack{t=1,\ldots,N \\ t\,:\,m_t=M}} f(x_\star) - f^{(M)}(x_t) & \text{if we have queried at the } M^{\text{th}} \text{ fidelity at least once,} \\ +\infty & \text{otherwise.} \end{cases} \tag{4.9}$$

The above reduces to S_n (1.1) when we only have access to $f^{(M)}$ with $n = N = \lfloor \Lambda/\lambda^{(M)} \rfloor$.

Before we proceed, we note that it is customary in the bandit literature to analyse *cumulative regret*. The definition of cumulative regret depends on the application at hand [125] and our results can be extended to to many sensible notions of cumulative regret. However, both to simplify exposition and since our focus in this paper is optimisation, we stick to simple regret.

Challenges: We conclude this section with a commentary on some of the challenges in multi-fidelity optimisation using Figure 4.5 for illustration. For simplicity, we will focus on 2 fidelities when we have one approximation $f^{(1)}$ to an expensive function $f^{(2)}$. For now assume that (unrealistically) $f^{(1)}$ and its optimum $x_\star^{(1)}$ are known. Typically $x_\star^{(1)}$ is suboptimal for $f^{(2)}$. A seemingly straightforward solution might be to search for $x_\star$ in an appropriate subset, such as a neighborhood of $x_\star^{(1)}$. However, if this neighborhood is too small, we might miss the optimum $x_\star$ (green region in Figure 4.8(a)). A crucial challenge for multi-fidelity methods is to not get stuck at the optimum of a lower fidelity. While exploiting information from lower fidelities, it is also important to *explore* sufficiently at higher fidelities. In our experiments, we demonstrate that naive strategies which do not do so could get stuck at the optimum of a lower fidelity. Alternatively, if we pick a very large subset (Figure 4.8(b)) we might not miss $x_\star$; however, it defeats the objectives of the multi-fidelity set up where the goal is to use the approximation to be prudent about where we query $f^{(2)}$. Figure 4.5(c) displays a seemingly sensible subset, but it remains to

be seen how it is chosen. Further, this subset might not even be a neighborhood as illustrated in Figure 4.5(d), where $f^{(1)}, f^{(2)}$ are multi-modal and the optima are in different modes. In such cases, an appropriate algorithm should explore all such modes. On top of the above, an algorithm does not actually know $f^{(1)}$. A sensible algorithm should explore $f^{(1)}$ and simultaneously identify the above subset, either implicitly or explicitly, for exploration at the second fidelity $f^{(2)}$. Finally, it is also important to note that $f^{(1)}$ is not simply a noisy version of $f^{(2)}$; this setting is more challenging as an algorithm needs to explicitly account for the bias in the approximations.

4.2.2 The Multi-fidelity Gaussian Process Upper Confidence Bound (MF-GP-UCB) Algorithm

We now propose MF-GP-UCB, which extends GP-UCB to the multi-fidelity setting. Like GP-UCB, MF-GP-UCB will also maintain a UCB for $f^{(M)}$ obtained via the previous queries at *all* fidelities. Denote the posterior GP mean and standard deviation of $f^{(m)}$ conditioned *only* on the previous queries at fidelity m by $\mu_t^{(m)}, \sigma_t^{(m)}$ respectively (2.1). Then define,

$$\varphi_t^{(m)}(x) = \mu_{t-1}^{(m)}(x) + \beta_t^{1/2}\sigma_{t-1}^{(m)}(x) + \zeta^{(m)}, \quad \forall\, m, \qquad \varphi_t(x) = \min_{m=1,\ldots,M} \varphi_t^{(m)}(x). \tag{4.10}$$

For appropriately chosen β_t, $\mu_{t-1}^{(m)}(x) + \beta_t^{1/2}\sigma_{t-1}^{(m)}(x)$ will upper bound $f^{(m)}(x)$ with high probability. By **A2** and (4.10), $\varphi_t^{(m)}(x)$ upper bounds $f^{(M)}(x)$ for all m. We have M such bounds, and their minimum $\varphi_t(x)$ gives the best upper bound for $f^{(M)}$. Following UCB strategies such as GP-UCB, our next query is at the maximiser of this UCB, $x_t = \operatorname{argmax}_{x\in\mathcal{X}} \varphi_t(x)$.

Next we need to decide which fidelity to query at. Consider any $m < M$. The $\zeta^{(m)}$ constraints on $f^{(m)}$ restrict the value of $f^{(M)}$ – the confidence band $\beta_t^{1/2}\sigma_{t-1}^{(m)}$ for $f^{(m)}$ is lengthened by $\zeta^{(m)}$ to obtain confidence on $f^{(M)}$. If $\beta_t^{1/2}\sigma_{t-1}^{(m)}(x_t)$ for $f^{(m)}$ is large, it means that we have not constrained $f^{(m)}$ sufficiently well at x_t and should query at the m^{th} fidelity. On the other hand, querying indefinitely in the same region to reduce the uncertainty $\beta_t^{1/2}\sigma_{t-1}^{(m)}$ at the m^{th} fidelity in that region will not help us much as the $\zeta^{(m)}$ elongation caps off how much we can learn about $f^{(M)}$ from $f^{(m)}$; i.e. even if we knew $f^{(m)}$ perfectly, we will only have constrained $f^{(M)}$ to within a $\pm\zeta^{(m)}$ band. Our algorithm captures this simple intuition. Having selected x_t, we begin by checking at the first fidelity. If $\beta_t^{1/2}\sigma_{t-1}^{(1)}(x_t)$ is smaller than a threshold $\gamma^{(1)}$, we proceed to the second fidelity. If at any stage $\beta_t^{1/2}\sigma_{t-1}^{(m)}(x_t) \geq \gamma^{(m)}$ we query at fidelity $m_t = m$. If we proceed all the way to fidelity M, we query at $m_t = M$. The choices for $\gamma^{(m)}$, which we discuss in detail in Chapters 4.2.3 and 4.2.4, should account for the trade-off between the cost $\lambda^{(m)}$ expended at fidelity m and the information obtainable via the approximation. We summarise the resulting procedure in Algorithm 4.

Before we proceed, we make an essential observation. The posterior for any $f^{(m)}(x)$ conditioned on previous queries at *all* fidelities $\bigcup_{\ell=1}^{M} \mathcal{D}_t^{(\ell)}$ is not Gaussian due to the $\zeta^{(m)}$ constraints (**A2**). However, $|f^{(m)}(x) - \mu_{t-1}^{(m)}(x)| < \beta_t^{1/2}\sigma_{t-1}^{(m)}(x)$ holds with high probability, since, by conditioning only on queries at the m^{th} fidelity we have Gaussianity for $f^{(m)}(x)$. (See Lemma 33, Chapter 4.5.2).

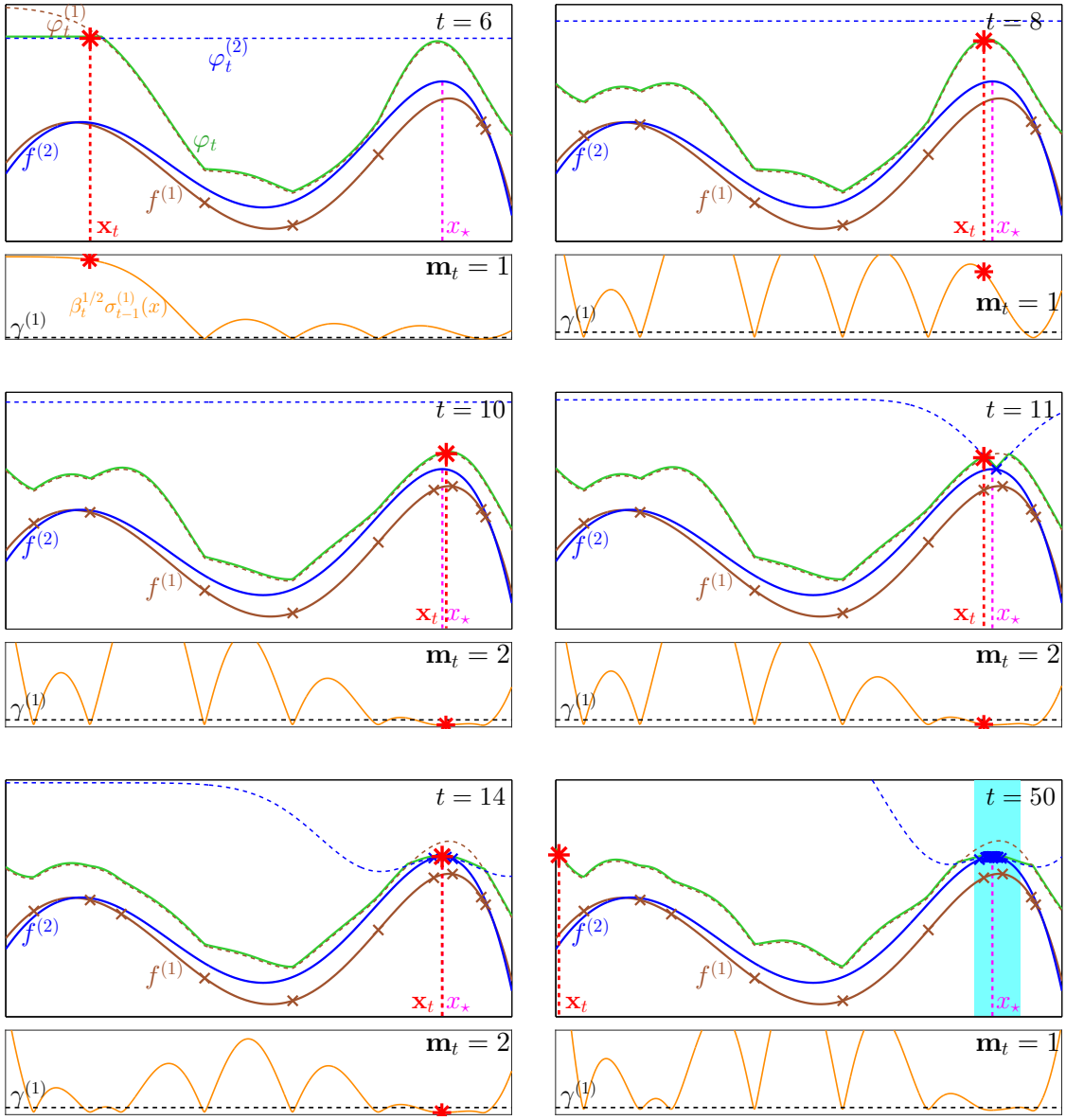


Figure 4.6: The 6 panels illustrate an execution of MF-GP-UCB in 2 fidelities at times $t = 6, 8, 10, 11, 14, 50$. In each panel, the top figure illustrates the upper bounds and selection of x_t while the bottom figure illustrates the selection of m_t. We have initialised MF-GP-UCB with 5 random points at the first fidelity. In the top figures, the solid lines in brown and blue are $f^{(1)}, f^{(2)}$ respectively, and the dashed lines are $\varphi_t^{(1)}, \varphi_t^{(2)}$. The solid green line is $\varphi_t = \min(\varphi_t^{(1)}, \varphi_t^{(2)})$. The small crosses are queries from 1 to $t-1$ and the red star is the maximiser of φ_t, i.e. the next query x_t. $x_\star$, the optimum of $f^{(2)}$ is shown in magenta. In the bottom figures, the solid orange line is $\beta_t^{1/2}\sigma_{t-1}^{(1)}$ and the dashed black line is $\gamma^{(1)}$. When $\beta_t^{1/2}\sigma_{t-1}^{(1)}(x_t) \leq \gamma^{(1)}$ we play at fidelity $m_t = 2$ and otherwise at $m_t = 1$. The cyan region in the last panel is the good set $\mathcal{X}_g$ described in Chapter 4.2.3.

Algorithm 4 MF-GP-UCB from Kandasamy et al. [123, 124]

Inputs: kernel κ, bounds $\{\zeta^{(m)}\}_{m=1}^{M}$, thresholds $\{\gamma^{(m)}\}_{m=1}^{M}$.

- For $m = 1, \ldots, M$: $\quad \mathcal{D}_0^{(m)} \leftarrow \varnothing, \quad (\mu_0^{(m)}, \sigma_0^{(m)}) \leftarrow (\mathbf{0}, \kappa^{1/2})$.
- for $t = 1, 2, \ldots$
  1. $x_t \leftarrow \text{argmax}_{x \in \mathcal{X}} \ \varphi_t(x)$. See (4.10) for φ_t and Chapters 4.2.3, 4.2.4 for β_t.
  2. $m_t = \min \{ m \,|\beta_t^{1/2} \sigma_{t-1}^{(m)}(x_t) \geq \gamma^{(m)} \ \text{ or } \ m = M\}$.
  3. $y_t \leftarrow$ Query $f^{(m_t)}$ at x_t.
  4. Update $\mathcal{D}_t^{(m_t)} \leftarrow \mathcal{D}_{t-1}^{(m_t)} \cup \{(x_t, y_t)\}$. Obtain $\mu_t^{(m_t)}, \sigma_t^{(m_t)}$ conditioned on $\mathcal{D}_t^{(m_t)}$ (2.1). Set $\mathcal{D}_t^{(m)} \leftarrow \mathcal{D}_{t-1}^{(m)}$, $\mu_t^{(m)} \leftarrow \mu_{t-1}^{(m)}$, $\sigma_t^{(m)} \leftarrow \sigma_{t-1}^{(m)}$ for $m \neq m_t$.

An Illustration of MF-GP-UCB: Figure 4.6 illustrates MF-GP-UCB via a simulation on a 2–fidelity problem. At the initial stages, MF-GP-UCB is mostly exploring $\mathcal{X}$ in the first fidelity. $\beta_t^{1/2}\sigma_{t-1}^{(1)}$ is large and we are yet to constrain $f^{(1)}$ well to proceed to $m = 2$. At $t = 10$, we have constrained $f^{(1)}$ sufficiently well at a region around the optimum. $\beta_t^{1/2}\sigma_{t-1}^{(1)}(x_t)$ falls below $\gamma^{(1)}$ and we query at $m_t = 2$. Notice that once we do this (at $t = 11$), $\varphi_t^{(2)}$ dips to change φ_t in that region. At $t = 14$, MF-GP-UCB has identified the maximum $x_\star$ with just 4 queries to $f^{(2)}$. The region shaded in cyan in the last figure is the "good set" $\mathcal{X}_g$, which we alluded to in Chapter 4.2.1. We will define it formally and explain its significance in the multi-fidelity set up shortly. Our analysis predicts that most second fidelity queries in MF-GP-UCB will be be confined to this set (roughly) and the simulation corroborates this claim. For example, in the last figure, at $t = 50$, the algorithm decides to explore at a point far away from the optimum. However, this query occurs in the first fidelity since we have not sufficiently constrained $f^{(1)}(x_t)$ in this region and $\beta_t^{1/2}\sigma_{t-1}^{(1)}(x_t)$ is large. The key idea is that it is *not necessary* to query such regions at the second fidelity as the first fidelity alone is enough to conclude that it is suboptimal. In addition, observe that in a large portion of $\mathcal{X}$, φ_t is given by $\varphi_t^{(1)}$ except in a small neighborhood around $x_\star$, where it is given by $\varphi_t^{(2)}$.

Next we present our main theoretical results. We wish to remind the reader that a table of notations is available in Appendix A.

4.2.3 Theoretical Results

First and foremost, we will show that condition **A2** occurs with positive probability when we sample the functions from a GP. The following lemma shows that $\mathbb{P}_{\mathcal{GP}}(\mathbf{A2}) = \xi_{\mathbf{A2}} > 0$ which establishes that the generative mechanism is valid. The proof is given in Chapter 4.5.

Lemma 18. *Let $f^{(1)}, \ldots, f^{(M)}$ be sampled from $\mathcal{GP}(0, \kappa)$ and* **A2** *denote the event $\{\|f^{(M)} - f^{(m)}\|_\infty \leq \zeta^{(m)}, \forall m \leq M-1\}$. Then,*

$$\mathbb{P}_{\mathcal{GP}}(\mathbf{A2}) = \xi_{\mathbf{A2}} \ \geq \ Q\left(\frac{\zeta^{(M-1)}}{2}\right) \cdot \prod_{m=1}^{M-1} Q\left(\frac{\zeta^{(m)}}{2}\right) \tag{4.11}$$

Here Q is from Lemma 7. $\xi_{\mathbf{A2}} > 0$ since each of the terms in the product are positive.

We are now ready to present our theoretical results. We begin with an informal yet intuitive introuduction to our theorems in $M = 2$ fidelities.

A Preview of our Theorems

We begin an informal yet intuitive introduction to our theorems in $M = 2$ fidelities. In this subsection, we will ignore constants and polylog terms when they are dominated by other terms. $\lesssim, \gtrsim, \asymp$ denote inequality and equality ignoring constants. When $A \subset \mathcal{X}$, we will denote its complement by $\overline{A}$.

Fundamental to the 2-fidelity problem is the good set $\mathcal{X}_g = \{x \in \mathcal{X}; f(x_\star) - f^{(1)}(x) \leq \zeta^{(1)}\}$. $\mathcal{X}_g$ is a high-valued region for $f^{(2)}(x)$: for all $x \in \mathcal{X}_g$, $f^{(2)}(x)$ is at most $2\zeta^{(1)}$ away from the optimum. If a multi-fidelity strategy were to use *all* its second fidelity queries only in $\mathcal{X}_g$, then, by Theorem 1, the regret will only have $\Psi_n(\mathcal{X}_g)$ dependence after n high fidelity queries. In contrast, a strategy that only operates at the highest fidelity, such as GP-UCB, will have $\Psi_n(\mathcal{X})$ dependence. When $\zeta^{(1)}$ is small, i.e. when $f^{(1)}$ is a good approximation to $f^{(2)}$, $\mathcal{X}_g$ will be much smaller than $\mathcal{X}$. Then, $\Psi_n(\mathcal{X}_g) \ll \Psi_n(\mathcal{X})$, and the multi-fidelity strategy will have significantly better regret than a single fidelity strategy. Alas, achieving this somewhat ideal goal is not possible without perfect knowledge of the approximation. However, with MF-GP-UCB we can come quite close. As we will show shortly, *most* second fidelity queries will be confined to the slightly inflated good set $\widetilde{\mathcal{X}}_{g,\rho} = \{x \in \mathcal{X}; f(x_\star) - f^{(1)}(x) \leq \zeta^{(1)} + 3\gamma^{(1)}\}$. The following lemma bounds the number of first and second fidelity evaluations in $\widetilde{\mathcal{X}}_{g,\rho}$ and its complement $\overline{\widetilde{\mathcal{X}}_{g,\rho}}$. We denote the number of queries at the m^{th} fidelity in a set $A \subset \mathcal{X}$ within the first n time steps by $T_n^{(m)}(A)$.

Lemma 19 (Informal, Bounding the number of evaluations for $M = 2$)**.** *Let $\mathcal{X} \subset [0, r]^d$. Consider* MF-GP-UCB *after n total evaluations at either fidelity. Let $T_n^{(m)}(A)$ denote the number of fidelity m queries in some set $A \subset \mathcal{X}$ in n steps. Then,*

$$T_n^{(1)}\left(\overline{\mathcal{X}}_g\right) \lesssim \text{polylog}(n) \cdot \Pi(\widetilde{\mathcal{X}}_{g,\rho}), \qquad T_n^{(1)}(\widetilde{\mathcal{X}}_{g,\rho}) \lesssim \frac{\text{polylog}(n)}{{\gamma^{(1)}}^2} \cdot \Pi(\widetilde{\mathcal{X}}_{g,\rho}),$$

$$T_n^{(2)}\left(\overline{\mathcal{X}}_g\right) \lesssim \tau_n \cdot \Pi\left(\overline{\mathcal{X}}_g\right), \qquad T_n^{(2)}(\widetilde{\mathcal{X}}_{g,\rho}) \asymp n.$$

Here $\Pi(A) = |A|$ for discrete A and $\Pi(A) = \text{vol}(A)$ for continuous A. The bound for $T_n^{(2)}\left(\overline{\mathcal{X}}_g\right)$ holds for any sublinear increasing sequence $\{\tau_n\}_{n \geq 1}$

The above lemma will be useful for two reasons. First, the bounds on $T_n^{(2)}(\cdot)$ show that most second fidelity queries are inside $\widetilde{\mathcal{X}}_{g,\rho}$; the number of such expensive queries outside $\widetilde{\mathcal{X}}_{g,\rho}$ is small. This *strong* result is only possible in the multi-fidelity setting. From the results of Srinivas et al. [235], we can infer that the best achievable bound on the number of plays for GP-UCB inside a suboptimal set is $\asymp n^{1/2}$ for the SE kernel and even worse for the Matérn kernel. For

example, in the simulation of Figure 4.6, all queries to $f^{(2)}$ are in fact confined to $\mathcal{X}_g$ which is a subset of $\widetilde{\mathcal{X}}_{g,\rho}$. This allows us to obtain regret that scales with $\Psi_n(\widetilde{\mathcal{X}}_{g,\rho})$ as explained above. Second, we will use Lemma 19 to control N, the (random) number of queries by MF-GP-UCB within capital Λ. Let $n_\Lambda = \lfloor \Lambda/\lambda^{(2)} \rfloor$ be the (non-random) number of queries by a single fidelity method operating only at the second fidelity. As $\lambda^{(1)} < \lambda^{(2)}$, N could be large for an arbitrary multi-fidelity method. However, using the bounds on $T_n^{(1)}(\cdot)$ we can show that N is $\asymp n_\Lambda$ when Λ is larger than some value Λ_0. Below, we detail the main ingredients in the proof of Lemma 19.

- $T_n^{(1)}(\widetilde{\mathcal{X}}_{g,\rho})$: By the design of our algorithm, MF-GP-UCB will begin querying $f^{(1)}$. To achieve finite regret we need to show that we will eventually query $f^{(2)}$. For any region in $\widetilde{\mathcal{X}}_{g,\rho}$ the switching condition of step 2 in Algorithm 4 ensures that we do not query that region indefinitely. That is, if we keep querying a certain region, the first fidelity GP uncertainty $\beta_t^{1/2}\sigma_{t-1}^{(m)}$ will reduce below $\gamma^{(1)}$ in that region. We will discuss the implications of the choice of $\gamma^{(1)}$ at the end of this subsection and in Chapter 4.2.4.
- $T_n^{(1)}\left(\overline{\mathcal{X}}_g\right)$: For queries to $f^{(1)}$ outside $\widetilde{\mathcal{X}}_{g,\rho}$, we use the following reasoning: as $f^{(1)}$ is small outside $\widetilde{\mathcal{X}}_{g,\rho}$, it is unlikely to contain the UCB maximiser and be selected in step 1 of Algorithm 4 several times.
- $T_n^{(2)}\left(\overline{\mathcal{X}}_g\right)$: We appeal to previous first fidelity queries. If we are querying at the second fidelity at a certain region, it can only be because the first fidelity confidence band is small. This implies that there must be several first fidelity queries in that region which in turn implies that we can learn about $f^{(1)}$ with high confidence. As $f^{(1)}$ alone would tell us that any point in $\widetilde{\mathcal{X}}_{g,\rho}$ is suboptimal for $f^{(2)}$, the maximiser of the UCB is unlikely to lie in this region frequently. Hence, we will not query outside $\widetilde{\mathcal{X}}_{g,\rho}$ often.

It follows from the above that the number of second fidelity queries in $\widetilde{\mathcal{X}}_{g,\rho}$ scales $T_n^{(2)}(\widetilde{\mathcal{X}}_{g,\rho}) \asymp n$. We can now invoke techniques from Srinivas et al. [235] to control the regret using the MIG. However, we can use the MIG of $\widetilde{\mathcal{X}}_{g,\rho}$ since most second fidelity evaluations are in $\widetilde{\mathcal{X}}_{g,\rho}$. This allows us to obtain a tighter bound on $R(\Lambda)$ of the following form.

Theorem 20 (Informal, Regret of MF-GP-UCB for $M = 2$)**.** *Let $\mathcal{X} \subset [0, r]^d$. Then there exists Λ_0 depending only on $\gamma^{(1)}$, $\lambda^{(1)}$ and the approximation $f^{(1)}$ such that, for all $\Lambda > \Lambda_0$ the following holds with high probability.*

$$S(\Lambda) \lesssim \sqrt{\frac{\beta_{n_\Lambda}\Psi_{n_\Lambda}(\widetilde{\mathcal{X}}_{g,\rho})}{n_\Lambda}}$$

It is instructive to compare the above rates against that for GP-UCB in Theorem 1. By dropping the common and sub-dominant terms, the rate for GP-UCB is $\Psi_{n_\Lambda}^{1/2}(\mathcal{X})$ whereas for MF-GP-UCB it is $\Psi_{n_\Lambda}^{1/2}(\widetilde{\mathcal{X}}_{g,\rho})$. Therefore, whenever the approximation is very good ($\mathrm{vol}(\widetilde{\mathcal{X}}_{g,\rho}) \ll \mathrm{vol}(\mathcal{X})$) the rates for MF-GP-UCB are very appealing. When the approximation worsens and $\mathcal{X}_g, \widetilde{\mathcal{X}}_{g,\rho}$ become larger, the bound decays gracefully. In the worst case, MF-GP-UCB is never worse than GP-UCB up to constant terms for $\Lambda \geq \Lambda_0$. The Λ_0 term is required since at the initial stages, MF-GP-UCB will be exploring $f^{(1)}$ before proceeding to $f^{(2)}$, at which stage its regret will still

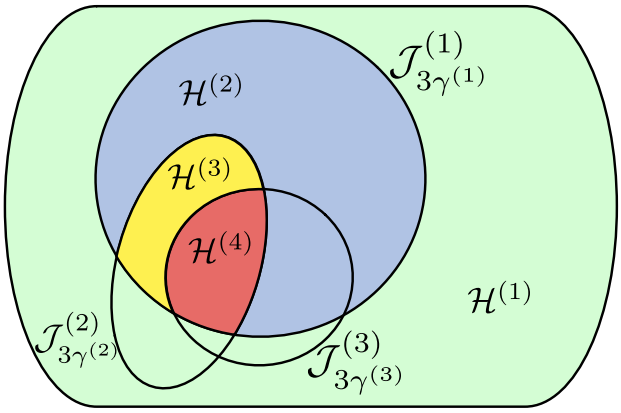


Figure 4.7: Illustration of the partition $\mathcal{H}^{(m)}$'s for a $M = 4$ fidelity problem. The sets $\mathcal{J}^{(m)}_{\zeta^{(m)}}$ are indicated next to their boundaries. The sets $\mathcal{H}^{(1)}, \mathcal{H}^{(2)}, \mathcal{H}^{(3)}, \mathcal{H}^{(4)}$ are shown in green, blue, yellow and red respectively. Most of the capital invested at points in $\mathcal{H}^{(m)}$ will be due to queries to the m^{th} fidelity function $f^{(m)}$.

be $+\infty$. The costs $\lambda^{(1)}, \lambda^{(2)}$ get factored into the result via the $\Lambda > \Lambda_0$ condition. If $\lambda^{(1)}$ is large, for fixed $\gamma^{(1)}$, a larger amount of capital is spent at the first fidelity, so Λ_0 will be large.

Now let us analyse the effect of the parameter $\gamma^{(1)}$ on the result. At first sight, large $\gamma^{(1)}$ seems to increase the size of $\widetilde{\mathcal{X}}_{g,\rho}$ which would suggest that we should keep it as small as possible. However, smaller $\gamma^{(1)}$ also increases Λ_0; intuitively, if $\gamma^{(1)}$ is too small, then one will wait for a long time in step 2 of Algorithm 4 for $\beta_t^{1/2}\sigma^{(1)}_{t-1}$ to decrease without proceeding to $f^{(2)}$. As one might expect, an "optimal" choice of $\gamma^{(1)}$ depends on how large a Λ_0 we are willing to tolerate; i.e. how long we are willing to wait investigating the cheap approximation. Moreover, if the approximation is extremely cheap, it makes sense to use very small $\gamma^{(1)}$ and learn as much as possible about $f^{(2)}$ from $f^{(1)}$. However, it also depends on other problem dependent quantities such as $\mathcal{X}_g$. In Chapter 4.2.3 we describe a choice for $\gamma^{(1)}$ based on $\lambda^{(1)}, \lambda^{(2)}$ and $\zeta^{(1)}$ that aims to balance the cost spent at each fidelity. In our experiments however, we found that more aggressive choices for these threshold values $\gamma^{(m)}$ perform better in practice. We describe one such technique Chapter 4.2.4.

For general M, we will define a hierarchy of good sets, the complement of which will be eliminated when we proceed from one fidelity to the next. At the highest fidelity, we will be querying mostly inside a small subset of $\mathcal{X}$ informed by the approximations $f^{(1)}, \ldots, f^{(M-1)}$. We will formalise these intuitions in the next two subsections.

Discrete $\mathcal{X}$

We first analyse the case when $\mathcal{X}$ is a discrete subset of $[0, r]^d$. Denote $\Delta^{(m)}(x) = f(x_\star) - f^{(m)}(x) - \zeta^{(m)}$ and $\mathcal{J}^{(m)}_\eta = \{x \in \mathcal{X}; \Delta^{(m)}(x) \leq \eta\}$. Note that $\Delta^{(m)} > 0$ for all m by our assumptions. Central to our analysis will be the partitioning $(\mathcal{H}^{(m)})_{m=1}^M$ of $\mathcal{X}$. First define $\mathcal{H}^{(1)} = \overline{\mathcal{J}}^{(1)}_{3\gamma} = \{x : f^{(1)}(x) < f(x_\star) - \zeta^{(1)} - 3\gamma^{(1)}\}$ to be the arms whose $f^{(1)}$ value is at least $\zeta^{(1)} + 3\gamma^{(1)}$ below the optimum $f(x_\star)$. Then recursively define,

$$\mathcal{H}^{(m)} = \overline{\mathcal{J}}^{(m)}_{3\gamma} \cap \left(\bigcap_{\ell=1}^{m-1} \mathcal{J}^{(\ell)}_{3\gamma} \right) \quad \text{for } 2 \leq m \leq M-1, \qquad \mathcal{H}^{(M)} = \bigcap_{\ell=1}^{M-1} \mathcal{J}^{(\ell)}_{3\gamma}. \tag{4.12}$$

In addition to the above, we will also find it useful to define the sets "above" $\mathcal{H}^{(m)}$ as $\widehat{\mathcal{H}}^{(m)} = \bigcup_{\ell=m+1}^{M} \mathcal{H}^{(\ell)}$ and the sets "below" $\mathcal{H}^{(m)}$ as $\breve{\mathcal{H}}^{(m)} = \bigcup_{\ell=1}^{m-1} \mathcal{H}^{(\ell)}$. Our analysis reveals that most of the capital invested at points in $\mathcal{H}^{(m)}$ will be due to queries to the m^{th} fidelity function $f^{(m)}$. $\breve{\mathcal{H}}^{(m)}$ is the set of points that can be excluded from queries at fidelities m and beyond due to information from lower fidelities. $\widehat{\mathcal{H}}^{(m)}$ are points that will be queried at fidelities higher than m several times. In the 2 fidelity setting described in Chapter 4.2.3, $\widetilde{\mathcal{X}}_{g,\rho} = \mathcal{H}^{(2)}$ and $\overline{\widetilde{\mathcal{X}}_{g,\rho}} = \mathcal{H}^{(1)} = \breve{\mathcal{H}}^{(2)}$. We have illustrated these sets in Figure 4.7.

Recall that $n_\Lambda = \lfloor \Lambda/\lambda^{(M)} \rfloor$ is the number of queries by a single-fidelity method; it is a lower bound on N, the number of queries by a multi-fidelity method. Similarly, $\overline{n}_\Lambda = \lfloor \Lambda/\lambda^{(1)} \rfloor$ will be an upper bound on N. We will now define two quantities Λ_1, Λ_2 where $\Lambda_1 < \Lambda_2$. We will show improved simple regret over GP-UCB when the capital Λ is larger than these quantities, with the $\Lambda > \Lambda_2$ regime being better by an additive $\log(\lambda^{(M)}/\lambda^{(1)})$ factor over the $\Lambda > \Lambda_1$ case. Formally, we define Λ_1 to be the smallest Λ satisfying the following condition,

$$\sum_{m=2}^{M} \lambda^{(m)} |\mathcal{H}^{(m-1)}| \; + \; \sum_{m=1}^{M-1} \lambda^{(m)} |\mathcal{H}^{(m)} \cup \widehat{\mathcal{H}}^{(m)}| \left\lceil \frac{\eta^2}{{\gamma^{(m)}}^2} \beta_{\overline{n}_\Lambda} \right\rceil \; \leq \; \frac{\Lambda}{2}, \tag{4.13}$$

and Λ_2 to be the smallest Λ satisfying the following condition,

$$\lambda^{(M)} |\mathcal{X}| \; + \; \lambda^{(M)} \sum_{m=1}^{M-1} |\mathcal{H}^{(m)} \cup \widehat{\mathcal{H}}^{(m)}| \left\lceil \frac{\eta^2}{{\gamma^{(m)}}^2} \beta_{\overline{n}_\Lambda} \right\rceil \; \leq \; \frac{\Lambda}{2}. \tag{4.14}$$

We can find such Λ_1, Λ_2, since for fixed $\gamma^{(m)}$'s, in both cases, the right side is linear in Λ and the left is logarithmic since $\beta_n \asymp \mathcal{O}(\log(n))$ and $\overline{n}_\Lambda \asymp \Lambda$. Since $\{\mathcal{H}^{(m)}\}_{m=1}^{M}$ form a partition of $\mathcal{X}$ and $\lambda^{(1)} < \cdots < \lambda^{(M)}$, we see that $\Lambda_1 < \Lambda_2$. Recall that at the initial stages, MF-GP-UCB has infinite simple regret since the evaluations are at lower fidelities. $\Lambda > \Lambda_1$ indicates the phase where $\Theta(n_\Lambda)$ evaluations have been made inside $\mathcal{H}^{(M)}$, but the total number of evaluations N could be much larger. When $\Lambda > \Lambda_2$, we have reached a phase where N is also in $\Theta(n_\Lambda)$.

Moreover, note that when the approximations are good, i.e. the sets $\mathcal{H}^{(m)}$ are small, both Λ_1 and Λ_2 are small. Λ_1 is also small when the approximations are cheap, i.e. $\lambda^{(m)}$'s are small. Therefore, the cheaper and better the approximations, we have to wait less time (for fixed $\gamma^{(m)}$) before MF-GP-UCB starts querying at the M^{th} fidelity and achieves good regret.

We now state our main theorem for discrete $\mathcal{X}$. To simplify the analysis, we will introduce an additional condition in the fidelity selection criterion in step 2 of Algorithm 4. We will always evaluate $f^{(m)}$ at x_t only if x_t has been evaluated at all lower fidelities, $1, \ldots, m-1$; precisely, that $m_t = \min_m \{ m \,|\beta_t^{1/2} \sigma_{t-1}^{(m)}(x_t) \geq \gamma^{(m)} \;\text{ or }\; m = M \;\text{ or }\; T_n^{(m)}(x_t) = 0\}$. Both this condition, and the dependence of Λ_2 on $|\mathcal{X}|$ in (4.14) are an artefact of our analysis. They arise only because we do not account for the correlations between the arms in our discrete analysis; doing so requires us to make assumptions about the locations of the arms in $[0, r]^d$. We will not need this condition or have Λ_2 depend on $|\mathcal{X}|$ for the continuous case.

Theorem 21. *Let $\mathcal{X}$ be a discrete subset of $[0,r]^d$. Let $f^{(m)} \sim \mathcal{GP}(\mathbf{0},\kappa)$ for all m. Assume that $f^{(m)}$'s satisfy assumptions* **A1**, **A2** *and κ satisfies Assumption 1. Pick $\delta \in (0,1)$ and run* MF-GP-UCB *(Algorithm 4) with $\beta_t = 2\log\left(M|\mathcal{X}|\pi^2 t^2/(3\xi_{\mathbf{A2}}\delta)\right)$. Then, we have the following bounds on $S(\Lambda)$ with $\mathbb{P}$-probability greater than $1-\delta$.*

$$\text{for all } \Lambda > \Lambda_1, \quad S(\Lambda) \;\leq\; \sqrt{\frac{2C_1\beta_{\overline{n}_\Lambda}\Psi_{n_\Lambda}(\mathcal{H}^{(M)})}{n_\Lambda}}$$

$$\text{for all } \Lambda > \Lambda_2, \quad S(\Lambda) \;\leq\; \sqrt{\frac{2C_1\beta_{2n_\Lambda}\Psi_{n_\Lambda}(\mathcal{H}^{(M)})}{n_\Lambda}}$$

Here $C_1 = 8/\log(1+\eta^2)$ is a constant, $n_\Lambda = \lfloor \Lambda/\lambda^{(M)} \rfloor$, $\overline{n}_\Lambda = \lfloor \Lambda/\lambda^{(1)} \rfloor$, and $\xi_{\mathbf{A2}}$ is from (4.11).

The difference between the two results is the $\beta_{\overline{n}_\Lambda}$ dependence in the former setting and β_{n_Λ} in the latter; the latter bound is better by an additive $\log(\lambda^{(M)}/\lambda^{(1)})$ term, but we have to wait for longer. Dropping constant and polylog terms and comparing to the result in Theorem 1 reveals that we outperform GP-UCB by a factor of $\sqrt{\Psi_{n_\Lambda}(\mathcal{H}^{(M)})/\Psi_{n_\Lambda}(\mathcal{X})} \asymp \sqrt{\mathrm{vol}(\mathcal{H}^{(M)})/\mathrm{vol}(\mathcal{X})}$ asymptotically. The set $\mathcal{H}^{(M)}$ from (4.12) is determined by the $\zeta^{(1)}, \ldots \zeta^{(M-1)}$ values, the approximations $f^{(1)}, \ldots, f^{(M-1)}$ and the parameters $\gamma^{(1)}, \ldots, \gamma^{(M-1)}$. The better the approximations, the smaller the set $\mathcal{H}^{(M)}$ and there is more advantage over single fidelity strategies. In Figure 4.8, we have shown the ratio $\mathrm{vol}(\mathcal{H}^{(2)})/\mathrm{vol}(\mathcal{X})$ for a two fidelity problem as $\zeta^{(1)}$ decreases—the figure corroborates our claim that the rates improve as the $\zeta^{(m)}$ values decrease. As the approximations worsen, the advantage to multi-fidelity optimisation diminishes as expected, but we are never worse than GP-UCB up to constant factors.

A few remarks are in order. First, note that the dependence on n_Λ (or equivalently Λ) is the same for both GP-UCB and MF-GP-UCB. In fact, one should not expect multi-fidelity optimisation to yield "rate" improvements since such $\sqrt{1/n}$ dependencies are typical in the bandit literature [26, 223]. The multi-fidelity framework allows us to find a good region, i.e. $\mathcal{H}^{(M)}$, where the optimum exisits, and as such, we should expect the improvements to be in terms of the size of this set, relative to $\mathcal{X}$. Second, the bound is given in terms of $\mathcal{H}^{(M)}$ which, as illustrated by Figure 4.8, gives us insight into the types of gains we can expect from multi-fidelity optimisation. However, $\mathcal{H}^{(M)}$ is a random quantity and obtaining high probability bounds on its volume could shed more light on the gains of our multi-fidelity optimisation framework; this is an interesting avenue for future work.

Choice of $\gamma^{(m)}$. It should be noted that an "optimal" choice of $\gamma^{(m)}$ depends on the available budget, i.e. how long we are willing to wait before achieving non-trivial regret. If we are willing to wait long, we can afford to choose small $\gamma^{(m)}$ and consequently have better guarantees on the regret. This optimal choice also depends on several unknown problem dependent factors – such as the sizes of the sets $\mathcal{H}^{(m)}$. In Kandasamy et al. [125], the choice $\gamma^{(m)} = \zeta^{(m)}\sqrt{\lambda^{(m)}/\lambda^{(m+1)}}$ was used which ensures that for an arm $x \in \mathcal{H}^{(m)}$, the cost spent at lower fidelities $1, \ldots, m-1$ is not more than the cost spent at fidelity m. Beyond this intuitive property, this choice further achieves a lower bound on the K-armed multi-fidelity problem. The same choice for $\gamma^{(m)}$ here ensures that the cost spent at the lower fidelities is not more than an upper bound on the cost spent at

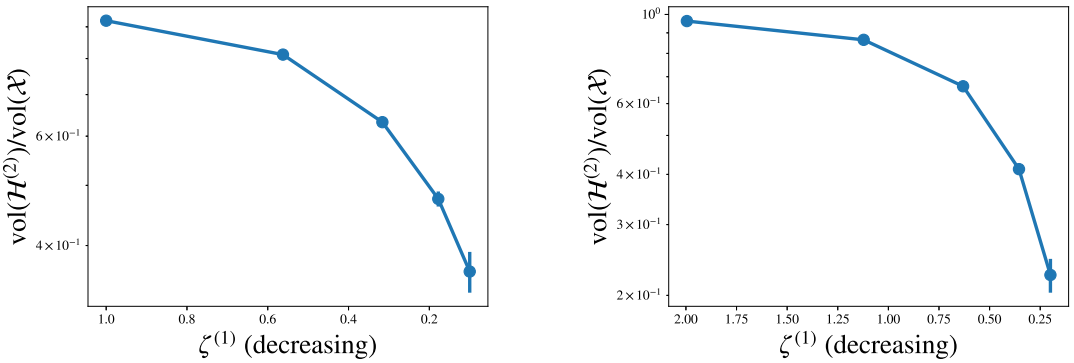


Figure 4.8: Empirically computed values for the ratio $\text{vol}(\mathcal{H}^{(2)})/\text{vol}(\mathcal{X})$ for a one dimensional (left) and two dimensional (right) 2-fidelity problem. For this, the samples $f^{(1)}, f^{(2)}$ were generated using the generative mechanism of Chapter 4.2.1, under the stipulated value for $\zeta^{(1)}$. In both cases, we used an SE kernel with bandwidth 1 and scale parameter 1. The y-axis is the mean value for the ratio over several samples and the x-axis is $\zeta^{(1)}$. In both cases, we used $\gamma^{(1)} = \zeta^{(1)}/3$, and approximated the continuous domain with a uniform grid of size 10^4. The figure indicates that as the approximation improves, i.e. $\zeta^{(1)}$ decreases, the ratio decreases and consequently, we get better bounds.

fidelity m – we have elaborated more in Remark 3 after our proofs. We have empirically demonstrated the effect of different choice of $\gamma^{(m)}$ values via an experiment in Figure 4.12. Building on these ideas, an explicit prescription for the choice of $\gamma^{(m)}$ is bound to be a fruitful avenue of research, and we leave this to future work. In the meanwhile, in Chapter 4.2.4, we describe a heuristic for adaptively choosing $\gamma^{(m)}$ adaptively which worked well in our experiments.

Continuous and Compact $\mathcal{X}$

We define the sets $\mathcal{H}^{(m)}, \widehat{\mathcal{H}}^{(m)}$ for $m = 1, \ldots, M$ as in the discrete case. Let $\{\nu_n\}_{n\geq 0}$ be any sublinear sequence such that $\nu_n \to \infty$. Let

$$\mathcal{H}^{(m)}_{\tau,n} = \left\{ x \in \mathcal{X} : \ \mathrm{B}_2\big(x, r\sqrt{d}/\nu_n^{\frac{1}{2d}}\big) \cap \mathcal{H}^{(m)}_{\tau} \neq \varnothing \quad \wedge \quad x \notin \widehat{\mathcal{H}}^{(m)} \right\}$$

to be a ν_n-dependent L_2 dilation of $\mathcal{H}^{(m)}_{\tau,n}$ by $r\sqrt{d}/\nu_n^{\frac{1}{2d}}$. Here, $\mathrm{B}_2(x,\epsilon)$ is an L_2 ball of radius ϵ centred at x. Notice that as $n \to \infty$, $\mathcal{H}^{(m)}_{\tau,n} \to \mathcal{H}^{(m)}_{\tau}$. Similar to the discrete case, we define Λ_1 to be the smallest Λ satisfying the following the condition,

$$\lambda^{(M)}\nu_{\overline{n}_\Lambda} \ + \ C_\kappa \eta^2 \beta^{p+1}_{\overline{n}_\Lambda} \sum_{m=1}^{M-1} \lambda^{(m)} \frac{\text{vol}(\mathcal{H}^{(m)}_{n_\Lambda} \cup \widehat{\mathcal{H}}^{(m)})}{{\gamma^{(m)}}^{2p}} \ \leq \ \frac{\Lambda}{2}, \tag{4.15}$$

and Λ_2 to be the smallest Λ satisfying the following condition,

$$\lambda^{(M)}\nu_{\overline{n}_\Lambda} \ + \ C_\kappa \eta^2 \beta^{p+1}_{\overline{n}_\Lambda} \lambda^{(M)} \sum_{m=1}^{M-1} \frac{\text{vol}(\mathcal{H}^{(m)}_{n_\Lambda} \cup \widehat{\mathcal{H}}^{(m)})}{{\gamma^{(m)}}^{2p}} \ \leq \ \frac{\Lambda}{2}. \tag{4.16}$$

Here $p = 1/2$ for the SE kernel and $p = 1$ for the Matérn kernel. C_κ is a kernel dependent constant elucidated in our proofs; for the SE kernel, $C_\kappa = 2^{2+d/2}(d\kappa_0/h^2)^{d/2}$ where κ_0, h are parameters of the kernel. Via a reasoning similar to the discrete case we see that $\Lambda_1 < \Lambda_2$. Our main theorem is as follows.

Theorem 22. *Let $\mathcal{X} \subset [0,r]^d$ be compact and convex. Let $f^{(m)} \sim \mathcal{GP}(\mathbf{0}, \kappa)\ \forall\, m$, and satisfy assumptions* **A1**, **A2**. *Let κ satisfy Assumption 1 with some constants a, b. Pick $\delta \in (0,1)$ and run* MF-GP-UCB *(Algorithm 4) with*

$$\beta_t = 2\log\left(\frac{M\pi^2 t^2}{2\xi_{\mathbf{A2}}\delta}\right) + 4d\log(t) + \max\left\{0\,,\ 2d\log\left(brd\sqrt{\log\left(\frac{6Mad}{\xi_{\mathbf{A2}}\delta}\right)}\right)\right\}.$$

Then, we have the following bounds on $S(\Lambda)$ with $\mathbb{P}$-probability greater than $1-\delta$.

$$\text{for all } \Lambda > \Lambda_1, \quad S(\Lambda) \ \leq\ \sqrt{\frac{2C_1\beta_{\overline{n}_\Lambda}\Psi_{n_\Lambda}(\mathcal{H}^{(M)}_{n_\Lambda})}{n_\Lambda}} + \frac{\pi^2}{3n_\Lambda}$$

$$\text{for all } \Lambda > \Lambda_2, \quad S(\Lambda) \ \leq\ \sqrt{\frac{2C_1\beta_{2n_\Lambda}\Psi_{n_\Lambda}(\mathcal{H}^{(M)}_{n_\Lambda})}{n_\Lambda}} + \frac{\pi^2}{3n_\Lambda}$$

Here $C_1 = 8/\log(1+\eta^2)$ is a constant, $n_\Lambda = \lfloor \Lambda/\lambda^{(M)} \rfloor$, and $\overline{n}_\Lambda = \lfloor \Lambda/\lambda^{(1)} \rfloor$.

Note that the sets $\mathcal{H}^{(M)}_{\tau,n_\Lambda}$ depend on the sublinear increasing sequence $\{\nu_n\}_{n\geq 0}$ – the theorem is valid for any such choice of ν_n. The comparison of the above bound against GP-UCB is similar to the discrete case. The main difference is that we have an additional dilation of $\mathcal{H}^{(M)}_\tau$ to $\mathcal{H}^{(M)}_{\tau,n_\Lambda}$ which occurs due to a covering argument in our analysis. Recall that $\mathcal{H}^{(m)}_{\tau,n_\Lambda} \to \mathcal{H}^{(m)}_\tau$ as $\Lambda \to \infty$. The bound is determined by the MIG of the set $\mathcal{H}^{(M)}_{\tau,n_\Lambda}$, which is small when the approximations are good.

4.2.4 Implementation Details

We describe implementation details for MF-GP-UCB and other BO methods that were used in the experiments in the next subsection. In addition to some standard techniques in the Bayesian optimisation literature, we describe the heuristics used to set the $\gamma^{(m)}, \zeta^{(m)}$ parameters of our method.

Initialisation: Following recommendations in Brochu et al. [23] all GP methods were initialised with uniform random queries using an initialisation capital Λ_0. For single fidelity methods, we used it at the M^{th} fidelity, whereas for multi-fidelity methods we used $\Lambda_0/2$ at the first fidelity and $\Lambda_0/2$ at the second fidelity.

Kernel: In all our experiments, we used the SE kernel. We initialise the kernel by maximising the GP marginal likelihood [203] on the initial sample and then update the kernel every 25 iterations using marginal likelihood.

Choice of β_t: β_t, as specified in Theorems 1, 22 has unknown constants and tends to be too conservative in practice. Following Kandasamy et al. [120] we use $\beta_t = 0.2d\log(2t)$ which captures the dominant dependencies on d and t.

Maximising φ_t: We used the DiRect algorithm [112].

Choice of $\zeta^{(m)}$'s: Algorithm 4 assumes that the $\zeta^{(m)}$'s are given with the problem description, which is hardly the case in practice. In our implementation, instead of having to deal with $M-1$, $\zeta^{(m)}$ values we will assume $\|f^{(m)} - f^{(m-1)}\|_\infty \leq \zeta$. This satisfies assumption **A2** with $(\zeta^{(1)}, \zeta^{(2)}, \ldots, \zeta^{(M-1)}) = ((M-1)\zeta, (M-2)\zeta, \ldots, \zeta)$. This allows us to work with only one value of ζ. We initialise ζ to a small value, 1% of the range of initial queries. Whenever we query at any fidelity $m > 1$ we also check the posterior mean of the $(m-1)^{\text{th}}$ fidelity. If $|f^{(m)}(x_t) - \mu_{t-1}^{(m-1)}(x_t)| > \zeta$, we query again at x_t, but at the $(m-1)^{\text{th}}$ fidelity. If $|f^{(m)}(x_t) - f^{(m-1)}(x_t)| > \zeta$, we update ζ to twice the violation.

Choice of $\gamma^{(m)}$'s: The role of the $\gamma^{(m)}$ values at each fidelity is to ensure that we do not spend too much effort at the lower fidelities, where if $\gamma^{(m)}$ is too small, MF-GP-UCB spends a large number of queries at fidelity m to reduce the variance below $\gamma^{(m)}$. This might cause MF-GP-UCB to spend an unnecessarily large number of evaluations at fidelity m. Hence, we start with small values for all $\gamma^{(m)}$. However, if the algorithm does not query above the m^{th} fidelity for more than $\lambda^{(m+1)}/\lambda^{(m)}$ iterations, we double $\gamma^{(m)}$. All $\gamma^{(m)}$ values were initialised to 1% of the range of initial queries.

Whilst the first four choices are standard in the BO literature [23, 232], our methods for selecting the $\zeta^{(m)}$ and $\gamma^{(m)}$ parameters are heuristic in nature. We obtained robust implementations of MF-GP-UCB with little effort in tweaking these choices. In fact, we found our implementation was able to recover even from fairly bad approximations at the lower fidelities (see experiment in Figure 4.11). We believe that other reasonable heuristics can also be used in place of our choices here, and a systematic investigation into protocols for the same will be a fruitful avenue for future research.

4.2.5 Experiments

We present experiments for compact and continuous $\mathcal{X}$ since it is the more practically relevant setting. We compare MF-GP-UCB to the following baselines. **Single fidelity methods:** GP-UCB; GP-EI: the expected improvement criterion for BO [113]; DiRect: the dividing rectangles method [112]. **Multi-fidelity methods:** MF-NAIVE: a naive baseline where we use GP-UCB to query at the *first* fidelity a large number of times and then query at the last fidelity at the points queried at $f^{(1)}$ in decreasing order of $f^{(1)}$-value; MF-SKO: the multi-fidelity sequential kriging method from [100]. Previous works on multi-fidelity methods (including MF-SKO) had not made their code available and were not straightforward to implement. We discuss this more at the end of this section along with some other single and multi-fidelity baselines we tried but excluded in the comparison to avoid clutter in the figures. We also detail some design choices and hyperparameters for the baselines.

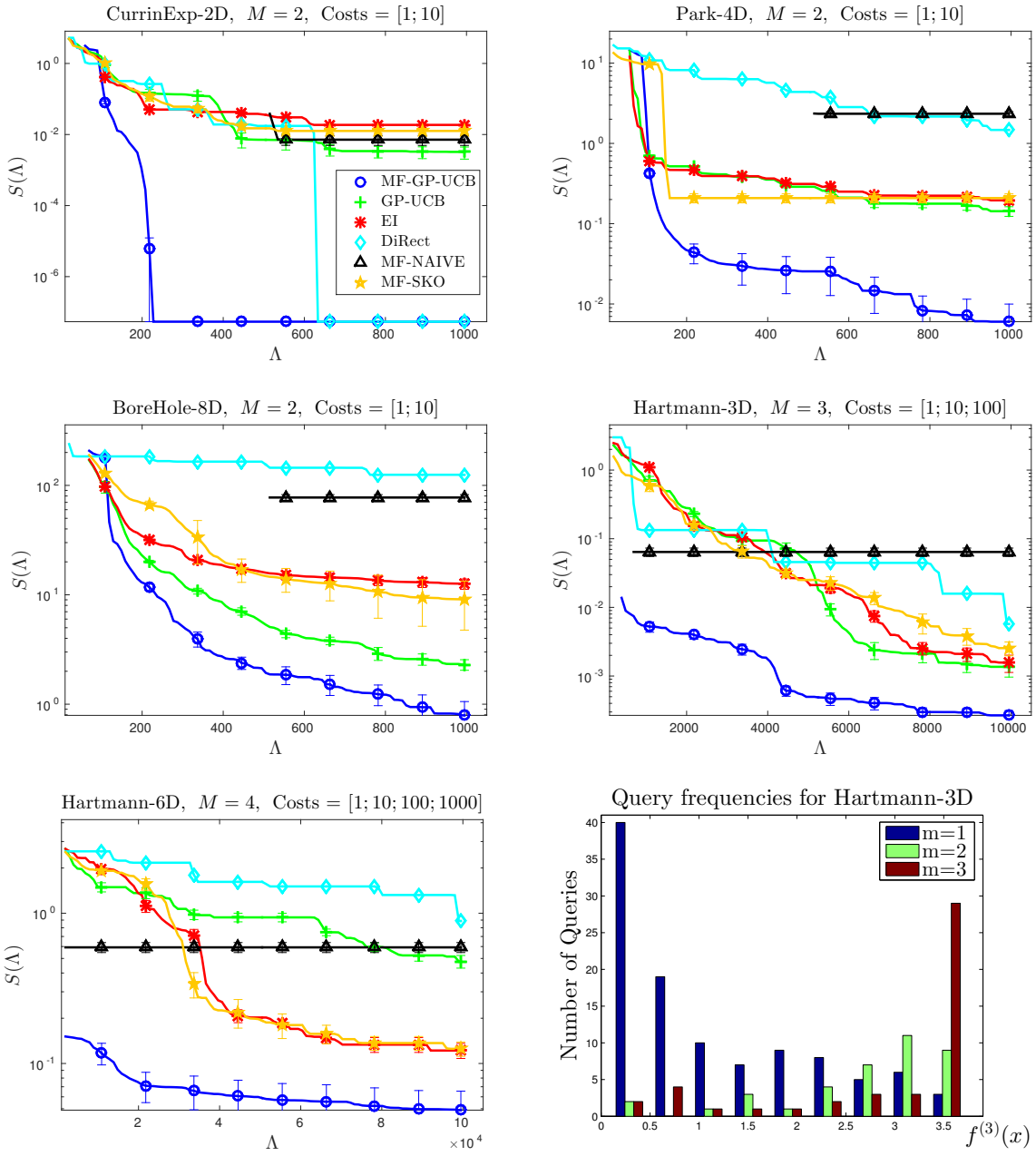


Figure 4.9: The simple regret $S(\Lambda)$ (4.9) against the spent capital Λ on the synthetic functions. The title states the function, its dimensionality, the number of fidelities and the costs we used for each fidelity in the experiment; for example, in the fourth panel, we used $M = 3$ fidelities, with costs $\lambda^{(1)} = 1, \lambda^{(2)} = 10, \lambda^{(3)} = 100$ on the 3 dimensional Hartmann function. All curves barring DiRect (which is a deterministic), were produced by averaging over 20 experiments. The error bars indicate one standard error. All figures follow the legend in the first figure for the Currin exponential function. The last panel shows the number of queries at different function values at each fidelity for the Hartmann-3D example.

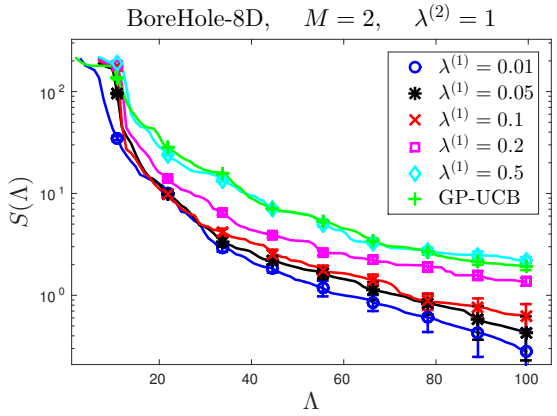


Figure 4.10: The performance of our implementation of MF-GP-UCB for different values of $\lambda^{(1)}$ in the 2 fidelity Borehole experiment. Our implementation uses the techniques and heuristics described in Chapter 4.2.4. In all experiments we used $\lambda^{(2)} = 1$. We have also shown the curve for GP-UCB for reference.

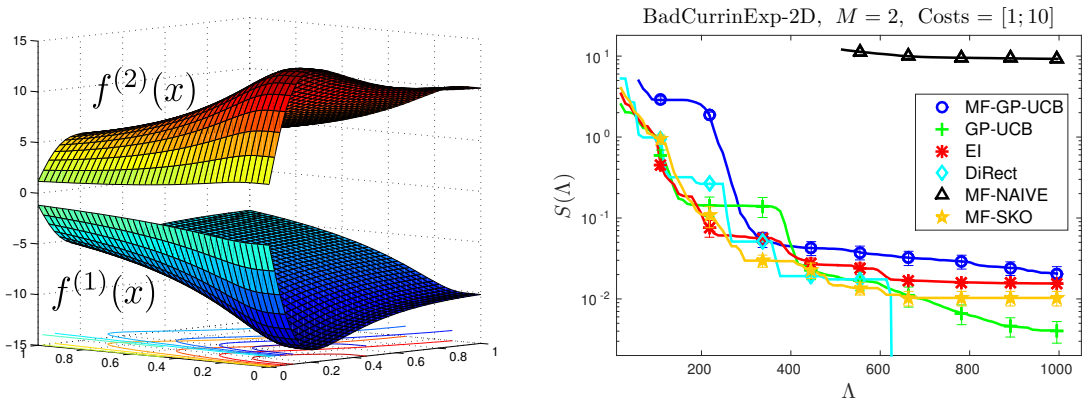
Synthetic Examples

We begin with a series of synthetic experiments, designed to demonstrate the applicability and limitations of MF-GP-UCB. We use the Currin exponential ($d = 2$), Park ($d = 4$) and Borehole ($d = 8$) functions in $M = 2$ fidelity experiments and the Hartmann functions in $d = 3$ and 6 with $M = 3$ and 4 fidelities respectively. The first three functions are taken from previous multi-fidelity literature [274] while we tweaked the Hartmann functions to obtain the lower fidelities for the latter two cases. At the end of this section, we provide the formulae for these functions and the approximations used for the lower fidelities. We show the simple regret $S(\Lambda)$ against capital Λ in Figure 4.9. The number of fidelities and the costs used for each fidelity are also given in Figure 4.9. MF-GP-UCB outperforms other baselines on all problems.

The last panel of Figure 4.9 shows a histogram of the number of queries at each fidelity after 184 queries of MF-GP-UCB, for different ranges of $f^{(3)}(x)$ for the Hartmann-3D function. Many of the queries at the low $f^{(3)}$ values are at fidelity 1, but as we progress they decrease and the second fidelity queries increase. The third fidelity dominates very close to the optimum but is used sparingly elsewhere. This corroborates the prediction in our analysis that MF-GP-UCB uses low fidelities to explore and successively higher fidelities at promising regions to zero in on $x_\star$. (Also see Figure 4.6.)

A common occurrence with MF-NAIVE was that once we started querying at fidelity M, the regret barely decreased. The diagnosis in all cases was the same: it was stuck around the maximum of $f^{(1)}$ which is suboptimal for $f^{(M)}$. This suggests that while we have cheap approximations, the problem is by no means trivial. As explained previously, it is also important to *explore* at higher fidelities to achieve good regret. The efficacy of MF-GP-UCB when compared to single fidelity methods is that it confines this exploration to a small set containing the optimum. In our experiments we found that MF-SKO did not consistently beat other single fidelity methods. Despite our best efforts to reproduce MF-SKO, we found it to be quite brittle. We also tried another multi-fidelity method and found that it did not perform as desired (See Chapter 4.2.5 for details).

Effect of the cost of the approximations: We now test the effect the cost of the approximation on performance. Figure 4.10 shows the results when MF-GP-UCB was run on the 2-fidelity Borehole experiment for different costs for the approximation $f^{(1)}$. We fixed $\lambda^{(2)} = 1$ and varied



(a) (b)

Figure 4.11: (a): the functions used in the Bad Currin Exponential experiment where $f^{(1)} = -f^{(2)}$. (b): the simple regret for this experiment. See caption under Figure 4.9 for more details.

$\lambda^{(1)}$ between 0.01 to 0.5. As $\lambda^{(1)}$ increases, the performance worsens as expected. At $\lambda^{(1)} = 0.5$ it is indistinguishable from GP-UCB as the overhead of managing 2 fidelities becomes significant when compared to the improvements of using the approximation.

Bad Approximations: It is natural to ask how MF-GP-UCB performs with bad approximations at lower fidelities. We found that our implementation with the heuristics suggested in Chapter 4.2.4 to be quite robust. We demonstrate this using the Currin exponential function, but using the negative of $f^{(2)}$ as the first fidelity approximation, i.e. $f^{(1)}(x) = -f^{(2)}(x)$. Figure 4.11 illustrates $f^{(1)}$, $f^{(2)}$ and gives the simple regret $S(\Lambda)$. Understandably, it loses to the single fidelity methods since the first fidelity queries are wasted and it spends some time at the second fidelity recovering from the bad approximation. However, it eventually is able to achieve low regret.

Effect of threshold values on MF-GP-UCB: We now demonstrate the effect of different choices for $\gamma^{(1)}$ on MF-GP-UCB as described in Algorithm 4. We use the 3 dimensional Hartmann function in a 2 fidelity set up where $\zeta^{(1)} \approx 0.112$, $\lambda^{(1)} = 1$ and $\lambda^{(2)} = 10$. The implementation follows the description in Chapter 4.2.4, except that the true $\zeta^{(1)}$ value is made known to MF-GP-UCB and the threshold value $\gamma^{(1)}$ is kept fixed at values $0.03, 0.1, 0.3, 1.0$. The result is shown in Figure 4.12. We see that as $\gamma^{(1)}$ decreases the curves start later in the figure indicating that MF-GP-UCB spends more time at the approximation $f^{(1)}$ before proceeding to $f^{(2)}$; however, the simple regret is also generally better for smaller $\gamma^{(1)}$. Therefore, if we have a large computational budget and are willing to wait longer, we can choose small $\gamma^{(m)}$ values and achieve better simple regret.

Model Selection & Astrophysics Experiments

We now present results on three hyperparameter tuning tasks and a maximum likelihood inference task in Astrophysics. We compare methods on computation time since that is the "cost"

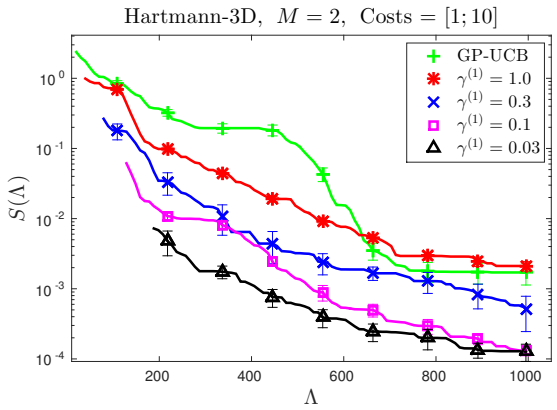


Figure 4.12: The performance of MF-GP-UCB for different choices of *fixed* threshold values $\gamma^{(1)}$. The curves were averaged over 20 independent runs and in this figure, they start when at least 10 of the 20 runs have queried at least once at the top (second) fidelity. This experiment was run on the $3-$dimensional Hartmann function in the two fidelity set up where $\zeta^{(1)} \approx 0.112$. The true $\zeta^{(1)}$ value was made known to MF-GP-UCB.

in all experiments. We include the processing time for each method in the comparison (i.e. the cost of determining the next query). The results are given in Figure 4.13 where MF-GP-UCB outperforms other baselines on all tasks. The experimental set up for each optimisation problem is described below.

Classification using SVMs (SVM): We trained a Support vector classifier on the magic gamma dataset using the sequential minimal optimisation algorithm to an accuracy of 10^{-12}. The goal is to tune the kernel bandwidth and the soft margin coefficient in the ranges $(10^{-3}, 10^{1})$ and $(10^{-1}, 10^{5})$ respectively on a dataset of size 2000. We set this up as a $M = 2$ fidelity experiment with the entire training set at the second fidelity and 500 points at the first. Each query to $f^{(m)}$ required 5-fold cross validation on the respective training sets.

Regression using additive kernels (SALSA): We used the SALSA method for additive kernel ridge regression [117] on the 4-dimensional coal power plant dataset. We tuned the 6 hyperparameters –the regularisation penalty, the kernel scale and the kernel bandwidth for each dimension– each in the range $(10^{-3}, 10^{4})$ using 5-fold cross validation. This experiment used $M = 3$ and $2000, 4000, 8000$ points at each fidelity respectively.

Viola & Jones face detection (V&J): The Viola & Jones cascade face classifier [254], which uses a cascade of weak classifiers, is a popular method for face detection. To classify an image, we pass it through each classifier. If at any point the classifier score falls below a threshold, the image is classified as negative. If it passes through the cascade, then it is classified as positive. One of the more popular implementations comes with OpenCV and uses a cascade of 22 weak classifiers. The threshold values in the OpenCV implementation are pre-set based on some heuristics and there is no reason to think they are optimal for a given face detection problem. The goal is to tune these 22 thresholds by optimising them over a training set. We modified the OpenCV implementation to take in the thresholds as parameters. As our domain $\mathcal{X}$ we chose a neighbourhood around the configuration used in OpenCV. We set this up as an $M = 2$ fidelity experiment where the second fidelity used 3000 images from the Viola and Jones face database and the first used just 300. Interestingly, on an independent test set, the configurations found by MF-GP-UCB consistently achieved over 90% accuracy while the OpenCV configuration achieved only 87.4% accuracy.

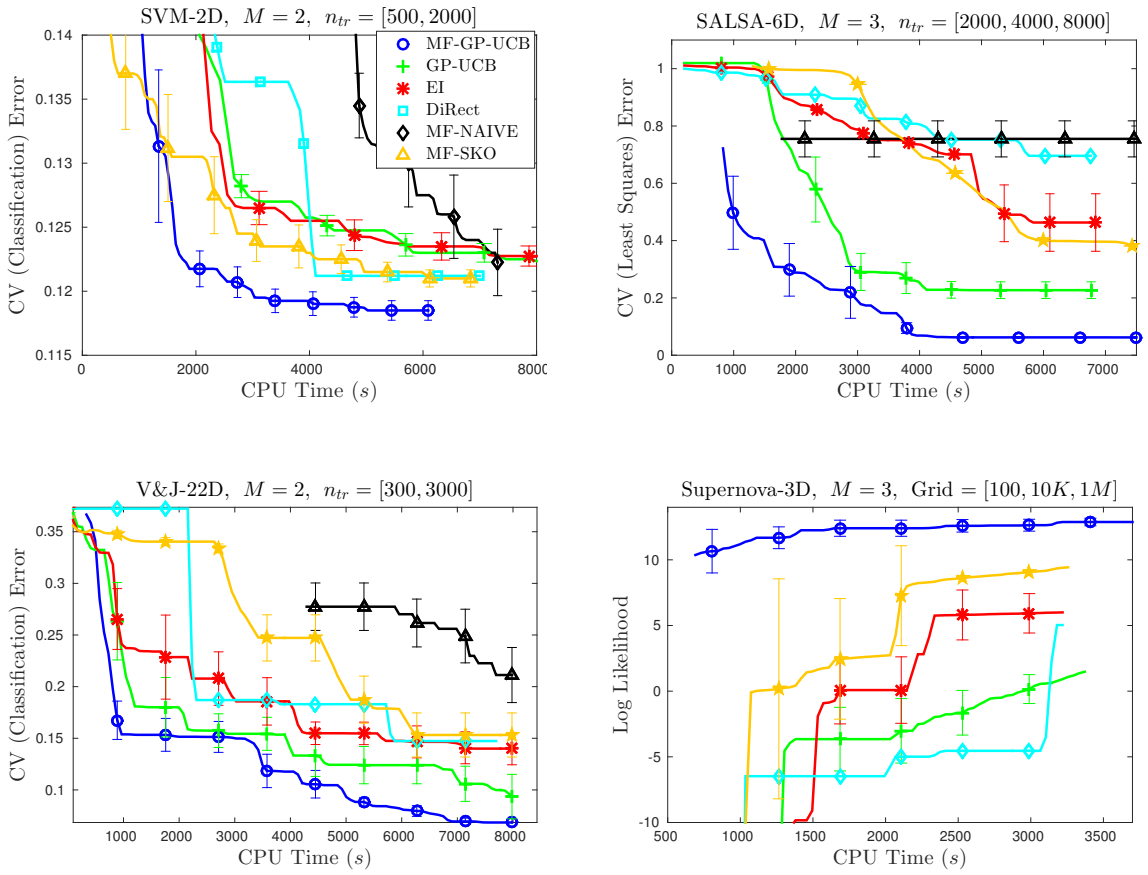


Figure 4.13: Results on the real experiments. The first three figures are hyperparameter tuning tasks while the last is an astrophysical maximum likelihood problem. The title states the experiment, dimensionality (number of hyperparameters or cosmological parameters) and the number of fidelities. For the three hyperparameter tuning tasks we plot the best cross validation error (lower is better) and for the astrophysics task we plot the highest log likelihood (higher is better). For the hyperparameter tuning tasks we obtained the lower fidelities by using smaller training sets, indicated by n_{tr} in the figures and for the astrophysical problem we used a coarser grid for numerical integration, indicated by "Grid". MF-NAIVE is not visible in the last experiment because it performed very poorly. All curves were produced by averaging over 10 experiments. The error bars indicate one standard error. The lengths of the curves are different in time as we ran each method for a pre-specified number of iterations and they concluded at different times.

Type Ia Supernovae (Supernova): We use Type Ia supernovae data from Davis et al [50] for maximum likelihood inference on 3 cosmological parameters, the Hubble constant $H_0 \in (60, 80)$, the dark matter fraction $\Omega_M \in (0, 1)$ and the dark energy fraction $\Omega_\Lambda \in (0, 1)$. Unlike typical parametric maximum likelihood problems we see in machine learning, the likelihood is only available as a black-box. It is computed using the Robertson–Walker metric Davis et al [50] which requires a (one dimensional) numerical integration for each sample in the dataset. We set this up as a $M = 3$ fidelity task. At the third fidelity, the integration was performed using the trapezoidal rule on a grid of size 10^6. For the first and second fidelities, we used grids of size $10^2, 10^4$ respectively. The goal is to maximise the likelihood at the third fidelity.

Some Implementation Details of other Baselines

For MF-NAIVE we limited the number of first fidelity evalutions to $\max\left(\frac{1}{2}\frac{\Lambda}{\lambda^{(1)}}, 500\right)$ where Λ was the total budget used in the experiment. The 500 limit was set to avoid unnecessary computation – for all of these problems, 500 queries are not required to find the maximum. While there are other methods for multi-fidelity optimisation (discussed under Related Work) none of them had made their code available nor were their methods straightforward to implement - this includes MF-SKO.

A straightforward way to incorporate lower fidelity information to GP-UCB and GP-EI is to share the same kernel parameters. This way, the kernel κ can be learned by by jointly maximising the marginal likelihood. While the idea seems natural, we got mixed results in practice. On some problems this improved the performance of all GP methods (including MF-GP-UCB), but on others all performed poorly. One explanation is that while lower fidelities approximate function values, they are not always best described by the same kernel. The results presented do not use lower fidelities to learn κ as it was more robust. For MF-GP-UCB, each $\kappa^{(m)}$ was learned independently using only the queries at fidelity m.

In addition to the baselines presented in the figures, we also compared our method to the following methods. The first two are single fidelity and the last two are mutlti-fidelity methods.

- Querying uniformly at random at the highest fidelity and taking the maximum. On all problems this performed worse than other methods.
- A variant of MF-NAIVE where instead of GP-UCB we queried at the first fidelity uniformly at random. On some problems this did better than querying with GP-UCB, probably since unlike GP-UCB it was not stuck at the maximum of $f^{(1)}$. However, generally it performed worse.
- The multi-fidelity method from Forrester, Alexander I. J. and Sóbester, András and Keane, Andy J. [60] also based on GPs. We found that this method did not perform as desired: in particular, it barely queried beyond the first fidelity.

Description of Synthetic Experiments

The following are the descriptions of the synthetic functions used. The first three functions and their approximations were taken from [274].

Currin exponential function: The domain is the two dimensional unit cube $\mathcal{X} = [0,1]^2$. The second and first fidelity functions are,

$$f^{(2)}(x) = \left(1 - \exp\left(\frac{-1}{2x_2}\right)\right)\left(\frac{2300x_1^3 + 1900x_1^2 + 2092x_1 + 60}{100x_1^3 + 500x_1^2 + 4x_1 + 20}\right),$$
$$f^{(1)}(x) = \frac{1}{4}f^{(2)}(x_1 + 0.05, x_2 + 0.05) + \frac{1}{4}f^{(2)}(x_1 + 0.05, \max(0, x_2 - 0.05)) +$$
$$\frac{1}{4}f^{(2)}(x_1 - 0.05, x_2 + 0.05) + \frac{1}{4}f^{(2)}(x_1 - 0.05, \max(0, x_2 - 0.05)).$$

Park function: The domain is $\mathcal{X} = [0,1]^4$. The second and first fidelity functions are,

$$f^{(2)}(x) = \frac{x_1}{2}\left(\sqrt{1 + (x_2 + x_3^2)\frac{x_4}{x_1^2}} - 1\right) + (x_1 + 3x_4)\exp(1 + \sin(x_3)),$$
$$f^{(1)}(x) = \left(1 + \frac{\sin(x_1)}{10}\right)f^{(2)}(x) - 2x_1^2 + x_2^2 + x_3^2 + 0.5.$$

Borehole function: The second and first fidelity functions are,

$$f^{(2)}(x) = \frac{2\pi x_3(x_4 - x_6)}{\log(x_2/x_1)\left(1 + \frac{2x_7x_3}{\log(x_2/x_1)x_1^2x_8} + \frac{x_3}{x_5}\right)},$$
$$f^{(1)}(x) = \frac{5x_3(x_4 - x_6)}{\log(x_2/x_1)\left(1.5 + \frac{2x_7x_3}{\log(x_2/x_1)x_1^2x_8} + \frac{x_3}{x_5}\right)}.$$

The domain of the function is $[0.05, 0.15; 100, 50K; 63.07K, 115.6K; 990, 1110; 63.1, 116; 700, 820; 1120, 1680; 9855, 12045]$. We first linearly transform the variables to lie in $[0,1]^8$.

Hartmann-3D function: The M^{th} fidelity function is $f^{(M)}(x) = \sum_{i=1}^4 \alpha_i \exp\left(-\sum_{j=1}^3 A_{ij}(x_j - P_{ij})^2\right)$ where $A, P \in \mathbb{R}^{4\times 3}$ are fixed matrices given below and $\alpha = [1.0, 1.2, 3.0, 3.2]$. For the lower fidelities we use the same form except changing α to $\alpha^{(m)} = \alpha + (M - m)\delta$ where $\delta = [0.01, -0.01, -0.1, 0.1]$ and $M = 3$. The domain is $\mathcal{X} = [0,1]^3$.

$$A = \begin{bmatrix} 3 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3 & 10 & 30 \\ 0.1 & 10 & 35 \end{bmatrix}, \quad P = 10^{-4} \times \begin{bmatrix} 3689 & 1170 & 2673 \\ 4699 & 4387 & 7470 \\ 1091 & 8732 & 5547 \\ 381 & 5743 & 8828 \end{bmatrix} \tag{4.17}$$

Hartmann-6D function: The 6-D Hartmann function takes the same form as the 3-D case except $A, P \in \mathbb{R}^{4\times 6}$ are as given below. We use the same modifications as above to obtain the lower

fidelities using $M = 4$.

$$A = \begin{bmatrix} 10 & 3 & 17 & 3.5 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{bmatrix}, \tag{4.18}$$

$$P = 10^{-4} \times \begin{bmatrix} 1312 & 1696 & 5569 & 124 & 8283 & 5886 \\ 2329 & 4135 & 8307 & 3736 & 1004 & 9991 \\ 2348 & 1451 & 3522 & 2883 & 3047 & 6650 \\ 4047 & 8828 & 8732 & 5743 & 1091 & 381 \end{bmatrix}$$

4.3 Multi-fidelity GP Bandits with Continuous Approximations

In this section, we study multi-fidelity GP bandits with a continuous spectrum of approximations. In addition to this generalisation, here we also consider approximation conditions of a more Bayesian flavour different to the uniform bound conditions in Chapters 4.1 and 4.2.

To motivate this set up, consider tuning a classification algorithm over a space of hyperparameters $\mathcal{X}$ by maximising a validation set accuracy. The algorithm is to be trained using N_{max} data points via an iterative algorithm for T_{max} iterations. However, we wish to use fewer training points $N < N_{\text{max}}$ and/or fewer iterations $T < T_{\text{max}}$ to approximate the validation accuracy. We can view validation accuracy as a function $g : [1, N_{\text{max}}] \times [1, T_{\text{max}}] \times \mathcal{X} \to \mathbb{R}$ where evaluating $g(N, T, x)$ requires training the algorithm with N points for T iterations with the hyperparameters x. If the training complexity of the algorithm is quadratic in data size and linear in the number of iterations, then the cost of this evaluation is $\lambda(N, T) = \mathcal{O}(N^2 T)$. Our goal is to find the optimum when $N = N_{\text{max}}$, and $T = T_{\text{max}}$, i.e. we wish to maximise $f(x) = g(N_{\text{max}}, T_{\text{max}}, x)$.

In this setting, while N, T are technically discrete choices, they are more naturally viewed as coming from a continuous 2 dimensional *fidelity space*, $[1, N_{\text{max}}] \times [1, T_{\text{max}}]$. One might hope that cheaper queries to $g(N, T, \cdot)$ with N, T less than $N_{\text{max}}, T_{\text{max}}$ can be used to learn about $g(N_{\text{max}}, T_{\text{max}}, \cdot)$ and consequently optimise it using less overall cost. Indeed, this is the case with many machine learning algorithms where cross validation performance tends to vary smoothly with data set size and number of iterations. Therefore, one may use cheap low fidelity experiments with small (N, T) to discard bad hyperparameters and deploy expensive high fidelity experiments with large (N, T) only in a small but promising region. The main theoretical result of this section (Theorem 23) shows that our proposed algorithm, BOCA, exhibits precisely this behaviour.

Continuous approximations also arise in simulation studies: where simulations can be carried out at varying levels of granularity, on-line advertising: where an ad can be controlled by continuous parameters such as display time or target audience, and several other experiment design tasks.

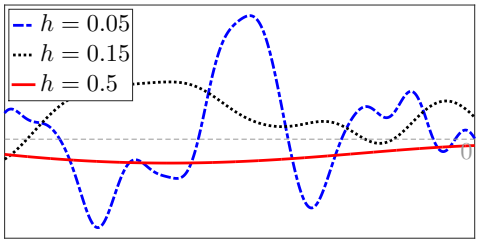


Figure 4.14: Samples drawn from a GP with 0 mean and SE kernel with bandwidths $h = 0.01, h = 0.15, 0.5$. Samples tend to be smoother across the domain for large bandwidths.

In fact, in many multi-fidelity papers, the finite approximations were obtained by discretising a continuous space [100, 123]. Here, we study a Bayesian optimisation technique that is directly designed for continuous fidelity spaces and is potentially applicable to more general spaces. Our main contributions in this section are,

1. A novel setting and model for multi-fidelity optimisation with continuous approximations using GP assumptions. We develop a novel algorithm, BOCA, for this setting.
2. A theoretical analysis characterising the behaviour and regret bound for BOCA.
3. An empirical study which demonstrates that BOCA outperforms alternatives, both multi-fidelity and otherwise, on a series of synthetic problems and real examples in hyperparameter tuning and astrophysics.

4.3.1 Problem Formalism

As before, for simplicity, we will take both $\mathcal{X}$ and $\mathcal{Z}$ to be compact Euclidean spaces. We begin with a brief review of *radial kernels*, which are common choices for prior covariances in GPs. Some examples are the squared exponential (SE) and Matérn kernels. Using a radial kernel means that the prior covariance can be written as $\kappa(x, x') = \kappa_0 \phi(\|x - x'\|)$ and depends only on the distance between x and x'. Here, the scale parameter κ_0 captures the magnitude f could deviate from μ. The function $\phi : \mathbb{R}_+ \to \mathbb{R}_+$ is a decreasing function with $\|\phi\|_\infty = \phi(0) = 1$. In this section, we will use the SE kernel in a running example to convey the intuitions in our methods. For the SE kernel, $\phi(r) = \phi_h(r) = \exp(-r^2/(2h^2))$, where $h \in \mathbb{R}_+$, called the *bandwidth* of the kernel, controls the smoothness of the GP. When h is large, the samples drawn from the GP tend to be smoother as illustrated in Figure 4.14. We will reference this observation frequently in the text.

Continuous Approximations: In this set up, we will let f be a slice of a function g that lies in a larger space. Precisely, we will assume the existence of a fidelity space $\mathcal{Z}$ and a function $g : \mathcal{Z} \times \mathcal{X} \to \mathbb{R}$ defined on the cartesian product of the fidelity space and domain. The function f which we wish to maximise is related to g via $f(\cdot) = g(z_\bullet, \cdot)$, where $z_\bullet \in \mathcal{Z}$. For instance, in the hyperparameter tuning example, $\mathcal{Z} = [1, N_{\text{max}}] \times [1, T_{\text{max}}]$ and $z_\bullet = [N_{\text{max}}, T_{\text{max}}]$. Our goal is to find a maximiser $x_\star \in \operatorname{argmax}_x f(x) = \operatorname{argmax}_x g(z_\bullet, x)$. We have illustrated this setup in Fig. 4.15. In the rest of the manuscript, the term "fidelities" will refer to points z in the fidelity space $\mathcal{Z}$. As before, we will impose the following smoothness/cost assumptions on the fidelities.

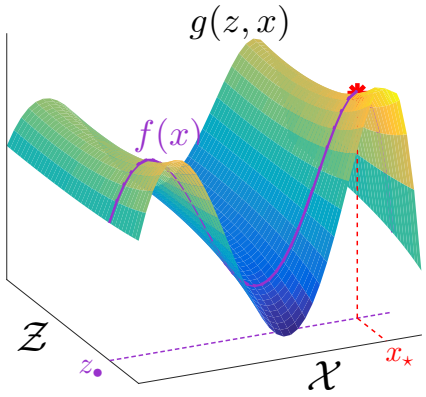


Figure 4.15: $g : \mathcal{Z} \times \mathcal{X} \to \mathbb{R}$ is a function defined on the product space of the fidelity space $\mathcal{Z}$ and domain $\mathcal{X}$. The purple line is $f(x) = g(z_\bullet, x)$. We wish to find the maximiser $x_\star \in \text{argmax}_{x \in \mathcal{X}} f(x)$. The multi-fidelity framework is attractive when g is smooth across $\mathcal{Z}$ as illustrated in the figure.

1. *There exist fidelities $z \in \mathcal{Z}$ where evaluating g is cheaper than evaluating at $z_\bullet$.* To this end, we will associate a *known* cost function $\lambda : \mathcal{Z} \to \mathbb{R}_+$. In the hyperparameter tuning example, $\lambda(z) = \lambda(N, T) = \mathcal{O}(N^2 T)$. It is helpful to think of $z_\bullet$ as being the most expensive fidelity, i.e. maximiser of λ, and that $\lambda(z)$ decreases as we move away from $z_\bullet$. However, this notion is strictly not necessary for our algorithm or results.

2. *The cheap $g(z, \cdot)$ evaluation gives us information about $g(z_\bullet, \cdot)$.* This is true if g is smooth across the fidelity space as illustrated in Fig. 4.15. As we will describe shortly, this smoothness can be achieved by modelling g as a GP with an appropriate kernel for the fidelity space $\mathcal{Z}$.

In the above setup, a multi-fidelity algorithm is a sequence of query-fidelity pairs $\{(z_t, x_t)\}_{t \geq 0}$ where, at time t, the algorithm chooses $z_t \in \mathcal{Z}$ and $x_t \in \mathcal{X}$, and observes $y_t = g(z_t, x_t) + \epsilon$ where $\mathbb{E}[\epsilon] = 0$. The choice of (z_t, x_t) can of course depend on the previous fidelity-query-observation triples $\{(z_i, x_i, y_i)\}_{i=1}^{t-1}$.

Multi-fidelity Simple Regret: We provide bounds on the simple regret $S(\Lambda)$ of a multi-fidelity optimisation method after it has spent capital Λ of a resource. As before, we will aim to provide *any capital* bounds, meaning that an algorithm would be expected to do well for *all* values of (sufficiently large) Λ. Say we have made N queries to g within capital Λ, i.e. N is the *random* quantity such that $N = \max\{n \geq 1 : \sum_{t=1}^{n} \lambda(z_t) \leq \Lambda\}$. While the cheap evaluations at $z \neq z_\bullet$ are useful in guiding search for the optimum of $g(z_\bullet, \cdot)$, there is no reward for optimising a cheaper $g(z, \cdot)$. Accordingly, we define the simple regret after capital Λ as,

$$S(\Lambda) = \begin{cases} \min_{\substack{t \in \{1,\ldots,N\} \\ \text{s.t } z_t = z_\bullet}} f(x_\star) - f(x_t) & \text{if we have queried at } z_\bullet, \\ +\infty & \text{otherwise.} \end{cases} \tag{4.19}$$

This definition is similar in spirit to the definition in Chapter 4.2 and reduces to the single fidelity definition (1.1) when we only query g at $z_\bullet$. It is also similar to the definition in Kandasamy et al. [123], but unlike them, we do not impose additional boundedness constraints on f or g.

Assumptions: As we will be primarily focusing on continuous and compact domains and fidelity spaces, going forward we will assume, without any loss of generality, that $\mathcal{X} = [0, 1]^d$ and $\mathcal{Z} = [0, 1]^p$. We discuss non-continuous settings briefly at the end of Chapter 4.3.2. In keeping

with similar work in the Bayesian optimisation literature, we will assume $g \sim \mathcal{GP}(\mathbf{0}, \kappa)$ and upon querying at (z, x) we observe $y = g(z, x) + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \eta^2)$. $\kappa : (\mathcal{Z} \times \mathcal{X})^2 \to \mathbb{R}$ is the prior covariance defined on the product space. We will study κ of the following form,

$$\kappa([z, x], [z', x']) \;=\; \kappa_0 \, \phi_{\mathcal{Z}}(\|z - z'\|) \, \phi_{\mathcal{X}}(\|x - x'\|). \tag{4.20}$$

Here, $\kappa_0 \in \mathbb{R}_+$ is the scale parameter and $\phi_{\mathcal{Z}}, \phi_{\mathcal{X}}$ are radial kernels defined on $\mathcal{Z}, \mathcal{X}$ respectively. The fidelity space kernel $\phi_{\mathcal{Z}}$ is an important component in this work. It controls the smoothness of g across the fidelity space and hence determines how much information the lower fidelities provide about $g(z_\bullet, \cdot)$. For example, suppose that $\phi_{\mathcal{Z}}$ was a SE kernel. A favourable setting for a multi-fidelity method would be for $\phi_{\mathcal{Z}}$ to have a large bandwidth $h_{\mathcal{Z}}$ as that would imply that g is very smooth across $\mathcal{Z}$. We will see that $h_{\mathcal{Z}}$ determines the behaviour and theoretical guarantees of BOCA in a natural way when $\phi_{\mathcal{Z}}$ is the SE kernel. To formalise this notion, we will define the *information gap* as follows, $\xi : \mathcal{Z} \to [0, 1]$.

$$\xi(z) = \sqrt{1 - \phi_{\mathcal{Z}}(\|z - z_\bullet\|)^2}. \tag{4.21}$$

One interpretation of $\xi(z)$ is that it measures the gap in information about $g(z_\bullet, \cdot)$ when we query at $z \neq z_\bullet$. That is, it is the price we have to pay, in information, for querying at a cheap fidelity. Observe that ξ increases when we move away from $z_\bullet$ in the fidelity space. For the SE kernel, it can be shown[1] $\xi(z) \approx \frac{\|z - z_\bullet\|}{h_{\mathcal{Z}}}$. For large $h_{\mathcal{Z}}$, g is smoother across $\mathcal{Z}$ and we can expect the lower fidelities to be more informative about f; as expected the information gap ξ is small for large $h_{\mathcal{Z}}$. If $h_{\mathcal{Z}}$ is small and g is not smooth, the gap ξ is large and lower fidelities are not as informative.

Before we present our algorithm for the above setup, we will introduce notation for the posterior GPs for g and f. Let $\mathcal{D}_n = \{(z_i, x_i, y_i)\}_{i=1}^n$ be n fidelity, query, observation values from the GP g, where y_i was observed when evaluating $g(z_i, x_i)$. We will denote the posterior mean and standard deviation of g conditioned on $\mathcal{D}_n$ by ν_n and τ_n respectively (ν_n, τ_n can be computed from (2.1) by replacing $x \leftarrow [z, x]$). Therefore $g(z, x)|\mathcal{D}_n \sim \mathcal{N}(\nu_n(z, x), \tau_n^2(z, x))$ for all $(z, x) \in \mathcal{Z} \times \mathcal{X}$. We will further denote

$$\mu_n(\cdot) = \nu_n(z_\bullet, \cdot), \qquad \sigma_n(\cdot) = \tau_n(z_\bullet, \cdot), \tag{4.22}$$

to be the posterior mean and standard deviation of $g(z_\bullet, \cdot) = f(\cdot)$. It follows that $f|\mathcal{D}_n$ is also a GP and satisfies $f(x)|\mathcal{D}_n \sim \mathcal{N}(\mu_n(x), \sigma_n^2(x))$ for all $x \in \mathcal{X}$.

4.3.2 Bayesian Optimisation with Continuous Approximations (BOCA)

BOCA is a sequential strategy to select a domain point $x_t \in \mathcal{X}$ and fidelity $z_t \in \mathcal{Z}$ at time t based on previous observations. At time t, we will first construct an upper confidence bound φ_t for the function f we wish to optimise. It takes the form,

$$\varphi_t(x) = \mu_{t-1}(x) + \beta_t^{1/2} \sigma_{t-1}(x). \tag{4.23}$$

[1] Strictly, $\xi(z) \leq \|z - z_\bullet\|/h_{\mathcal{Z}}$, but the inequality is tighter for larger $h_{\mathcal{Z}}$. In any case, ξ is strictly decreasing with $h_{\mathcal{Z}}$.

Recall from (4.22) that μ_{t-1} and σ_{t-1} are the posterior mean and standard deviation of f using the observations from the previous $t-1$ time steps at all fidelities, i.e. the entire $\mathcal{Z} \times \mathcal{X}$ space. We will specify β_t in Theorems 23, 38. Following other UCB algorithms, our next point x_t in the domain $\mathcal{X}$ for evaluating g is a maximiser of φ_t, i.e. $x_t \in \operatorname{argmax}_{x \in \mathcal{X}} \varphi_t(x)$.

Next, we need to determine the fidelity $z_t \in \mathcal{Z}$ to query g. For this we will first select a subset $\mathcal{Z}_t(x_t)$ of $\mathcal{Z}$ as follows,

$$\mathcal{Z}_t(x_t) = \Big\{ z \in \mathcal{Z} \ : \ \lambda(z) < \lambda(z_\bullet), \ \ \tau_{t-1}(z, x_t) > \gamma(z), \ \ \xi(z) > \beta_t^{-1/2} \|\xi\|_\infty \Big\}, \qquad (4.24)$$

$$\text{where} \quad \gamma(z) = \sqrt{\kappa_0}\, \xi(z) \left(\frac{\lambda(z)}{\lambda(z_\bullet)} \right)^q.$$

Here, ξ is the information gap function in (4.21) and τ_{t-1} is the posterior standard deviation of g, and p, d are the dimensionalities of $\mathcal{Z}, \mathcal{X}$. The exponent q depends on the kernel used for $\phi_{\mathcal{Z}}$. For e.g., for the SE kernel, $q = 1/(p+d+2)$. We filter out the fidelities we consider at time t using three conditions as specified above. We elaborate on these conditions in more detail in Chapter 4.3.2. If $\mathcal{Z}_t$ is not empty, we choose the cheapest fidelity in this set, i.e. $z_t \in \operatorname{argmin}_{z \in \mathcal{Z}_t} \lambda(z)$. If $\mathcal{Z}_t$ is empty, we choose $z_t = z_\bullet$.

We have summarised the resulting procedure below in Algorithm 5. An important advantage of BOCA is that it only requires specifying the GP hyperparameters for g such as the kernel κ. In practice, this can be achieved by various effective heuristics such as maximising the GP marginal likelihood or cross validation which are standard in most BO methods. In contrast, MF-GP-UCB of Kandasamy et al. [123] requires tuning several other hyperparameters.

Algorithm 5 BOCA from Kandasamy et al. [127]

Input: kernel κ.

- Set $\nu_0(\cdot) \leftarrow \mathbf{0}$, $\tau_0(\cdot) \leftarrow \kappa(\cdot,\cdot)^{1/2}$, $\mathcal{D}_0 \leftarrow \varnothing$.
- for $t = 1, 2, \ldots$
  1. $x_t \leftarrow \operatorname{argmax}_{x \in \mathcal{X}} \varphi_t(x)$. See (4.23)
  2. $z_t \leftarrow \operatorname{argmin}_{z \in \mathcal{Z}_t(x_t) \cup \{z_\bullet\}} \lambda(z)$. See (4.24)
  3. $y_t \leftarrow$ Query g at (z_t, x_t).
  4. $\mathcal{D}_t \leftarrow \mathcal{D}_{t-1} \cup \{(z_t, x_t, y_t)\}$. Update posterior mean ν_t, and standard deviation τ_t for g conditioned on $\mathcal{D}_t$.

Fidelity Selection Criterion

We will now provide an intuitive justification for the three conditions in the selection criterion for z_t, i.e., equation (4.24). The first condition, $\lambda(z) < \lambda(z_\bullet)$ is fairly obvious; since we wish to optimise $g(z_\bullet, \cdot)$ and since we are not rewarded for queries at other fidelities, there is no reason to consider fidelities that are more expensive than $z_\bullet$.

The second condition, $\tau_{t-1}(z, x_t) > \gamma(z)$ says that we will only consider fidelities where the posterior variance is larger than a threshold $\gamma(z) = \sqrt{\kappa_0}\xi(z)(\lambda(z)/\lambda(z_\bullet))^q$, which depends critically

on two quantities, the cost function λ and the information gap ξ. As a first step towards parsing this condition, observe that a reasonable multi-fidelity strategy should be inclined to query cheap fidelities and learn about g before querying expensive fidelities. As $\gamma(z)$ is monotonically increasing in $\lambda(z)$, it becomes easier for a cheap z to satisfy $\tau_{t-1}(z, x_t) > \gamma(z)$ and be included in $\mathcal{Z}_t$ at time t. Moreover, since we choose z_t to be the minimiser of λ in $\mathcal{Z}_t$, a cheaper fidelity will always be chosen over expensive ones if included in $\mathcal{Z}_t$. Second, if a particular fidelity z is far away from $z_\bullet$, it probably contains less information about $g(z_\bullet, \cdot)$. Again, a reasonable multi-fidelity strategy should be discouraged from making such queries. This is precisely the role of the information gap ξ which is increasing with $\|z - z_\bullet\|$. As z moves away from $z_\bullet$, $\gamma(z)$ increases and it becomes harder to satisfy $\tau_{t-1}(z, x_t) > \gamma(z)$. Therefore, such a z is less likely to be included in $\mathcal{Z}_t(x_t)$ and be considered for evaluation. Our analysis reveals that setting γ as in (4.24) is a reasonable trade off between cost and information in the approximations available to us; cheaper fidelities cost less, but provide less accurate information about the function f we wish to optimise. It is worth noting that the second condition is similar in spirit to Kandasamy et al. [123] who proceed from a lower to higher fidelity only when the lower fidelity variance is smaller than a threshold. However, while they treat the threshold as a hyperparameter, we are able to explicitly specify theoretically motivated values.

The third condition in (4.24) is $\xi(z) > \|\xi\|_\infty / \beta_t^{1/2}$. Since ξ is increasing as we move away from $z_\bullet$, it says we should exclude fidelities inside a (small) neighbourhood of $z_\bullet$. Recall that if $\mathcal{Z}_t$ is empty, BOCA will choose $z_\bullet$ by default. But when it is not empty, we want to prevent situations where we get arbitrarily close to $z_\bullet$ but not actually query *at* $z_\bullet$. Such pathologies can occur when we are dealing with a continuum of fidelities and this condition forces BOCA to pick $z_\bullet$ instead of querying very close to it. Observe that since β_t is increasing with t, this neighborhood is shrinking with time and therefore the algorithm will eventually have the opportunity to evaluate fidelities close to $z_\bullet$.

Extensions: While we have focused on continuous $\mathcal{Z}$, many of the ideas here can be extended to other settings. If $\mathcal{Z}$ is a discrete subset of $[0, 1]^p$ our work extends straightforwardly. We reiterate that this will not be the same as the finite fidelity MF-GP-UCB algorithm as the assumptions are different. In particular, Kandasamy et al. [123] are not able to effectively share information across fidelities as we do. We also believe that Algorithm 5 can be extended to arbitrary fidelity spaces $\mathcal{Z}$ provided that a kernel can be defined on $\mathcal{Z}$. Our results can also be extended to discrete domains $\mathcal{X}$ and various other kernels for $\phi_\mathcal{X}$ by adopting techniques from Srinivas et al. [235].

4.3.3 Theoretical Analysis

We now present our main theoretical results for BOCA. In our analysis of BOCA we show that most queries to g at fidelity $z_\bullet$ will be confined to a small subset of $\mathcal{X}$ which contains the optimum $x_\star$. Precisely, for sufficiently large capital Λ, for any $\alpha \in (0, 1)$, we show there exists $\rho > 0$ such that the number of queries *outside* the following set $\mathcal{X}_\rho$ is less than n_Λ^α.

$$\mathcal{X}_\rho = \left\{ x \in \mathcal{X} \,:\, f(x_\star) - f(x) \leq 2\rho\sqrt{\kappa_0}\, \|\xi\|_\infty \right\}. \tag{4.25}$$

Here, ξ is from (4.21). While it is true that any optimisation algorithm would eventually query extensively in a neighbourhood around the optimum, a strong result of the above form is not always possible. For instance, for GP-UCB, the best achievable bound on the number of queries in any set that does not contain $x_\star$ is $n_\Lambda^{1/2}$. The fact that $\mathcal{X}_\rho$ exists relies crucially on the multi-fidelity assumptions and that our algorithm leverages information from lower fidelities when querying at $z_\bullet$. As ξ is small when g is smooth across $\mathcal{Z}$, the set $\mathcal{X}_\rho$ will be small when the approximations are highly informative about $g(z_\bullet, \cdot)$. For e.g., when $\phi_\mathcal{Z}$ is a SE kernel, we have $\mathcal{X}_\rho \approx \{x \in \mathcal{X} : f(x_\star) - f(x) \leq 2\rho\sqrt{\kappa_0 p}/h_\mathcal{Z}\}$. When $h_\mathcal{Z}$ is large and g is smooth across $\mathcal{Z}$, $\mathcal{X}_\rho$ is small as the right side of the inequality is smaller. As BOCA confines most of its evaluations to this small set containing $x_\star$, we will be able to achieve much better regret than GP-UCB. When $h_\mathcal{Z}$ is small and g is not smooth across $\mathcal{Z}$, the set $\mathcal{X}_\rho$ becomes large and the advantage of multi-fidelity optimisation diminishes. One can similarly argue that for the Matérn kernel, as the parameter ν increases, g will be smoother across $\mathcal{Z}$, and $\mathcal{X}_\rho$ becomes smaller yielding better bounds on the regret. To simplify the ensuing discussion, we provide an informal statement of our main theoretical result below. $\lesssim, \asymp$ denotes inequality and equality ignoring constant and polylog terms. A formal statement is available in Theorem 38 in Chapter 4.6.

Theorem 23 (Informal, Regret of BOCA)**.** *Let $g \sim \mathcal{GP}(0, \kappa)$ where κ satisfies (4.20). Choose $\beta_t \asymp d\log(t/\delta)$. Then, for sufficiently large Λ and for all $\alpha \in (0, 1)$, there exists ρ depending on α such that the following bound holds with probability at least $1 - \delta$.*

$$S(\Lambda) \lesssim \sqrt{\frac{\Psi_{n_\Lambda}(\mathcal{X}_\rho)}{n_\Lambda}} + \sqrt{\frac{\Psi_{n_\Lambda^\alpha}(\mathcal{X})}{n_\Lambda^{2-\alpha}}}$$

In the above bound, the latter term vanishes fast due to the $n_\Lambda^{-(1-\alpha/2)}$ dependence. When comparing this with Theorem 1, we see that we outperform GP-UCB by a factor of $\sqrt{\Psi_{n_\Lambda}(\mathcal{X}_\rho)/\Psi_{n_\Lambda}(\mathcal{X})} \asymp \sqrt{\mathrm{vol}(\mathcal{X}_\rho)/\mathrm{vol}(\mathcal{X})}$ asymptotically. If g is smooth across the fidelity space, $\mathcal{X}_\rho$ is small and the gains over GP-UCB are significant. If g becomes less smooth across $\mathcal{Z}$, the bound decays gracefully, but we are never worse than GP-UCB up to constant factors. This bound also has similarities to Theorems 21 and 22 in Chapter 4.2, where we demonstrate better bounds than GP-UCB by showing that the regret is dominated by queries inside a good set which contains the optimum. However, the characterisation of this set is different in both instances due to different approximation conditions.

4.3.4 Experiments

We compare BOCA to the following four baselines: (i) GP-UCB, (ii) the GP-EI criterion in BO [113], (iii) MF-GP-UCB from Chapter 4.2 which uses only a finite number of approximations, (iv) MF-SKO, the multi-fidelity sequential kriging optimisation method from Huang et al. [100]. All methods are based on GPs and we use the SE kernel for both the fidelity space and domain. The first two are not multi-fidelity methods, while the last two are finite multi-fidelity

methods[2]. Kandasamy et al. [123] also study some naive multi-fidelity algorithms and demonstrate that they do not perform well; as such we will not consider such alternatives here. In all our experiments, the fidelity space was designed to be $\mathcal{Z} = [0,1]^p$ with $z_\bullet = \mathbf{1}_p = [1,\ldots,1] \in \mathbb{R}^p$ being the most expensive fidelity. For MF-GP-UCB and MF-SKO, we used 3 fidelities (2 approximations) where the approximations were obtained at $z = 0.3331_p$ and $z = 0.6671_p$ in $\mathcal{Z}$. Empirically, we found that both algorithms did reasonably well with 1-3 approximations, but did not perform well with a large number of approximations (> 5); even the original papers restrict experiments to 1-3 approximations. Implementation details for all methods are given in at the end of this section.

Synthetic Experiments

The results for the synthetic experiments are given in Figure 4.16. The title of each figure states the function used, and the dimensionalities p, d of the fidelity space and domain. To reflect the setting in our theory, we add Gaussian noise to the function value when observing g at any (z, x). This makes the problem more challenging than standard global optimisation problems where function evaluations are not noisy. The functions g, the cost functions λ and the noise variances η^2 are given at the end of this section.

The first two panels in Figure 4.16 are simple sanity checks. In both cases, $\mathcal{Z} = [0,1]$, $\mathcal{X} = [0,1]$ and the functions were sampled from GPs. The GP was made known to all methods, i.e. all methods used the true GP in picking the next point. In the first panel, we used an SE kernel with bandwidth 0.1 for $\phi_{\mathcal{X}}$ and 1.0 for $\phi_{\mathcal{Z}}$. g is smooth across $\mathcal{Z}$ in this setting, and BOCA outperforms other baselines. The curve starts mid-way as BOCA is yet to query at $z_\bullet$ up until that point. The second panel uses the same set up as the first except we used bandwidth 0.01 for $\phi_{\mathcal{Z}}$. Even though g is highly un-smooth across $\mathcal{Z}$, BOCA does not perform poorly. This corroborates a claim that we made earlier that BOCA can naturally adapt to the smoothness of the approximations. The other multi-fidelity methods suffer in this setting.

In the next three figures, we use some standard benchmarks for global optimisation. We modify them to obtain g and add noise to the observations. As the kernel and other GP hyperparameters are unknown, we learn them by maximising the marginal likelihood every 25 iterations. We outperform all methods on all problems except in the case of the Borehole function where MF-GP-UCB does better.

The last synthetic experiment is the Branin function. We used the same set up as above, but use 10 fidelities for MF-GP-UCB and MF-SKO where the k^{th} fidelity is obtained at $z = \frac{k}{10}\mathbf{1}_p$ in the fidelity space. Notice that the performance of finite fidelity methods deteriorate. In particular, as MF-GP-UCB does not share information across fidelities, the approximations need to be designed carefully for the algorithm to work well. The more natural modelling assumptions in BOCA prevent such pitfalls. We next present two real examples in astrophysics and hyperparameter

[2]To our knowledge, the only other work that applies to continuous approximations is Klein et al. [139] which was developed specifically for hyperparameter tuning. Further, their implementation is not made available and is not straightforward to implement.

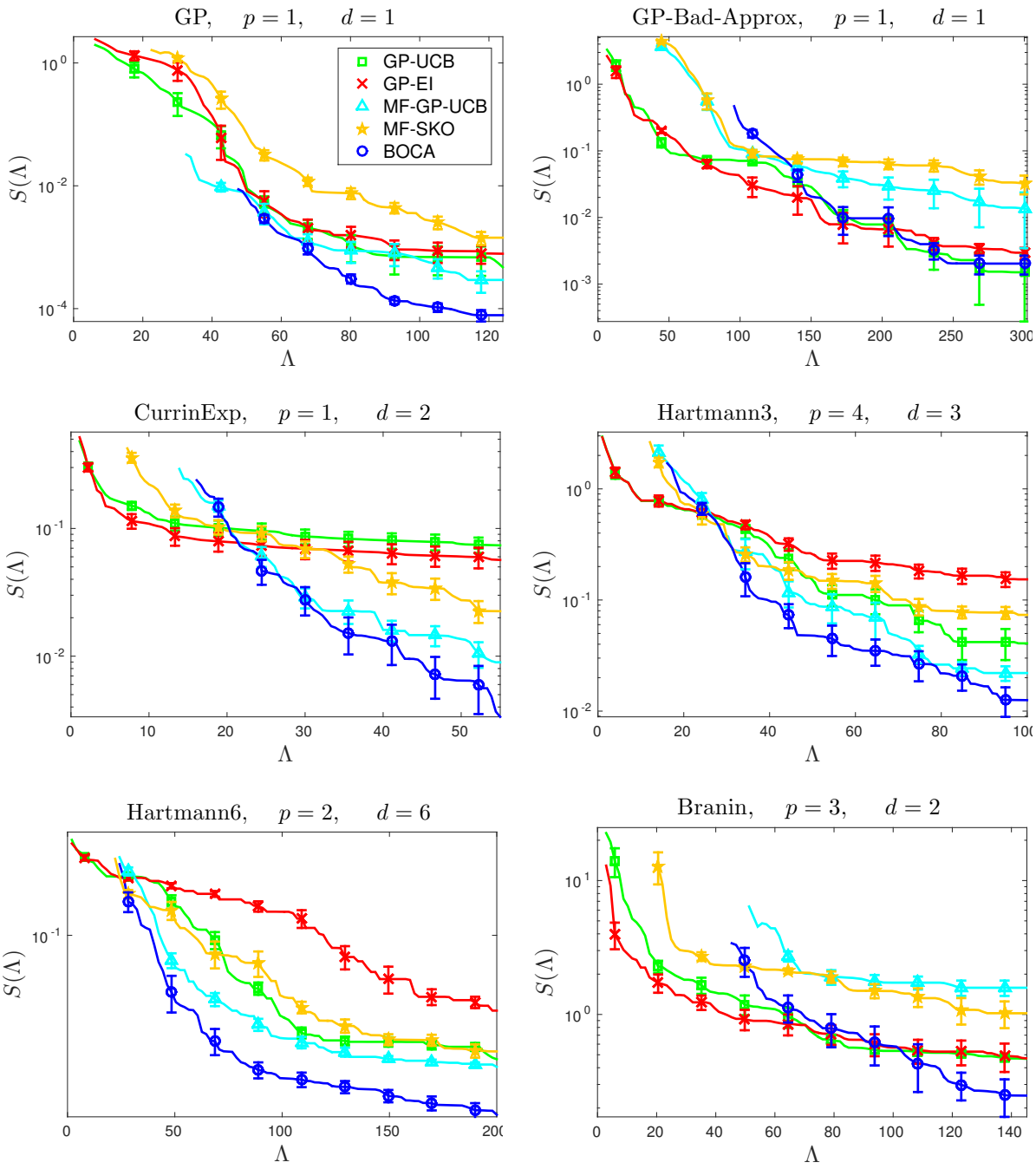
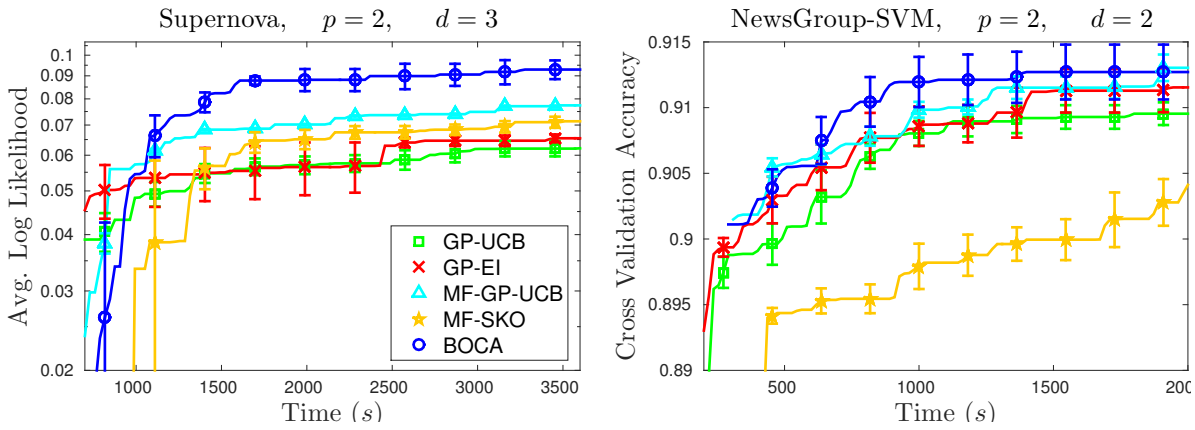


Figure 4.16: Results on 6 synthetic problems where we plot the simple regret $S(\Lambda)$ (lower is better) against the capital Λ. The title states the function used, and the fidelity and domain dimesions. For the first two figures we used capital $30\lambda(z_\bullet)$, therefore a method which only queries at $g(z_\bullet, \cdot)$ can make at most 30 evaluations. For the third figure we used $50\lambda(z_\bullet)$, for the fourth $100\lambda(z_\bullet)$ and for the last $50\lambda(z_\bullet)$ to reflect the dimensionality d of $\mathcal{X}$. The curves for the multi-fidelity methods start mid-way since they have not queried at $z_\bullet$ up until that point. All curves were produced by averaging over 20 experiments and the error bars indicate one standard error.



(a) (b)

Figure 4.17: Results on the supernova (a) and news group experiments (b). We have plotted the maximum value (higher is better) against wall clock time. Both curves were produced by averaging over 10 xperiments each. The error bars indicate one standard error.

tuning. We do *not* add noise to the observations, but treat it as optimisation tasks, where the goal is to maximise the function.

Astrophysical Maximum Likelihood Inference

We use data on TypeIa supernova for maximum likelihood inference on 3 cosmological parameters, the Hubble constant $H_0 \in (60, 80)$, the dark matter fraction $\Omega_M \in (0, 1)$ and dark energy fraction $\Omega_\Lambda \in (0, 1)$; hence $d = 3$. The likelihood is given by the Robertson-Walker metric, the computation of which requires a one dimensional numerical integration for each point in the dataset. Unlike typical maximum likelihood problems, here the likelihood is only accessible via point evaluations. We use the dataset from Davis et al [50] which has data on 192 supernovae. We construct a $p = 2$ dimensional multi-fidelity problem where we can choose between data set size $N \in [50, 192]$ and perform the integration on grids of size $G \in [10^2, 10^6]$ via the trapezoidal rule. As the cost function for fidelity selection, we used $\lambda(N, G) = NG$ as the computation time is linear in both parameters. Our goal is to maximise the average log likelihood at $z_\bullet = [192, 10^6]$. For the finite fidelity methods we use three fidelities with the approximations available at $z = [97, 2.15 \times 10^3]$ and $z = [145, 4.64 \times 10^4]$ (which correspond to 0.3331_p and 0.6671_p after rescaling as in the synthetic experiments). The results are given in Figure 4.17(a) where we plot the maximum average log likelihood against wall clock time as that is the cost in this experiment. The plot includes the time taken by each method to tune the GPs and determine the next points/fidelities for evaluation.

Support Vector Classification with 20 news groups

We use the 20 news groups dataset [111] in a text classification task. We obtain the bag of words representation for each document, convert them to tf-idf features and feed them to a support vec-

tor classifier. The goal is to tune the regularisation penalty and the temperature of the rbf kernel both in the range $[10^{-2}, 10^{3}]$; hence $d = 2$. The support vector implementation was taken from scikit-learn. We set this up as a 2 dimensional multi-fidelity problem where we can choose a dataset size $N \in [5000, 15000]$ and the number of training iterations $T \in [20, 100]$. Each evaluation takes the given dataset of size N and splits it up into 5 to perform 5-fold cross validation. As the cost function for fidelity selection, we used $\lambda(N, T) = NT$ as the training/validation complexity is linear in both parameters. Our goal is to maximise the cross validation accuracy at $z_\bullet = [15000, 100]$. For the finite fidelity methods we use three fidelities with the approximations available at $z = [8333, 47]$ and $z = [11667, 73]$. The results are given in Figure 4.17(b) where we plot the average cross validation accuracy against wall clock time.

Implementation Details

We describe some of our implementation details of BOCA and other BO methods below.

Domain and Fidelity space: Given a problem with arbitrary domain $\mathcal{X}$ and $\mathcal{Z}$, we mapped them to $[0, 1]^d$ and $[0, 1]^p$ by appropriately linear transforming the coordinates.

Initialisation: Following recommendations in Brochu et al. [23] all GP methods were initialised with uniform random queries with $\Lambda/10$ capital, where Λ is the total capital used in the experiment. For GP-UCB and GP-EI all queries were initialised at $z_\bullet$ whereas for the multi-fidelity methods, the fidelities were picked at random from the available fidelities.

GP Hyperparameters: Except in the first two experiments of Figure 4.16, the GP hyperparameters were learned after initialisation by maximising the GP marginal likelihood [203] and then updated every 25 iterations. We use an SE kernel for both $\phi_{\mathcal{X}}$ and $\phi_{\mathcal{Z}}$ and instead of using one bandwidth for the entire fidelity space and domain, we learn a bandwidth for each dimension separately. We learn the kernel scale, bandwidths and noise variance using marginal likelihood. The mean of the GP is set to be the median of the observations.

Choice of β_t: β_t, as specified in Theorem 23 has unknown constants and tends to be too conservative in practice [235]. Following intuitions described previously we set it to be $\beta_t = 0.5d \log(2\ell t + 1)$. Here, ℓ is the effective L_1 diameter of $\mathcal{X}$ and is computed by scaling each dimension by the inverse of the bandwidth of the SE kernel for that dimension.

Maximising φ_t: We used the DiRect algorithm [112].

Fidelity selection: Since we only worked in low dimensional fidelity spaces, the set $\mathcal{Z}_t$ was constructed in practice by obtaining a finely sampled grid of $\mathcal{Z}$ and then filtering out those which satisfied the 3 conditions in (4.24). In the second condition of (4.24), the threshold $\gamma(z)$ can be multiplied up to a constant factor, i.e $c\gamma(z)$ without affecting our theoretical results. In practice, we started with $c = 1$ but we updated it every 20 iterations via the following rule: if the algorithm has queried $z_\bullet$ more than 75% of the time in the last 20 iterations, we decrease it to $c/2$ and if it queried less than 25% of the time we increase it to $2c$. But the c value is always clipped inbetween 0.1 and 20. In practice we observed that the value for c usually stabilised around 1

and 8 although in some experiments it shot up to 20. Changing c this way resulted in slightly better performance in practice.

Description of Synthetic Functions

The following are the synthetic functions used in the paper.

GP Samples: For the GP samples in the first two experiments of Figure 4.16 we used an SE kernel with bandwidth 0.1 for $\phi_{\mathcal{X}}$. For $\phi_{\mathcal{Z}}$ we used bandwidths 1 and 0.01 for the first and second experiments respectively. The function was constructed by obtaining the GP function values on a 50×50 grid in the two dimensional $\mathcal{Z} \times \mathcal{X}$ space and then interpolating for evaluations in between via bivariate splines. For both experiments we used $\eta^2 = 0.05$ and the cost function $\lambda(z) = 0.2 + 6z^2$.

Currin exponential function: The domain is the two dimensional unit cube $\mathcal{X} = [0,1]^2$ and the fidelity was $\mathcal{Z} = [0,1]$ with $z_\bullet = 1$. We used $\lambda(z) = 0.1 + z^2$, $\eta^2 = 0.5$ and,

$$g(z,x) = \left(1 - 0.1(1-z)\exp\left(\frac{-1}{2x_2}\right)\right)\left(\frac{2300x_1^3 + 1900x_1^2 + 2092x_1 + 60}{100x_1^3 + 500x_1^2 + 4x_1 + 20}\right).$$

Hartmann functions: We used $g(z,x) = \sum_{i=1}^{4}(\alpha_i - \alpha_i'(z))\exp\big(-\sum_{j=1}^{3} A_{ij}(x_j - P_{ij})^2\big)$. Here $\alpha = [1.0, 1.2, 3.0, 3.2]$ and α_i' was set as $\alpha_i'(z) = 0.1(1 - z_i)$ if $i \leq p$ for $i = 1, 2, 3, 4$. A and P are as given in equations (4.17) and (4.18) for the 3 and 6 dimensional cases respectively. We constructed the $p = 4$ and $p = 2$ Hartmann functions for the 3 and 6 dimensional cases respectively this way. When $z = z_\bullet = \mathbf{1}_p$, this reduces to the usual Hartmann function commonly used as a benchmark in global optimisation. For the 3 dimensional case we used $\lambda(z) = 0.05 + (1 - 0.05)z_1^3 z_2^2$ as the cost function and $\eta^2 = 0.01$. for the 6 dimensional case we used $\lambda(z) = 0.05 + (1 - 0.05)z_1^3 z_2^2 z_3^{1.5} z_4^1$ and $\eta^2 = 0.05$.

4.4 Proofs for Theoretical Results in Chapter 4.1

4.4.1 Proof Outline

We first outline our plan of attack for the proofs of both the upper and lower bounds.

Upper Bound - Theorem 16

First we analyse MF-UCB after n plays (at any fidelity) and control the number of plays of an arm at various fidelities depending on which $\mathcal{X}^{(m)}$ it belongs to. To that end we prove the following.

Lemma 24. (Bounding $\mathbb{E}[T_{k,n}^{(m)}]$ – Informal) *After n time steps of* MF-UCB *for any $k \in \mathcal{K}$,*

$$T_{k,n}^{(\ell)} \lesssim \frac{\log(n)}{\psi(\gamma^{(m)})}, \quad \forall\, \ell < [\![k]\!], \qquad \mathbb{E}[T_{k,n}^{([\![k]\!])}] \lesssim \frac{\log(n)}{\psi(\Delta_k^{([\![k]\!])}/2)}, \qquad \mathbb{E}[T_{k,n}^{(>[\![k]\!])}] \leq \mathcal{O}(1).$$

The bounds above are illustrated in Table 4.1. Let $\tilde{R}_k(\Lambda) = \sum_{m=1}^{M} \lambda^{(m)} \Delta_k^{(M)} T_{k,N}^{(m)}$ be the regret incurred due to arm k and $\tilde{R}_{kn} = \mathbb{E}[\tilde{R}_k(\Lambda)|N = n]$. Using Lemma 24 we have,

$$\frac{\tilde{R}_{kn}}{\Delta_k^{(M)} \log(n)} \lesssim \sum_{\ell=1}^{[\![k]\!]-1} \frac{\lambda^{(\ell)}}{\psi(\gamma^{(m)})} + \frac{\lambda^{([\![k]\!])}}{\psi(\Delta_k^{([\![k]\!])}/2)} + o(1) \tag{4.26}$$

The next step will be to control the number of plays N within capital Λ which will bound $\mathbb{E}[\log(N)]$. While $\Lambda/\lambda^{(1)}$ is an easy bound, we will see that for MF-UCB, N will be on the order of $n_\Lambda = \Lambda/\lambda^{(M)}$. For this we will use the following high probability bounds on $T_{k,n}^{(m)}$.

Lemma 25. (Bounding $\mathbb{P}(T_{k,n}^{(m)} > \cdot\,)$ – Informal) *After n time steps of* MF-UCB *for any $k \in \mathcal{K}$,*

$$\mathbb{P}\left(T_{k,n}^{([\![k]\!])} \gtrsim x \cdot \frac{\log(n)}{\psi(\Delta_k^{([\![k]\!])}/2)}\right) \lesssim \frac{1}{n^{x\rho-1}}, \qquad \mathbb{P}\left(T_{k,n}^{(>[\![k]\!])} > x\right) \lesssim \frac{1}{x^{\rho-2}}.$$

We bound the number of plays at fidelities less than M via Lemma 25 and obtain $n/2 > \sum_{m=1}^{M-1} Q_n^{(m)}$ with probability greater than, say δ, for all $n \geq n_0$. By setting $\delta = 1/\log(\Lambda/\lambda^{(1)})$, we get $\mathbb{E}[\log(N)] \lesssim \log(n_\Lambda)$. The actual argument is somewhat delicate since δ depends on Λ.

This gives as an expression for the regret due to arm k to be of the form (4.40) where n is replaced by n_Λ. Then we we argue that the regret incurred by an arm k at fidelities less than $[\![k]\!]$ (first term in the RHS of (4.40)) is dominated by $\lambda^{([\![k]\!])}/\psi(\Delta_k^{([\![k]\!])})$ (second term). This is possible due to the design of the sets $\mathcal{X}^{(m)}$ and Assumption 2. While Lemmas 24, 25 require only $\rho > 2$, we need $\rho > 4$ to ensure that $\sum_{m=1}^{M-1} Q_n^{(m)}$ remains sublinear when we plug-in the probabilities from Lemma 25. $\rho > 2$ is attainable with a more careful design of the sets $\mathcal{X}^{(m)}$. The $\Lambda > \Lambda_0$ condition is needed because initially MF-UCB is playing at lower fidelities and for small Λ, N could be much larger than n_Λ.

Lower Bound - Theorem 17

First we show that for an arm k with $\Delta_k^{(p)} > 0$ and $\Delta_k^{(\ell)} \leq 0$ for all $\ell < p$, any strategy should satisfy

$$R_k(\Lambda) \gtrsim \log(n_\Lambda)\, \Delta_k^{(M)} \left[\min_{\ell \geq p, \Delta_k^{(\ell)} > 0} \frac{\lambda^{(\ell)}}{{\Delta_k^{(\ell)}}^2}\right]$$

where R_k is the regret incurred due to arm k. The proof uses a change of measure argument. The modification has Bernoulli distributions with mean $\tilde{\mu}_k^{(\ell)}, \ell = 1, \ldots, M$ where $\tilde{\mu}_k^{(\ell)} = \mu_k^{(\ell)}$ for all

	$\mathcal{X}^{(1)}$	$\mathcal{X}^{(2)}$	$\mathcal{X}^{(m)}$	$\mathcal{X}^{(M)}$	$\mathcal{X}_\star$
$\mathbb{E}[T_{k,n}^{(1)}]$	$\frac{\log(n)}{\psi(\Delta_k^{(1)})}$	$\frac{\log(n)}{\psi(\gamma^{(1)})}$	$\cdots \quad \frac{\log(n)}{\psi(\gamma^{(1)})} \quad \cdots$	$\frac{\log(n)}{\psi(\gamma^{(1)})}$	$\frac{\log(n)}{\psi(\gamma^{(1)})}$
$\mathbb{E}[T_{k,n}^{(2)}]$		$\frac{\log(n)}{\psi(\Delta_k^{(2)})}$	$\cdots \quad \frac{\log(n)}{\psi(\gamma^{(2)})} \quad \cdots$	$\frac{\log(n)}{\psi(\gamma^{(2)})}$	$\frac{\log(n)}{\psi(\gamma^{(2)})}$
$\vdots$ $\mathbb{E}[T_{k,n}^{(m)}]$ $\vdots$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\cdots \quad \frac{\log(n)}{\psi(\Delta_k^{(m)})} \quad \cdots$	$\frac{\log(n)}{\psi(\gamma^{(m)})}$	$\frac{\log(n)}{\psi(\gamma^{(m)})}$
$\mathbb{E}[T_{k,n}^{(M)}]$			$\mathcal{O}(1)$	$\frac{\log(n)}{\psi(\Delta_k^{(M)})}$	$\Omega(n)$

Table 4.1: Bounds on the expected number of plays for each $k \in \mathcal{X}^{(m)}$ (columns) at each fidelity (rows) after n time steps (i.e. n plays at any fidelity) in MF-UCB.

$\ell < m$. Then we push $\tilde{\mu}_k^{(\ell)}$ slightly above $\mu_\star - \zeta^{(\ell)}$ from $\ell = m$ all the way to M where $\tilde{\mu}_k^{(M)} > \mu_\star$. To control the probabilities after changing to $\tilde{\mu}_k^{(\ell)}$ we use the conditions in Assumption 3. Then for $k \in \mathcal{X}^{(m)}$ we argue that $\lambda^{(\ell)} {\Delta_k^{(\ell)}}^2 \gtrsim \lambda^{(m)} / {\Delta_k^{(m)}}^2$ using, once again the design of the sets $\mathcal{X}^{(m)}$. This yields the separate results for $k \in \mathcal{X}_{\checkmark}^{(m)}, \mathcal{X}_{\times}^{(m)}$.

4.4.2 Proof of Upper Bound

We will repeatedly use the following result in the proof of the upper bound.

Lemma 26. *For all $u > 0$, $\sum_{t=u+1}^{\infty} \mathbb{P}(\mathcal{B}_{k_\star,t} < \mu_\star) \leq \frac{M\nu}{\rho-2} \frac{1}{u^{\rho-2}}$.*

Proof. The proof is straightforward using a union bound.

$$\begin{aligned}
\mathbb{P}(\mathcal{B}_{k_\star,t} < \mu_\star) &= \mathbb{P}(\exists m \in \{1,\dots,M\},\ \exists 1 \leq s \leq t-1,\ \ \mathcal{B}_{k_\star,t}^{(m)}(s) \leq \mu_\star) \qquad (4.27)\\
&= \sum_{m=1}^{M} \sum_{s=1}^{t-1} \mathbb{P}\left(\overline{X}_{k_\star,s}^{(m)} - \mu_{k_\star}^{(m)} < \mu_\star - \mu_{k_\star}^{(m)} - \zeta^{(m)} - \psi^{-1}\left(\frac{\rho \log(t)}{s} \right) \right) \\
&\leq \sum_{m=1}^{M} \sum_{s=1}^{t-1} \nu t^{-\rho} \leq\ M\nu t^{1-\rho}
\end{aligned}$$

In the third step we have used $\mu_\star - \mu_k^{(m)} \leq \zeta^{(m)}$. The result follows by bounding the sum with the integral $\sum_{t=u+1}^{\infty} t^{1-\rho} \leq \int_u^{\infty} t^{1-\rho} = u^{2-\rho}/(\rho-2)$. □

We first prove Lemma 24, stated formally below.

Lemma 27. *Let $m \leq M$ and consider any arm $k \in \mathcal{X}^{(m)}$. After n time steps of (γ,ρ)-MF-UCB with $\rho > 2$ and $\gamma > 0$, we have the following bounds on $\mathbb{E}[T_{k,n}^{(\ell)}]$ for $\ell = 1,\dots,M$.*

$$T_{k,n}^{(\ell)} \leq \frac{\rho \log(n)}{\psi(\gamma^{(m)})} + 1,\ \ \forall\, \ell < m, \qquad \mathbb{E}[T_{k,n}^{(m)}] \leq \frac{\rho \log(n)}{\psi(\Delta_k^{(m)}/2)} + \kappa_\rho, \qquad \mathbb{E}[T_{k,n}^{(>m)}] \leq \kappa_\rho.$$

Here, $\kappa_\rho = 1 + \frac{\nu}{2} + \frac{M\nu}{\rho - 2}$ is a constant.

Proof. As n is fixed in this proof, we will write $\mathbb{E}[\cdot], \mathbb{P}(\cdot)$ for $\mathbb{E}[\cdot|N = n], \mathbb{P}(\cdot|N = n)$. Let $\phi_t^{(m)} = \lfloor \frac{\rho \log(t)}{\psi(\gamma^{(m)})} \rfloor$. By design of the algorithm we won't play any arm more than $\phi_n^{(m)} + 1$ times at any $m < M$. To see this, assume we have already played $\phi_n^{(m)} + 1$ times at any $t < n$. Then,

$$\psi^{-1}\left(\frac{\rho \log(t)}{T_{k,t-1}^{(m)}}\right) < \psi^{-1}\left(\frac{\rho \log(t)}{\rho \log(n)} \psi(\gamma^{(m)})\right) \leq \gamma^{(m)},$$

and we will proceed to the $(m+1)^{\text{th}}$ fidelity in step 2 of Algorithm 3. This gives the first part of the theorem. For any $\ell \geq m$ we can avoid the $\frac{1}{\psi(\gamma^{(m)})}$ dependence to obtain tighter bounds.

For the case $\ell = m$, our analysis follows usual multi-armed bandit analyses [9, 25]. For any $u \leq n$, we can bound $T_{k,n}^{(m)}$ via $T_{k,n}^{(m)} \leq u + \sum_{t=u+1}^{n} Z_{k,t,u}^{(m)}$ where $Z_{k,t,u}^{(m)} = \mathbb{1}\{m_t = m \wedge I_t = k \wedge T_{k,t-1}^{(m)} \geq u\}$. We relax $Z_{k,t,u}^{(m)}$ further via,

$$\begin{aligned}
Z_{k,t,u}^{(m)} &\leq \mathbb{1}\{T_{k,t-1}^{(\ell)} > \phi_t^{(\ell)} \; \forall \ell \leq m-1 \quad \wedge \quad u \leq T_{k,t-1}^{(m)} \leq \phi_t^{(m)} \quad \wedge \quad \mathcal{B}_{k,t} > \mathcal{B}_{k_\star,t}\} \\
&\leq \mathbb{1}\{T_{k,t-1}^{(m)} \geq u \quad \wedge \quad \mathcal{B}_{k,t}^{(m)}(T_{k,t-1}^{(m)}) \geq \mu_\star\} \; + \; \mathbb{1}\{\mathcal{B}_{k_\star,t} < \mu_\star\} \\
&\leq \mathbb{1}\{\exists\, u \leq s \leq t-1 : \; \mathcal{B}_{k,t}^{(m)}(s) > \mu_\star\} \; + \; \mathbb{1}\{\mathcal{B}_{k_\star,t} < \mu_\star\}.
\end{aligned}$$

This yields, $\mathbb{E}[T_{k,n}^{(m)}] \leq u + \sum_{t=u+1}^{n} \sum_{s=u}^{t-1} \mathbb{P}(\mathcal{B}_{k,t}^{(m)}(s) > \mu_\star) + \sum_{t=u+1}^{n} \mathbb{P}(\mathcal{B}_{k_\star,t} < \mu_\star)$. The third term in this summation is bounded by $M\nu/(\rho - 2)$ using Lemma 26. To bound the second, choose $u = \lceil \rho \log(n)/\psi(\Delta_k^{(m)}/2) \rceil$. Then,

$$\begin{aligned}
\mathbb{P}(\mathcal{B}_{k,t}^{(m)}(s) > \mu_\star) &= \mathbb{P}\left(\overline{X}_{k,s}^{(m)} - \mu_k^{(m)} > \mu_\star - \mu_k^{(m)} - \zeta^{(m)} - \psi^{-1}\left(\frac{\rho \log(t)}{s}\right)\right) \\
&\leq \mathbb{P}(\overline{X}_{k,s}^{(m)} - \mu_k^{(m)} > \Delta_k^{(m)}/2) \leq \nu \exp\left(-s\psi\left(\frac{\Delta_k^{(m)}}{2}\right)\right) \leq \nu n^{-\rho} \qquad (4.28)
\end{aligned}$$

In the second and last steps we have used $\psi^{-1}(\rho \log(t)/s) < \psi^{-1}(\rho \log(t)/u) \leq \Delta_k^{(m)}/2$ since ψ^{-1} is increasing and $u > \rho \log(n)/\psi(\Delta_k^{(m)}/2)$. Since there are at most n^2 terms in the summation, the second term is bounded by $\nu n^{2-\rho}/2 \leq \nu/2$. Collecting the terms gives the bound on $\mathbb{E}[T_{k,n}^{(m)}]$.

To bound $T_{k,n}^{(>m)}$ we write $T_{k,n}^{(>m)} \leq u + \sum_{t=u+1}^{n} Z_{k,t,u}^{(>m)}$ where

$$\begin{aligned}
Z_{k,t,u}^{(>m)} &= \mathbb{1}\{m_t > m \quad \wedge \quad I_t = k \quad \wedge \quad T_{k,t-1}^{(>m)} \geq u\} \\
&\leq \mathbb{1}\{T_{k,t-1}^{(\ell)} > \phi_t^{(\ell)} \, \forall \ell \leq m \quad \wedge \quad \mathcal{B}_{k,t} > \mathcal{B}_{k_\star,t} \quad \wedge \quad T_{k,t-1}^{(>m)} \geq u\} \\
&\leq \mathbb{1}\{T_{k,t-1}^{(m)} > \phi_t^{(m)} \quad \wedge \quad \mathcal{B}_{k,t}^{(m)}(T_{k,t-1}^{(m)}) > \mu_\star\} + \mathbb{1}\{\mathcal{B}_{k_\star,t} < \mu_\star\} \\
&\leq \mathbb{1}\{\exists\, \phi_t^{(m)} + 1 \leq s \leq t-1 : \; \mathcal{B}_{k,t}^{(m)}(s) > \mu_\star\} + \mathbb{1}\{\mathcal{B}_{k_\star,t} < \mu_\star\}
\end{aligned}$$

This yields, $\mathbb{E}[T_{k,n}^{(>m)}] \leq u + \sum_{t=u+1}^{n} \sum_{s=\phi_t^{(m)}+1}^{t-1} \mathbb{P}(\mathcal{B}_{k,t}^{(m)}(s) > \mu_\star) \; + \; \sum_{t=u+1}^{n} \mathbb{P}(\mathcal{B}_{k_\star,t} < \mu_\star)$. The inner term inside the double summation can be bounded via,

$$\begin{aligned}
\mathbb{P}(\mathcal{B}_{k,t}^{(m)}(s) > \mu_\star) &= \mathbb{P}\left(\overline{X}_{k,s}^{(m)} - \mu_k^{(m)} > \mu_\star - \mu_k^{(m)} - \zeta^{(m)} - \psi^{-1}\Big(\frac{\rho\log(t)}{s}\Big)\right) \\
&\leq \mathbb{P}(\overline{X}_{k,s}^{(m)} - \mu_k^{(m)} > \Delta_k^{(m)} - \gamma^{(m)}) \leq \nu\exp(-s\psi(\Delta_k^{(m)} - \gamma^{(m)})) \\
&\leq \nu\exp\left(-\frac{\psi(\Delta_k^{(m)} - \gamma^{(m)})}{\psi(\gamma^{(m)})}\rho\log(t)\right) \leq \nu t^{-\rho} \qquad (4.29)
\end{aligned}$$

The second step follows from $s > \phi_t^{(m)} > \rho\log(t)/\psi(\gamma^{(m)})$ and the last step uses $\psi(\Delta_k^{(m)} - \gamma^{(m)}) > \psi(\gamma^{(m)})$ when $\Delta_k^{(m)} > 2\gamma^{(m)}$. To bound the summation, we use $u = 1$ and bound it by an integral: $\sum_{t=u+1}^{n} t^{-\rho+1} \leq 1/(2u^{\rho-2}) \leq 1/2$. Collecting the terms gives the bound on $\mathbb{E}[T_{k,n}^{(>m)}]$. □

We now prove Lemma 25, stated formally below.

Lemma 28. *Consider any arm $k \in \mathcal{X}^{(m)}$. For (γ,ρ)-MF-UCB with $\rho > 2$ and $\gamma > 0$, we have the following concentration results for $\ell = 1,\ldots,M$ for any $x \geq 1$.*

$$\begin{aligned}
\mathbb{P}\left(T_{k,n}^{(m)} > x\left(1 + \frac{\rho\log(n)}{\psi(\Delta_k^{(m)}/2)}\right)\right) &\leq \frac{\nu\tilde{\kappa}_{k,\rho}^{(m)}}{(x\cdot\log(n))^{\rho-1}} + \frac{\nu}{n^{x\rho-1}}. \\
\mathbb{P}\left(T_{k,n}^{(>m)} > x\right) &\leq \frac{M\nu}{\rho-1}\frac{1}{x^{\rho-1}} + \frac{1}{(\rho-2)x^{\rho-2}}
\end{aligned}$$

Here, $\tilde{\kappa}_{k,\rho}^{(m)} = \frac{M}{\rho-1}\left(\frac{\psi(\Delta_k^{(m)}/2)}{\rho}\right)^{\rho-1}$.

Proof. For the first inequality, we modify the analysis in Audibert et al. [9] to the multi-fidelity setting. We begin with the following observation for all $u \in \mathbb{N}$.

$$\begin{aligned}
&\{\forall\, t:\, u+1 \leq t \leq n, \mathcal{B}_{k,t}^{(m)}(u) \leq \mu_\star\} \cap \qquad (4.30) \\
&\bigcap_{m=1}^{M}\{\forall\, 1 \leq s \leq n-u : \mathcal{B}_{k_\star,u+s}^{(m)}(s) > \mu_\star\} \implies T_{k,n}^{(m)} \leq u
\end{aligned}$$

To prove this, consider $s^{(m)}, m = 1,\ldots,M$ such that $s^{(1)} \geq 1$, $s^{(m)} \geq 0, \forall m \neq 1$. For all $u + \sum_{m=1}^{M} s^{(m)} \leq t \leq n$ and for all $\ell = 1,\ldots,M$ we have

$$\mathcal{B}_{k_\star,t}^{(\ell)}(s^{(\ell)}) \geq \mathcal{B}_{k_\star,u+s}^{(\ell)}(s^{(\ell)}) \; > \; \mu_\star \geq \mathcal{B}_{k,t}^{(m)}(u) \geq \mathcal{B}_{k,t}^{(m)}(T_{k,t-1}^{(m)}).$$

This means that arm k will not be the $\mathcal{B}_{k,t}$ maximiser at any time $u < t < n$ and consequently it won't be played more than $u+1$ times at the m^{th} fidelity. Via the union bound we have,

$$\mathbb{P}(T_{k,n}^{(m)} > u) \leq \sum_{t=u+1}^{n}\mathbb{P}(\mathcal{B}_{k,t}^{(m)}(u) > \mu_\star) + \sum_{m=1}^{M}\sum_{s=1}^{n-u}\mathbb{P}(\mathcal{B}_{k_\star,u+s}^{(m)}(s) < \mu_\star).$$

We will use $u = \lceil x(1+\rho\log(n)/\psi(\Delta_k^{(m)}/2))\rceil$. Bounding the inner term of the second double summation closely mimics the calculations in (4.27) via which it can be shown $\mathbb{P}(\mathcal{B}^{(m)}_{k_\star,u+s}(s) < \mu_\star) \leq \nu(u+s)^{-\rho}$. The second term is then bounded by an integral as follows,

$$\sum_{m=1}^{M}\sum_{s=1}^{n-u}\mathbb{P}(\mathcal{B}^{(m)}_{k_\star,u+s}(s) < \mu_\star) \leq M\sum_{s=1}^{n-u}\nu(u+s)^{-\rho} \leq M\nu\int_u^n t^{-\rho} \leq \frac{M\nu u^{1-\rho}}{\rho-1} \leq \frac{\nu\tilde{\kappa}^{(m)}_{k,\rho}}{(x\cdot\log(n))^{\rho-1}}$$

The inner term of the first summation mimics the calculations in (4.28). Noting that $s > x\rho\log(n)/\ \psi(\Delta_k^{(m)}/2)$ it can be shown $\mathbb{P}(\mathcal{B}^{(m)}_{k,t}(u) > \mu_\star) \leq \nu n^{-\rho x}$ which bounds the outer summation by $\nu n^{-\rho x+1}$. This proves the first concentration result.

For the second, we begin with the following observation for all $u \in \mathbb{N}$.

$$\{\forall\, t:\ u+1 \leq t \leq n,\ \ \mathcal{B}^{(m)}_{k,t}(T^{(m)}_{k,t-1}) \leq \mu_\star \quad\vee\quad T^{(m)}_{k,t-1} \leq \phi^{(m)}_t\} \cap \bigcap_{m=1}^{M}\{\forall\, 1 \leq s \leq n-u : \mathcal{B}^{(m)}_{k_\star,u+s}(s) > \mu_\star\} \quad\Longrightarrow\quad T^{(>m)}_{k,n} \leq u \tag{4.31}$$

To prove this first note that when $T^{(m)}_{k,t-1} \leq \phi^{(m)}_t$ we will play at the m^{th} fidelity or lower. Otherwise, consider $s^{(m)}, m = 1,\ldots,M$ such that $s^{(1)} \geq 1$ and $s^{(m)} \geq 0, \forall m$. For all $u+\sum_{m=1}^{M} s^{(m)} \leq t \leq n$ and for all $\ell = 1,\ldots,M$ we have

$$\mathcal{B}^{(\ell)}_{k_\star,t}(s^{(\ell)}) \geq \mathcal{B}^{(\ell)}_{k_\star,u+s}(s^{(\ell)}) \ > \ \mu_\star \geq \mathcal{B}^{(m)}_{k,t}(T^{(m)}_{k,t-1}).$$

This means that arm k will not be played at time t and consequently for any $t > u$. After a further relaxation we get,

$$\mathbb{P}(T^{(>m)}_{k,n} > u) \leq \sum_{t=u+1}^{n}\sum_{s=\phi^{(m)}_t+1}^{t-1}\mathbb{P}(\mathcal{B}^{(m)}_{k,t}(s) > \mu_\star) + \sum_{m=1}^{M}\sum_{s=1}^{n-u}\mathbb{P}(\mathcal{B}^{(m)}_{k_\star,u+s}(s) < \mu_\star)$$

The second summation is bounded via $\frac{M\nu}{(\rho-1)u^{\rho-1}}$. Following an analysis similar to (4.29), the inner term of the first summation can be bounded by $\nu t^{-\rho}$ which bounds the first term by $u^{2-\rho}/(\rho-2)$. The result follows by using $u = x$ in (4.31). □

Proof of Theorem 16

We are now ready to prove the upper bound. We first establish the following Lemma.

Lemma 29 (Regret of MF-UCB)**.** *Let $\rho > 4$. There exists Λ_0 depending on $\lambda^{(1)}, \lambda^{(M)}$ such that for all $\Lambda > \Lambda_0$, (γ,ρ)-MF-UCB satisfies,*

$$\mathbb{E}[R(\Lambda)] \leq \mu_\star\lambda^{(M)} + \sum_{k=1}^{K}\Delta_k^{(M)}\Bigg(\sum_{\ell=1}^{[k]-1}\lambda^{(\ell)}\frac{\rho(\log(n_\Lambda)+c)}{\psi(\gamma^{(\ell)})} +$$

$$\lambda^{(\llbracket k \rrbracket)} \frac{\rho(\log(n_\Lambda) + c)}{\psi(\Delta_k^{(\llbracket k \rrbracket)}/2)} \;+\; \mu_\star \kappa_\rho \lambda^{(M)} \Bigg)$$

Here $c = 1 + \log(2)$ *and* $\kappa_\rho = 1 + \frac{\nu}{\rho - 2} + \frac{M\nu}{\rho - 2}$ *are constants.*

Proof. Denote the set of arms "above" $\mathcal{X}^{(m)}$ by $\widehat{\mathcal{K}}^{(m)} = \bigcup_{\ell=m+1}^{M} \mathcal{X}^{(\ell)}$ and those "below" $\mathcal{X}^{(m)}$ by $\check{\mathcal{K}}^{(m)} = \bigcup_{\ell=1}^{m-1} \mathcal{X}^{(\ell)}$. We first observe,

$$\Bigg(\forall\, m \le M-1,\ \forall k \in \mathcal{X}^{(m)},\ \ T_{k,n}^{(m)} \le x \Bigg(1 + \frac{\rho \log(n)}{\psi(\Delta_k^{(m)}/2)} \Bigg) \quad \wedge \quad T_{k,n}^{(>m)} \le y \Bigg) \tag{4.32}$$

$$\implies \sum_{m=1}^{M-1} Q_n^{(m)} \;\le\; Ky \;+\; \sum_{m=1}^{M-1} \sum_{k \in \mathcal{X}^{(m)}} x \Bigg(1 + \frac{\rho \log(n)}{\psi(\Delta_k^{(m)}/2)} \Bigg) + \sum_{m=1}^{M-1} |\widehat{\mathcal{K}}^{(m)}| \Bigg(1 + \frac{\rho \log(n)}{\psi(\gamma^{(m)})} \Bigg)$$

To prove this we first note that the LHS of (4.32) is reducible to,

$$\forall\, m \le M-1,\ \ Q_n^{(m)} \le \sum_{k \in \check{\mathcal{K}}^{(m)}} T_{k,n}^{(m)} \;+\; \sum_{k \in \mathcal{X}^{(m)}} x \Bigg(1 + \frac{\rho \log(n)}{\psi(\Delta_k^{(m)}/2)} \Bigg) + \sum_{k \in \widehat{\mathcal{K}}^{(m)}} \Bigg(1 + \frac{\rho \log(n)}{\psi(\gamma^{(m)})} \Bigg)$$

The statement follows by summing the above from $m = 1, \ldots, M-1$ and rearranging the $T_{k,n}^{(>m)}$ terms to obtain,

$$\begin{aligned} \sum_{m=1}^{M-1} \sum_{k \in \check{\mathcal{K}}^{(m)}} T_{k,n}^{(m)} &= \sum_{m=1}^{M-1} \sum_{\ell=1}^{m-1} \sum_{k \in \mathcal{X}^{(\ell)}} T_{k,n}^{(m)} = \sum_{m=1}^{M-2} \sum_{k \in \mathcal{X}^{(m)}} \sum_{\ell=m+1}^{M-1} T_{k,n}^{(\ell)} \le \sum_{m=1}^{M-2} \sum_{k \in \mathcal{X}^{(m)}} T_{k,n}^{(>m)} \\ &\le\; (K - |\mathcal{X}^{(M-1)} \cup \mathcal{X}^{(M)} \cup \mathcal{X}_\star|) y \;\le\; Ky. \end{aligned}$$

Now for the given Λ under consideration, define $\delta_\Lambda = \frac{1}{\log(\Lambda/\lambda^{(1)})}$. In addition define,

$$x_{n,\delta} = \max \Bigg(\;\; 1 \;\;,\;\; \frac{1}{\rho} \Bigg(3 + \frac{\log(2\nu\pi^2 K/(3\delta))}{\log(n)} \Bigg) \;\;,\;\; \Bigg(\frac{2\pi^2 K \nu M}{3(\rho-1)\delta} \Bigg)^{\frac{1}{\rho-1}} \frac{\psi(\Delta_k^{(m)}/2)}{\rho} n^{\frac{2}{\rho-1}} \Bigg).$$

$$y_{n,\delta} = \max \Bigg(\;\; 1 \;\;,\;\; \Bigg(\frac{2\pi^2 K M \nu}{3(\rho-1)\delta} \Bigg)^{\frac{1}{\rho-1}} n^{\frac{2}{\rho-1}} \;\;,\;\; \Bigg(\frac{\pi^2 K}{3\delta} \Bigg)^{\frac{1}{\rho-2}} n^{\frac{2}{\rho-2}} \Bigg).$$

Now choose $n_{0,\Lambda}$ to be the smallest n such that the following holds for all $n \ge n_{0,\Lambda}$.

$$\frac{n}{2} \;\ge\; K y_{n,\delta_\Lambda} \;+\; \sum_{m=1}^{M-1} \sum_{k \in \mathcal{X}^{(m)}} x_{n,\delta_\Lambda} \Bigg(1 + \frac{\rho \log(n)}{\psi(\Delta_k^{(m)}/2)} \Bigg) + \sum_{m=1}^{M-1} \sum_{k \in \widehat{\mathcal{K}}^{(m)}} 1 + \frac{\rho \log(n)}{\psi(\gamma)}, \tag{4.33}$$

For such an $n_{0,\Lambda}$ to exist, for a given Λ, we need both x_n, y_n sublinear. This is true since $\rho > 4$. In addition, observe that $n_{0,\Lambda}$ grows only polylogarithmically in Λ since (4.33) reduces to $n^p \gtrsim (\log(\Lambda))^{1/2}$ where $p > 0$ depends on our choice of ρ.

By (4.32), the RHS of (4.33) is an upper bound on the number of plays at fidelities lower than M. Therefore, for all $n \geq n_{0,\Lambda}$,

$$\begin{aligned}
\mathbb{P}\left(Q_n^{(M)} < \frac{n}{2}\right) &\leq \sum_{m=1}^{M-1} \sum_{k \in \mathcal{X}^{(m)}} \mathbb{P}\left(T_{k,n}^{(m)} > x_{n,\delta}\left(1 + \frac{\rho \log(n)}{\psi(\Delta_k^{(m)}/2)}\right)\right) + \mathbb{P}\left(T_{k,n}^{(>m)} > y_{n,\delta}\right) \\
&\leq \sum_{m=1}^{M-1} \sum_{k \in \mathcal{X}^{(m)}} \frac{\nu}{n^{\rho x_{n,\delta_\Lambda} - 1}} + \frac{\nu \tilde{\kappa}_{k,\rho}^{(m)}}{(x_{n,\delta_\Lambda} \log(n))^{\rho-1}} + \frac{\nu M}{(\rho-1) y_{n,\delta_\Lambda}^{\rho-1}} + \frac{1}{2 y_{n,\delta_\Lambda}^{\rho-2}} \\
&\leq K\left(4 \times \frac{3\delta}{2Kn^2\pi^2}\right) \leq \frac{6\delta}{n^2\pi^2}. \qquad (4.34)
\end{aligned}$$

The last step follows from the fact that each of the four terms inside the summation in the second line are $\leq 3\delta/(2Kn^2\pi^2)$. For the last term we have used that $(\rho-2)/2 > 1$ and that $3\delta/(\pi^2 K)$ is smaller than 1. Note that the double summation just enumerates over all arms in $\mathcal{K}$.

We can now specify the conditions on Λ_0. Λ_0 should be large enough so that for all $\Lambda \geq \Lambda_0$, we have $\lfloor \Lambda/\lambda^{(M)} \rfloor \geq n_{0,\Lambda}$. Such an Λ_0 exists since $n_{0,\Lambda}$ grows only polylogarithmically in Λ. This ensures that we have played a sufficient numer of rounds to apply the concentration result in (4.34).

Let the (random) expended capital after n rounds of MF-UCB be $\Omega(n)$. Let $\mathcal{E} = \{\exists n \geq n_{0,\Lambda} : \Omega(n) < n\lambda^{(M)}/2\}$. Since $\Omega(n) \geq \lambda^{(M)} Q_n^{(M)}$, by using the union bound on (4.34) we have $\mathbb{P}(\mathcal{E}) \leq \delta_\Lambda$. Therefore,

$$\mathbb{P}\left(N > \frac{2\Lambda}{\lambda^{(M)}}\right) = \mathbb{P}\left(N > \frac{2\Lambda}{\lambda^{(M)}} \middle| \mathcal{E}\right) \underbrace{\mathbb{P}(\mathcal{E})}_{\leq \delta_\Lambda} + \underbrace{\mathbb{P}\left(N > \frac{2\Lambda}{\lambda^{(M)}} \middle| \mathcal{E}^c\right)}_{=0} \mathbb{P}(\mathcal{E}^c) < \delta_\Lambda$$

The last step uses the following reasoning: Conditioned on $\mathcal{E}^c$, $n > 2\Omega(n)/\lambda^{(M)}$ is false for $n > n_{0,\Lambda}$. In particular, it is true for the *random* number of plays N since $\Lambda > \Lambda_0 \implies N \geq n_{0,\Lambda}$. Now, clearly $\Lambda > \Omega(N)$ and therefore $N > 2\Lambda/\lambda^{(M)}$ is also false.

By noting that $n_\Lambda = \Lambda/\lambda^{(M)}$ and that $\log(\Lambda/\lambda^{(1)})$ is always an upper bound on $\log(N)$, we have,

$$\mathbb{E}[\log(N)] \leq \log(2n_\Lambda)\mathbb{P}(N < 2n_\Lambda) + \log\left(\frac{\Lambda}{\lambda^{(1)}}\right)\mathbb{P}(N > 2n_\Lambda) \leq \log(n_\Lambda) + 1 + \log(2) \qquad (4.35)$$

Lemma 29 now follows by an application of Lemma 24. First we condition on $N = n$ to obtain,

$$\begin{aligned}
\mathbb{E}[R(\Lambda)|N=n] &\leq \mu_\star \lambda^{(M)} + \sum_{k=1}^{K} \sum_{m=1}^{M} \Delta_k^{(M)} \lambda^{(m)} T_{k,n}^{(m)} \\
&\leq \mu_\star \lambda^{(M)} + \sum_{k=1}^{K} \Delta_k^{(M)} \left(\sum_{\ell=1}^{[k]-1} \lambda^{(\ell)} \frac{\rho \log(n)}{\psi(\gamma^{(m)})} + \lambda^{([k])} \frac{\rho \log(n)}{\psi(\Delta_k^{([k])}/2)} + \kappa_\rho \lambda^{(M)} \right)
\end{aligned}$$

The theorem follows by plugging in the above in $\mathbb{E}[R(\Lambda)] = \mathbb{E}[\mathbb{E}[R(\Lambda)|N]]$ and using the bound for $\mathbb{E}[\log(N)]$ in (4.35). □

We can now bound the regret for MF-UCB.

Proof of Theorem 16. Recall that $\psi(\gamma^{(m)}) = \frac{\lambda^{(m)}}{\lambda^{(m+1)}}\psi(\zeta^{(m)})$. Plugging this into Lemma 29 we get

$$\begin{aligned}\mathbb{E}[R(\Lambda)] \leq\ & \mu_\star\lambda^{(M)} + \sum_{k=1}^{K}\Delta_k^{(M)}\left(\sum_{\ell=1}^{[k]-1}\lambda^{(\ell+1)}\frac{\rho(\log(n_\Lambda)+c)}{\psi(\zeta^{(\ell)})}\right. \\ &\left. + \lambda^{(\llbracket k\rrbracket)}\frac{\rho(\log(n_\Lambda)+c)}{\psi(\Delta_k^{(\llbracket k\rrbracket)}/2)} + \kappa_\rho\lambda^{(M)}\right) \\ \leq\ & \mu_\star\lambda^{(M)} + \sum_{k=1}^{K}\Delta_k^{(M)}\cdot\lambda^{(\llbracket k\rrbracket)}\rho(\log(n_\Lambda)+c)\left(\frac{2}{\psi(\zeta^{(\llbracket k\rrbracket-1)})}+\frac{1}{\psi(\Delta_k^{(\llbracket k\rrbracket)}/2)}\right) \\ &+ \Delta_k^{(M)}\kappa_\rho\lambda^{(M)}\end{aligned}$$

The second step uses Assumption 2. The theorem follows by noting that for any $k \in \mathcal{X}^{(m)}$ and $\ell < m$, $\Delta_k^{(m)} = \Delta_k^{(\ell)} + \zeta^{(\ell)} - \zeta^{(m)} + \mu_k^{(\ell)} - \mu_k^{(M)} + \mu_k^{(M)} - \mu_k^{(m)} \leq 2\gamma^{(\ell)} + 2\zeta^{(\ell)} \leq 4\zeta^{(\ell)}$. Therefore $1/\psi(\Delta_k^{(m)}) > c_1/\psi(\zeta^{(\ell)})$ where c_1 depends on ψ (for sub-Gaussian distributions, $c_1 = 1/16$). □

4.4.3 Proof of Lower Bound

The regret R_k incurred by any multi-fidelity strategy after capital Λ due to a suboptimal arm k is,

$$R_k(\Lambda) = \Delta_k^{(M)}\sum_{m=1}^{M}\lambda^{(m)}T_{k,N}^{(m)},$$

here N is the total number of plays. We then have, $R(\Lambda) = \sum_k R_k(\Lambda)$. For what follows, for an arm k and any fidelity m denote $\mathrm{KL}_k^{(m)} = \mathrm{KL}(\mu_k^{(m)}\|\mu_\star - \zeta^{(m)})$. The following lemma provides an asymptotic lower bound on R_k.

Lemma 30. *Consider any set of Bernoulli reward distributions with $\mu_\star \in (1/2, 1)$ and $\zeta^{(1)} < 1/2$. For any k with $\Delta_k^{(\ell)} < 0$ for all $\ell < p$ and $\Delta_k^{(p)} > 0$, there exists a problem dependent constant c_p such that any strategy satisfying Assumption 3 must satisfy,*

$$\liminf_{\Lambda\to\infty}\frac{R_k(\Lambda)}{\log(n_\Lambda)} \geq c_p'\,\Delta_k^{(M)}\min_{\ell\geq p,\Delta_k^{(\ell)}>0}\frac{\lambda^{(\ell)}}{{\Delta_k^{(\ell)}}^2}$$

Proof. For now we will fix $N = n$ and consider any game after n rounds. Our proof we will modify the reward distributions of the given arm k for all $\ell \geq p$ and show that any algorithm satisfying Assumption 3 will not be able to distinguish between both problems with high probability. Since the KL divergence is continuous, for any $\epsilon > 0$ we can choose $\tilde{\mu}_k^{(p)} \in (\mu_\star - \zeta^{(p)}, \mu_\star - \zeta^{(p)} + \min_{\ell<p} -\Delta_k^{(\ell)})$ such that $\mathrm{KL}(\mu_k^{(p)} \| \tilde{\mu}_k^{(p)}) < (1+\epsilon)\mathrm{KL}(\mu_k^{(p)} \| \mu_\star - \zeta^{(p)}) = (1+\epsilon)\mathrm{KL}_k^{(p)}$.

The modified construction for arm k, will also have Bernoulli distributions with means $\tilde{\mu}_k^{(1)}$, $\tilde{\mu}_k^{(2)}, \ldots, \tilde{\mu}_k^{(M)}$. $\tilde{\mu}_k^{(p)}$ will be picked to satisfy the two constraints above and for the remaining fidelities,

$$\tilde{\mu}_k^{(\ell)} = \mu_k^{(\ell)} \quad \text{for } \ell < p, \qquad \tilde{\mu}_k^{(\ell)} = \tilde{\mu}_k^{(p)} + \zeta^{(p)} - \zeta^{(\ell)} \quad \text{for } \ell > p.$$

Now note that for $\ell < p$, $\tilde{\mu}_k^{(M)} - \tilde{\mu}_k^{(\ell)} = \tilde{\mu}_k^{(p)} + \zeta^{(p)} - \mu_k^{(\ell)} < \mu_\star - \zeta^{(p)} - \Delta_k^{(\ell)} + \zeta^{(p)} - \mu_k^{(\ell)} = \zeta^{(\ell)}$; similarly, $\tilde{\mu}_k^{(M)} - \tilde{\mu}_k^{(\ell)} = \tilde{\mu}_k^{(p)} + \zeta^{(p)} - \mu_k^{(\ell)} > \mu_\star - \mu_k^{(\ell)} > \mu_k^{(M)} - \mu_k^{(\ell)} > -\zeta^{(\ell)}$. For $\ell > p$, $\tilde{\mu}_k^{(M)} - \tilde{\mu}_k^{(\ell)} = \zeta^{(\ell)}$. Hence, the modified construction satisfies the conditions on the lower fidelities laid out in Chapter 4.1.1 and we can use Assumption 3. Further $\tilde{\mu}_k^{(M)} > \mu_\star$, so k is the optimal arm in the modified problem. Now we use a change of measure argument.

Following Bubeck and Cesa-Bianchi [25], Lai and Robbins [151], denote the expectations, probabilities and distribution in the original problem as $\mathbb{E}, \mathbb{P}, P$ and in the modified problem as $\tilde{\mathbb{E}}, \tilde{\mathbb{P}}, \tilde{P}$. Denote a sequence of observations when playing arm k at by $\{Z_{k,t}^{(\ell)}\}_{t\geq 0}$ and define,

$$L_k^{(\ell)}(s) = \sum_{t=1}^{s} \log\left(\frac{\mu_k^{(\ell)} Z_{k,t}^{(\ell)} + (1-\mu_k^{(\ell)})(1-Z_{k,t}^{(\ell)})}{\tilde{\mu}_k^{(\ell)} Z_{k,t}^{(\ell)} + (1-\tilde{\mu}_k^{(\ell)})(1-Z_{k,t}^{(\ell)})}\right) = \sum_{t:Z_{k,t}^{(\ell)}=1} \log\frac{\mu_k^{(\ell)}}{\tilde{\mu}_k^{(\ell)}} + \sum_{t:Z_{k,t}^{(\ell)}=0} \log\frac{1-\mu_k^{(\ell)}}{1-\tilde{\mu}_k^{(\ell)}}.$$

Observe that $\mathbb{E}[s^{-1}L_k^{(\ell)}(s)] = \mathrm{KL}(\mu_k^{(\ell)} \| \tilde{\mu}_k^{(\ell)})$. Let A be any event in the σ-field generated by the observations in the game.

$$\begin{aligned}\tilde{\mathbb{P}}(A) &= \int \mathbb{1}(A)\mathrm{d}\tilde{P} = \int \mathbb{1}(A) \prod_{\ell=p}^{M} \left(\prod_{i=1}^{T_{k,n}^{(\ell)}} \frac{\tilde{\theta}_k^{(\ell)}(Z_{k,i}^{(\ell)})}{\theta_k^{(\ell)}(Z_{k,i}^{(\ell)})}\right)\mathrm{d}P \\ &= \mathbb{E}\left[\mathbb{1}(A)\exp\left(-\sum_{\ell\geq p} L_k^{(\ell)}(T_{k,n}^{(\ell)})\right)\right] \end{aligned} \tag{4.36}$$

Now let $f_n^{(\ell)} = C\log(n)$ for all ℓ such that $\Delta_k^{(\ell)} < 0$ and $f_n^{(\ell)} = \frac{1}{M-p}\frac{1-\epsilon}{\mathrm{KL}(\mu_k^{(p)} \| \tilde{\mu}_k^{(p)})}\log(n)$ otherwise. (Recall that $\Delta_k^{(\ell)} < 0$ for all $\ell < p$ and $\Delta_k^{(p)} > 0$). C is a large enough constant that we will specificy shortly. Define the following event A_n.

$$A_n = \left\{T_{k,n}^{(\ell)} \leq f_n^{(\ell)},\ \forall \ell \quad \wedge \quad L_k^{(\ell)}(T_{k,n}^{(\ell)}) \leq \frac{1}{M-p}(1-\epsilon/2)\log(n),\ \forall \ell : \Delta_k^{(\ell)} > 0\right\}$$

By (4.36) we have $\tilde{\mathbb{P}}(A_n) \geq \mathbb{P}(A_n)n^{-(1-\epsilon/2)}$. Since k is the unique optimal arm in the modified

construction, by Assumptions 3 we have $\forall\, a > 0$,

$$\tilde{\mathbb{P}}(A_n) \leq \mathbb{P}\Big(\sum_m T_{k,n}^{(m)} < \Theta(\log(n))\Big) \leq \frac{\mathbb{E}\big[n - \sum_m T_{k,n}^{(m)}\big]}{n - \Theta(\log(n))} \in o(n^{a-1})$$

By choosing $a < \epsilon/2$ we have $\mathbb{P}(A_n) \to 0$ as $n \to \infty$. Next, we upper bound the probability of A_n in the original problem as follows,

$$\mathbb{P}(A_n) \geq \mathbb{P}\Bigg(\underbrace{T_{k,n}^{(\ell)} \leq f_n^{(\ell)},\ \forall \ell}_{A_{n,1}} \quad \wedge \quad \underbrace{\max_{s \leq f_n} L_k^{(\ell)}(s) \leq \frac{1}{M-p}(1-\epsilon/2)\log n, \forall \ell : \Delta_k^{(\ell)} > 0}_{A_{n,2}}\Bigg)$$

We will now show that $A_{n,2}$ remains large as $n \to 0$. Writing $A_{n,2} = \bigcap_{\ell:\Delta_k^{(\ell)}>0} A_{n,2,\ell}$, we have

$$\mathbb{P}(A_{n,2,\ell}) = \mathbb{P}\left(\frac{f_n^{(\ell)}(M-p)}{(1-\epsilon)\log(n)} \cdot \frac{1}{f_n^{(\ell)}} \max_{s \leq f_n^{(\ell)}} L_k^{(m)}(s) \leq \frac{1-\epsilon/2}{1-\epsilon}\right).$$

As $f_n^{(\ell)} \to \infty$, by the strong law of large numbers $\frac{1}{f_n^{(\ell)}} \max_{s \leq f_n^{(\ell)}} L_k^{(m)}(s) \to \mathrm{KL}(\mu_k^{(m)} \| \tilde{\mu}_k^{(m)})$. After substituting for $f_n^{(\ell)}$ and repeating for all ℓ, we get $\lim_{n\to\infty} \mathbb{P}(A_{n,2}) = 1$. Therefore, $\mathbb{P}(A_{n,1}) \leq o(1)$. To conclude the proof, we upper bound $\mathbb{E}[R_k(\Lambda)]$ as follows,

$$\begin{aligned}
\frac{\mathbb{E}[R_k(\Lambda)]}{\Delta_k^{(M)}} &\geq \mathbb{P}(\exists\, \ell \,\text{s.t.} T_{k,N}^{(\ell)} > f_N^{(\ell)}) \cdot \mathbb{E}[R_k(\Lambda) \,|\, \exists\, \ell \,\text{s.t.} T_{k,N}^{(\ell)} > f_N^{(\ell)}] \geq \mathbb{P}(\overline{A_{n,1}}) \cdot \min_\ell f_{n_\Lambda}^{(\ell)} \lambda^{(\ell)} \\
&\geq (1-o(1)) \min_{\ell \geq p} \frac{(1-\epsilon)\log(n_\Lambda)\lambda^{(\ell)}}{(M-p)\mathrm{KL}(\mu_k^{(\ell)} \| \tilde{\mu}_k^{(\ell)})} \geq \frac{\log(n_\Lambda)}{M-p}(1-o(1))\frac{1-\epsilon}{1+\epsilon} \min_{\ell > m} \frac{\lambda^{(\ell)}}{\mathrm{KL}_k^{(\ell)}}
\end{aligned}$$

Above, the second step uses the fact that $N \geq n_\Lambda$ and $\log$ is increasing. In the third step, we have chosen $C > \max_{\ell \geq p} \lambda^{(\ell)} \Delta_k^{(M)} / \mathrm{KL}(\mu_k^{(\ell)} \| \tilde{\mu}_k^{(\ell)})$ for $\ell < p$ large enough so that the minimiser will be at $\ell \geq p$. The lemma follows by noting that the statements holds for all $\epsilon > 0$ and that for Bernoulli distributions with parameters μ_1, μ_2, $\mathrm{KL}(\mu_1 \| \mu_2) \leq (\mu_1 - \mu_2)^2/(\mu_2(1-\mu_2))$. The constant given in the theorem is $c'_p = \frac{1}{M-p} \min_{\ell > p}(\mu_\star - \zeta^{(\ell)})(1 - \mu_\star + \zeta^{(\ell)})$. □

We can now use the above Lemma to prove Theorem 17.

Proof of Theorem 17. Let $k \in \mathcal{X}_\checkmark^{(m)}$. We will use Lemma 30 with $p = m$. It is sufficient to show that $\lambda^{(\ell)}/{\Delta_k^{(\ell)}}^2 \gtrsim \lambda^{(m)}/{\Delta_k^{(m)}}^2$ for all $\ell > m$. First note that

$$\begin{aligned}
\Delta_k^{(\ell)} &= \mu_\star - \mu_k^{(m)} - \zeta^{(m)} + \mu_k^{(m)} - \mu_\star + \mu_\star - \mu_k^{(\ell)} + \zeta^{(m)} - \zeta^{(\ell)} \\
&\leq \Delta_k^{(m)} + 2\zeta^{(m)} \leq 2\Delta_k^{(m)} \sqrt{\frac{\lambda^{(m+1)}}{\lambda^{(m)}}}
\end{aligned}$$

Here the last step uses that $\Delta_k^{(m)} > 2\gamma^{(m)} = \sqrt{\lambda^{(m)}/\lambda^{(m+1)}}\zeta^{(m)}$. Here we have used $\psi(\epsilon) = 2\epsilon^2$ which is just Hoeffding's inequality. Therefore, $\frac{\lambda^{(m)}}{{\Delta_k^{(m)}}^2} \leq 4\frac{\lambda^{(m+1)}}{{\Delta_k^{(\ell)}}^2} \leq 4\frac{\lambda^{(\ell)}}{{\Delta_k^{(\ell)}}^2}$.

When $k \in \mathcal{X}_{\boldsymbol{\chi}}^{(m)}$, we use Lemma 30 with $p = \ell_0 = \min\{\ell; \Delta_k^{(\ell)} > 0\}$. However, by repeating the same argument as above, we can eliminate all $\ell > m$. Hence, we only need to consider ℓ such that $\ell_0 \leq \ell \leq m$ and $\Delta_k^{(\ell)} > 0$ in the minimisation of Lemma 30. This is precisely the set $\mathcal{L}_m(k)$ given in the theorem. The theorem follows by repeating the above argument for all arms $k \in \mathcal{K}$. The constant c_p in Theorem 17 is $c_p'/4$ where c_p' is from Lemma 30. □

4.5 Proofs of Theoretical Results in Chapter 4.2

In this section, we present the proofs of our main theorems in Chapter 4.2. While it is self contained, the reader will benefit from first reading the more intuitive discussion in Chapter 4.2.3. Our goal in this section is to bound the simple regret $S(\Lambda)$ given in (4.9).

4.5.1 Set Up & Notation

Recall that N is the random number of plays within capital Λ. While $N \leq \lfloor \Lambda/\lambda^{(1)} \rfloor$ is a trivial upper bound for N, this will be too loose for our purposes. In fact, we will show that after a sufficiently large number of queries at any fidelity, the number of queries at fidelities smaller than M will be sublinear in N. Hence $N \in \mathcal{O}(n_\Lambda)$ where $n_\Lambda = \lfloor \Lambda/\lambda^{(M)} \rfloor$ is the number of plays by any algorithm that operates only at the highest fidelity.

We introduce some notation to keep track of the evaluations at each fidelity in MF-GP-UCB. After n steps, we will have queried multiple times at any of the M fidelities. $T_n^{(m)}(x)$ denotes the number of queries at $x \in \mathcal{X}$ at fidelity m after n steps. $T_n^{(m)}(A)$ denotes the same for a subset $A \subset \mathcal{X}$. $\mathcal{D}_n^{(m)} = \{(x_t, y_t)\}_{t:m_t=m}$ is the set of query-value pairs at the m^{th} fidelity until time n.

Roadmap: To bound $S(\Lambda)$ in both the discrete and continuous settings, we will begin by studying the algorithm after n evaluations at any fidelity and analyse the following quantity,

$$\tilde{R}_n = \sum_{\substack{t:m_t=M \\ x_t \in \mathcal{Z}}} \left(f(x_\star) - f^{(M)}(x_t)\right) \tag{4.37}$$

Readers familiar with the bandit literature will see that this is similar to the notion of *cumulative regret*, except we only consider queries at the M^{th} fidelity and inside a set $\mathcal{Z} \subset \mathcal{X}$. $\mathcal{Z}$ contains the optimum and generally has high value for the payoff function $f^{(M)}(x)$; it will be determined by the approximations provided via the lower fidelity evaluations. We will show that most of the M^{th} fidelity evaluations will be inside $\mathcal{Z}$ in the multi-fidelity setting, and hence, the regret for MF-GP-UCB will scale with $\Psi_n(\mathcal{Z})$ instead of $\Psi_n(\mathcal{X})$ as is the case for GP-UCB. Finally, to convert this bound in terms of n to one that depends on Λ, we show that both the total number

of evaluations N and the number of highest fidelity evaluations $T_N^{(M)}(\mathcal{X})$ are on the order of n_Λ when Λ is sufficiently large. For this, we bound the number of plays at the lower fidelities (see Lemma 19). Then $S(\Lambda)$ can be bounded by,

$$S(\Lambda) \;\leq\; \frac{1}{T_N^{(M)}(\mathcal{X})}\tilde{R}_N \;\lesssim\; \frac{1}{n_\Lambda}\tilde{R}_{n_\Lambda}. \tag{4.38}$$

Before we proceed, we will prove a series of results that will be necessary in our proofs of Theorems 21 and 22. We first prove Lemma 18.

Proof of Lemma 18. Let $\mathbf{A2}' = \left\{\|f^{(M)}\|_\infty \leq \zeta^{(M-1)}/2 \;\cap\; \bigcap_{m=1}^{M-1}\|f^{(m)}\|_\infty \leq \zeta^{(m)}/2\right\}$. It is straightforward to see that $\mathbf{A2}' \subset \mathbf{A2}$ since for any $m \leq M-1$,

$$\|f^{(M)} - f^{(m)}\|_\infty \leq \|f^{(M)}\|_\infty + \|f^{(m)}\|_\infty \leq \zeta^{(M-1)}/2 + \zeta^{(m)}/2 \leq \zeta^{(m)}.$$

Hence, $\mathbb{P}_{\mathcal{GP}}(\mathbf{A2}) \geq \mathbb{P}_{\mathcal{GP}}(\mathbf{A2}')$. We can now bound,

$$\mathbb{P}_{\mathcal{GP}}(\mathbf{A2}') \;=\; \mathbb{P}_{\mathcal{GP}}\big(\|f^{(M)}\|_\infty \leq \zeta^{(M-1)}/2\big) \cdot \prod_{m=1}^{M-1}\mathbb{P}_{\mathcal{GP}}\big(\|f^{(m)}\|_\infty \leq \zeta^{(m)}/2\big) \;\geq\; \xi_{\mathbf{A2}}.$$

Here the equality in the first step comes from the observation that the $f^{(m)}$'s are independent under the $\mathbb{P}_{\mathcal{GP}}$ probability. The last inequality comes from Lemm 7. ■

Remark 2. It is worth noting that the above bound is a fairly conservative lower bound on $\xi_{\mathbf{A2}}$ since $\mathbf{A2}'$ essentially requires that all samples $f^{(m)}$ be small so as to make the differences $f^{(M)} - f^{(m)}$ small. We can obtain a more refined bound on $\xi_{\mathbf{A2}}$ by noting that $f^{(M)} - f^{(m)} \sim \mathcal{GP}(\mathbf{0}, 2\kappa)$ and following proofs for bounding the supremum of a GP (e.g. Theorem 5.4 in Adler [4], Theorem 4 in Ghosal and Roy [72]). This leads to smaller values for β_t in Theorems 21 and 22 and consequently better constants in our bounds. However, this analysis will require accounting for correlations when analysing multiple GPs which is beyond the scope and tangential to the goals of this paper. Moreover, from a practical perspective it would not result in anything actionable since many quantities in the expression for β_t are already unknown in practice, even for GP-UCB. It is also worth noting that the dependence of $\xi_{\mathbf{A2}}$ on our regret bounds is mild since it appears as a $\sqrt{\log(1/\xi_{\mathbf{A2}})}$ term.

Next, Lemma 31 provides a way to bound the probability of an event under our prior (**A1** and **A2**) using the probability of the event when the functions are sampled from a GP (**A1** only).

Lemma 31. *Let E be a $\mathbb{P}_{\mathcal{GP}}$-measurable event. Then, $\mathbb{P}(E) \leq \xi_{\mathbf{A2}}^{-1}\mathbb{P}_{\mathcal{GP}}(E)$.*

Proof This follows via a straightforward application of Bayes' rule, shown below. The last step uses Lemma 18 and that the intersection of two sets is at most as large as either set.

$$\mathbb{P}(E) = \mathbb{P}_{\mathcal{GP}}(E|\mathbf{A2}) = \frac{\mathbb{P}_{\mathcal{GP}}(E \cap \mathbf{A2})}{\mathbb{P}_{\mathcal{GP}}(\mathbf{A2})} \leq \frac{1}{\xi_{\mathbf{A2}}}\mathbb{P}_{\mathcal{GP}}(E).$$

■

For our analysis, we will also need to control the sum of conditional standard deviations for queries in a subset $A \subset \mathcal{X}$. We provide the lemma below, whose proof is based of a similar result in Srinivas et al. [235].

Lemma 32. *Let $f \sim \mathcal{GP}(0, \kappa)$, $f : \mathcal{X} \to \mathbb{R}$ and each time we query at any $x \in \mathcal{X}$ we observe $y = f(x) + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \eta^2)$. Let $A \subset \mathcal{X}$. Assume that we have queried f at n points, $(x_t)_{t=1}^n$ of which s points are in A. Let σ_{t-1} denote the posterior variance at time t, i.e. after $t-1$ queries. Then, $\sum_{x_t \in A} \sigma^2_{t-1}(x_t) \leq \frac{2}{\log(1+\eta^{-2})} \Psi_s(A)$.*

Proof Let $A_s = \{z_1, z_2, \ldots, z_s\}$ be the queries inside A in the order they were queried. Now, assuming that we have only queried inside A at A_s, denote by $\tilde{\sigma}_{t-1}(\cdot)$, the posterior standard deviation after $t-1$ such queries. Then,

$$\begin{aligned}\sum_{t:x_t \in A} \sigma^2_{t-1}(x_t) \;&\leq\; \sum_{t=1}^{s} \tilde{\sigma}^2_{t-1}(z_t) \;\leq\; \sum_{t=1}^{s} \eta^2 \frac{\tilde{\sigma}^2_{t-1}(z_t)}{\eta^2} \;\leq\; \sum_{t=1}^{s} \frac{\log(1+\eta^{-2}\tilde{\sigma}^2_{t-1}(z_t))}{\log(1+\eta^{-2})} \\ &\leq\; \frac{2}{\log(1+\eta^{-2})} I(y_{A_s}; f_{A_s})\end{aligned}$$

Queries outside A will only decrease the variance of the GP so we can upper bound the first sum by the posterior variances of the GP with only the queries in A. The third step uses the inequality $u^2/v^2 \leq \log(1+u^2)/\log(1+v^2)$ with $u = \tilde{\sigma}_{t-1}(z_t)/\eta$ and $v = 1/\eta$ and the last step uses Lemma 5 in Chapter 2.4. The result follows from the fact that $\Psi_s(A)$ maximises the mutual information among all subsets of size s. ■

4.5.2 Discrete $\mathcal{X}$

Proof of Theorem 21. Without loss of generality, we can assume that MF-GP-UCB is run indefinitely with different (x_t, m_t, y_t) values at each time step t, but will analyse the queries until capital Λ is spent. Let N denote the (random) number of queries within Λ, i.e. the quantity satisfying $N = \max\{n \geq 1; \sum_{t=1}^n \lambda^{(m_t)} \leq \Lambda\}$. Note that $\mathbf{supp}\,(N) \subset \{n \in \mathbb{N} : n_\Lambda \leq n \leq \overline{n}_\Lambda\}$. In our analysis, we will first analyse MF-GP-UCB after n steps and control the regret and the number of lower fidelity evaluations.

Bounding the regret after n evaluations: We will need the following lemma to establish that $\varphi_t(x)$ upper bounds $f^{(M)}(x)$. The proof is given in Chapter 4.5.2.

Lemma 33. *Pick $\delta \in (0,1)$ and choose $\beta_t \geq 2\log\left(\frac{M|\mathcal{X}|\pi^2 t^2}{3\xi_{\mathbf{A2}}\delta}\right)$. Then, with probability at least $1-\delta/2$, for all $t \geq 1$, for all $x \in \mathcal{X}$ and for all $m \in \{1, \ldots, M\}$, we have*

$$\left|f^{(m)}(x) - \mu^{(m)}_{t-1}(x)\right| \;\leq\; \beta_t^{1/2} \sigma^{(m)}_{t-1}(x).$$

First note the following bound on the instantaneous regret when $m_t = M$,

$$\begin{aligned} f(x_\star) - f^{(M)}(x_t) &\leq \varphi_t(x_\star) - (\mu_{t-1}^{(M)}(x_t) - \beta_t^{1/2}\sigma_{t-1}^{(M)}(x_t)) \\ &\leq \varphi_t(x_t) - (\mu_{t-1}^{(M)}(x_t) - \beta_t^{1/2}\sigma_{t-1}^{(M)}(x_t)) \;\leq\; 2\beta_t^{1/2}\sigma_{t-1}^{(M)}(x_t). \end{aligned} \tag{4.39}$$

The first step uses that $\varphi_t^{(m)}(x)$ is an upper bound for $f^{(M)}(x)$ by Lemma 33 and the assumption **A2**, and hence so is the minimum $\varphi_t(x)$. The second step uses that x_t was the maximiser of $\varphi_t(x)$ and the third step that $\varphi_t^{(M)}(x) \geq \varphi_t(x)$. To control $\tilde{R}_n$, we will use $\mathcal{Z} = \mathcal{H}^{(M)}$ in (4.37) and invoke Lemma 44. Applying the Cauchy Schwarz inequality yields,

$$\begin{aligned} \tilde{R}_n^2 &\leq T_n^{(m)}(\mathcal{H}^{(M)}) \sum_{\substack{t:m_t=M \\ x_t\in\mathcal{H}^{(M)}}} \left(f(x_\star) - f^{(M)}(x_t)\right)^2 \leq T_n^{(m)}(\mathcal{H}^{(M)}) \sum_{\substack{t:m_t=M \\ x_t\in\mathcal{H}^{(M)}}} 4\beta_t(\sigma_{t-1}^{(m)}(x_t))^2 \\ &\leq C_1 T_n^{(m)}(\mathcal{H}^{(M)})\beta_n \Psi_{T_n^{(m)}(\mathcal{H}^{(M)})}(\mathcal{H}^{(M)}). \end{aligned} \tag{4.40}$$

Here $C_1 = 8/\log(1+\eta^{-2})$.

Bounding the number of evaluations: Lemma 34, given below, bounds the number of evaluations at different fidelities in different regions of $\mathcal{X}$. This will allow us to bound, among other things, the total number of plays N and the number of M^{th} fidelity evaluations outside $\mathcal{Z}$. The proof of Lemma 34 is given in Chapter 4.5.2. Recall that $T_n^{(m)}(x)$ denotes the number of queries at point $x \in \mathcal{X}$ at fidelity m. Similarly, we will denote $T_n^{(>m)}(x)$ to denote the number of queries at point x at fidelities larger than m.

Lemma 34. *Pick $\delta \in (0,1)$ and set $\beta_t = 2\log\left(\frac{M|\mathcal{X}|\pi^2 t^2}{3\xi_{\mathbf{A2}}\delta}\right)$. Further assume $\varphi_t(x_\star) \geq f(x_\star)$. Consider any $x \in \mathcal{H}^{(m)}\backslash\{x_\star\}$ for $m < M$. We then have the following bounds on the number of queries at any given time step n,*

$$\begin{aligned} T_n^{(\ell)}(x) &\leq \frac{\eta^2}{{\gamma^{(m)}}^2}\beta_n + 1, \quad \textit{for } \ell < m, \\ \mathbb{P}\left(T_n^{(m)}(x) > \left\lceil 5\Big(\frac{\eta}{\Delta^{(m)}(x)}\Big)^2\beta_n\right\rceil\right) &\leq \frac{3\delta}{2\pi^2}\frac{1}{|\mathcal{X}|n^2}, \\ \mathbb{P}\left(T_n^{(>m)}(x) > u\right) &\leq \frac{3\delta}{2M\pi^2}\frac{1}{|\mathcal{X}|u}. \end{aligned}$$

First whenever $\varphi_t(x_\star) \geq f(x_\star)$, by using the union bound on the second result of Lemma 34,

$$\mathbb{P}\left(\exists\, n \geq 1,\ \exists\, m \in \{1,\ldots,M\},\ \exists\, x \in \mathcal{H}^{(m)}\backslash\{x_\star\},\ T_n^{(m)}(x) > \left\lceil 5\Big(\frac{\eta}{\Delta^{(m)}(x)}\Big)^2\beta_n\right\rceil\right) \leq \frac{\delta}{4}.$$

Here we have used $\sum n^{-2} = \pi^2/6$. The last two quantifiers just enumerates over all $x \in \mathcal{X}\backslash\{x_\star\}$. Similarly, applying the union bound for $u = 1$ on the third result, we have, for any given n,

$$\mathbb{P}\left(\exists\, m \in \{1,\ldots,M\},\ \exists\, x \in \mathcal{H}^{(m)},\ T_n^{(>m)}(x) > 1\right) \leq \frac{3\delta}{2\pi^2} < \frac{\delta}{4}.$$

	$\mathcal{H}^{(1)}$	$\mathcal{H}^{(2)}$	$\mathcal{H}^{(m)}$	$\mathcal{H}^{(M)}\setminus\{x_\star\}$
$T_n^{(1)}(x)$	$\frac{5\eta^2}{{\Delta^{(1)}(x)}^2}\beta_n + 1$	$\frac{\eta^2}{{\gamma^{(1)}}^2}\beta_n + 1$	$\dots$ $\frac{\eta^2}{{\gamma^{(1)}}^2}\beta_n + 1$ $\dots$	$\frac{\eta^2}{{\gamma^{(1)}}^2}\beta_n + 1$
$T_n^{(2)}(x)$		$\frac{5\eta^2}{{\Delta^{(2)}(x)}^2}\beta_n + 1$	$\dots$ $\frac{\eta^2}{{\gamma^{(2)}}^2}\beta_n + 1$ $\dots$	$\frac{\eta^2}{{\gamma^{(2)}}^2}\beta_n + 1$
$\vdots$ $T_n^{(m)}(x)$ $\vdots$	1	1	$\vdots$ $\dots$ $\frac{5\eta^2}{{\Delta^{(m)}(x)}^2}\beta_n + 1$ $\dots$ $\vdots$	$\vdots$ $\frac{\eta^2}{{\gamma^{(m)}}^2}\beta_n + 1$ $\vdots$
$T_n^{(M)}(x)$			1	$\frac{5\eta^2}{{\Delta^{(M)}(x)}^2}\beta_n + 1$

Table 4.2: Bounds on the number of queries for each $x \in \mathcal{H}^{(m)}$ (columns) at each fidelity (rows). The bound for $T_n^{(M)}(x)$ in $\mathcal{H}^{(M)}$ holds for all arms except the optimal arm $x_\star$ (note $\Delta^{(M)}(x_\star) = 0$).

We will apply the above result for $n = \lfloor \Lambda/\lambda^{(1)} \rfloor$ and observe that $T_n^{(>m)}(x)$ is non-decreasing in n. Hence,

$$\mathbb{P}\left(\forall\, n \le \Lambda/\lambda^{(1)},\ \forall\, m \in \{1,\dots,M\},\ \forall\, x \in \mathcal{H}^{(m)},\ \ T_n^{(>m)}(x) \ \le\ 1 \right) > 1 - \frac{\delta}{4}. \tag{4.41}$$

The condition for Lemma 34 holds with probability at least $1-\delta/2$ (by Lemma 33), and therefore the above bounds hold together with probability $> 1-\delta$. We have tabulated these bounds in Table 4.2. We therefore have the following bound on the number of fidelity m $(< M)$ plays $T_n^{(m)}(\mathcal{X})$,

$$\begin{aligned} T_n^{(m)}(\mathcal{X}) \ &\le\ T_n^{(m)}(\breve{\mathcal{H}}^{(m)}) \ +\ \sum_{x\in\mathcal{H}^{(m)}} \left\lceil \frac{5\eta^2}{{\Delta^{(m)}(x)}^2}\beta_n \right\rceil \ +\ |\widehat{\mathcal{H}}^{(m)}| \left\lceil \frac{\eta^2}{{\gamma^{(m)}}^2}\beta_n \right\rceil \\ &\le\ T_n^{(m)}(\breve{\mathcal{H}}^{(m)}) \ +\ |\mathcal{H}^{(m)} \cup \widehat{\mathcal{H}}^{(m)}| \left\lceil \frac{\eta^2}{{\gamma^{(m)}}^2}\beta_n \right\rceil \qquad (4.42) \\ &\le\ |\mathcal{H}^{(m-1)}| \ +\ |\mathcal{H}^{(m)} \cup \widehat{\mathcal{H}}^{(m)}| \left\lceil \frac{\eta^2}{{\gamma^{(m)}}^2}\beta_n \right\rceil \qquad (4.43) \end{aligned}$$

The second step uses that $\Delta^{(m)}(x) \ge 3\gamma^{(m)}$ for $x \in \mathcal{H}^{(m)}$ and the last step uses the modification to the discrete algorithm which ensures that we will always play an arm at a lower fidelity before we play it at a higher fidelity. Hence, for an arm in $\mathcal{H}^{(m)}$, the 1 play at fidelities larger than m will be played at fidelity $m+1$.

Proof of first result: First consider the total cost $\Lambda'(n)$ expended at fidelities $1,\dots,M-1$ and at the M^{th} fidelity outside of $\mathcal{H}^{(M)}$ after n evaluations. Using (4.43), we have,

$$\begin{aligned} \Lambda'(n) \ &=\ \sum_{m=1}^{M-1} \lambda^{(m)} T_n^{(m)}(\mathcal{X}) \ +\ \lambda^{(M)} T_n^{(M)}(\breve{\mathcal{H}}^{(M)}) \\ &\le\ \sum_{m=2}^{M} \lambda^{(m)} |\mathcal{H}^{(m-1)}| \ +\ \sum_{m=1}^{M-1} \lambda^{(m)} |\mathcal{H}^{(m)} \cup \widehat{\mathcal{H}}^{(m)}| \left\lceil \frac{\eta^2}{{\gamma^{(m)}}^2}\beta_n \right\rceil. \end{aligned}$$

Since $N \leq \overline{n}_\Lambda$, we have for all $n \in \mathbf{supp}\,(N)$, $\Lambda'(n)$ is less than the LHS of (4.13) and hence less than $\Lambda/2$. Therefore, the amount of cost spent at the M^{th} fidelity inside $\mathcal{H}^{(M)}$ is at least $\Lambda/2$ and since each such evaluation expends $\lambda^{(M)}$, we have $T_N^{(M)}(\mathcal{H}^{(M)}) \geq n_\Lambda/2$. Therefore using (4.40) we have,

$$S(\Lambda) \leq \frac{1}{T_N^{(M)}(\mathcal{H}^{(M)})}\tilde{R}_N \leq \sqrt{\frac{C_1\beta_N \Psi_{T_N^{(M)}(\mathcal{H}^{(M)})}(\mathcal{H}^{(M)})}{T_N^{(M)}(\mathcal{H}^{(M)})}} \leq \sqrt{\frac{2C_1\beta_{\overline{n}_\Lambda}\Psi_{n_\Lambda}(\mathcal{H}^{(M)})}{n_\Lambda}}.$$

Here, we have used $N \leq \overline{n}_\Lambda$ and that $n_\Lambda \geq T_N^{(M)}(\mathcal{H}^{(M)}) \geq n_\Lambda/2$.

Proof of second result: Using (4.42), the total number of queries at fidelities less than M and the number of M^{th} fidelity queries outside of $\mathcal{H}^{(M)}$ can be bounded as follows,

$$\sum_{m=1}^{M-1}\sum_{x\in\mathcal{X}} T_n^{(m)}(x) \;+\; T_n^{(M)}(\widetilde{\mathcal{H}}^{(M)}) \;\leq\; |\mathcal{X}| \;+\; \sum_{m=1}^{M-1} |\mathcal{H}^{(m)} \cup \widehat{\mathcal{H}}^{(m)}| \left\lceil \frac{\eta^2}{{\gamma^{(m)}}^2}\beta_n \right\rceil. \tag{4.44}$$

The first term of the RHS above follows via (4.41) and the following argument. In particular, this does not use the additional condition on the discrete algorithm – we will use a similar argument in the continuous domain setting.

$$\sum_{m=1}^{M}\sum_{x\in\widetilde{\mathcal{H}}^{(m)}} T_n^{(m)}(x) = \sum_{m=1}^{M}\sum_{\ell=1}^{m-1}\sum_{x\in\mathcal{H}^{(\ell)}} T_n^{(m)}(x) \leq \sum_{m=1}^{M-1}\sum_{x\in\mathcal{H}^{(m)}} T_n^{(>m)}(x) \;\leq\; |\mathcal{X}|. \tag{4.45}$$

Let the LHS of (4.44) be A and the RHS be B when $n = N$. When $\Lambda > \Lambda_2$, by (4.14) and using the fact that $N \leq \overline{n}_\Lambda$, we have $B < n_\Lambda/2 < N/2$. Since $N = A + T_N^{(M)}(\mathcal{H}^{(M)})$, we have $T_N^{(M)}(\mathcal{H}^{(M)}) > N/2 > n_\Lambda/2$. Further, since the total expended budget after N rounds $\Lambda(N)$ satisfies $\Lambda(N) \geq T_N^{(M)}(\mathcal{H}^{(M)})\lambda^{(M)} > \lambda^{(M)}N/2$, we also have $N < 2n_\Lambda$. Putting these results together we have for all $\Lambda > \Lambda_2$,

$$S(\Lambda) \leq \sqrt{\frac{C_1\beta_N \Psi_{T_N^{(M)}(\mathcal{H}^{(M)})}(\mathcal{H}^{(M)})}{T_N^{(M)}(\mathcal{H}^{(M)})}} \leq \sqrt{\frac{2C_1\beta_{2n_\Lambda}\Psi_{n_\Lambda}(\mathcal{H}^{(M)})}{n_\Lambda}}.$$

■

Remark 3. ***Choice of*** $\gamma^{(m)}$***:*** As described in the main text, the optimal choice for $\gamma^{(m)}$ depends on the available budget and unknown problem dependent quantities. However the choice $\gamma^{(m)} = \sqrt{\lambda^{(m)}/\lambda^{(m+1)}}\zeta^{(m)}$ ensures that for any $x \in \mathcal{H}^{(m)}$, the bounds on the number of plays in Table 4.2 are on the same order for fidelities m and below. To see this, consider any $\ell < m$. Then,

$$\Delta^{(m)}(x) = \Delta^{(\ell)}(x) + \zeta^{(\ell)} - \zeta^{(m)} + f^{(\ell)}(x) - f^{(M)}(x) + f^{(M)}(x) - f^{(m)}(x) \leq 3\gamma^{(\ell)} + 2\zeta^{(\ell)} \leq 5\zeta^{(\ell)}.$$

We therefore have,

$$\lambda^{(\ell)} \cdot \frac{\eta^2}{{\gamma^{(\ell)}}^2} = \lambda^{(\ell+1)}\frac{\eta^2}{{\zeta^{(\ell)}}^2} \leq 5\left(\lambda^{(m)} \cdot \frac{5\eta^2}{{\Delta^{(m)}(x)}^2}\right)$$

Above, by Table 4.2, the left most expression is an upper bound on the cost spent at fidelity ℓ and the term inside the parantheses is an upper bound on the cost spent at fidelity m. Hence, the capital spent at the lower fidelities is within a constant factor of this bound. In the K-armed setting [125], we showed a $\mathcal{O}(\eta^2/\Delta^{(m)}(x)^2)$ lower bound on the number of plays at the m^{th} fidelity as well; such a result is not straightforward in the GP setting due to correlations between arms.

Proof of Lemma 33

This is a straightforward argument using Gaussian concentration and the union bound. Consider any given m, t, x.

$$\begin{aligned}
&\mathbb{P}\left(|f^{(m)}(x) - \mu_{t-1}^{(m)}(x)| > \beta_t^{1/2}\sigma_{t-1}^{(m)}(x)\right) \\
&\quad= \frac{1}{\xi_{\mathbf{A2}}}\mathbb{P}_{\mathcal{GP}}\left(|f^{(m)}(x) - \mu_{t-1}^{(m)}(x)| > \beta_t^{1/2}\sigma_{t-1}^{(m)}(x)\right) \\
&\quad= \frac{1}{\xi_{\mathbf{A2}}}\mathbb{E}_{\mathcal{GP}}\left[\mathbb{E}_{\mathcal{GP}}\left[\mathbb{1}\left\{|f^{(m)}(x) - \mu_{t-1}^{(m)}(x)| > \beta_t^{1/2}\sigma_{t-1}^{(m)}(x)\right\} \;\middle|\; \mathcal{D}_{t-1}^{(m)}\right]\right] \\
&\quad= \frac{1}{\xi_{\mathbf{A2}}}\mathbb{E}_{\mathcal{GP}}\left[\mathbb{E}_{\mathcal{GP}}\left[\mathbb{1}\left\{|f^{(m)}(x) - \mu_{t-1}^{(m)}(x)| > \beta_t^{1/2}\sigma_{t-1}^{(m)}(x)\right\} \;\middle|\; \mathcal{D}_{t-1}^{(m)}\right]\right] \\
&\quad= \frac{1}{\xi_{\mathbf{A2}}}\mathbb{E}_{\mathcal{GP}}\left[\mathbb{P}_{Z\sim\mathcal{N}(0,1)}\left(|Z| > \beta_t^{1/2}\right)\right] \;\leq\; \frac{1}{\xi_{\mathbf{A2}}}\exp\left(\frac{\beta_t}{2}\right) \;=\; \frac{3\delta}{M|\mathcal{X}|\pi^2 t^2}.
\end{aligned}$$

The first step uses Lemma 31. In the second step we have conditioned w.r.t $\mathcal{D}_{t-1}^{(m)}$ which allows us to use Lemma 4. Recall that conditioning on all queries will not be a Gaussian due to the $\zeta^{(m)}$ constraints. The statement follows via a union bound over all $m \in \{1, \ldots, M\}$, $x \in \mathcal{X}$ and all t and noting that $\sum_t t^{-2} = \pi^2/6$. ■

Proof of Lemma 34

First consider any $\ell < m$. Assume that we have already queried $\lceil \eta^2\beta_n/{\gamma^{(m)}}^2 \rceil$ times at any $t \leq n$. Since the Gaussian variance after s observations is η^2/s and that queries elsewhere will only decrease the conditional variance we have, $\kappa_{t-1}^{(\ell)}(x,x) \leq \eta^2/T_{t-1}^{(\ell)}(x) < {\gamma^{(m)}}^2/\beta_n$. Therefore, $\beta_t^{1/2}\sigma_{t-1}^{(\ell)}(x) < \beta_n^{1/2}\sigma_{t-1}^{(\ell)}(x) < \gamma^{(m)}$ and by the design of our algorithm we will not play at the ℓ^{th} fidelity at time t for all t until n. This establishes the first result.

To bound $T_n^{(m)}(x)$ we first observe,

$$\begin{aligned}
\mathbb{1}\{T_n^{(m)}(x) > u\} \;\leq\; &\mathbb{1}\{\exists t : u+1 \leq t \leq n :\; \varphi_t(x) \text{ was maximum} \;\wedge \\
&\quad \beta_t^{1/2}\sigma_{t-1}^{(\ell)}(x) < \gamma^{(m)},\, \forall\, \ell < m \;\wedge\; \beta_t^{1/2}\sigma_{t-1}^{(m)}(x) \geq \gamma^{(m)} \wedge \\
&\quad T_{t-1}^{(m)}(x) \geq u\} \\
\leq\; &\mathbb{1}\{\exists t : u+1 \leq t \leq n :\; \varphi_t(x) > \varphi_t(x_\star) \quad\wedge\quad T_{t-1}^{(m)}(x) \geq u\}
\end{aligned}$$

$$\leq \quad \mathbb{1}\big\{\exists t : u+1 \leq t \leq n : \ \varphi_t^{(m)}(x) > f(x_\star) \quad \wedge \quad T_{t-1}^{(m)}(x) \geq u\big\}. \quad (4.46)$$

The first line just enumerates the conditions in our algorithm for it to have played x at time t at fidelity m. In the second step we have relaxed some of those conditions, noting in particular that if $\varphi_t(\cdot)$ was maximised at x then it must be larger than $\varphi_t(x_\star)$. The last step uses the fact that $\varphi_t^{(m)}(x) \geq \varphi_t(x)$ and the assumption on $\varphi_t(x_\star)$. Consider the event $\{\varphi_t^{(m)}(x) > f(x_\star) \wedge T_{t-1}^{(m)}(x) \geq u\}$. We will choose $u = \lceil 5\eta^2\beta_n/\Delta^{(m)}(x)^2 \rceil$ and bound its probability via,

$$\begin{aligned}
&\mathbb{P}\Big(\varphi_t^{(m)}(x) > f(x_\star) \wedge T_{t-1}^{(m)}(x) \geq u\Big) \\
&\quad = \frac{1}{\xi_{\mathbf{A2}}}\mathbb{P}_{\mathcal{GP}}\Big(\mu_{t-1}^{(m)}(x) + \beta_t^{1/2}\sigma_{t-1}^{(m)}(x) + \zeta^{(m)} > f(x_\star) \quad \wedge \quad T_{t-1}^{(m)}(x) \geq u\Big) \\
&\quad = \frac{1}{\xi_{\mathbf{A2}}}\mathbb{P}_{\mathcal{GP}}\Bigg(\mu_{t-1}^{(m)}(x) - f^{(m)}(x) > \underbrace{f(x_\star) - f^{(m)}(x) - \zeta^{(m)}}_{\Delta^{(m)}(x)} - \beta_t^{1/2}\sigma_{t-1}^{(m)}(x) \wedge \\
&\qquad\qquad T_{t-1}^{(m)}(x) > u\Bigg) \\
&\quad \leq \frac{1}{\xi_{\mathbf{A2}}}\mathbb{P}_{\mathcal{GP}}\Big(\mu_{t-1}^{(m)}(x) - f^{(m)}(x) > (\sqrt{5}-1)\beta_n^{1/2}\sigma_{t-1}^{(m)}(x)\Big) \\
&\quad \leq \frac{1}{\xi_{\mathbf{A2}}}\mathbb{P}_{Z\sim\mathcal{N}(0,1)}\Big(Z > \frac{(\sqrt{5}-1)^2}{2}\beta_n^{1/2}\Big) \\
&\quad \leq \frac{1}{\xi_{\mathbf{A2}}}\frac{1}{2}\exp\left(-\frac{3}{4}\beta_n\right) = \frac{1}{\xi_{\mathbf{A2}}}\frac{1}{2}\left(\frac{3\xi_{\mathbf{A2}}\delta}{M|\mathcal{X}|\pi^2}\right)^{\frac{3}{2}} n^{-3} \leq \frac{1}{2}\frac{3\delta}{M|\mathcal{X}|\pi^2}n^{-3}
\end{aligned}$$

Above in the third step we have used, if $u \geq 5\eta^2\beta_n/\Delta^{(m)}(x)^2$, then $\Delta^{(m)}(x) \geq \sqrt{5}\beta_n^{1/2}\sigma_{t-1}^{(m)}(x)$ and that $\beta_n \geq \beta_t$. The fourth step uses Lemma 4 after conditioning on $\mathcal{D}_{t-1}^{(m)}$, the fifth step uses $(\sqrt{5}-1)^2 > 3/2$ and the last step uses $3\delta/|\mathcal{X}|\pi^2 < 1$. Using the union bound on (4.46), we get $\mathbb{P}(T_n^{(m)}(x) > u) \leq \sum_{t=u+1}^n \mathbb{P}(\varphi_t^{(m)}(x) > f(x_\star) \wedge T_{t-1}^{(m)}(x) \geq u)$. Now (4.46) implies that $\mathbb{P}(T_n^{(m)}(x) > u) \leq \sum_{t=u+1}^n \mathbb{P}(\varphi_t^{(m)}(x) > f(x_\star) \wedge T_{t-1}^{(m)}(x) \geq u)$. The second inequality of the lemma follows by noting that there are at most n terms in the summation.

Finally, for the third inequality we observe

$$\mathbb{P}(T_n^{(>m)}(x) > u) \ \leq \ \mathbb{P}\big(\exists t : u+1 \leq t \leq n \,;\ \varphi_t^{(m)}(x) > f(x_\star) \quad \wedge \quad \beta_t^{1/2}\sigma_{t-1}^{(m)}(x) < \gamma^{(m)}\big). \quad (4.47)$$

As before, we have used that if x is to be queried at time t, then $\varphi_t(x)$ should be at least larger than $\varphi_t(x_\star)$ which is larger than $f(x_\star)$ due to the assumption in the theorem. The second condition is necessary to ensure that the switching procedure proceeds beyond the m^{th} fidelity. It is also necessary to have $\beta_t^{1/2}\sigma_{t-1}^{(\ell)}(x) < \gamma^{(\ell)}$ for $\ell < m$, but we have relaxed them. We first bound the probability of the event $\{\varphi_t^{(m)}(x) > f(x_\star) \quad \wedge \quad \beta_t^{1/2}\sigma_{t-1}^{(m)}(x) < \gamma^{(m)}\}$.

$$\mathbb{P}\big(\varphi_t^{(m)}(x) \ > \ f(x_\star) \wedge \beta_t^{1/2}\sigma_{t-1}^{(m)}(x) < \gamma^{(m)}\big)$$

$$
\begin{aligned}
&= \frac{1}{\xi_{\mathbf{A2}}}\mathbb{P}_{\mathcal{GP}}\big(\varphi_t^{(m)}(x) \;>\; f(x_\star) \;\wedge\; \beta_t^{1/2}\sigma_{t-1}^{(m)}(x) < \gamma^{(m)}\big) \\
&= \frac{1}{\xi_{\mathbf{A2}}}\mathbb{P}_{\mathcal{GP}}\big(\mu_{t-1}^{(m)}(x) - f^{(m)}(x) \;>\; \Delta^{(m)}(x) - \beta_t^{1/2}\sigma_{t-1}^{(m)}(x) \;\wedge\; \beta_t^{1/2}\sigma_{t-1}^{(m)}(x) < \gamma^{(m)}\big) \\
&\leq \frac{1}{\xi_{\mathbf{A2}}}\mathbb{P}_{\mathcal{GP}}\big(\mu_{t-1}^{(m)}(x) - f^{(m)}(x) \;>\; 2\gamma^{(m)} - \beta_t^{1/2}\sigma_{t-1}^{(m)}(x) \quad\wedge\quad \beta_t^{1/2}\sigma_{t-1}^{(m)}(x) < \gamma^{(m)}\big) \\
&\leq \frac{1}{\xi_{\mathbf{A2}}}\mathbb{P}_{\mathcal{GP}}\big(\mu_{t-1}^{(m)}(x) - f^{(m)}(x) \;>\; \beta_t^{1/2}\sigma_{t-1}^{(m)}(x)\big) \\
&\leq \frac{1}{\xi_{\mathbf{A2}}}\mathbb{P}_{Z\sim\mathcal{N}(0,1)}\Big(Z \;>\; \beta_t^{1/2}\Big) \;\leq\; \frac{1}{\xi_{\mathbf{A2}}}\frac{1}{2}\exp\left(-\frac{1}{2}\beta_t\right) \\
&= \frac{1}{\xi_{\mathbf{A2}}}\frac{1}{2}\left(\frac{3\xi_{\mathbf{A2}}\delta}{M|\mathcal{X}|\pi^2}\right)t^{-2} \;\leq\; \frac{1}{2}\frac{3\delta}{M|\mathcal{X}|\pi^2}t^{-2}
\end{aligned}
$$

Here, the second step uses that for all $x \in \mathcal{H}^{(m)}$, $\Delta^{(m)}(x) > 3\gamma^{(m)} > 2\gamma^{(m)}$ and the third step uses the second condition. Using the union bound on (4.47) and bounding the sum by an integral gives us,

$$
\begin{aligned}
\mathbb{P}(T_n^{(>m)}(x) > u) \;&\leq\; \sum_{t=u+1}^{n}\frac{1}{2}\frac{3\delta}{M|\mathcal{X}|\pi^2}t^{-2} \;\leq\; \frac{1}{2}\frac{3\delta}{M|\mathcal{X}|\pi^2}\int_u^\infty t^{-2}\mathrm{d}t \\
&\leq\; \frac{1}{2}\;\frac{3\delta}{M|\mathcal{X}|\pi^2}\;\frac{1}{u}.
\end{aligned}
$$

■

4.5.3 Compact and Convex $\mathcal{X}$

To prove theorem 22 we will require a fairly delicate set up for the continuous setting. Given a sequence $\{\nu_n\}_{n\geq 0}$, at time n we will consider a $r\sqrt{d}/(2\nu_n^{1/2d})$-covering of the space $\mathcal{X}$ of size $\nu_n^{1/2}$. For instance, if $\mathcal{X} = [0,r]^d$ a sufficient discretisation would be an equally spaced grid having $\nu_n^{1/2d}$ points per side. Let $\{a_{i,n}\}_{i=1}^{n^{\frac{\alpha}{2}}}$ be the points in the covering, $F_n = \{A_{i,n}\}_{i=1}^{n^{\frac{\alpha}{2}}}$ be the "cells" in the covering, i.e. $A_{i,n}$ is the set of points which are closest to $a_{i,n}$ in $\mathcal{X}$ and the union of all sets $A_{i,n}$ in F_n is $\mathcal{X}$. Next we will define another partitioning of the space similar using this covering. First let $F_n^{(1)} = \{A_{i,n} \in F_n :\; A_{i,n} \subset \mathcal{J}^{(1)}_{\max(\tau,\rho\gamma)}\}$. Next,

$$
F_n^{(m)} = \left\{ A_{i,n} \in F_n :\; A_{i,n} \subset \overline{\mathcal{J}}^{(m)}_{\max(\tau,\rho\gamma)} \;\wedge\; A_{i,n} \notin \bigcup_{\ell=1}^{m-1} F_n^{(\ell)} \right\} \quad \text{for } \; 2 \leq m \leq M-1. \tag{4.48}
$$

Note that $F_n^{(m)} \subset F_n^{(m)}$. We define the following *disjoint* subsets $\{\mathcal{F}_n^{(m)}\}_{m=1}^{M-1}$ of $\mathcal{X}$ via $\mathcal{F}_n^{(m)} = \bigcup_{A_{i,n}\in F_n^{(m)}} A_{i,n}$. We have illustrated $\bigcup_{\ell=1}^{m-1}\mathcal{F}_n^{(\ell)}$ with respect to $\mathcal{H}_\tau^{(m)}$ and $\mathcal{H}_{\tau,n}^{(m)}$ in Figure 4.18. By observing that $\mathcal{H}_{\tau,n}^{(1)} = \mathcal{H}^{(1)}$ and that $\overline{\mathcal{H}_{\tau,n}^{(m)} \cup \widehat{\mathcal{H}}^{(m)}} \subset \bigcup_{\ell=1}^{m-1}\mathcal{F}_n^{(\ell)}$ (see Figure 4.18) we have

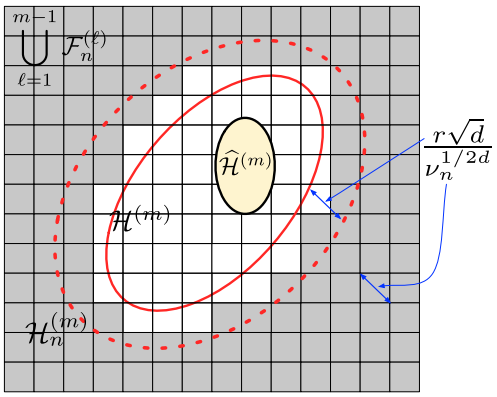


Figure 4.18: Illustration of the sets $\{\mathcal{F}_n^{(\ell)}\}_{\ell=1}^{m-1}$ with respect to $\mathcal{H}_\tau^{(m)}$. The grid represents a $r\sqrt{d}/n^{1/(2d)}$ covering of $\mathcal{X}$. The yellow region is $\widehat{\mathcal{H}}_\tau^{(m)}$. The area enclosed by the solid red line (excluding $\widehat{\mathcal{H}}_\tau^{(m)}$) is $\mathcal{H}_\tau^{(m)}$. $\mathcal{H}_{\tau,n}^{(m)}$, shown by a dashed red line, is obtained by dilating $\mathcal{H}_\tau^{(m)}$ by $r\sqrt{d}/n^{\alpha/2d}$. The grey shaded region represents $\bigcup_{\ell=1}^{m-1}\mathcal{F}_n^{(\ell)}$. By our definition, $\bigcup_{\ell=1}^{m-1}\mathcal{F}_n^{(\ell)}$ contains the cells which are entirely outside $\mathcal{H}_\tau^{(m)}$. However, the inflation $\mathcal{H}_{\tau,n}^{(m)}$ is such that $\widehat{\mathcal{H}}_\tau^{(m)} \cup \mathcal{H}_{\tau,n}^{(m)} \cup \bigcup_{\ell=1}^{m-1}\mathcal{F}_n^{(\ell)} = \mathcal{X}$. We further note that as $n \to \infty$, $\mathcal{H}_{\tau,n}^{(m)} \to \mathcal{H}_\tau^{(m)}$.

the following,

$$\forall m \in \{1,\ldots,M\}, \quad T_n^{(m)}(\mathcal{X}) \leq \left(\sum_{\ell=1}^{m-1} T_n^{(m)}(\mathcal{F}_n^{(\ell)})\right) + T_n^{(m)}(\mathcal{H}_{\tau,n}^{(m)}) + T_n^{(m)}(\widehat{\mathcal{H}}^{(m)}). \quad (4.49)$$

We are now ready to prove Theorem 22. We will denote the ε covering number of a set $A \subset \mathcal{X}$ in the $\|\cdot\|_2$ metric by $\Omega^{(\varepsilon)}(A)$.

Proof of Theorem 22. As in the discrete case, we will first control the regret and the number of lower fidelity evaluations by controlling each term in (4.49).

Bounding the regret after n evaluations: We will need the following lemma whose proof is given in Chapter 4.5.2.

Lemma 35. *For β_t as given in Theorem 22, the following holds with probability $> 1 - 5\delta/6$.*

$$\forall\, m \in \{1,\ldots,M\}, \ \forall\, t \geq 1, \quad \Delta^{(m)}(x_t) = f(x_\star) - f^{(m)}(x_t) \leq 2\beta_t \sigma_{t-1}^{(m)}(x_t) + 1/t^2.$$

As in the discrete setting, we set $\mathcal{Z} = \mathcal{H}_{\tau,n}^{(M)}$ in (4.37) to bound $\tilde{R}_n$. Using $m = M$ in Lemma 35 and using calculations similar to the discrete case yields,

$$\tilde{R}_n \leq \sum_{\substack{m_t = M \\ x_t \in \mathcal{Z}}} \left(2\beta_t^{1/2}\sigma_{t-1}^{(M)}(x_t) + \frac{1}{t^2}\right) \leq \sqrt{C_1 T_n^{(m)}(\mathcal{H}_{\tau,n}^{(M)})\beta_n \Psi_{T_n^{(m)}(\mathcal{H}_{\tau,n}^{(m)})}(\mathcal{H}_{\tau,n}^{(m)})} + \frac{\pi^2}{6}. \quad (4.50)$$

Here $C_1 = 8/\log(1+\eta^{-2})$. We have also used the fact $\sum_{t>0} t^{-2} = \frac{\pi^2}{6}$.

Bounding the number of evaluations: The following lemma will be used to bound the number of plays in $\mathcal{H}_{\tau,n}^{(m)} \cup \widehat{\mathcal{H}}^{(m)}$. The proof is given in Chapter 4.5.3.

Lemma 36. *Let $f \sim \mathcal{GP}(\mathbf{0}, \kappa)$, $f : \mathcal{X} \to \mathbb{R}$ and we observe $y = f(x) + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \eta^2)$. Let $A \subset \mathcal{X}$ such that its L_2 diameter $\mathrm{diam}(A) \leq D$. Say we have n queries $(x_t)_{t=1}^n$ of which s*

points are in A. Then the posterior variance of the GP, $\kappa'(x,x)$ at any $x \in A$ satisfies

$$\kappa'(x,x) \leq \begin{cases} C_{SE}D^2 + \frac{\eta^2}{s} & \text{if } \kappa \text{ is the SE kernel,} \\ C_{Mat}D + \frac{\eta^2}{s} & \text{if } \kappa \text{ is the Matérn kernel,} \end{cases}$$

for appropriate kernel dependent constants C_{SE}, C_{Mat}.

First consider the SE kernel. At time t consider any $\varepsilon_n = \frac{\gamma^{(m)}}{\sqrt{8C_{SE}\beta_n}}$ covering $(B_i)_{i=1}^{\varepsilon_n}$ of $\mathcal{H}_{\tau,n}^{(m)} \cup \widehat{\mathcal{H}}^{(m)}$. The number of queries inside any B_i of this covering at time n will be at most $\left\lceil \frac{2\eta^2}{{\gamma^{(m)}}^2}\beta_n \right\rceil$. To see this, assume we have already queried this many times inside B_i at time $t \leq n$. By Lemma 36 the maximum variance in A_i can be bounded by

$$\max_{x \in A_i} \kappa_{t-1}^{(m)}(x,x) \leq C_{SE}(2\varepsilon_n)^2 \ + \ \frac{\eta^2}{T_t^{(m)}(A_i)} \leq \frac{{\gamma^{(m)}}^2}{\beta_n}.$$

Therefore, $\beta_t^{1/2}\sigma_{t-1}^{(m)}(x) \leq \beta_n^{1/2}\sigma_{t-1}^{(m)}(x) < \gamma^{(m)}$ and we will not query inside A_i until time n. A similar result is obtained for the Matérn kernel by setting $\varepsilon_n = \frac{{\gamma^{(m)}}^2}{4C_{Mat}\beta_n}$. Therefore we have,

$$\begin{aligned} T_n^{(m)}(\mathcal{H}_{\tau,n}^{(m)} \cup \widehat{\mathcal{H}}^{(m)}) \ &\leq \ \Omega_{\varepsilon_n}(\widehat{\mathcal{H}}^{(m)} \cup \widehat{\mathcal{H}}^{(m)}) \left\lceil \frac{2\eta^2}{{\gamma^{(m)}}^2}\beta_n \right\rceil \\ &\leq \ C_\kappa \eta^2 \beta_n^{p+1} \frac{\mathrm{vol}(\mathcal{H}_{\tau,n}^{(m)} \cup \widehat{\mathcal{H}}^{(m)})}{{\gamma^{(m)}}^{2p}}. \end{aligned} \tag{4.51}$$

Here $C_\kappa = 2^{2+d/2}(dC_{SE})^{\frac{d}{2}}$ and $p = 1/2$ for the SE kernel while $C_\kappa = 2^{2+d}(C_{Mat})^d d^{d/2}$ and $p = 1$ for the Matérn kernel. We have also used the fact that $\lceil k \rceil \leq 2k$ for large enough k and the following bound for a δ-packing in the Euclidean metric $\Omega_\delta(A) \leq \mathrm{vol}(A)d^{d/2}/(2^{d/2}\delta^d)$.

Next, we will bound $T_n^{(m)}(\overline{\mathcal{H}_{\tau,n}^{(m)} \cup \widehat{\mathcal{H}}^{(m)}})$ by controlling. $T_n^{(>m)}(\mathcal{F}_n^{(m)})$. To that end we provide the following Lemma whose proof is given in Chapter 4.5.3.

Lemma 37. *Consider any $A_{i,n} \in F_n^{(m)}$ where $F_n^{(m)}$ is as defined in (4.48). Let β_t be as given in Theorem 22. Then for all $n' \geq u \geq (3\eta)^{-2/3}$ we have,*

$$\mathbb{P}(T_{n'}^{(>m)}(A_{i,n}) > u) \ \leq \ \frac{\delta}{\pi^2} \cdot \frac{1}{u}$$

Using the above result with $n' = \overline{n}_\Lambda$ gives us the result for all $n' \leq \overline{n}_\Lambda$ since $T_{n'}^{(>m)}(A_{i,n})$ is nondecreasing with n. Setting $u = \max\{(3\eta)^{-2/3}, \nu_n^{1/2}\}$, and applying the union bound over all $m \in \{1, \ldots, M\}$ and $A_{i,n} \in F_n^{(m)}$, yields the following bound for all $n' \leq \overline{n}_\Lambda$,

$$\mathbb{P}\left(\exists\, m \in \{1,\ldots,M\},\, T_{n'}^{(>m)}(\mathcal{F}_n^{(m)}) > |F_n^{(m)}|\nu_n^{1/2}\right) \ \leq \ \sum_{m=1}^{M} \mathbb{P}\left(T_{n'}^{(>m)}(\mathcal{F}_n^{(m)}) > |F_n^{(m)}|\nu_n^{1/2}\right)$$

$$\leq \sum_{m=1}^{M} \sum_{A_{i,n} \in F_n^{(m)}} \mathbb{P}\left(T_{n'}^{(>m)}(A_{i,n}) > \nu_n^{1/2}\right) \leq \sum_{m=1}^{M} |F_n^{(m)}| \frac{\delta}{\pi^2} \frac{1}{\nu_n^{1/2}}$$
$$\leq |F_n| \frac{\delta}{\pi^2} \frac{1}{\nu_n^{1/2}} = \frac{\delta}{\pi^2} \leq \frac{\delta}{6}. \tag{4.52}$$

Henceforth, all statements we make will make use of the bounds above and will hold with probability $> 1 - \delta$ for all $n \in \mathbf{supp}\,(N)$.

Proof of first result: Consider the cost $\Lambda'(n)$ spent at fidelities $1, \ldots, M-1$ and at the M^{th} fidelity outside of $\mathcal{H}_{\tau,n}^{(M)}$ after n evaluations.

$$\begin{aligned}
\Lambda'(n) &= \sum_{m=1}^{M-1} \lambda^{(m)} T_n^{(m)}(\mathcal{X}) + \lambda^{(M)} T_n^{(M)}(\mathcal{H}_{\tau,n}^{(M)}) \\
&= \sum_{m=1}^{M} \lambda^{(m)} \left(\sum_{\ell=1}^{m-1} T_n^{(m)}(\mathcal{F}_n^{(\ell)}) \right) + \sum_{m=1}^{M-1} \lambda^{(m)} T_n^{(m)}(\mathcal{H}_{\tau,n}^{(m)} \cup \widehat{\mathcal{H}}^{(m)}) \\
&\leq \lambda^{(M)} \nu_n + C_\kappa \eta^2 \beta_n^{p+1} \sum_{m=1}^{M-1} \lambda^{(m)} \frac{\mathrm{vol}(\mathcal{H}_{\tau,n}^{(m)} \cup \widehat{\mathcal{H}}^{(m)})}{{\gamma^{(m)}}^{2p}}
\end{aligned}$$

The second step uses (4.49). The third step uses (4.51), (4.52), and the following argument,

$$\sum_{m=1}^{M} \left(\sum_{\ell=1}^{m-1} T_n^{(m)}(\mathcal{F}_n^{(\ell)}) \right) \leq \sum_{m=1}^{M-1} T_n^{(>m)}(\mathcal{F}_n^{(\ell)}) \leq \sum_{m=1}^{M-1} |F_n^{(m)}| \nu_n^{1/2} \leq \nu_n^{1/2} |F_n| \leq \nu_n. \tag{4.53}$$

The remainder of the proof follows similar to the discrete case. Noting that $n_\Lambda \leq n \leq \overline{n}_\Lambda$ and that $\mathcal{H}_{\tau,n}^{(m)}$ is shrinking with n, we can conclude that $\Lambda'(n)$ is less than the LHS of (4.16). Therefore, $T_N^{(M)}(\mathcal{H}_{\tau,n}^{(M)}) \geq n_\Lambda/2$ and hence,

$$S(\Lambda) \leq \sqrt{\frac{C_1 \beta_N \Psi_{T_N^{(M)}(\mathcal{H}_{\tau,n}^{(m)})}(\mathcal{H}_{\tau,n}^{(m)})}{T_N^{(M)}(\mathcal{H}_{\tau,n}^{(M)})} + \frac{\pi^2}{6 T_N^{(M)}(\mathcal{H}_{\tau,n}^{(M)})}} \leq \sqrt{\frac{2 C_1 \beta_{\overline{n}_\Lambda} \Psi_{n_\Lambda}(\mathcal{H}_{n_\Lambda}^{(M)})}{n_\Lambda}} + \frac{\pi^2}{3 n_\Lambda}.$$

Proof of second result: As in the discrete case, we bound the number of queries at fidelity $m < M$ and the M^{th} fidelity queries outside $\mathcal{H}_{\tau,n}^{(M)} \cup \mathcal{H}^{(M)}$ as follows.

$$\begin{aligned}
\sum_{m=1}^{M-1} T_n^{(m)}(\mathcal{X}) + T_n^{(M)}(\overline{\mathcal{H}_{\tau,n}^{(M)}}) &\leq \sum_{m=1}^{M} \left(\sum_{\ell=1}^{m-1} T_n^{(m)}(\mathcal{F}_n^{(\ell)}) \right) + \sum_{m=1}^{M-1} T_n^{(m)}(\mathcal{H}_{\tau,n}^{(m)} \cup \widehat{\mathcal{H}}^{(m)}) \\
&\leq \nu_n + C_\kappa \beta_n^{p+1} \sum_{m=1}^{M-1} \frac{\mathrm{vol}(\mathcal{H}_{\tau,n}^{(M)} \cup \mathcal{H}^{(M)})}{{\gamma^{(m)}}^{2p}}
\end{aligned} \tag{4.54}$$

The first step uses (4.49) while the second step uses (4.51) and (4.53). Once again, similar to the discrete case we can argue that for all $\Lambda > \Lambda_2$, the RHS B of (4.54) satisfies $B < n_\Lambda/2 < N/2$, the M^{th} fidelity plays in $\mathcal{H}_{\tau,n}^{(M)}$ satisfies $T_N^{(M)}(\mathcal{H}_{\tau,n}^{(M)}) > N/2 > n_\Lambda/2$, and the number of plays satisfies $N \leq 2n_\Lambda$. Combining this with (4.50) gives us the following for all $n \leq n_\Lambda$,

$$S(\Lambda) \leq \sqrt{\frac{C_1 \beta_N \Psi_{T_N^{(M)}(\mathcal{H}_{\tau,n}^{(m)})}(\mathcal{H}_{\tau,n}^{(m)})}{T_N^{(M)}(\mathcal{H}_{\tau,n}^{(M)})} + \frac{\pi^2}{6T_N^{(M)}(\mathcal{H}_{\tau,n}^{(M)})}} \leq \sqrt{\frac{2C_1 \beta_{2n_\Lambda} \Psi_{n_\Lambda}(\mathcal{H}_{n_\Lambda}^{(M)})}{n_\Lambda} + \frac{\pi^2}{3n_\Lambda}}.$$

■

Proof of Lemma 35

The first part of the proof mimics the arguments in Lemmas 5.6, 5.7 of Srinivas et al. [235]. By Assumption 1 for any given $m \in \{1, \ldots, M\}$ and $i \in \{1, \ldots, d\}$ we have,

$$\mathbb{P}_{\mathcal{GP}}\left(\left| \frac{\partial f^{(m)}(x)}{\partial x_i} \right| > b\sqrt{\log\left(\frac{6Mad}{\xi_{\mathbf{A2}}\delta}\right)} \right) \leq \frac{\xi_{\mathbf{A2}}\delta}{6Md}$$

Then, by the union bound and Lemma 31 we have,

$$\mathbb{P}\Bigg(\forall m \in \{1, \ldots, M\}, \forall i \in \{1, \ldots, d\}, \forall x \in \mathcal{X}, \ \left| \frac{\partial f^{(m)}(x)}{\partial x_i} \right| < b\sqrt{\log\left(\frac{6Mad}{\xi_{\mathbf{A2}}\delta}\right)} \Bigg) \geq 1 - \frac{\delta}{6}.$$

Now we construct a discretisation F_t of $\mathcal{X}$ of size $(\nu_t)^d$ such that we have for all $x \in \mathcal{X}$, $\|x - [x]_t\|_1 \leq rd/\nu_t$. Here $[x]_t$ is the closest point to x in the discretisation. (Note that this is different from the discretisation appearing in Theorem 22 even though we have used the same notation). By choosing $\nu_t = t^2 brd\sqrt{\log(6Mad/(\xi_{\mathbf{A2}}\delta))}$ and using the above we have

$$\forall x \in \mathcal{X}, \quad |f^{(m)}(x) - f^{(m)}([x]_t)| \leq b\log(6Mad/\delta)\|x - [x]_t\|_1 \leq 1/t^2 \tag{4.55}$$

for all $f^{(m)}$'s with probability $> 1 - \delta/6$.

Noting that $\beta_t \geq 2\log(M|F_t|\pi^2 t^2/2\delta)$ for the given choice of ν_t we have the following with probability $> 1 - \delta/3$.

$$\forall t \geq 1, \ \forall m \in \{1, \ldots, M\}, \ \forall a \in F_t, \quad |f^{(m)}(a) - \mu_{t-1}^{(m)}(a)| \leq \beta_t^{1/2}\sigma_{t-1}^{(m)}(a). \tag{4.56}$$

The proof mimics that of Lemma 33 using the same conditioning argument. However, instead of a fixed set over all t, we change the set at which we have confidence based on the discretisation. Similarly we can show that with probability $> 1 - \delta/3$ we also have confidence on the decisions x_t at all time steps. Precisely,

$$\forall t \geq 1, \ \forall m \in \{1, \ldots, M\}, \quad |f^{(m)}(x_t) - \mu_{t-1}^{(m)}(x_t)| \leq \beta_t^{1/2}\sigma_{t-1}^{(m)}(x_t). \tag{4.57}$$

Using (4.55),(4.56) and (4.57) the following statements hold with probability $> 1 - 5\delta/6$. First we can upper bound $f(x_\star)$ by,

$$f(x_\star) \;\le\; f^{(m)}(x_\star) + \zeta^{(m)} \;\le\; f^{(m)}([x_\star]_t) + \zeta^{(m)} + \frac{1}{t^2} \;\le\; \varphi_t^{(m)}([x_\star]_t) + \frac{1}{t^2}. \tag{4.58}$$

Since the above holds for all m, we have $f(x_\star) \le \varphi_t([x_\star]_t) + 1/t^2$. Now, using similar calculations as (4.39) we bound $\Delta^{(m)}(x_t)$.

$$\begin{aligned}
\Delta^{(m)}(x_t) =\; & f(x_\star) - f^{(m)}(x_t) - \zeta^{(m)} \\
\le\; & \varphi_t([x_\star]_t) + \frac{1}{t^2} - f^{(m)}(x_t) - \zeta^{(m)} \;\le\; \varphi_t(x_t) - f^{(m)}(x_t) - \zeta^{(m)} + \frac{1}{t^2} \\
\le\; & \varphi_t^{(m)}(x_t) - \mu_{t-1}^{(m)}(x_t) + \beta_t^{1/2}\sigma_{t-1}^{(m)}(x_t) - \zeta^{(m)} + \frac{1}{t^2} \;\le\; 2\beta_t^{1/2}\sigma_{t-1}^{(m)}(x_t) + \frac{1}{t^2}.
\end{aligned}$$

■

Proof of Lemma 36

Since the posterior variance only decreases with more observations, we can upper bound $\kappa'(x,x)$ for any $x \in A$ by considering its posterior variance with only the s observations in A. Further the maximum variance within A occurs if we pick 2 points x_1, x_2 that are distance D apart and have all observations at x_1; then x_2 has the highest posterior variance. Therefore, we will bound $\kappa'(x,x)$ for any $x \in A$ with $\kappa(x_2,x_2)$ in the above scenario. Let $\kappa_{\mathcal{Z}} = \kappa(x,x)$ and $\kappa(x,x') = \kappa_{\mathcal{Z}}\phi(\|x - x'\|_2)$, where $\phi(\cdot) \le 1$ depends on the kernel. Denote the gram matrix in the scenario described above by $\Delta = \kappa_{\mathcal{Z}}\mathbf{1}\mathbf{1}^\top + \eta^2 I$. Then using the Sherman-Morrison formula on the posterior variance (2.1),

$$\begin{aligned}
\kappa'(x,x) \;\le\; \kappa'(x_2,x_2) \;=\; & \kappa(x_2,x_2) - \left[\kappa(x_1,x_2)\mathbf{1}\right]^\top \Delta^{-1}\left[\kappa(x_1,x_2)\mathbf{1}\right] \\
= \; & \kappa_{\mathcal{Z}} - \kappa_{\mathcal{Z}}^2\phi^2(D)\mathbf{1}^\top\left[\kappa_{\mathcal{Z}}\mathbf{1}\mathbf{1}^\top + \eta^2 I\right]^{-1}\mathbf{1} \\
= \; & \kappa_{\mathcal{Z}} - \kappa_{\mathcal{Z}}\phi^2(D)\mathbf{1}^\top\left[\frac{\kappa_{\mathcal{Z}}}{\eta^2}I - \frac{\left(\frac{\kappa_{\mathcal{Z}}}{\eta^2}\right)^2\mathbf{1}\mathbf{1}^\top}{1 + \frac{\kappa_{\mathcal{Z}}}{\eta^2}s}\right]\mathbf{1} \\
= \; & \kappa_{\mathcal{Z}} - \kappa_{\mathcal{Z}}\phi^2(D)\left(\frac{\kappa_{\mathcal{Z}}}{\eta^2}s - \frac{\left(\frac{\kappa_{\mathcal{Z}}}{\eta^2}\right)^2 s^2}{1 + \frac{\kappa_{\mathcal{Z}}}{\eta^2}s}\right) \\
= \; & \kappa_{\mathcal{Z}} - \kappa_{\mathcal{Z}}\phi^2(D)\frac{s}{\frac{\eta^2}{\kappa_{\mathcal{Z}}} + s} \;=\; \frac{1}{1 + \frac{\eta^2}{\kappa_{\mathcal{Z}}s}}\left(\kappa_{\mathcal{Z}} - \kappa_{\mathcal{Z}}\phi^2(D) + \frac{\eta^2}{s}\right) \\
\le \; & \kappa_{\mathcal{Z}}(1 - \phi^2(D)) + \frac{\eta^2}{s}.
\end{aligned}$$

For the SE kernel $\phi^2(D) = \exp\left(\frac{-D^2}{2h^2}\right)^2 = \exp\left(\frac{-D^2}{h^2}\right) \le 1 - \frac{D^2}{h^2}$. Plugging this into the bound above retrieves the first result with $C_{SE} = \kappa_{\mathcal{Z}}/h^2$. For the Matérn kernel we use a Lipschtiz

constant L_{Mat} of ϕ. Then $1-\phi^2(D) = (1-\phi(D))(1+\phi(D)) \leq 2(\phi(0)-\phi(D)) \leq 2L_{Mat}D$. We get the second result with $C_{Mat} = 2\kappa_{\mathcal{Z}}L_{Mat}$. Since the SE kernel decays fast, we get a stronger result on its posterior variance which translates to a better bound in our theorems. ■

Proof of Lemma 37

First, we will invoke the same discretisation used in the proof of Lemma 35 via which we have $\varphi_t([x_\star]_t) \geq f(x_\star) - 1/t^2$ (4.58). (Therefore, Lemma 37 holds only with probability $> 1-\delta/6$, but this event has already been accounted for in Lemma 35.) Let $b_{i,n,t} = \operatorname{argmax}_{x\in A_{i,n}} \varphi_t(x)$ be the maximiser of the upper confidence bound in $A_{i,n}$ at time t. Note that the discretisation is fixed ahead of time and $b_{i,n,t}$ is deterministic given the data $\{(x_t, m_t, y_t)\}_{i=1}^{t-1}$ at time t. Now using the relaxation $x_t \in A_{i,n} \implies \varphi_t(b_{i,n,t}) > \varphi_t([x_\star]_t) \implies \varphi_t^{(m)}(b_{i,n,t}) > f(x_\star) - 1/t^2$ and proceeding,

$$\begin{aligned}
\mathbb{P}(T_{n'}^{(>m)}(A_{i,n}) > u) \;\leq\; & \frac{1}{\xi_{\mathbf{A2}}}\mathbb{P}_{\mathcal{GP}}\big(\exists t : u+1 \leq t \leq n,\;\; \varphi_t^{(m)}(b_{i,n,t}) > f(x_\star) - 1/t^2 \;\;\wedge \\
& \qquad\qquad \beta_t^{1/2}\sigma_{t-1}^{(m)}(b_{i,n,t}) < \gamma^{(m)}\big) \qquad (4.59)\\
\leq\; & \frac{1}{\xi_{\mathbf{A2}}}\sum_{t=u+1}^{n'} \mathbb{P}_{\mathcal{GP}}\big(\mu_{t-1}^{(m)}(b_{i,n,t}) - f^{(m)}(b_{i,n,t}) > \Delta^{(m)}(b_{i,n,t}) - \beta_t^{1/2}\sigma_{t-1}^{(m)}(b_{i,n,t}) - 1/t^2 \;\;\wedge \\
& \qquad\qquad \beta_t^{1/2}\sigma_{t-1}^{(m)}(b_{i,n,t}) < \gamma^{(m)}\big) \\
\leq\; & \frac{1}{\xi_{\mathbf{A2}}}\sum_{t=u+1}^{n'} \mathbb{P}_{\mathcal{GP}}\big(\mu_{t-1}^{(m)}(b_{i,n,t}) - f^{(m)}(b_{i,n,t}) > 2\beta_t^{1/2}\sigma_{t-1}^{(m)}(b_{i,n,t}) - 1/t^2\big) \\
\leq\; & \frac{1}{\xi_{\mathbf{A2}}}\sum_{t=u+1}^{n'} \mathbb{P}_{Z\sim\mathcal{N}(0,1)}\left(Z > \beta_t^{1/2}\right) \;\leq\; \sum_{t=u+1}^{n'} \frac{1}{\xi_{\mathbf{A2}}}\frac{1}{2}\exp\left(\frac{-\beta_t}{2}\right) \\
\leq\; & \frac{1}{\xi_{\mathbf{A2}}}\frac{1}{2}\left(\frac{2\xi_{\mathbf{A2}}\delta}{M\pi^2}\right)\sum_{t=u+1}^{n'} t^{-2} \;\leq\; \frac{\delta}{M\pi^2}\frac{1}{u}
\end{aligned}$$

In the second step we have rearranged the terms and used the definition of $\Delta^{(m)}(x)$. In the third step, as $A_{i,n} \subset \overline{\mathcal{J}}^{(m)}_{\max(\tau,\rho\gamma)}$, we have $\Delta^{(m)}(b_{i,n,t}) > 3\gamma^{(m)} > 3\beta_t^{1/2}\sigma_{t-1}^{(m)}(b_{i,n,t})$. The last step bounds the sum by an integral. For the fourth step, we have used, $t > u \geq 1/(3\eta)^{2/3}$, $\beta_t > 2\log(M\pi^2t^2/2\delta) > (3/2)^2$, and $\sigma_{t-1}^{(m)}(b_{i,n,t}) > \eta/\sqrt{t}$ to conclude,

$$t > \frac{1}{(3\eta)^{2/3}} \implies \frac{3t^{3/2}}{2} > \frac{1}{2\eta} \implies t^{3/2}\beta_t^{1/2} > \frac{1}{2\eta} \implies 2\beta_t^{1/2}\sigma_{t-1}^{(m)} > \frac{1}{t^2}.$$

■

4.6 Proofs of Theoretical Results in Chapter 4.3

We will first state a formal version of Theorem 23. Recall from the main text where we stated that most evaluations at $z_\bullet$ are inside the following set $\mathcal{X}_\rho$.

$$\mathcal{X}_\rho = \{x \in \mathcal{X} : f(x_\star) - f(x) \leq 2\rho\sqrt{\kappa_0}\|\xi\|_\infty\}.$$

This is not entirely accurate as it hides a dilation that arises due to a covering argument in our proofs. Precisely, we will show that after n queries at any fidelity, BOCA will use most of the $z_\bullet$ evaluations in $\mathcal{X}_{\rho,n}$ defined below using $\mathcal{X}_\rho$.

$$\mathcal{X}_{\rho,n} = \left\{ x \in \mathcal{X} : \mathrm{B}_2\left(x, \sqrt{d}/n^{\alpha/2d}\right) \cap \mathcal{X}_{\rho,n} \neq \varnothing \right\} \tag{4.60}$$

Here $\mathrm{B}_2(x, \epsilon)$ is an L_2 ball of radius ϵ centred at x. $\mathcal{X}_{\rho,n}$ is a dilation of $\mathcal{X}_\rho$ by $\sqrt{d}/n^{\alpha/2d}$. Notice that for all $\alpha > 0$, as $n \to \infty$, $\mathcal{X}_{\rho,n}$ approaches $\mathcal{X}_\rho$ at a polynomial rate. We now state our main theorem below.

Theorem 38. *Let $\mathcal{Z} = [0,1]^p$ and $\mathcal{X} = [0,1]^d$. Let $g \sim \mathcal{GP}(\mathbf{0}, \kappa)$ where κ is of the form (4.20). Let $\phi_\mathcal{X}$ satisfy Assumption 1 with some constants $a, b > 0$. Pick $\delta \in (0,1)$ and run* BOCA *with*

$$\beta_t = 2\log\left(\frac{\pi^2 t^2}{2\delta}\right) + 4d\log(t) + \max\left\{0\,,\, 2d\log\left(brd\log\left(\frac{6ad}{\delta}\right)\right)\right\}.$$

Then, for all $\alpha \in (0,1)$ there exists ρ, Λ_0 such that with probability at least $1-\delta$ we have for all $\Lambda \geq \Lambda_0$,

$$S(\Lambda) \;\leq\; \sqrt{\frac{2C_1\beta_{2n_\Lambda}\Psi_{2n_\Lambda}(\mathcal{X}_{\rho,n})}{n_\Lambda}} \;+\; \sqrt{\frac{2C_1\beta_{2n_\Lambda}\Psi_{2n_\Lambda^\alpha}(\mathcal{X})}{n_\Lambda^{2-\alpha}}} \;+\; \frac{\pi^2}{6n_\Lambda}.$$

Here $C_1 = 8/\log(1+\eta^2)$ is a constant and $n_\Lambda = \lfloor \Lambda/\lambda(z_\bullet) \rfloor$. ρ satisfies $\rho > \rho_0 = \max\{2, 1 + \sqrt{(1+2/\alpha)/(1+d)}\}$.

In addition to the dilation, Theorem 23 in Chapter 4.3.2 also suppresses the constants and polylog terms. The next three subsections are devoted to proving the above theorem. In Chapter 4.6.1 we describe some discretisations for $\mathcal{Z}$ and $\mathcal{X}$ which we will use in our proofs. Chapter 4.6.2 gives some lemmas we will need and Chapter 4.6.3 gives the proof.

4.6.1 Set Up & Notation

Notation: Let $U \subset \mathcal{Z} \times \mathcal{X}$. $T_n(U)$ will denote the number of queries by BOCA at points $(z,x) \in U$ within n time steps. When $A \subset \mathcal{Z}$ and $B \subset \mathcal{X}$, we will overload notation to denote $T_n(A,B) = T_n(A \times B)$. For $z \in \mathcal{Z}$, $[> z]$ will denote the fidelities which are more expensive than z, i.e. $[> z] = \{z' \in \mathcal{Z} : \lambda(z') > \lambda(z)\}$.

We will require a fairly delicate set up before we can prove Theorem 38. Let $\alpha > 0$. All sets described in the rest of this subsection are defined with respect to α. First define

$$\tilde{\mathcal{H}}_n = \{(z, x) \in \mathcal{Z} \times \mathcal{X} : f(x_\star) - f(x) < 2\rho\beta_n^{1/2}\sqrt{\kappa_0}\xi(z)\},$$

where recall from (4.21), $\xi(z) = \sqrt{1 - \phi_{\mathcal{Z}}^2(\|z - z_\bullet\|)}$ is the information gap function. We next define $\mathcal{H}'_n$ to be an L_2 dilation of $\tilde{\mathcal{H}}_n$ in the $\mathcal{X}$ space, i.e.

$$\mathcal{H}'_n = \{(z, x) \in \mathcal{Z} \times \mathcal{X} : \mathrm{B}_2\big(x, \sqrt{d}/n^{\alpha/2d}\big) \cup \tilde{\mathcal{H}}_n \neq \varnothing\}.$$

Finally, we define $\mathcal{H}_n$ to be the intersection of $\mathcal{H}'_n$ with all fidelities satisfying the third condition in (4.24). That is,

$$\mathcal{H}_n = \mathcal{H}'_n \cap \Big\{(z, x) \in \mathcal{Z} \times \mathcal{X} : \xi(z) > \|\xi\|_\infty/\beta_n^{1/2}\Big\}. \tag{4.61}$$

In our proof we will use the second condition in (4.24) to control the number of queries in $\mathcal{H}_n$.

To control the number of queries outside $\mathcal{H}_n$ we first introduce a $\frac{\sqrt{d}}{2n^{\frac{\alpha}{2d}}}$-covering of the space $\mathcal{X}$ of size $n^{\alpha/2}$. If $\mathcal{X} = [0, 1]^d$, a sufficient covering would be an equally spaced grid having $n^{\frac{\alpha}{2d}}$ points per side. Let $\{a_{i,n}\}_{i=1}^{n^{\frac{\alpha}{2}}}$ be the points in the covering. $A_{i,n} \subset \mathcal{X}$ to be the points in $\mathcal{X}$ which are closest to $a_{i,n}$ in $\mathcal{X}$. Therefore $F_n = \{A_{i,n}\}_{i=1}^{n^{\frac{\alpha}{2}}}$ is a partition of $\mathcal{X}$.

Now define $Q_t : 2^{\mathcal{X}} \to 2^{\mathcal{Z}}$ to be the following function which maps subsets of $\mathcal{X}$ to subsets of $\mathcal{Z}$.

$$Q_t(A) = \Big\{z \in \mathcal{Z} : \ \forall\, x \in A, \quad f(x_\star) - f(x) \geq 2\rho\beta_t^{1/2}\sqrt{\kappa_0}\xi(z)\Big\}. \tag{4.62}$$

That is, Q_t maps $A \subset \mathcal{X}$ to fidelities where the information gap ξ is smaller than $(f(x_\star) - f(x))/(2\rho\beta_t^{1/2})$ for all $x \in A$. Next we define $\theta_t : 2^{\mathcal{X}} \to \mathcal{Z}$, to be the cheapest fidelity in $Q_t(A)$ for a subset $A \in \mathcal{X}$.

$$\theta_t(A) \ = \ \operatorname*{arginf}_{z \in Q_t(A)} \lambda(z). \tag{4.63}$$

We will see that BOCA will not query inside an $A_{i,n} \in F_n$ at fidelities larger than $\theta_t(A_{i,n})$ too many times (see Lemma 42). That is, $T_n([> \theta_n(A_{i,n})], A_{i,n})$ will be small. We now define $\mathcal{F}_n$ as follows,

$$\mathcal{F}_n = \bigcup_{A_{i,n} \subset \mathcal{X} \setminus \mathcal{X}_{\rho,n}} [> \theta_n(A_{i,n})] \times A_{i,n}. \tag{4.64}$$

That is, we first choose $A_{i,n}$'s that are completely outside $\mathcal{X}_{\rho,n}$ and take their cross product with fidelities more expensive than $\theta_t(A_{i,n})$. By design of the above sets, and using the third condition in (4.24) we can bound the total number of queries as follows,

$$n = T_n(\mathcal{Z}, \mathcal{X}) \ \leq \ T_n(\{z_\bullet\}, \mathcal{X}_{\rho,n}) + T_n(\mathcal{F}_n) + T_n(\mathcal{H}_n)$$

We will show that the last two terms on the right hand side are small for BOCA and consequently, the first term will be large. But first, we establish a series of technical results which will be useful in proving theorem 38.

4.6.2 Some Technical Lemmas

The first lemma proves that the UCB φ_t in (4.23) upper bounds $f(x_t)$ on all the domain points $\{x_t\}_{t\geq 1}$ chosen for evaluation.

Lemma 39. *Let $\beta_t > 2\log(\pi^2 t^2/2\delta)$. Then, with probability $> 1-\delta/3$, we have*

$$\forall\, t \geq 1, \quad |f(x_t) - \mu_{t-1}(x_t)| \;\leq\; \beta_t^{1/2}\sigma_{t-1}(x_t).$$

Proof: This is a straightforward argument using Lemma 4 and the union bound. At $t \geq 1$,

$$\begin{aligned}\mathbb{P}\Big(|f(x) - \mu_{t-1}(x)| > \beta_t^{1/2}\sigma_{t-1}(x)\Big) &= \mathbb{E}\left[\mathbb{E}\left[|f(x) - \mu_{t-1}(x)| > \beta_t^{1/2}\sigma_{t-1}(x) \;\middle|\; \mathcal{D}_{t-1}\right]\right] \\ &= \mathbb{E}\left[\mathbb{P}_{Z\sim\mathcal{N}(0,1)}\left(|Z| > \beta_t^{1/2}\right)\right] \leq \exp\left(\frac{-\beta_t}{2}\right) = \frac{2\delta}{\pi^2 t^2}.\end{aligned}$$

In the first step we have conditioned w.r.t $\mathcal{D}_{t-1} = \{(z_i, x_i, y_i)\}_{i=1}^{t-1}$ which allows us to use Lemma 4 as $f(x)|\mathcal{D}_{t-1} \sim \mathcal{N}(\mu_{t-1}(x), \sigma_{t-1}^2(x))$. The statement follows via a union bound over all $t \geq 0$ and the fact that $\sum_t t^{-2} = \pi^2/6$. ■

Next we show that the GP sample paths are well behaved and that $\varphi_t(x)$ upper bounds $f(x)$ on a sufficiently dense subset at each time step. For this we use the following lemma.

Lemma 40. *Let β_t be as given in Theorem 38. Then for all t, there exists a discretisation G_t of $\mathcal{X}$ of size $(t^2 brd\sqrt{6ad/\delta})^d$ such that the following hold.*

- *Let $[x]$ be the closest point to $x \in \mathcal{X}$ in the discretisation. With probability $> 1-\delta/6$, we have*
$$\forall\, t \geq 1, \quad \forall\, x \in \mathcal{X}, \quad |f(x) - f([x]_t)| \leq 1/t^2.$$
- *With probability $> 1-\delta/3$, for all $t \geq 1$ and for all $a \in G_t$, $|f(a) - \mu_{t-1}(a)| \leq \beta_t^{1/2}\sigma_{t-1}(a)$.*

Proof: The first part of the proof, which we skip here, uses the regularity condition for $\phi_{\mathcal{X}}$ in Assumption 1 and mimics the argument in Lemmas 5.6, 5.7 of Srinivas et al. [235]. The second part mimics the proof of Lemma 39 and uses the fact that $\beta_t > 2\log(|G_t|\pi^2 t^2/2\delta)$. ■

The discretisation in the above lemma is different to the coverings introduced in Chapter 4.6.1. The next lemma is about the information gap function in (4.21).

Lemma 41. *Let $g \sim \mathcal{GP}(0, \kappa)$, $g : \mathcal{Z}\times\mathcal{X} \to \mathbb{R}$ and κ is of the form (4.20). Suppose we have s observations from g. Let $z \in \mathcal{Z}$ and $x \in \mathcal{X}$. Then $\tau_{t-1}(z, x) < \alpha$ implies $\sigma_{t-1}(x) < \alpha + \sqrt{\kappa_0}\xi(z)$.*

Proof: The proof uses the observation that for radial kernels, the maximum difference between the variances at two points u_1 and u_2 occurs when all s observations are at u_2 or vice versa. Now we use $u_1 = (z, x)$ and $u_2 = (z_\bullet, x)$ and apply Lemma 43 to obtain $\tau_{t-1}^2(z_\bullet, x) \leq \kappa_0(1 - \phi_{\mathcal{Z}}(\|z_\bullet - z\|))^2 + \frac{\eta^2/s}{1+\frac{\eta^2}{s\kappa_0}}$. However, As $\tau_{t-1}^2(z, x) = \frac{\eta^2/s}{1+\frac{\eta^2}{s\kappa_0}}$ when all observations are at (z, x) and noting that $\sigma_{t-1}^2(x) = \tau_{t-1}^2(z_\bullet, x)$, we have $\sigma_{t-1}^2(x) \leq \kappa_0(1 - \phi_{\mathcal{Z}}(\|z_\bullet - z\|))^2 + \tau_{t-1}^2(z, x)$. Since the above situation characterised the maximum difference between $\sigma_{t-1}^2(x)$ and $\tau_{t-1}^2(z, x)$, this

inequality is valid for any general observation set. The proof is completed using the elementary inequality $a^2 + b^2 \leq (a+b)^2$ for $a, b > 0$. ■

We are now ready to prove Theorem 38. The plan of attack is as follows. We will analyse BOCA after n time steps and bound the number of plays at fidelities $z \neq z_\bullet$ and outside $\mathcal{X}_{\rho,n}$ at $z_\bullet$. Then we will show that for sufficiently large Λ, the number of *random* plays N is bounded by $2n_\Lambda$ with high probability. Finally we use techniques from Srinivas et al. [235], specifically the maximum information gain, to control the simple regret. However, unlike them we will obtain a tighter bound as we can control the regret due to the sets $\mathcal{X}_{\rho,n}$ and $\mathcal{X} \setminus \mathcal{X}_{\rho,n}$ separately.

4.6.3 Proof of Main Result

Let $\alpha > 0$ be given. We invoke the sets $\mathcal{X}_{\rho,n}, \mathcal{H}_n, \mathcal{F}_n$ in equations (4.60), (4.61), (4.64) for the given α. The following lemma establishes that for any $A \subset \mathcal{X}$, we will not query inside A at fidelities larger than $\theta_t(A)$ (4.63) too many times. The proof is given in Chapter 4.6.3.

Lemma 42. *Let $A \subset \mathcal{X}$ which does not contain the optimum. Let ρ, β_t be as given in Theorem 38. Then for all $u > \max\{3, (2(\rho - \rho_0)\eta)^{-2/3}\}$, we have*

$$\mathbb{P}\Big(T_n([> \theta_t(A)], A) \; > \; u\Big) \; \leq \; \frac{\delta}{\pi^2} \frac{1}{u^{1+4/\alpha}}$$

To bound $T(\mathcal{F}_n)$, we will apply Lemma 42 with $u = n^{\alpha/2}$ on all $A_{i,n} \in F_n$ satisfying $A_{i,n} \subset \mathcal{X} \setminus \mathcal{X}_{\rho,n}$. Since $\mathcal{X}_\rho \subset \mathcal{X}_{\rho,n}$, $A_{i,n}$ does not contain the optimum. As $\mathcal{F}_n$ is the union of such sets (4.64), we have for all n (larger than a constant),

$$\begin{aligned}
\mathbb{P}(T(\mathcal{F}_n) > n^\alpha) \; &\leq \; \mathbb{P}\Big(\exists A_{i,n} \subset \mathcal{X} \setminus \mathcal{X}_{\rho,n}, \;\; T_n([> \theta_t(A_{i,n})], A_{i,n}) \; > \; n^{\alpha/2}\Big) \\
&\leq \sum_{\substack{A_{i,n} \in F_n \\ A_{i,n} \subset \mathcal{X} \setminus \mathcal{X}_{\rho,n}}} \mathbb{P}\Big(T_n([> \theta_t(A_{i,n})], A_{i,n}) \; > \; n^{\alpha/2}\Big) \leq |F_n| \frac{\delta}{\pi^2} \frac{1}{n^{\alpha/2+2}} \leq \frac{\delta}{\pi^2} \frac{1}{n^2}
\end{aligned}$$

Now applying the union bound over all n, we get $\mathbb{P}(\forall\, n \geq 1, \; T(\mathcal{F}_n) > n^\alpha) \; \leq \; \delta/6$.

Now we will bound the number of plays in $\mathcal{H}_n$ using the second condition in (4.24). We begin with the following Lemma. The proof mimics the argument in Lemma 11 of Kandasamy et al. [123] who prove a similar result for GPs defined on just the domain, i.e. $f \sim \mathcal{GP}(\mathbf{0}, \kappa)$ where $f : \mathcal{X} \to \mathbb{R}$.

Lemma 43. *Let $A \subset \mathcal{Z} \times \mathcal{X}$ and the L_2 diameter of A in $\mathcal{X}$ be $D_\mathcal{X}$ and that in $\mathcal{Z}$ be $D_\mathcal{Z}$. Suppose we have n evaluations of g of which s are in A. Then for any $(z, x) \in A$, the posterior variance τ'^2 satisfies,*

$$\tau'^2(z, x) \leq \kappa_0(1 - \phi^2_\mathcal{Z}(D_\mathcal{Z})\phi^2_\mathcal{X}(D_\mathcal{X})) + \frac{\eta^2}{s}.$$

Let $\lambda_r = \lambda_{min}/\lambda(z_\bullet)$ where $\lambda_{min} = \min_{z\in\mathcal{Z}} \lambda(z)$. If the maximum posterior variance in a certain region is smaller than $\gamma(z)$, then we will not query within that region by the second condition in (4.24). Further by the third condition, since we will only query at fidelities satisfying $\xi(z) > \|\xi\|_\infty/\beta_n^{1/2}$, it is sufficient to show that the posterior variance is bounded by $\kappa_0\|\xi\|_\infty^2\lambda_r^{2q}/\beta_n$ at time n to prove that we will not query again in that region. For this we can construct a covering of $\mathcal{H}_n$ such that $1 - \phi_{\mathcal{Z}}^2(D_{\mathcal{Z}})\phi_{\mathcal{X}}^2(D_{\mathcal{X}}) < \frac{1}{2}\|\xi\|_\infty^2\lambda_r^{2q}/\beta_n$. For any $A \subset \mathcal{Z}\times\mathcal{X}$, the covering number, which we denote $\Omega_n(A)$ of this construction will typically be poly-logarithmic in n (See Remark 4 below). Now if there are $\frac{2\beta_n\eta^2}{\lambda_r^{2q}\|\xi\|_\infty^2\kappa_0} + 1$ queries inside a ball in this covering, the posterior variance, by Lemma 43 will be smaller than $\kappa_0\|\xi\|_\infty^2\lambda_r^{2q}/\beta_n$. Therefore, we will not query any further inside this ball. Hence, the total number of queries in $\mathcal{H}_n$ is $T_n(\mathcal{H}_n) \leq C_2\Omega_n(\mathcal{H}_n)\frac{\beta_n}{\lambda_r^{2q}} \leq C_3\text{vol}(\mathcal{H}_n)\frac{\text{polylog}(n)}{\text{poly}(\lambda_r)}$ for appropriate constants C_2, C_3. (Also see Remark 5).

Next, we will argue that the number of queries for sufficiently large Λ, is bounded by $n_\Lambda/2$ where, recall $n_\Lambda = \lfloor\Lambda/\lambda(z_\bullet)\rfloor$. This simply follows from the bounds we have for $T_n(\mathcal{F}_n)$ and $T_n(\mathcal{H}_n)$.

$$T_n(\mathcal{Z}\setminus\{z_\bullet\},\mathcal{X}) \leq T_n(\mathcal{F}_n) + T_n(\mathcal{H}_n) \leq n^\alpha + \mathcal{O}(\text{polylog}(n)).$$

Since the right hand side is sub-linear in n, we can find n_0 such that for all n_0, $n/2$ is larger than the right hand side. Therefore for all $n \geq n_0$, $T_n(\{z_\bullet\},\mathcal{X}) > n/2$. Since our bounds hold with probability $> 1-\delta$ for all n we can invert the above inequality to bound N, the random number of queries after capital Λ. We have $N \leq 2\Lambda/\lambda(z_\bullet)$. We only need to make sure that $N \geq n_0$ which can be guaranteed if $\Lambda > \Lambda_0 = n_0\lambda(z_\bullet)$.

The final step of the proof is to bound the simple regret after n time steps in BOCA. This uses techniques that are now standard in GP bandit optimisation, so we only provide an outline. We will need the following Lemma, whose proof is given in Chapter 4.6.3.

Lemma 44. *Assume that we have queried g at n points, $(z_t, x_t)_{t=1}^n$ of which s points are in $\{z_\bullet\}\times A$ for any $A \subset \mathcal{X}$. Let σ_{t-1} denote the posterior variance of f at time t, i.e. after $t-1$ queries. Then, $\sum_{x_t\in A, z_t=z_\bullet}\sigma_{t-1}^2(x_t) \leq \frac{2}{\log(1+\eta^{-2})}\Psi_s(A)$. Here $\Psi_s(A)$ is the MIG of $\phi_{\mathcal{X}}$ after s queries to A as given in Definition 1.*

We now define the quantity R_n below. Readers familiar with the GP bandit literature might see that it is similar to the notion of cumulative regret, but we only consider queries at $z_\bullet$.

$$R_n = \sum_{\substack{t=1\\ z_t=z_\bullet}}^n f(x_\star) - f(x_t) \quad = \quad \sum_{\substack{z_t=z_\bullet\\ x_t\in\mathcal{X}_{\rho,n}}} f(x_\star) - f(x_t) \quad + \quad \sum_{\substack{z_t=z_\bullet\\ x_t\notin\mathcal{X}_{\rho,n}}} f(x_\star) - f(x_t). \tag{4.65}$$

For any $A \subset \mathcal{X}$ we can use Lemmas 39, 40, and 44 and the Cauchy Schwartz inequality to obtain,

$$\sum_{\substack{z_t=z_\bullet\\ x_t\in A}} f(x_\star) - f(x_t) \leq \sqrt{C_1 T_n(z_\bullet, A)\beta_n\Psi_{T_n(z_\bullet,A)}(A)} + \sum_{\substack{z_t=z_\bullet\\ x_t\in A}}\frac{1}{t^2}. \tag{4.66}$$

For the first term in (4.65), we set $A = \mathcal{X}_{\rho,n}$ in (4.66) and use the trivial bound $T_n(z_\bullet, \mathcal{X}_{\rho,n}) \leq n$. For the second term we note that $\{z_\bullet\}\times(\mathcal{X}\setminus\mathcal{X}_{\rho,n}) \subset \mathcal{F}_n$ and hence, $T_n(z_\bullet, \mathcal{X}\setminus\mathcal{X}_{\rho,n}) \leq T_n(\mathcal{F}_n) \leq$

n^α. As $A \subset B \implies \Psi_n(A) \leq \Psi_n(B)$, we have $R_n \leq \sqrt{C_1 n \beta_n \Psi_n(\mathcal{X}_{\rho,n})} + \sqrt{C_1 n^\alpha \beta_n \Psi_{n^\alpha}(\mathcal{X})} + \pi^2/6$. Now, using the fact that $N \leq 2n_\Lambda$ for large enough N we have,

$$R_N \leq \sqrt{2C_1 n_\Lambda \beta_{2n_\Lambda} \Psi_{2n_\Lambda}(\mathcal{X}_{\rho,n})} \; + \; \sqrt{2^\alpha C_1 n_\Lambda^\alpha \beta_{2n_\Lambda} \Psi_{2n_\Lambda^\alpha}(\mathcal{X})} \; + \; \frac{\pi^2}{6}.$$

The theorem now follows from the fact that $S(\Lambda) \leq \frac{1}{N} R_N$ by definition and that $N \geq n_\Lambda$. The failure instances arise out of Lemmas 39, 40 and the bound on $T_n(\mathcal{F}_n)$, the summation of whose probabilities are bounded by δ. ■

Remark 4 (Construction of covering for the SE kernel)**.** We demonstrate that such a construction is always possible using the SE kernel. Using the inequality $e^{-x} \geq 1 - x$ for $x > 0$ we have,

$$1 - \phi_{\mathcal{X}}^2(D_{\mathcal{X}})\phi_{\mathcal{Z}}^2(D_{\mathcal{Z}}) < \frac{D_{\mathcal{X}}^2}{h_{\mathcal{X}}^2} + \frac{D_{\mathcal{Z}}^2}{h_{\mathcal{Z}}^2}$$

where $D_{\mathcal{Z}}, D_{\mathcal{X}}$ will be the L_2 diameters of the balls in the covering. Now let $h = \min\{h_{\mathcal{Z}}, h_{\mathcal{X}}\}$ and choose

$$D_{\mathcal{X}} = D_{\mathcal{Z}} = \frac{h}{2} \frac{\|\xi\|_\infty}{\beta_n^{1/2}} \lambda_r^q,$$

via which we have $1 - \phi_{\mathcal{Z}}^2(z)\phi_{\mathcal{X}}^2(x) < \frac{1}{2}\xi(\sqrt{p})^2 \lambda_r^{2q}/\beta_n$ as stated in the proof. Noting that $\beta_n \asymp \log(n)$, using standard results on covering numbers, we can show that the size of this covering will be $\log(n)^{\frac{d+p}{2}}/\lambda_r^{q(d+p)}$. A similar argument is possible for Matérn kernels, but the exponent on $\log(n)$ will be worse.

Remark 5 (Choice of q for SE kernel)**.** From the arguments in our proof and Remark 4, we have that the number of plays in a set $S \subset (\mathcal{Z} \times \mathcal{X})$ is $T(S) \leq \text{vol}(S) \log(n)^{\frac{d+p+2}{2}} \left(\frac{\lambda(z_\bullet)}{\lambda_{min}}\right)^{q(p+d+2)}$. However, we chose to work work λ_{min} mostly to simplify the proof. It is not hard to see that for $A \subset \mathcal{X}$ and $B \subset \mathcal{Z}$ if $\lambda(z) \approx \lambda'$ for all $z \in B$, then $T_n(B, A) \approx \text{vol}(B \times A) \log(n)^{\frac{d+p+2}{2}} \left(\frac{\lambda(z_\bullet)}{\lambda'}\right)^{q(p+d+2)}$. As the capital spent in this region is $\lambda' T_n(A, B)$, by picking $q = 1/(p + d + 2)$ we ensure that the capital expended for a certain $A \subset \mathcal{X}$ at all fidelities is roughly the same, i.e. for any A, the capital density in fidelities z such that $\lambda(z) < \lambda(\theta_t(A))$ will be roughly the same. Recall that in Chapter 4.4, we showed that doing so achieves a nearly minimax optimal strategy for cumulative regret in K-armed bandits. While it is not clear that this is the best strategy for optimisation under GP assumptions, it did reasonably well in our experiments. We leave it to future work to resolve this.

Proof of Lemma 42

For brevity, we will denote $\theta = \theta_t(A)$. We will invoke the discretisation G_t used in Lemma 40 via which we have $\varphi_t([x_\star]_t) \geq f(x_\star) - 1/t^2$ for all $t \geq 1$. Let $b = \text{argmax}_{x \in A} \varphi_t(x)$ be the maximiser of the upper confidence bound φ_t in A at time t. Now note that, $x_t \in A \implies \varphi_t(b) > \varphi_t([x_\star]_t) \implies \varphi_t(b) > f(x_\star) - 1/t^2$. We therefore have,

$$\mathbb{P}\big(T_n([> \theta], A) > u\big) \leq \mathbb{P}\big(\exists t : u + 1 \leq t \leq n, \;\; \varphi_t(b) > f(x_\star) - 1/t^2 \;\; \wedge \;\; \tau_{t-1}(\theta, b) < \gamma(\theta)\big)$$

$$\leq \sum_{t=u+1}^{n} \mathbb{P}\big(\mu_{t-1}(b) - f(b) > f(x_\star) - f(b) - \beta_t^{1/2}\sigma_{t-1}(b) - 1/t^2 \\ \wedge \quad \tau_{t-1}(\theta, b) < \gamma(\theta)\big) \tag{4.67}$$

We now note that

$$\tau_{t-1}(\theta, b) < \gamma(\theta) \implies \sigma_{t-1}(b) < \gamma(\theta) + \sqrt{\kappa_0}\xi(\theta) \leq 2\sqrt{\kappa_0}\xi(\theta) \leq \frac{1}{\beta_t^{1/2}\rho}(f(x_\star) - f(b)).$$

The first step uses Lemma 41. The second step uses the fact that

$$\gamma(\theta) = \sqrt{\kappa_0}\xi(\theta)(\lambda(z)/\lambda(z_\bullet))^{1/(p+d+2)} \leq \sqrt{\kappa_0}\xi(\theta)$$

and the last step uses the definition of $Q_t(A)$ in (4.62) whereby we have $f(x_\star) - f(x) \geq 2\rho\beta_t^{1/2}\sqrt{\kappa_0}\xi(\theta)$. Now plugging this back into (4.67), we can bound each term in the summation by,

$$\mathbb{P}\big(\mu_{t-1}(b) - f(b) > (\rho-1)\beta_t^{1/2}\sigma_{t-1}(b) - 1/t^2\big) \leq \mathbb{P}_{Z\sim\mathcal{N}(0,1)}\left(Z > (\rho_0 - 1)\beta_t^{1/2}\right) \\ \leq \frac{1}{2}\exp\left(\frac{(\rho_0-1)^2}{2}\beta_t\right) \leq \frac{1}{2}\left(\frac{2\delta}{\pi^2}\right)^{(\rho_0-1)^2} t^{-(\rho_0-1)^2(2+2d)} \leq \frac{\delta}{\pi^2}t^{-(\rho_0-1)^2(2+2d)}. \tag{4.68}$$

In the first step we have used the following facts, $t > u \geq \max\{3,\ (2(\rho-\rho_0)\eta)^{-2/3}\}$, $\pi^2/2\delta > 1$ and $\sigma_{t-1}(b) > \eta/\sqrt{t}$ to conclude,

$$(\rho-\rho_0)\frac{\eta\sqrt{4\log(t)}}{\sqrt{t}} > \frac{1}{t^2} \implies (\rho-\rho_0)\cdot\sqrt{2\log\left(\frac{\pi^2t^2}{2\delta}\right)}\cdot\frac{\eta}{\sqrt{t}} > \frac{1}{t^2} \\ \implies (\rho-\rho_0)\beta_t^{1/2}\sigma_{t-1}(b) > \frac{1}{t^2}.$$

The second step of (4.68) uses Lemma 4, the third step uses the conditions on $\beta_t^{1/2}$ as given in theorem 38 and the last step uses the fact that $\pi^2/2\delta > 1$. Now plug (4.68) back into (4.67). The result follows by bounding the sum by an integral and noting that $\rho_0 > 2$ and $\rho_0 \geq 1 + \sqrt{(1+2/\alpha)/(1+d)}$. ■

Proof of Lemma 44

Let $A_s = \{u_1, u_2, \ldots, u_s\}$ be the queries in $\{z_\bullet\} \times A$ in the order they were queried. Now, assuming that we have queried g only inside $\{z_\bullet\} \times A$, denote by $\tilde{\sigma}_{t-1}(\cdot)$, the posterior standard deviation after $t-1$ such queries. Then,

$$\sum_{t:x_t\in A, z_t = z_\bullet} \sigma_{t-1}^2(x_t) \leq \sum_{t=1}^{s}\tilde{\sigma}_{t-1}^2(u_t) \leq \sum_{t=1}^{s}\eta^2\frac{\tilde{\sigma}_{t-1}^2(u_t)}{\eta^2} \leq \sum_{t=1}^{s}\frac{\log(1+\eta^{-2}\tilde{\sigma}_{t-1}^2(u_t))}{\log(1+\eta^{-2})} \\ \leq \frac{2}{\log(1+\eta^{-2})}I(y_{A_s}; f_{A_s}).$$

Queries outside $\{z_\bullet\} \times A$ will only decrease the variance of the GP so we can upper bound the first sum by the posterior variances of the GP with only the queries in $\{z_\bullet\} \times A$. The third step uses the inequality $u^2/v^2 \leq \log(1+u^2)/\log(1+v^2)$. The result follows from the fact that $\Psi_s(A)$ maximises the mutual information among all subsets of size s. ■

Chapter 5

Parallel Bandits

Traditional methods for bandits and Bayesian optimisation are studied in the sequential setting, where a decision maker needs to wait for the current evaluation (arm-pull) to finish before recommending the next evaluation. However, more often than not, in practical applications, one has the ability to make multiple evaluations in parallel. For example, in hyperparameter tuning, with modern computing infrastructures, we have the ability to evaluate several hundred hyperparameters in parallel. The training time for each hyperparameter is influenced by a myriad of factors, including contention on shared compute resources and the actual hyperparameter choices, so it typically exhibits significant variability. Our goal is to find a set of hyperparameters that achieve low validation error, in a short amount of time. Similarly, in drug discovery, each x characterises a candidate drug and $f(x)$ measures various qualities such as the potency, specificity, and solubility of the drug via an expensive *in vitro* or *in vivo* test. Today, high throughput screening equipment can test several thousand candidate drugs at the same time.

Addressing this problem in the above and several other applications with parallel function evaluations, we design and analyse new algorithms for parallel bandits and Bayesian optimisation. Our algorithms are synchronous and asynchronous parallel versions of Thompson Sampling (TS), which we call synTS and asyTS , respectively. These algorithms are conceptually simple, easy to implement, and also scale to large number of parallel evaluations. In a departure from prior work on parallel BO, we explicitly model evaluation times and study the relationship between optimisation performance and time, in addition to the more standard relationship between optimisation and the number of function evaluations. Our main contributions in this chapter are,

1. A theoretical analysis demonstrating that both synTS and asyTS making n evaluations distributed among M workers is almost as good as if the n evaluations were made in sequence.
2. We introduce and analyse simple regret with time as a resource in parallel settings. Under this definition, asyTS outperforms the synchronous and sequential versions up to constant factors.
3. Empirically, we demonstrate that TS significantly outperforms existing methods for parallel BO in both the synchronous and asynchronous settings on several synthetic problems and a hyperparameter tuning task.

Related Work

There has been a flurry of recent activity in parallelising BO [43, 52, 73, 78, 107, 134, 221, 257, 261, 262, 270]. In comparison to this prior work, our approach enjoys one or more of the following advantages.

1. **Asynchronicity:** The majority of work on parallel BO are in the synchronous (batch) setting. To our knowledge, only [73, 107, 257] can handle asynchronous parallelisation.

2. **Theoretical underpinnings:** Most methods for parallel BO do not come with theoretical guarantees, with the exception of some work using UCB techniques [43, 52, 134]. Crucially, to the best of our knowledge, no theoretical guarantees are available for asynchronous methods.

3. **Conceptual simplicity:** *All* of the above methods either introduce additional hyperparameters and/or ancillary computational subroutines. Some methods become computationally prohibitive when there are a large number of workers and must resort to approximations [107, 221, 257, 270]. In contrast, our approach is conceptually simple – a direct adaptation of the sequential TS algorithm to the parallel setting. Hence, it is robust in practice, especially with a large number of workers. Further, unlike existing methods, its computationally complexity does not increase with M and is exactly the same as the sequential version.

We mention that parallelised versions of TS have been explored to varying degrees in some applied domains of bandit and reinforcement learning research [96, 104, 186]. However, to our knowledge, we are the first to theoretically analyse parallel TS. More importantly, we are also the first to develop and study TS in an asynchronous parallel setting. Besides BO, there has been a line of work on online learning with delayed feedback (as we have in the parallel setting) [114, 200]. In addition, Jun et al. [116] study a best-arm identification problem when queries are issued in batches. But these papers only consider finite decision sets and do not model evaluation times to study trade-offs when time is viewed as the primary resource.

5.1 Preliminaries

Recall that our goal is to maximise an unknown function $f : \mathcal{X} \to \mathbb{R}$, by repeatedly obtaining noisy evaluations of f. As before, for simplicity of exposition we assume a compact Euclidean domain $\mathcal{X} \subset \mathbb{R}^d$. Similarly, we will assume that f is a sample from a Gaussian process [203] and that the noise, $\epsilon \sim \mathcal{N}(0, \eta^2)$, is i.i.d normal.

Our goal is to find the maximiser $x_\star = \operatorname{argmax}_{x \in \mathcal{X}} f(x)$ of f through repeated evaluations. In the BO literature, this is typically framed as minimising the *simple regret*, which is the difference between the optimal value $f(x_\star)$ and the best evaluation of the algorithm. Since f is a random quantity, so is its optimal value and hence the simple regret. This motivates studying the *Bayes simple regret* $\mathbb{E}[S_n]$, which is the expectation of the simple regret. Formally, we define the Bayes'

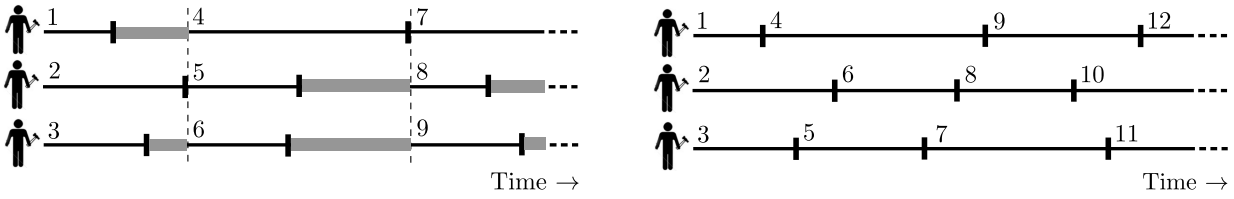


Figure 5.1: An illustration of the synchronous (left) and asynchronous (right) settings using $M = 3$ workers. The short vertical lines indicate when a worker finished its last evaluation. In the synchronous setting the grey shaded regions indicate idle time after a worker finishes its job. The horizontal location of a number indicates when the worker started its next evaluation while the number itself denotes the order in which the evaluation was dispatched by the algorithm.

simple regret as follows,

$$\mathbb{E}[S_n] \;=\; \mathbb{E}\Big[f(x_\star) - \max_{j=1,\ldots,n} f(x_j)\Big]. \tag{5.1}$$

The expectation in $\mathbb{E}[S_n]$ is with respect to the prior $f \sim \mathcal{GP}(\mathbf{0}, \kappa)$, the noise in the observations $\epsilon_j \sim \mathcal{N}(0, \eta^2)$, and any randomness of the algorithm. We focus on simple regret here mostly to simplify exposition; our proof also applies for the cumulative regret (1.1).

In many applications of BO, including hyperparameter tuning, the time required to evaluate the function is the dominant cost, and we are most interested in maximising f in a short period of time. Moreover, there is often considerable variability in the time required for different evaluations, caused by inherent differences between points in the domain, randomness of the environment, or other factors. For example, in the hyperparameter tuning application, unpredictable factors such as resource contention, initialisation, etc., may induce significant variability in evaluation times.

To adequately capture these settings, we model the time to complete an evaluation as a random variable, and measure performance in terms of the simple regret within a time budget, T. Specifically, letting $N = N(T)$ denote the (random) number of evaluations performed by an algorithm within time T, we define the simple regret S'_T as follows,

$$S'_T \;=\; \begin{cases} f(x_\star) - \max_{j \le N} f(x_j) & \text{if } N \ge 1 \\ \max_{x \in \mathcal{X}} |f(x_\star) - f(x)| & \text{otherwise} \end{cases}. \tag{5.2}$$

This definition is similar to (5.1), except, when an algorithm has not completed an evaluation yet, its simple regret is the worst possible value. We similarly define the Bayes' simple regret with time as the resource as $\mathbb{E}[S'_T]$, where the expectation now also includes the randomness in the evaluation times in addition to the three sources of randomness in $\mathbb{E}[S_n]$. In this work, we will model evaluation times as random variables independent from f; specifically we consider uniform, half-normal, or exponential random variables. While the model does not precisely capture all aspects of evaluation times observed in practice, we prefer it because (a) it is fairly general, (b) it leads to a clean algorithm and analysis, and (c) the resulting algorithm has good

performance on real applications, as we demonstrate in Chapter 5.4. Studying other models for the evaluation time is an intriguing question for future work and is discussed further in Chapter 9.

To our knowledge, all prior theoretical work for parallel BO [43, 52, 134], measures regret in terms of the total number of evaluations, i.e. $S_n, \mathbb{E}[S_n]$. However, explicitly modeling evaluation times and treating time as the main resource in the definition of regret is a better fit for applications and leads to new conclusions in the parallel setting.

Parallel BO: We are interested in parallel approaches for BO, where the algorithm has access to M workers that can evaluate f at different points in parallel. In this setup, we wish to differentiate between the synchronous and asynchronous settings, illustrated in Fig. 5.1. In the former, the algorithm issues a batch of M queries simultaneously, one per worker, and waits for all M evaluations to be completed before issuing the next batch. In contrast, in the asynchronous setting, a new evaluation may be issued as soon as a worker finishes its last job and becomes available. In the parallel setting, N in (5.2) will refer to the number of evaluations completed by *all* M workers.

Information accumulation vs worker utilisation: One of our goals in the theoretical analysis will be to quantify the trade-offs between information accumulation and worker utilisation in the sequential, synchronous parallel and asynchronous parallel settings. When comparing the three settings purely in terms of the number of evaluations, i.e. $\mathbb{E}[S_n]$, the parallel settings are naturally at a disadvantage: the sequential algorithm makes use of feedback from all its previous evaluations when issuing a query, whereas a parallel algorithm could be missing up to $M-1$ of them. As we will see however, for our TS algorithms, this difference is fairly small - the bounds for the parallel algorithms are only slightly worse than for sequential variants. The advantage in the parallel setting however, is that we will be able to complete more evaluations than a sequential version within an allotted time. One can make a similar argument to compare the synchronous and asynchronous settings. When issuing queries, a synchronous algorithm has more information about f, since all previous evaluations complete before a batch is selected, whereas asynchronous algorithms always issue queries with $M-1$ missing evaluations. For example, in Fig. 5.1, when dispatching the fourth job, the synchronous version uses results from the first three evaluations whereas the asynchronous version uses just the result of the first evaluation. However, in the synchronous setting, workers may sit idle for some time waiting for the other workers to finish. Foreshadowing our results in Theorem 48, when there is significant variability in evaluation times, worker utilisation is more important than information accumulation, and hence the asynchronous setting will enable better bounds on $\mathbb{E}[S'_T]$. Next, we present our algorithms.

5.2 Thompson Sampling for Parallel Bayesian Optimisation

A review of sequential TS: We briefly review Thompson sampling [246] for GPs from Chapter 2.2. At step j, TS samples x_j according to the posterior probability that it is the optimum. In GPs, this reduces to first drawing a sample g from the posterior for f conditioned on $\mathcal{D}_j$ and then

setting $x_j = \text{argmax}_x\, g(x)$ to be the maximiser of g. We then evaluate f at x_j. The resulting procedure is displayed in Algorithm 6.

Algorithm 6 seqTS from Thompson [246]

Require: Prior GP $\mathcal{GP}(\mathbf{0}, \kappa)$.
1: $\mathcal{D}_1 \leftarrow \varnothing, \quad \mathcal{GP}_1 \leftarrow \mathcal{GP}(\mathbf{0}, \kappa)$.
2: **for** $j = 1, 2, \ldots$ **do**
3: Sample $g \sim \mathcal{GP}_j$.
4: $x_j \leftarrow \text{argmax}_{x \in \mathcal{X}}\, g(x)$.
5: $y_j \leftarrow$ Query f at x_j.
6: $\mathcal{D}_{j+1} \leftarrow \mathcal{D}_j \cup \{(x_j, y_j)\}$.
7: Compute posterior $\mathcal{GP}_{j+1} = \mathcal{GP}(\mu_{\mathcal{D}_{j+1}}, \kappa_{\mathcal{D}_{j+1}})$ conditioned on $\mathcal{D}_{j+1}$. See (2.1).
8: **end for**

Asynchronous Parallel TS: For the asynchronously parallel setting, we propose a natural adaptation of the above algorithm. Precisely, when a worker finishes an evaluation, we update the posterior with the query-feedback pair, sample g from the posterior, and re-deploy the worker with an evaluation at $x_j = \text{argmax}_x\, g(x)$. The procedure, called asyTS, is displayed in Algorithm 7. In the first M steps, when at least one of the workers have not been assigned a job yet, the algorithm skips lines 6–19 and samples g from the prior GP, $\mathcal{GP}_1$, in line 6.

Algorithm 7 asyTS from Kandasamy et al. [129]

Require: Prior GP $\mathcal{GP}(\mathbf{0}, \kappa)$.
1: $\mathcal{D}_1 \leftarrow \varnothing, \quad \mathcal{GP}_1 \leftarrow \mathcal{GP}(\mathbf{0}, \kappa)$.
2: **for** $j = 1, 2, \ldots$ **do**
3: Wait for a worker to finish.
4: $\mathcal{D}_j \leftarrow \mathcal{D}_{j-1} \cup \{(x', y')\}$ where (x', y') are the worker's previous query/observation.
5: Compute posterior $\mathcal{GP}_j = \mathcal{GP}(\mu_{\mathcal{D}_j}, \kappa_{\mathcal{D}_j})$.
6: Sample $g \sim \mathcal{GP}_j, \quad x_j \leftarrow \text{argmax}\, g(x)$.
7: Re-deploy worker to evaluate f at x_j.
8: **end for**

Synchronous Parallel TS: To illustrate comparisons, we also introduce a synchronous parallel version, synTS, which operates as follows. At any given time, we wait for all M workers to finish their evaluations. Then, we update the posterior with the M query-feedback pairs, draw m samples $\{g_m\}_{m=1}^M$ and re-deploy all workers at the maxima of these samples, i.e. worker m evaluates f at $x_{j+m} = \text{argmax}_x\, g_m(x)$. The procedure, called synTS, is displayed in Algorithm 8.

We wish to highlight the main methodological differences of our algorithms with prior work for parallel BO. Since existing methods select points using deterministic criteria such as UCB or EI, they need to *explicitly* enforce diversity of query points so as to prevent the algorithm from picking the same or similar points for all M workers. Consequently, such methods introduce additional hyperparameters and/or potentially expensive computational routines. In contrast, asyTS

Algorithm 8 synTS from Kandasamy et al. [129]

Require: Prior GP $\mathcal{GP}(\mathbf{0}, \kappa)$.

$\mathcal{D}_1 \leftarrow \varnothing, \quad \mathcal{GP}_1 \leftarrow \mathcal{GP}(\mathbf{0}, \kappa)$.

for $j = 1, M+1, 2M+1 \ldots$ **do**

  Wait for all workers to finish.

  $\mathcal{D}_j \leftarrow \mathcal{D}_{j-M} \cup \{(x'_m, y'_m)\}_{m=1}^{M}$ where (x'_m, y'_m) are worker m's query/observation.

  Compute posterior $\mathcal{GP}_j = \mathcal{GP}(\mu_{\mathcal{D}_j}, \kappa_{\mathcal{D}_j})$.

  Draw m samples $g_m \sim \mathcal{GP}_j$, for $m = 1, \ldots, M$. Let $x_{j+m} \leftarrow \operatorname{argmax} g_m(x)$.

  Re-deploy worker m to evaluate f at x_{j+m}.

end for

and synTS are essentially the same as their sequential counterpart and their computational complexity does not increase with M. In addition to this computational advantage, this conceptual simplicity results in robust empirical performance in practice. Our theoretical analysis shows that a straightforward application of TS works because its inherent randomness is sufficient to avoid redundant function evaluations when managing M workers in parallel. This phenomenon is confirmed by our experiments in Chapter 5.4, where we see that explicitly encouraging diversity does not improve the performance of asyTS . We demonstrate this empirically by constructing a variant asyHTS of asyTS which employs one such diversity scheme found in the literature. asyHTS performs either about the same as or slightly worse than asyTS in the many experiments we study in Chapter 5.4.

5.3 Theoretical Results

We now present our theoretical contributions. We analyse the performance of parallelised TS both with the number of evaluations n and the time budget T as the resource. In particular, we study how these rates change with the number of workers M and demonstrate that as M increases, while $\mathbb{E}[S_n]$ worsens slightly for the parallel settings when compared to the sequential setting, $\mathbb{E}[S'_T]$ can improve dramatically. We provide theorem statements here to convey key intuitions, with all formal statements and proofs deferred to Chapter 5.5. We use $\asymp, \lesssim$ to denote equality/inequality up to constant factors that are common across all theorem statements.

Our first goal is to compare the simple regret $\mathbb{E}[S_n]$ after n evaluations for synTS and asyTS with that of seqTS. Recall from Theorem 2, we have for seqTS, $\mathbb{E}[S_n] \lesssim \sqrt{\Psi_n \log(n)/n}$.

The first theorem of this chapter, presented below in Theorem 45, bounds $\mathbb{E}[S_n]$ for synTS.

Theorem 45 (Informal. $\mathbb{E}[S_n]$ for synTS)**.** *Let* $f \sim \mathcal{GP}(\mathbf{0}, \kappa)$. *Then, for* synTS*,*

$$\mathbb{E}[S_n] \lesssim \frac{M\sqrt{\log(M)}}{n} + \sqrt{\frac{\Psi_n \log(n+M)}{n}}.$$

A comparison of the sequential and synchronously parallel bounds for TS reveals that, for reasons explained before in Chapter 5.1, seqTS outperforms synTS purely in terms of the number of evaluations n. However, for large n, the first term in the bound for synTS vanishes faster than the latter, and the dependence on M in the latter term is insignificant when $n \gg M$. Hence, the difference between the sequential and synchronous parallel algorithms is small and negligible for large n. We also note that the leading constant for the second term is the same as that in Theorem 2. This implies a powerful conclusion: synTS with M parallel workers is almost as good as the sequential version with as many evaluations.

To present the results for the asynchronous setting, we introduce the following quantity ξ_M, which bounds the information we can gain about f from the evaluations in progress. Assume that we have completed n evaluations to f at the points in $\mathcal{D}_n$ and that there are q evaluations in process at points in A_q. That is $\mathcal{D}_n, A_q \subset \mathcal{X}$, $|\mathcal{D}_n| = n$ and $|A_q| = q < M$. Then $\xi_M > 0$ satisfies the following for all $n \geq 1$,

$$\max_{A_q \subset \mathcal{X}, |A_q| < M} I(f; y_{A_q} | y_{\mathcal{D}_n}) \leq \frac{1}{2} \log(\xi_M). \tag{5.3}$$

Our next result is a bound on $\mathbb{E}[S_n]$ for asyTS.

Theorem 46 (Informal. $\mathbb{E}[S_n]$ for asyTS)**.** *Let $f \sim \mathcal{GP}(\mathbf{0}, \kappa)$. Then, for* asyTS*,*

$$\mathbb{E}[S_n] \lesssim \sqrt{\frac{\xi_M \Psi_n \log(n)}{n}}.$$

Unfortunately, this bound depends on ξ_M which can be quite large under general conditions. However, ξ_M is a well studied quantity in the GP literature; precisely, Desautels et al. [52], Krause et al. [144] show that ξ_M can be bounded by a kernel dependent constant C_κ by initially querying f using an uncertainty sampling procedure for γ_M samples. This sampling procedure, which iteratively samples the points with the largest variance in the GP, is asynchronously parallelisable. Desautels et al. [52] shows that for the Matérn kernel, with $\gamma_M \asymp \text{poly}(M)$, this procedure guarantees $C_\kappa \leq e^e$, and for the SE kernel, with $\gamma_M \asymp M\text{polylog}(M)$, we can achieve any constant $C_\kappa > 1$, depending on the order of the polylog term. These values for C_κ, γ_M are not absolute – by picking a larger γ_M we can achieve smaller C_κ. In all cases however, γ_M is at most polynomial in M and does not depend on n. We provide more details on the initialisation scheme along with its theoretical properties in Chapter 5.5.2. By initialising asyTS with this sampling scheme, we obtain the bound below.

Corollary 47 (Informal. $\mathbb{E}[S_n]$ for asyTS after initialisation)**.** *Let $f \sim \mathcal{GP}(\mathbf{0}, \kappa)$. By first initialising* asyTS *with an uncertainty sampling scheme [52, 144], we have,*

$$\mathbb{E}[S_n] \lesssim \frac{\gamma_M}{n} + \sqrt{\frac{C_\kappa \Psi_n \log(n)}{n}}.$$

Distribution	pdf $p(x)$	seqTS	synTS	asyTS
Unif(a,b)	$\frac{1}{b-a}$ for $x \in (a,b)$	$n_{\text{seq}} = \frac{2T}{b+a}$	$n_{\text{syn}} = M\frac{T(M+1)}{a+bM}$	$n_{\text{asy}} = Mn_{\text{seq}}$ $(> n_{\text{syn}})$
$\mathcal{HN}(\zeta^2)$	$\frac{\sqrt{2}}{\zeta\sqrt{\pi}} e^{-\frac{x^2}{2\zeta^2}}$ for $x > 0$	$n_{\text{seq}} = \frac{T\sqrt{\pi}}{\zeta\sqrt{2}}$	$n_{\text{syn}} \asymp \frac{Mn_{\text{seq}}}{\sqrt{\log(M)}}$	$n_{\text{asy}} = Mn_{\text{seq}}$
Exp(λ)	$\lambda e^{-\lambda x}$ for $x > 0$	$n_{\text{seq}} = \lambda T$	$n_{\text{syn}} \asymp \frac{Mn_{\text{seq}}}{\log(M)}$	$n_{\text{asy}} = Mn_{\text{seq}}$

Table 5.1: Descriptions of the random delay models analysed for the synchronous and asynchronous parallel set ups. The second column shows the probability density functions $p(x)$ for the uniform Unif(a,b), half-normal $\mathcal{HN}(\zeta^2)$, and exponential Exp(λ) distributions. The subsequent columns show the expected number of evaluations $n_{\text{seq}}, n_{\text{syn}}, n_{\text{asy}}$ for seqTS, synTS, and asyTS respectively with M workers. synTS always completes fewer evaluations than asyTS. For example, in the exponential case, the difference could be a $\log(M)$ factor.

Despite the dependence on the initialisation scheme and the constant term C_κ, Corollary 47 is encouraging: since the first term in the bound vanishes faster than the latter, up to constant factors, asyTS with M parallel workers is almost as good as seqTS.

That said, we believe that bounds similar to Theorem 45 should be obtainable for asyTS without the additional constant C_κ and without the initialisation scheme. For instance, asyTS performs very well in our experiments even though we do not use this initialisation scheme. We leave it to future work to resolve this gap.

Now that we have bounds on the regret as a function of the number of evaluations, we can turn to bounding $\mathbb{E}[S'_T]$, the simple regret with time as the main resource. For this, we consider three different random distribution models for the time to complete a function evaluation: uniform, half-normal, and exponential. We choose these three distributions since they exhibit three different notions of tail decay, namely bounded, sub-Gaussian, and sub-exponential[1]. Table 5.1 describes these distributions and states the expected number of evaluations $n_{\text{seq}}, n_{\text{syn}}, n_{\text{asy}}$ for seqTS, synTS, asyTS respectively with M workers in time T. The final theoretical result of this chapter, presented below, bounds $\mathbb{E}[S'_T]$ for the Thompson sampling variants.

Theorem 48 (Informal, Simple regret with time for TS)**.** *Let $f \sim \mathcal{GP}(0, \kappa)$ and assume that for* asyTS*, ξ_M is bounded by C_κ after suitable initialisation. Assume that the times taken for an evaluation are i.i.d random variables with either uniform, half-normal or exponential distributions. Let $n_{seq}, n_{syn}, n_{asy}$ be as given in Table 5.1. Then $n_{seq} \leq n_{syn} \leq n_{asy}$ and $\mathbb{E}[S'_T]$ can be upper bounded by the following terms for* seqTS*,* synTS*, and* asyTS*.*

$$\text{seqTS:} \quad \sqrt{\frac{\Psi_{n_{\text{seq}}} \log(n_{\text{seq}})}{n_{\text{seq}}}},$$

$$\text{synTS:} \quad \frac{M\sqrt{\log(M)}}{n_{\text{syn}}} + \sqrt{\frac{\Psi_{n_{\text{syn}}} \log(n_{\text{syn}} + M)}{n_{\text{syn}}}},$$

[1]While we study uniform, half-normal and exponential, analogous results for other distributions with similar tail behaviour are possible with the appropriate concentration inequalities. See Chapter 5.5.3.

$$\text{asyTS:} \quad \frac{\text{poly}(M)}{n_{\text{asy}}} + \sqrt{\frac{C_\kappa \Psi_{n_{\text{asy}}} \log(n_{\text{asy}})}{n_{\text{asy}}}}.$$

As the above bounds are decreasing with the number of evaluations and since $n_{\text{seq}} < n_{\text{syn}} < n_{\text{asy}}$, the bound for $\mathbb{E}[S'_T]$ shows the opposite trend to $\mathbb{E}[S_n]$: asyTS is better than synTS which is better than seqTS. While the difference between synTS and asyTS is only a constant factor for the uniform distribution, it grows with the number of workers M for heavier tailed distributions; $\sqrt{\log(M)}$ for the half-normal and $\log(M)$ for the exponential. Hence, as the number of workers M increases, asyTS becomes increasingly attractive when compared to synTS. Intuitively, when there is more variability in evaluations, workers may sit idle for longer in the synchronous setting and hence synTS will complete fewer evaluations than asyTS.

Synopsis: The take-aways of our theoretical analysis can be summarised as follows. Theorems 45 and 46 show that since the synchronous setting has more information than the asynchronous setting, it achieves a better bound for $\mathbb{E}[S_n]$. Therefore, if function evaluations deterministically take the same amount of time, the synchronous algorithm may be preferred. Further, in some applications, we are necessarily in the synchronous setting. For example, in pre-clinical drug discovery, high throughput screening equipment can test a few thousand compounds in parallel, but only in batches [102]. However, Theorem 48 contends that if there is significant variability in evaluation times, then it is prudent to be asynchronous despite the lack of information when compared to the synchronous setting.

5.4 Experiments

We compare parallelised TS with a comprehensive suite of parallel BO methods from the literature on a series of synthetic experiments and a hyperparameter tuning task on the CIFAR-10 dataset.

Methods: We compare asyTS to the following methods. *Synchronous Methods:* synRAND: synchronous random sampling, synTS: synchronous TS, synBUCB from [52], synUCBPE from [43]. *Asynchronous Methods:* asyRAND: asynchronous random sampling, asyHUCB: an asynchronous version of UCB with hallucinated observations [52, 73], asyUCB: asynchronous upper confidence bound [235], asyEI: asynchronous expected improvement [113], asyTS: asynchronous TS, asyHTS: asynchronous TS with hallucinated observations to explicitly encourage diversity. This last method is based on asyTS but bases the posterior on $\mathcal{D}_j \cup \{(x, \mu_{\mathcal{D}_j}(x))\}_{x \in F_j}$ in line 19 of Algorithm 7, where F_j are the points in evaluation by other workers at step j and $\mu_{\mathcal{D}_j}$ is the posterior mean conditioned on just $\mathcal{D}_j$; this preserves the mean of the GP, but shrinks the variance around the points in F_j. This method is inspired by [52, 73], who use such hallucinations for UCB/EI-type strategies so as to discourage picking points close to those that are already in evaluation. asyUCB and asyEI directly use the sequential UCB and EI criteria, since the asynchronous versions do not repeatedly pick the same point for all workers. asyHUCB adds hallucinated observations to encourage diversity and is similar to [73] and is also an asynchronous version of [52]. While there are other methods for parallel BO, many of them are either computationally quite

expensive and/or require tuning several hyperparameters which might affect performance. The BO implementation for the TS and other methods follow the guidelines outlined in Chapter 4.3.4.

Synthetic Experiments

We first present results on a suite of benchmarks for global optimisation. To better align with our theoretical analysis, we add Gaussian noise to the function value when querying. This makes the problem more challenging than standard global optimisation where evaluations are not noisy.

We present results on a series of global optimisation benchmarks with different values for the number of parallel workers M. The descriptions of these functions are available in, for e.g. [123]. To construct the high dimensional variants, we repeat the same function by cycling through different groups of coordinates and add them up. For e.g. the Hartmann12 function was constructed as $f(x_{1:12}) = g(x_{1:6}) + g(x_{7:12})$ where g is the Hartmann6 function. Similarly, for the Park2-16 function we used the Park2-function 4 times, for Hartmann18, we used Hartmann6 thrice, and for CurrinExp-14 we used the Currin-exponential function 7 times.

In these synthetic experiments, we model the evaluation "time" as a random variable that is drawn from either a uniform, half-normal, exponential, or Pareto[2] distribution. Each time a worker makes an evaluation, we also draw a sample from this time distribution and maintain a queue to simulate the different start and finish times for each evaluation. The results are presented in Figures 5.2, 5.3 where we plot the simple regret S'_T against (simulated) time T. The time distributions are indicated on the top of each figure. In all cases, the time distributions were constructed so that the expected time to complete one evaluation is 1 time unit. Therefore, for e.g. in the Hartmann6 problem, an asynchronous version would use roughly $12 \times 30 = 360$ evaluations while a synchronous version would use roughly $\frac{12\times30}{\log(8)} \approx 173$ evaluations.

In the CurrinExp, Park1, and Park2 experiments, all asynchronous methods perform roughly the same and outperform the synchronous methods. On most of the other problems, asyTS performs best among the asynchronous methods and synTS among the synchronous methods. asyHTS, which also uses hallucinated observations, performs about the same or slightly worse than asyTS , demonstrating that there is no need to explicitly encourage diversity in TS. It is worth emphasizing that the improvement of TS over other methods become larger as M increases (e.g. $M > 20$). We believe that the ability to scale robustly with the number of workers is primarily due to the conceptual simplicity of our approach.

In our final synthetic experiment, we corroborate the claims in Theorems 2, 45, and 46 by comparing the performance of seqTS, synTS, and asyTS in terms of the number of evaluations n on the Park1 function. The results, displayed in the first panel of Figure 5.2, confirm that when comparing solely in terms of n, the sequential version outperforms the parallel versions while synchronous does marginally better than asynchronous.

[2]A Pareto distribution with parameter k has a pdf which decays $p(x) \propto x^{-(k+1)}$.

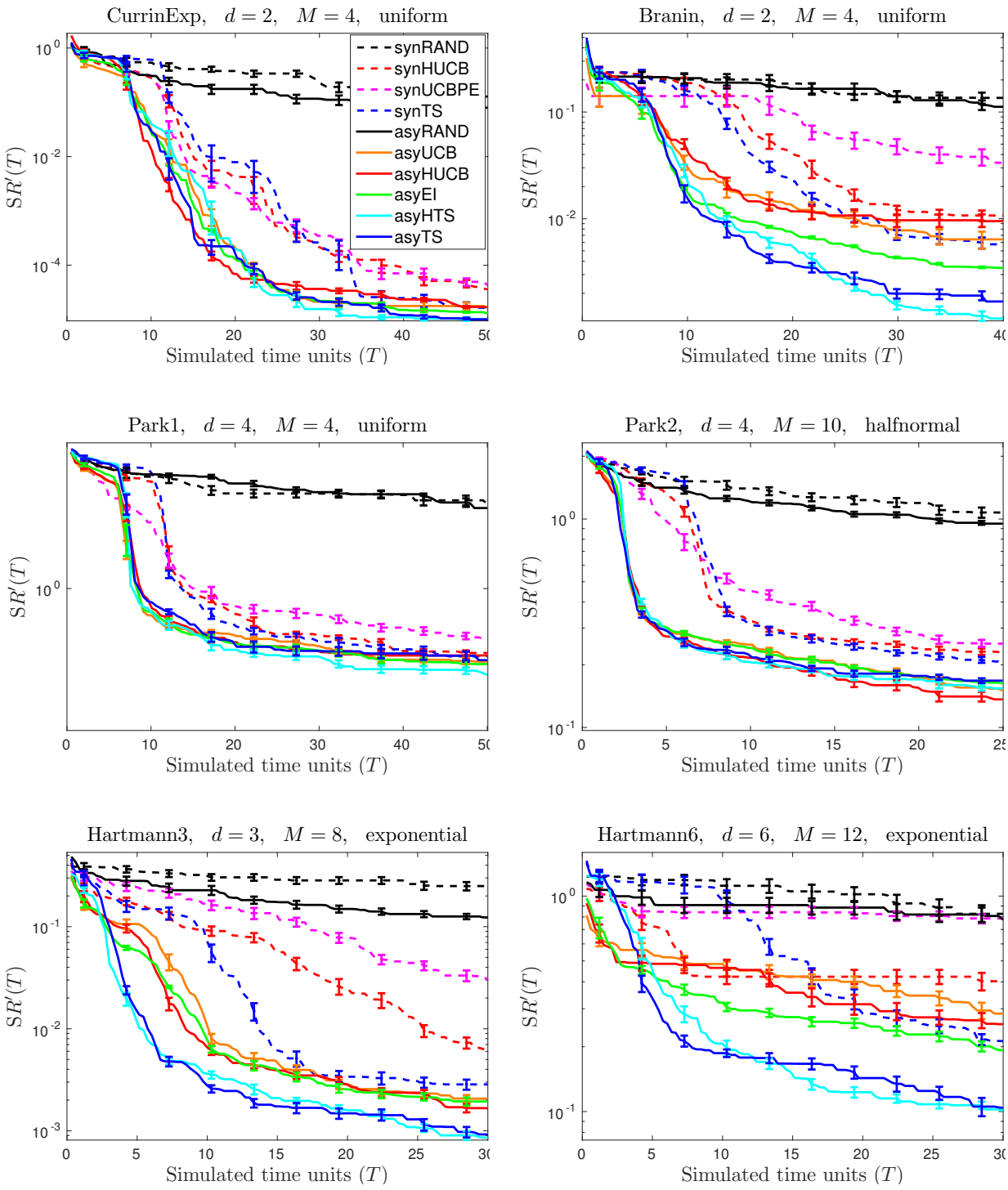


Figure 5.2: Results on the synthetic experiments. The title states the function used, its dimensionality d, the number of workers M and the distribution used for the time. All distributions were constructed so that the expected time for one evaluation was one time unit (for e.g., in the half normal $\mathcal{HN}(\zeta^2)$ in Table 5.1, we used $\zeta = \sqrt{\pi/2}$). The dotted lines depict synchronous methods while the solid lines are for asynchronous methods. The error bars indicate one standard error. All figures were averaged over at least 15 experiments.

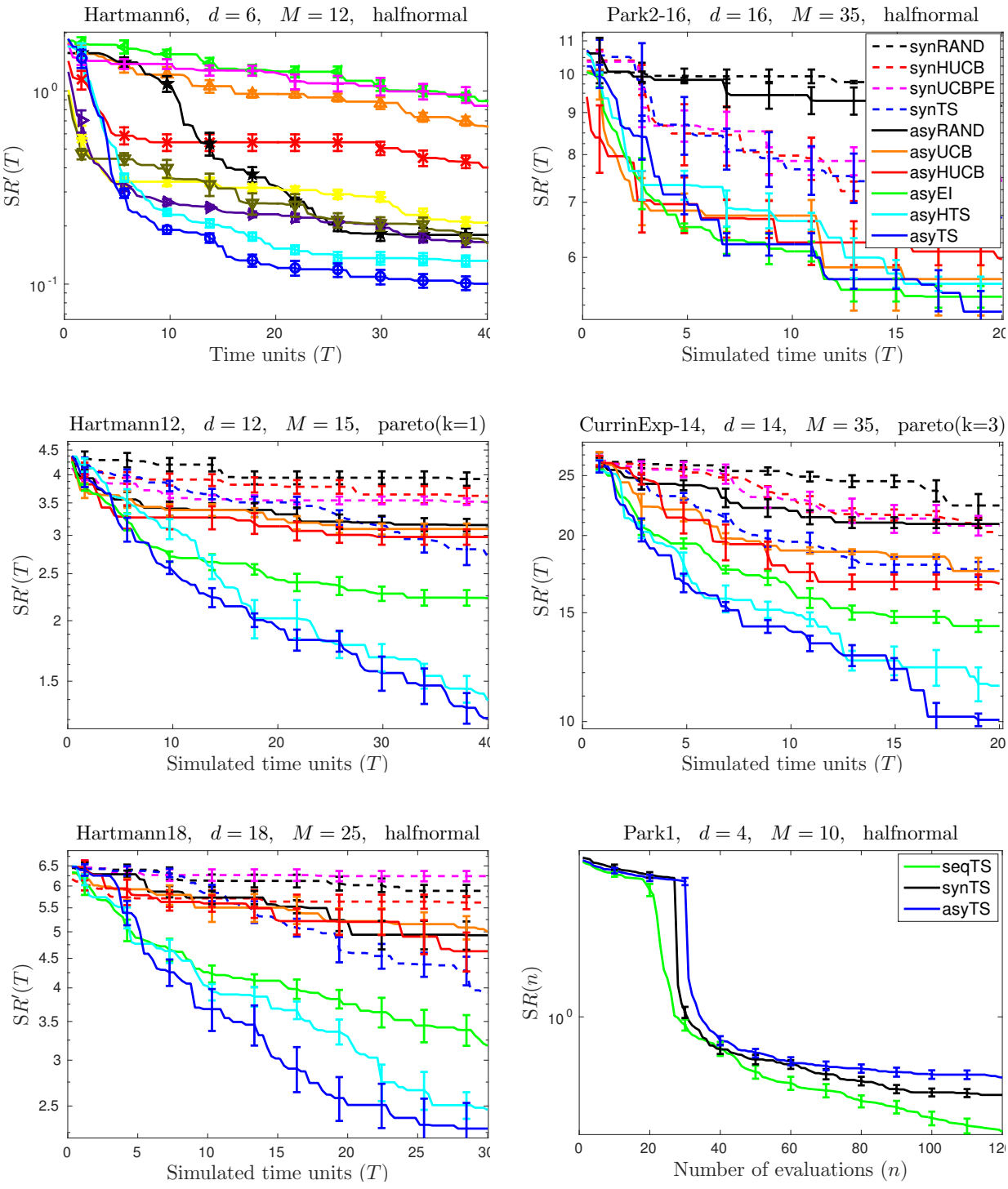


Figure 5.3: The first five panels are results on synthetic experiments. See caption under Figure 5.2 for more details. The last panel compares seqTS, synTS, and asyTS against the number of evaluations n.

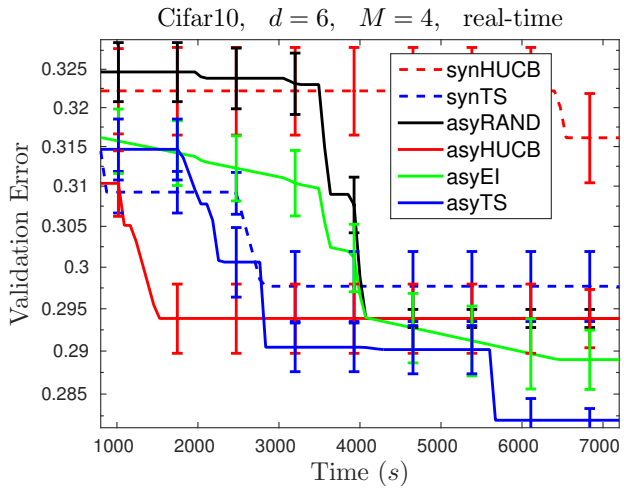


Figure 5.4: Cross validation results on the Cifar-10 experiment. The figure plots the best validation error (lower is better) vs time for each method. We have excluded some methods which performed poorly in the synthetic experiments due to the expensive nature of this experimental set up. The results presented are averaged over 9 experiments. Error bars indicate one standard error.

synBUCB	synTS	asyRAND	asyEI	asyHUCB	asyTS
25.63 ± 0.002	22.83 ± 1.01	23.93 ± 1.78	$\mathbf{19.49 \pm 0.21}$	22.14 ± 1.12	$\mathbf{19.53 \pm 0.11}$

Table 5.2: Test sesults on the Cifar-10 experiment. The table gives the test error (lower is better) on an independent test set of $10K$ images after training the best model chosen by each method for 80 epochs. The results presented are averaged over 9 experiments.

Image Classification on Cifar-10

We experiment with tuning hyperparameters of a 6 layer convolutional neural network on an image classification task on the Cifar-10 dataset [149]. The first 5 layers use convolutional filters while the last layer is a fully connected layer. We use skip connections as in the Resnet [91] between the first and third layers and then the third and fifth layers; when doing so, instead of just using an indentity transformation $\phi(x) = x$, we use a linear transformation $\phi(x) = Wx$ as the number of filters could be different at the beginning and end of a skip connection. The weights of W are also learned via back propagation as part of the training procedure. This modification to the Resnet was necessary in our set up as we are tuning the number of filters at each layer.

We tune the number of filters/neurons at each layer in the range $(16, 256)$ resulting in a 6 dimensional domain. Here, each function evaluation trains the model on 10K images for 20 epochs and computes the validation accuracy on a validation set of 10K images. Our implementation uses Tensorflow [2] and we use a parallel set up of $M = 4$ Titan X GPUs. The number of filters influences the training time which varied between ~ 4 to ~ 16 minutes depending on the size of the model. Note that this deviates from our theoretical analysis which treats function evaluation times as independent random variables, but it still introduces variability to evaluation times and demonstrates the robustness of our approach. Each method is given a budget of 2 hours to find the best model by optimising accuracy on a validation set. These evaluations are noisy since the result of each training procedure depends on the initial parameters of the network and other stochasticity in the training procedure. Since the true value of this function is unknown, we

simply report the best validation error achieved by each method. Due to the expensive nature of this experiment we only compare 6 of the above methods. The results are presented in Fig. 5.4. asyTS performs best on the validation error. The following are ranges for the number of evaluations for each method over 9 experiments:
- synchronous: synBUCB: 56 - 68, synTS: 56 - 68.
- asynchronous: asyRAND: 93 - 105, asyEI: 83 - 92, asyHUCB: 85 - 92, asyTS: 80 - 88.

While 20 epochs is insufficient to completely train a model, the validation error gives a good indication of how well the model would perform after sufficient training. In Table 5.2, we also give the error on a test set of 10K images after training the best model chosen by each algorithm to completion, i.e. for 80 epochs. asyTS and asyEI are able to recover the best models which achieve an accuracy of about 80%. While this falls short of state of the art results on Cifar-10 (for e.g. [91]), it is worth noting that we use only a small subset of the Cifar-10 dataset and a relatively small model. Nonetheless, it demonstrates the superiority of Thompson sampling as a method for parallelising BO over other baselines.

5.5 Proofs of Theoretical Results

5.5.1 Notation & Set up

We will require some set up in order to unify the analysis for the sequential, synchronously parallel and asynchronously parallel settings.

- The first is an indexing for function evaluations. This is illustrated for the synchronous and asynchronous parallel settings in Figure 5.1. In our analysis, index j or step j will refer to the j^{th} function evaluation dispatched by the algorithm. In the sequential setting this simply means that there were $j-1$ evaluations before the j^{th} . For synchronous strategies we index the first batch from $j = 1, \ldots, M$ and then the next batch $j = M+1, \ldots, 2M$ and so on as in Figure 5.1. In asynchronous settings, this might differ as each evaluation takes different amounts of time. For example, in Figure 5.1, the first worker finishes the $j = 1^{\text{st}}$ job and then starts the $j = 4^{\text{th}}$, while the second worker finishes the $j = 2^{\text{nd}}$ job and starts the $j = 6^{\text{th}}$.
- Next, we define $\mathcal{D}_j$ at step j of the algorithm to be the query-observation pairs (x_k, y_k) for function evaluations completed by step j. In the sequential setting $\mathcal{D}_j = \{(x_k, y_k) : k \in \{1, \ldots, j-1\}\}$ for all j. For the synchronous setting in Figure 5.1, $\mathcal{D}_1 = \mathcal{D}_2 = \mathcal{D}_3 = \varnothing$, $\mathcal{D}_4 = \mathcal{D}_5 = \mathcal{D}_6 = \{(x_k, y_k) : k \in \{1, 2, 3\}\}$, $\mathcal{D}_7 = \mathcal{D}_8 = \mathcal{D}_9 = \{(x_k, y_k) : k \in \{1, 2, 3, 4, 5, 6\}\}$ etc. Similarly, for the asynchronous setting, $\mathcal{D}_1 = \mathcal{D}_2 = \mathcal{D}_3 = \varnothing$, $\mathcal{D}_4 = \{(x_k, y_k) : k \in \{1\}\}$, $\mathcal{D}_5 = \{(x_k, y_k) : k \in \{1, 3\}\}$, $\mathcal{D}_6 = \{(x_k, y_k) : k \in \{1, 2, 3\}\}$, $\mathcal{D}_7 = \{(x_k, y_k) : k \in \{1, 2, 3, 5\}\}$ etc.
  Note that in the asynchronous setting $|\mathcal{D}_j| = j - M$ for all $j > M$. $\{\mathcal{D}_j\}_{j \geq 1}$ determines the filtration when constructing the posterior GP at every step j.
- Finally, in all three settings, $\mu_A : \mathcal{X} \to \mathbb{R}$ and $\sigma_A : \mathcal{X} \to \mathbb{R}_+$ will refer to the posterior mean and standard deviation of the GP conditioned on some evaluations A, i.e. $A \subset \mathcal{X} \times \mathbb{R}$ is a

set of (x, y) values and $|A| < \infty$. They can be computed by plugging in the (x, y) values in A to (2.1). For example, $\mu_{\mathcal{D}_j}, \sigma_{\mathcal{D}_j}$ will denote the mean and standard deviation conditioned on the completed evaluations, $\mathcal{D}_j$. Finally, when using our indexing scheme above we will also overload notation so that σ_{j-1} will denote the posterior standard deviation conditioned on evaluations from steps 1 to $j-1$. That is $\sigma_{j-1} = \sigma_A$ where $A = \{(x_k, y_k)\}_{k=1}^{j-1}$.

5.5.2 Bounding the regret in terms of the number of evaluations

In the remainder of this section, $\beta_n \in \mathbb{R}$ for all $n \geq 1$ will denote the following value.

$$\beta_n \;=\; 4(d+1)\log(n) + 2d\log(dab\sqrt{\pi}) \;\asymp\; d\log(n), \tag{5.4}$$

Here d is the dimension, a, b are from Assumption 1, and n will denote the number of evaluations. Our first theorem below is a bound on the simple regret for synTS after n completed evaluations.

Theorem 49. *Let $f \sim \mathcal{GP}(\mathbf{0}, \kappa)$ where $\kappa : \mathcal{X}^2 \to \mathbb{R}$ satisfies Assumption 1. Further, without loss of generality $\kappa(x, x') \leq 1$. Then for* synTS*, the Bayes simple regret after n evaluations satisfies,*

$$\mathbb{E}[S_n] \;\leq\; \frac{C_1}{n} \;+\; \frac{(M-1)\beta_{M-1}^{1/2}}{n} \;+\; \sqrt{\frac{C_2\beta_{n+M-1}\Psi_n}{n}},$$

where Ψ_n is the MIG in Definition 1, β_n is as defined in (5.4), ξ_M is from (5.3), and $C_1 = \pi^2/6 + \sqrt{2}\pi^{5/2}/12$, $C_2 = 2/\log(1+\eta^{-2})$ are constants.

Proof. Our proof is based on techniques from Russo and Van Roy [211] and Srinivas et al. [235]. As part of analysis, we will discretise $\mathcal{X}$ at each step j of the algorithm. Our discretisation ν_j, is obtained via a grid of $\tau_j = j^2 dab\sqrt{\pi}$ equally spaced points along each coordinate and has size $|\nu_j| = \tau_j^d$. It is easy to verify that ν_j satisfies the following property: for all $x \in \mathcal{X}$, $\|x - [x]_j\|_1 \leq d/\tau_j$, where $[x]_j$ is the closest point to x in ν_j. This discretisation is deterministically constructed ahead of time and does not depend on any of the random quantities in the problem.

For the purposes of our analysis, we define the Bayes cumulative regret after n evaluations as,

$$\mathbb{E}[R_n] = \mathbb{E}\left[\sum_{j=1}^{n} f(x_\star) - f(x_j)\right].$$

Here, just as in (5.1), the expectation is with respect to the randomness in the prior, observations and algorithm. Since the average is larger than the minimum, we have $\frac{1}{n}\sum_j f(x_\star) - f(x_j) \geq \min_j(f(x_\star) - f(x_j)) = S_n$; hence $\mathbb{E}[S_n] \leq \frac{1}{n}\mathbb{E}[R_n]$.

For the proof, we will use the following property of TS. The sampling distribution at time step j is $p(x_\star|\mathcal{D}_j)$. While TS is a randomised strategy, this distribution itself is constructed deterministically; the randomness comes from the algorithm. We will denote this external

source of randomness at step j by R_j. Now denote $U_j(\cdot) = \mu_{\mathcal{D}_j}(\cdot) + \beta_j^{1/2}\sigma_{\mathcal{D}_j}(\cdot)$ and $V_j(\cdot) = \mu_{\mathcal{D}_j}(\cdot) + \beta_{j+M-1}^{1/2}\sigma_{j-1}(\cdot)$. We begin by decomposing $\mathbb{E}[R_n]$ as follows.

$$
\begin{aligned}
\mathbb{E}[R_n] &= \sum_{j=1}^{n} \mathbb{E}\left[f(x_\star) - f(x_j)\right] \\
&\overset{(a)}{=} \sum_{j=1}^{n} \mathbb{E}\big[f(x_\star) - f([x_\star]_j) + f([x_\star]_j) - U_j([x_\star]_j) + U_j([x_\star]_j) - V_j([x_\star]_j) + V_j([x_\star]_j) \\
&\qquad\qquad - V_j([x_j]_j) + V_j([x_j]_j) - f([x_j]_j) + f([x_j]_j) - f(x_j)\big] \\
&= \underbrace{\sum_{j=1}^{n} \mathbb{E}\big[f(x_\star) - f([x_\star]_j)\big]}_{A_1} + \underbrace{\sum_{j=1}^{n} \mathbb{E}\big[f([x_j]_j) - f(x_j)\big]}_{A_2} \\
&\qquad + \underbrace{\sum_{j=1}^{n} \mathbb{E}\big[f([x_\star]_j) - U_j([x_\star]_j)\big]}_{A_3} + \underbrace{\sum_{j=1}^{n} \mathbb{E}\big[V_j([x_j]_j) - f([x_j]_j)\big]}_{A_4} \\
&\qquad + \underbrace{\sum_{j=1}^{n} \mathbb{E}\big[U_j([x_\star]_j) - V_j([x_\star]_j)\big]}_{A_5} + \underbrace{\sum_{j=1}^{n} \mathbb{E}\big[V_j([x_\star]_j) - V_j([x_j]_j)\big]}_{A_6}.
\end{aligned}
$$

In the first step we have added and subtracted $f([x_\star]_j), U_j([x_\star]_j), V_j([x_\star]_j), V_j([x_j]_j)$, and $f([x_j]_j)$ and in the second step we have separated the terms into the sums $A_1, \ldots, A_6$. We will first control A_5 and A_6 using properties of TS.

By conditioning on $\mathcal{D}_j, \{R_k\}_{k=1}^{j-1}$ we argue that j^{th} term in A_6 vanishes, i.e. $\mathbb{E}\big[V_j([x_\star]_j) - V_j([x_j]_j)\big] = \mathbb{E}\big[\mathbb{E}[V_j([x_\star]_j) - V_j([x_j]_j)|\mathcal{D}_j, \{R_k\}_{k<j}]\big] = 0$. For this, first note that as x_j is sampled from the posterior distribution for $x_\star$ conditioned on $\mathcal{D}_j$, both $x_j|\mathcal{D}_j$ and $x_\star|\mathcal{D}_j$ have the same distribution. Since $R_j \notin \{R_k\}_{k<j}$ and this randomness is independent of everything else, $x_\star$ and x_j are equal in distribution conditioned on $\mathcal{D}_j, \{R_k\}_{k<j}$. Now observe that V_j is deterministic conditioned on $\mathcal{D}_j, \{R_k\}_{k<j}$. This is because at step j, $\mu_{\mathcal{D}_j}$ is a function of past query points and observations $\mathcal{D}_j$, $\sigma_{\mathcal{D}_j}$ depends only on past query points and σ_{j-1} depends only on past query points and those currently in evaluation; the latter is also deterministic since we are conditioning on $\{R_k\}_{k<j}$. Now, as the discretisation ν_j is fixed ahead of time, $V_j([x_j]_j)$ and $V_j([x_\star]_j)$ are also equal in distribution given $\mathcal{D}_j, \{R_k\}_{k<j}$. Therefore, both quantities are also equal in expectation.

Now, let us bound A_5. Noting that each term inside the summation A_5 is $\mathbb{E}[U_j([x_\star]_j) - V_j([x_\star]_j)] = \mathbb{E}[\beta_j^{1/2}\sigma_{\mathcal{D}_j}([x_\star]_j) - \beta_{j+M}^{1/2}\sigma_{j-1}([x_\star]_j)]$ we have,

$$
A_5 = \sum_{j=1}^{M-1} \beta_j^{1/2}\mathbb{E}[\sigma_{\mathcal{D}_j}([x_\star]_j)] - \sum_{j=n-M+1}^{n} \mathbb{E}[\beta_{j+M-1}^{1/2}\sigma_{j-1}([x_\star]_j)] +
$$

$$\sum_{j=M}^{n-M} \beta_j^{1/2} \mathbb{E}\left[\sigma_{\mathcal{D}_j}([x_\star]_j) - \sigma_{j-M}([x_\star]_j)\right]$$

$$\leq (M-1)\beta_{M-1}^{1/2}\|\kappa\|_\infty^{1/2} + \sum_{j=M}^{n-M} \beta_j^{1/2} \mathbb{E}\left[\sigma_{\mathcal{D}_j}([x_\star]_j) - \sigma_{j-M}([x_\star]_j)\right] \tag{5.5}$$

In the first step we have simply rearranged the terms and in the second step we have bounded the first sum by it largest possible values and dropped the second sum. For the synchronous case, we always have $\mathcal{D}_j \supseteq \{(x_k, y_k) : k \leq j - M\}$. Hence, $\sigma_{\mathcal{D}_j} \leq \sigma_{j-M}$ pointwise for all j and each term is bounded above by 0. Finally, since we have assumed $\|\kappa\|_\infty = 1$, $A_6 \leq (M-1)\beta_{M-1}^{1/2}$.

To bound A_1, A_2 and A_3 we use the following Lemmas. The proofs are in Chapters 5.5.2 and 5.5.2.

Lemma 50. *At step j, for all $x \in \mathcal{X}$, $\mathbb{E}[|f(x) - f([x]_j)|] \leq \frac{1}{2j^2}$.*

Lemma 51. *At step j, for all $x \in \nu_j$, $\mathbb{E}[\mathbb{1}\{f(x) > U_j(x)\} \cdot (f(x) - U_j(x))] \leq \frac{1}{j^2\sqrt{2\pi}|\nu_j|}$.*

Using Lemma 50 and the fact that $\sum_j j^{-2} = \pi^2/6$, we have $A_1 + A_2 \leq \pi^2/6$. We bound A_3 via,

$$\begin{aligned}
A_3 \;&\leq\; \mathbb{E}\Bigg[\sum_{j=1}^{n} \mathbb{1}\{f([x_\star]_j) > U_j([x_\star]_j)\} \cdot (f([x_\star]_j) - U_j([x_\star]_j))\Bigg] \\
&\leq \sum_{j=1}^{n}\sum_{x\in\nu_j} \mathbb{E}\big[\, \mathbb{1}\{f(x) > U_j(x)\} \cdot (f(x) - U_j(x))\,\big] \\
&\leq \sum_{j=1}^{n}\sum_{x\in\nu_j} \frac{1}{j^2\sqrt{2\pi}|\nu_j|} = \frac{\sqrt{2\pi}}{12}
\end{aligned}$$

In the first step we upper bounded A_3 by only considering the positive terms in the summation. The second step bounds the term for $[x_\star]_j$ by the sum of corresponding terms for all $x \in \nu_j$. We then apply Lemma 51.

Finally, we bound each term inside the summation of A_4 as follows,

$$\begin{aligned}
\mathbb{E}[V_j([x_j]_j) - f([x_j]_j)] &= \mathbb{E}[\mu_{\mathcal{D}_j}([x_j]_j) + \beta_{j+M}^{1/2}\sigma_{j-1}([x_j]_j) - f([x_j]_j)] \\
&= \mathbb{E}[\mu_{\mathcal{D}_j}([x_j]_j) + \beta_{j+M}^{1/2}\sigma_{j-1}([x_j]_j) - \mathbb{E}[f([x_j]_j)|\mathcal{D}_j, \{R_k\}_{k<j}]] \\
&= \mathbb{E}[\beta_{j+M}^{1/2}\sigma_{j-1}([x_j]_j)]
\end{aligned} \tag{5.6}$$

Once again, we have used the fact that $\mu_{\mathcal{D}_j}, \sigma_{j-1}$ are deterministic given $\mathcal{D}_j, \{R_k\}_{k<j}$. Therefore,

$$\begin{aligned}
A_4 &\overset{(a)}{\leq} \beta_{n+M}^{1/2}\mathbb{E}\left[\sum_{j=1}^{n}\sigma_{j-1}([x_j]_j)\right] \overset{(b)}{\leq} \beta_{n+M}^{1/2}\mathbb{E}\left[\left(n\sum_{j=1}^{n}\sigma_j^2([x_j]_j)\right)^{1/2}\right] \\
&\overset{(c)}{\leq} \sqrt{\frac{2n\beta_{n+M}\Psi_n}{\log(1+\eta^{-2})}}
\end{aligned} \tag{5.7}$$

Here, (a) uses (5.6) and that β_j is increasing in j (5.4). (b) uses the Cauchy-Schwarz inequality and (c) uses Lemma 44. Putting the bounds for $A_1, \ldots, A_6$ together we get, $\mathbb{E}[R_n] \leq C_1 + (M-1)\beta_{M-1}^{1/2} + \sqrt{C_2 n \beta_n \Psi_n}$. The theorem follows from the relation $\mathbb{E}[S_n] \leq \frac{1}{n}\mathbb{E}[R_n]$. □

For our analysis of asyTS, we will need the following result from Desautels et al. [52]. Recall that the posterior variance of a GP does not depend on the observations.

Lemma 52 (Lemma 1 (modified) in [52])**.** *Let $f \sim \mathcal{GP}(0, \kappa)$. Let A, B be finite subsets of $\mathcal{X}$. Let $y_A \in \mathbb{R}^{|A|}$ and $y_B \in \mathbb{R}^{|B|}$ denote the observations when we evaluate f at A and B respectively. Further let σ_A, $\sigma_{A\cup B} : \mathcal{X} \to \mathbb{R}$ denote the posterior standard deviation of the GP when conditioned on A and $A \cup B$ respectively. Then,*

$$\text{for all } x \in \mathcal{X}, \quad \frac{\sigma_A(x)}{\sigma_{A\cup B}(x)} \leq \exp\left(I(f; y_B | y_A)\right)$$

The proof exactly mimics the proof in Desautels et al. [52]. Lemma 52 implies $\sigma_A(x) \leq \xi_M^{1/2}\sigma_{A\cup B}(x)$ where ξ_M is from (5.3).

The following result bounds the regret for asyTS in terms of the number of evaluations.

Theorem 53. *Assume the same setting and quantities as in Theorem 49. Then for* asyTS*, the Bayes simple regret after n evaluations satisfies,*

$$\mathbb{E}[S_n] \leq \frac{C_1}{n} + \sqrt{\frac{C_2 \xi_M \beta_n \Psi_n}{n}}.$$

Here, all quantities are as defined in Theorem 49.

Proof. We will first assume that the n evaluations completed are the the evaluations indexed $j = 1, \ldots, n$. Our proof will follow along the same lines as that for synTS, except we will use $U_j(\cdot) = V_j(\cdot) = \mu_{\mathcal{D}_j}(\cdot) + \beta_j^{1/2}\sigma_{\mathcal{D}_j}(\cdot)$. The terms A_1, A_2, A_3 are bound exactly the same way yielding $A_1 + A_2 + A_3 \leq C_1$. A_6 can be shown to be zero by conditioning on $\mathcal{D}_j$ and using a similar argument. $A_5 = 0$ since $U_j = V_j$. Hence, the only thing left to bound is A_4. Using a similar reasoning to (5.6), we have $\mathbb{E}[U_j([x_j]_j) - f([x_j]_j)] = \mathbb{E}[\beta_j^{1/2}\sigma_{\mathcal{D}_j}([x_j]_j)]$. Then,

$$\begin{aligned} A_4 &\overset{(a)}{\leq} \beta_n^{1/2} \sum_{j=1}^{n} \mathbb{E}[\sigma_{\mathcal{D}_j}([x_j]_j)] \overset{(b)}{\leq} \beta_n^{1/2}\xi_M^{1/2}\mathbb{E}\left[\sum_{j=1}^{n} \sigma_{j-1}([x_j]_j)\right] \\ &\overset{(c)}{\leq} \beta_n^{1/2}\xi_M^{1/2}\mathbb{E}\left[\left(n\sum_{j=1}^{n}\sigma_j^2([x_j]_j)\right)^{1/2}\right] \overset{(d)}{\leq} \sqrt{\frac{2\xi_M n \beta_n \Psi_n}{\log(1+\eta^{-2})}} \end{aligned} \tag{5.8}$$

Here, (a) uses that β_j is increasing in j (5.4). (c) uses the Cauchy-Schwarz inequality and (d) uses Lemma 44. For (b), first we note that $\mathcal{D}_j \subseteq \{(x^{(i)}, y^{(i)})\}_{i=1}^{j-1}$. In the asynchronous setting we will be missing up to $M-1$ evaluations during the first M steps and exactly $M-1$ evaluations thereafter. In either case, letting $A = \mathcal{D}_j$ and $B = \{(x^{(i)}, y^{(i)})\}_{i=1}^{j-1} \setminus \mathcal{D}_j$ in Lemma 52 we get,

$$\text{for all } x \in \mathcal{X}, \quad \sigma_{\mathcal{D}_j}(x) \leq \exp\left(I(f; y_B | y_{\mathcal{D}_j})\right)\sigma_{j-1}(x) \leq \xi_M^{1/2}\sigma_{j-1}(x). \tag{5.9}$$

The last step uses (5.3) and that $|B| < M$.

Now consider the case where the n evaluations completed are not the first n dispatched. Since A_1, A_2, A_3 are bounded by constants summing over all n we only need to worry about A_4. In step (a) of (5.8), we have bounded A_4 by the sum of posterior variances $\sigma_{j-1}([x_j]_j)$. Since $\sigma_{j'-1}([x_j]_j) < \sigma_{j-1}([x_j]_j)$ for $j' > j$, the sum for any n completed evaluations can be bound by the same sum for the first n evaluations dispatched. The result follows accordingly. □

Finally, we note that the bound for the sequential setting for Thompson sampling in Theorem 2 follows directly by setting $M = 1$ in Theorem 49. We state it formally below.

Corollary 54. *Assume the same setting and quantities as in Theorem 49. Then for* seqTS*, the Bayes' simple regret after n evaluations satisfies,*

$$\mathbb{E}[S_n] \;\leq\; \frac{C_1}{n} + \sqrt{\frac{C_2 \beta_n \Psi_n}{n}},$$

Proof of Lemma 50

Let $L = \sup_{i=1,\ldots,d} \sup_{x \in \mathcal{X}} \left|\frac{\partial f(x)}{\partial x_i}\right|$. By Assumption 1 and the union bound we have $\mathbb{P}(L \geq t) \leq da \exp^{-t^2/b^2}$. Let $x \in \mathcal{X}$. We bound,

$$\begin{aligned}\mathbb{E}[|f(x) - f([x]_j)|] \leq \mathbb{E}[L\|x - [x]_j\|_1] \leq \frac{d}{\tau_j}\mathbb{E}[L] = \frac{d}{\tau_j}\int_0^\infty \mathbb{P}(L \geq t)\mathrm{d}t \\ \leq \frac{d}{\tau_j}\int_0^\infty ae^{t^2/b^2}\mathrm{d}t = \frac{dab\sqrt{\pi}}{2\tau_j} = \frac{1}{2j^2}.\end{aligned}$$

The first step bounds the difference in the function values by the largest partial derivative and the L^1 distance between the points. The second step uses the properties of the discretisation ν_j and the third step uses the identity $\mathbb{E}X = \int \mathbb{P}(X > t)\mathrm{d}t$ for positive random variables X. The last step uses the value for τ_j specified in the main proof. □

Proof of Lemma 51

The proof is similar to Lemma 2 in [211], but we provide it here for completeness. We will use the fact that for $Z \sim \mathcal{N}(\mu, \sigma^2)$, we have $\mathbb{E}[Z\mathbb{1}(Z > 0)] = \frac{\sigma}{\sqrt{2\pi}}e^{-\mu^2/(2\sigma^2)}$. Noting that $f(x) - U_j(x)|\mathcal{D}_j \sim \mathcal{N}(-\beta_j^{1/2}\sigma_{\mathcal{D}_j}(x), \sigma^2_{\mathcal{D}_j}(x))$, we have,

$$\mathbb{E}[\mathbb{1}\{f(x) > U_j(x)\} \cdot (f(x) - U_j(x))|\mathcal{D}_j] = \frac{\sigma_{\mathcal{D}_j}(x)}{\sqrt{2\pi}}e^{\beta_j/2} \leq \frac{1}{\sqrt{2\pi}|\nu_j|j^2}.$$

Here, the last step uses that $\sigma_{\mathcal{D}_j}(x) \leq \kappa(x, x) \leq 1$ and that $\beta_j = 2\log(j^2|\nu_j|)$. □

On the Initialisation for asynchronous parallel TS

Description of the initialisation scheme: The initialisation scheme [52, 144] is an uncertainty sampling procedure designed to reduce the posterior variance throughout the domain $\mathcal{X}$. Here, we first pick the point with the largest prior GP variance, $x_1^{\text{init}} = \text{argmax}_x \kappa(x, x)$. We then iterate $x_j^{\text{init}} = \text{argmax}_{x \in \mathcal{X}} \kappa_{j-1}(x, x)$ where κ_{j-1} denotes the posterior kernel with the previous $j-1$ evaluations. As the posterior variance of a GP does not depend on the observations, this scheme is asynchronously parallelisable: simply pre-compute the evaluation points and then deploy them in parallel.

Bounds on $\mathbb{E}[S_n]$ after initialisation: Desautels et al. [52] provide bounds for C_κ as a function of γ_M for different kernels (see Chapter 5.3). During this initialisation phase the best bound we can achieve on the instantaneous regret is $\mathbb{E}[f(x_\star) - f(x_j)] \leq 2\,\Xi$. Applying Theorem 53 after initialisation, we have asyTS:

$$\mathbb{E}[S_n] \;\leq\; \frac{C_1}{n} + \frac{2\,\Xi\,\gamma_M}{n} + \sqrt{\frac{C_2 C_\kappa \Psi_n \log(n)}{n}} \tag{5.10}$$

A more rigorous proof will simply replace the unconditional mutual information in the definition of the MIG with the mutual information conditioned on the first γ_M evaluations.We conjecture that asyTS will not need this initialisation and wish to resolve this in future work.

5.5.3 Proofs for Parallel TS with Random Evaluation Times

The goal of this section is to prove Theorem 48. In Section 5.5.3 we derive some concentration results for uniform and half-normal distributes and their maxima. In Section 5.5.3 we do the same for exponential random variables. We put everything together in Section 5.5.3 to prove Theorem 48. We will use several results on sub-Gaussian and sub-exponential random variables which are reviewd in Chapter 2.4.

Results for Uniform and Half-normal Random Variables

In the next two lemmas, let $\{X_i\}_{i=1}^M$ denote a sequence of M i.i.d random variables and $Y = \max_i X_i$ be their maximum. We note that the results or techniques in Lemmas 55, 56 are not particularly new.

Lemma 55. *Let $X_i \sim \text{Unif}(a, b)$. Then $\mathbb{E}X_i = \theta$ and $\mathbb{E}Y = \theta + \frac{M-1}{M+1}\frac{b-a}{2}$ where $\theta = (a+b)/2$.*

Proof. The proof for $\mathbb{E}X_i$ is straightforward. The cdf of Y is $\mathbb{P}(Y \leq t) = \prod_{i=1}^M \mathbb{P}(X_i \leq t) =$

$(\frac{t-a}{b-a})^M$. Therefore its pdf is $p_Y(t) = M(t-a)^{M-1}/(b-a)^M$ and its expectation is

$$\begin{aligned}\mathbb{E}[Y] &= \int_a^b tM(t-a)^{M-1}/(b-a)^M \mathrm{d}t \\ &= \frac{a+bM}{M+1} = \theta + \frac{M-1}{M+1}\frac{b-a}{2}.\end{aligned}$$

□

Lemma 56. *Let $X_i \sim \mathcal{HN}(\zeta^2)$. Then $\mathbb{E}X_i = \zeta\sqrt{2/\pi}$ and $\mathbb{E}Y$ satisfies,*

$$\zeta K\sqrt{\log(M)} \leq \mathbb{E}Y \leq \zeta\sqrt{2\log(2M)}.$$

Here K is a universal constant. Therefore, $\mathbb{E}Y \in \Theta(\sqrt{\log(M)})\mathbb{E}X_i$.

Proof. The proof for $\mathbb{E}X_i$ just uses integration over the pdf $p_Y(t) = \frac{\sqrt{2}}{\sqrt{\pi\zeta^2}}e^{-\frac{t^2}{2\sigma^2}}$. For the second part, writing the pdf of $\mathcal{N}(0,\zeta^2)$ as $\phi(t)$ we have,

$$\begin{aligned}\mathbb{E}[e^{\lambda X_i}] &= 2\int_0^\infty e^{\lambda t}\phi(t)\mathrm{d}t \leq 2\int_{-\infty}^\infty e^{\lambda t}\phi(t)\mathrm{d}t \\ &= 2\mathbb{E}_{Z\sim\mathcal{N}(0,\zeta^2)}[e^{\lambda Z}] = 2e^{\zeta^2\lambda^2/2}.\end{aligned}$$

The inequality in the second step uses that the integrand is positive. Therefore, using Jensen's inequality and the fact that the maximum is smaller than the sum we get,

$$\begin{aligned}e^{\lambda\mathbb{E}[Y]} \leq \mathbb{E}[e^{\lambda Y}] \leq \sum_{i=1}^n \mathbb{E}[e^{\lambda X_i}] \leq 2Me^{\lambda^2\zeta^2/2} \\ \iff \mathbb{E}[Y] \leq \frac{1}{\lambda}\log(2M) + \frac{\zeta^2\lambda}{2}.\end{aligned}$$

Choosing $\lambda = \frac{\sqrt{2\log(2M)}}{\zeta}$ yields the upper bound.

The lower bound follows from Lemma 4.10 of Adler [4] which establishes a $K\sqrt{\log(M)}$ lower bound for M i.i.d standard normals $Z_1, \ldots, Z_M$. We can use the same lower bound since $|Z_i| \geq Z_i$. □

Lemma 57. *Suppose we complete a sequence of jobs indexed $j = 1, 2, \ldots$. The time taken for the jobs $\{X_j\}_{j\geq 1}$ are i.i.d with mean θ and sub-Gaussian parameter τ. Let $\delta \in (0,1)$, and N denote the number of completed jobs after time T. That is, N is the random variable such that $N = \max\{n \geq 1; \sum_{j=1}^n X_j \leq T\}$. Then, with probability greater than $1-\delta$, for all $\alpha \in (0,1)$, there exists $T_{\alpha,\delta}$ such that for all $T > T_{\alpha,\delta}$, $N \in \left(\frac{T}{\theta(1+\alpha)} - 1, \frac{T}{\theta(1-\alpha)}\right)$.*

Proof. We will first consider the total time taken $S_n = \sum_{i=1}^n X_i$ after n evaluations. Let $\varepsilon_n = \tau\sqrt{n\log(n^2\pi^2/(3\delta))}$ throughout this proof. Using Lemma 11, we have $\mathbb{P}(|S_n - n\theta| > \varepsilon_n) =$

$6\delta/(n\pi^2)$. By a union bound over all $n \geq 1$, we have that with with probability greater than δ, the following event $\mathcal{E}$ holds.

$$\mathcal{E} = \{\forall n \geq 1, \ \ |S_n - n\theta| \leq \varepsilon_n.\} \tag{5.11}$$

Since $\mathcal{E}$ is a statement about all time steps, it is also for true the random number of completed jobs N. Inverting the condition in (5.11) and using the definition of N, we have

$$N\theta - \epsilon_N \leq S_N \leq T \leq S_{N+1} \leq (N+1)\theta + \epsilon_{N+1}. \tag{5.12}$$

Now assume that there exists $T_{\alpha,\delta}$ such that for all $T \geq T_{\alpha,\delta}$ we have, $\epsilon_N \leq N\alpha\theta$. Since ε_n is sub-linear in n, it also follows that $\epsilon_{N+1} \leq (N+1)\alpha\theta$. Hence, $N\theta(1-\alpha) \leq T \leq (N+1)\theta(1+\alpha)$ and the result follows.

All that is left to do is to establish that such a $T_{\alpha,\delta}$ exists under event $\mathcal{E}$, for which we will once again appeal to (5.12). The main intuition is that as $\epsilon_N \asymp \sqrt{N\log(N)}$, the condition $\epsilon_N \leq N\alpha\theta$ is satisfied for N large enough. But N is growing with T, and hence it is satisfied for T large enough. More formally, since $\frac{N}{\epsilon_N} \asymp \frac{N+1}{\epsilon_{N+1}}$ using the upper bound for T it is sufficient to show $\frac{T}{\epsilon_{N+1}} \gtrsim \frac{1}{\alpha\theta}$ for all $T \geq T_{\alpha,\delta}$. But since $\epsilon_{N+1} \asymp \sqrt{N\log(N)}$ and the lower bound for T is $T \gtrsim N$, it is sufficient if $\frac{T}{\sqrt{T\log(T)}} \gtrsim \frac{1}{\alpha\theta}$ for all $T \geq T_{\alpha,\delta}$. This is achievable as the LHS is increasing with T and the RHS is constant. □

Our final result for the uniform and half-normal random variables follows as a consequence of Lemma 57.

Theorem 58. *Let the time taken X for completing an evaluation to f be a random variable.*

- *If $X \sim \text{Unif}(a,b)$, denote $\theta = (a+b)/2$, $\theta_M = \theta + \frac{M-1}{M+1}\frac{b-a}{2}$, and $\tau = (b-a)/2$.*
- *If $X \sim \mathcal{HN}(\tau^2)$, denote $\theta = \zeta\sqrt{2/\pi}$, $\theta_M = \theta \cdot \Theta(\sqrt{\log(M)})$, and $\tau = \zeta\pi/2$.*

Denote the number of evaluations within time T by sequential, synchronous parallel and asynchronous parallel algorithms by $N_{\text{seq}}, N_{\text{syn}}, N_{\text{asy}}$ respectivey. Let $\delta \in (0,1)$. Then, with probability greater than $1-\delta$, for all $\alpha \in (0,1)$, there exists $T_{\alpha,\delta}$ such that for all $T \geq T_{\alpha,\delta}$, we have each of the following,

$$\begin{aligned}
N_{\text{seq}} &\in \left(\frac{T}{\theta(1+\alpha)} - 1, \frac{T}{\theta(1-\alpha)}\right), \\
N_{\text{syn}} &\in \left(M\left\lfloor\frac{T}{\theta_M(1+\alpha)} - 1\right\rfloor, \frac{MT}{\theta_M(1-\alpha)}\right), \\
N_{\text{asy}} &\in \left(M\left\lfloor\frac{T}{\theta(1+\alpha)} - 1\right\rfloor, \frac{MT}{\theta(1-\alpha)}\right).
\end{aligned}$$

Proof. We first show τ sub-Gaussianity of X and $Y = \max_{j=1,\ldots,M} X_j$ when $X, X_1, \ldots, X_M$ are either uniform or half-normal. For the former, both X and Y are $\tau = (b-a)/2$ sub-Gaussian since they are bounded in $[a,b]$. For the Half-normal case, we note that $X = |Z|$

and $Y = \max_{j=1,\ldots,M} |Z_i|$ for some i.i.d. $\mathcal{N}(0, \zeta^2)$ variables $Z, \{Z_i\}_{i=1}^M$. Both are 1-Lipschitz functions of Z_i and $(Z_{i1}, \ldots, Z_{iM})$ respectively and $\tau = \zeta\pi/2$ sub-Gaussianity follows from Lemma 12.

Now in synchronous settings, the algorithm dispatches the k^{th} batch with evaluation times $\{(X_{k1}, \ldots, X_{kM})\}$. It releases its $(k+1)^{\text{th}}$ batch when all evalutions finish after time $Y_k = \max_{i=1,\ldots,M} X_{ki}$. The result for N_{syn} follows by applying Lemma 57 on the sequence $\{Y_k\}_{k\geq 1}$. For the sequential setting, each worker receives its $(k+1)^{\text{th}}$ job after completing its k^{th} evaluation in time X_k. We apply Lemma 57 on the sequence $\{X_k\}_{k\geq 1}$ for one worker to obtain that the number of jobs completed by this worker is $N_{\text{seq}} \in \left(\frac{T}{\theta(1+\alpha)} - 1, \frac{T}{\theta(1-\alpha)}\right)$. In the asynchronous setting, a worker receives his new job immediately after finishing his last. Applying the same argument as the sequential version to all workers but with $\delta \leftarrow \delta/M$ in Lemma 57 and the union bound yields the result for N_{asy}. □

Results for the Exponential Random Variable

The results here on exponential delays for function evaluations were primarily derived by Akshay Krishnamurthy. They are included here for the sake of completeness.

In this section we derive an analogous result to Theorem 58 for the case when the completion times are exponentially distributed. The main challenges stem from analysing the distribution of the maxima of a finite number of exponential random variables. Much of the analysis is based on results from Boucheron and Thomas [18] (See also chapter 6 of Boucheron et al. [19]).

In deviating from the notation used in Table 5.1, we will denote the parameter of the exponential distribution as θ, i.e. it has pdf $p(x) = \theta x^{-\theta x}$. The following fact about exponential random variables will be instrumental.

Fact 59. *Let $X_1, \ldots, X_n \sim \text{Exp}(\theta)$ iid. Also let $E_1, \ldots, E_n \sim \text{Exp}(\theta)$ iid and independent from X_1^n. If we define the order statistics $X_{(1)} \geq X_{(2)} \geq \ldots \geq X_{(n)}$ for $X_1, \ldots, X_n$, we have*

$$(X_{(n)}, \ldots, X_{(1)}) \sim \left(E_n/n, \ldots, \sum_{k=i}^{n} E_k/k, \ldots, \sum_{k=1}^{n} E_k/k \right).$$

Proof. This is Theorem 2.5 in [18] but we include a simple proof for completeness. We first must analyse the minimum of n exponentially distributed random variables. This is a simple calculation.

$$\mathbb{P}[\min_i X_i \geq t] = \prod_{i=1}^{n} \mathbb{P}[X_i \geq t] = \prod_{i=1}^{n} \exp(-\theta t) = \exp(-n\theta t)$$

This last expression is exactly the probability that an independent $\text{Exp}(n\theta)$ random variables is at least t.

This actually proves the first part, since $E_n/n \sim \mathrm{Exp}(n\theta)$. Now, using the memoryless property, conditioning on $X_{(n)} = x$ and $X_{(n)} = X_i$ for some i, we know that for $j \neq i$

$$\mathbb{P}[X_j \geq x' + x | X_{(n)} = x, X_{(n)} = X_i] = \exp(-\theta x').$$

Removing the conditioning on the index achieving $X_{(n)}$, and using the same calculation for the minimum, we now get

$$\mathbb{P}[X_{(n-1)} \geq x' + x | X_{(n)} = x] = \exp(-(n-1)\theta x').$$

Thus we have that $X_{(n-1)} - X_{(n)} \sim \mathrm{Exp}((n-1)\theta)$. The claim now follows by induction. □
As before the first step of the argument is to understand the expectation of the maximum.

Lemma 60. *Let $X_i \sim \mathrm{Exp}(\theta)$. Then $\mathbb{E}X_i = 1/\theta$ and $\mathbb{E}Y = h_M/\theta$ where $Y = \max_{i=1,\ldots,M} X_i$ is the maximum of the X_i's and $h_M = \sum_{i=1}^{M} i^{-1}$ is the M^{th} harmonic number.*

Proof. Using the relationship between the order statistics and the spacings in Fact 59 we get

$$\mathbb{E}\max_i X_i = \mathbb{E}X_{(1)} = \mathbb{E}\sum_{k=1}^{M} E_k/k = \sum_{k=1}^{M} \frac{1}{k\theta} = \frac{h_M}{\theta}.$$ □

Recall that $h_M \asymp \log(M)$ accounting for the claims made in Table 5.1 and the subsequent discussion.

While obtaining polynomial concentration is straightforward via Chebyshev's inequality it is insufficient for our purposes, since we will require a union bound over many events. However, we can obtain exponential concentration, although the argument is more complex. Our analysis is based on Herbst's argument, and a modified logarithmic Sobolev inequality, stated below in Theorem 61. To state the inequality, we first define the entropy $\mathrm{Ent}[X]$ of a random variable X as follows (not to be confused with Shannon entropy),

$$\mathrm{Ent}[X] \triangleq \mathbb{E}[X\log(X)] - \mathbb{E}[X]\log(\mathbb{E}[X]).$$

Theorem 61 (Modified logarithmic Sobolev inequality (Theorem 6.6 in [19]))**.** *Let $X_1, \ldots, X_n$ be independent random variables taking values in some space $\mathcal{X}$, $f : \mathcal{X}^n \to \mathbb{R}$, and define the random variable $Z = f(X_1, \ldots, X_n)$. Further let $f_i : \mathcal{X}^{n-1} \to \mathbb{R}$ for $i \in \{1, \ldots, n\}$ be arbitrary functions and $Z_i = f_i(X^{(i)}) = f_i(X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_n)$. Finally define $\tau(x) = e^x - x - 1$. Then for all $\lambda \in \mathbb{R}$*

$$\mathrm{Ent}[e^{\lambda Z}] \leq \sum_{i=1}^{n} \mathbb{E}[e^{\lambda Z}\tau(-\lambda(Z - Z_i))].$$

Application of the logarithmic Sobolev inequality in our case gives:

Lemma 62. *Let $X_1, \ldots, X_M \sim \mathrm{Exp}(\theta)$ iid, $E \sim \mathrm{Exp}(\theta)$ also independently and define $Z = \max_{i\in\{1,\ldots,M\}} X_i$ and $\mu = \mathbb{E}Z$. Define $\tau(x) = e^x - x - 1$ and $\psi(x) = \exp(x)\tau(-x) = 1 + (x-1)e^x$. Then for any $\lambda \in \mathbb{R}$*

$$\mathrm{Ent}[\exp\{\lambda(Z-\mu)\}] \leq \mathbb{E}[\exp\{\lambda(Z-\mu)\}] \times \mathbb{E}\psi(\lambda E),$$
$$\mathrm{Ent}[\exp\{\lambda(\mu-Z)\}] \leq \mathbb{E}[\exp\{\lambda(\mu-Z)\}] \times \mathbb{E}\tau(\lambda E).$$

Proof. We apply Theorem 61 with $Z = f(X_1, \dots, X_M) = \max_i X_i$ and $Z_i = f_i(X^{(i)}) = \max_{j \neq i} X_j$. Notice that in this case, $Z_i = Z$ except when X_i is the maximiser, in which case $Z_i = X_{(2)}$ the second largest of the samples. This applies here since the maximiser is unique with probability 1. Thus, Theorem 61 gives

$$\begin{aligned}
\operatorname{Ent}[\exp\{\lambda Z\}] &\leq \sum_{i=1}^{M} \mathbb{E}[\exp\{\lambda Z\}\tau(-\lambda(Z - Z_i))] \\
&= \mathbb{E}\left[\exp\{\lambda Z\}\tau(-\lambda(X_{(1)} - X_{(2)}))\right] \\
&= \mathbb{E}\left[\exp\{\lambda X_{(2)}\}\exp\{\lambda(X_{(1)} - X_{(2)})\}\tau(-\lambda(X_{(1)} - X_{(2)}))\right] \\
&= \mathbb{E}[\exp \lambda X_{(2)}]\mathbb{E}[\psi(\lambda E)] \leq \mathbb{E}[\exp \lambda X_{(1)}]\mathbb{E}[\psi(\lambda E)]
\end{aligned}$$

The first inequality is Theorem 61, while the first equality uses the definitions of f_i and the fact that $Z_i \neq Z$ for exactly one index i. The second equality is straightforward and the third uses Fact 59 to write $X_{(1)} - X_{(2)}$ as an independent $\operatorname{Exp}(\theta)$ random variable, which also allows us to split the expectation. Finally since $X_{(2)} \leq X_{(1)}$ almost surely, the final inequality follows. Multiplying both sides by $\exp(-\lambda\mu)$, which is non-random, proves the first inequality, since $\operatorname{Ent}(aX) = a\operatorname{Ent}(X)$.

The second inequality is similar. Set $Z = -\max_i X_i$ and $Z_i = -\max_{j \neq i} X_j$ and using the same argument, we get

$$\begin{aligned}
\operatorname{Ent}[\exp\{-\lambda X_{(1)}\}] &\leq \mathbb{E}[\exp\{-\lambda X_{(1)}\}\tau(\lambda(X_{(1)} - X_{(2)}))] \\
&= \mathbb{E}\left[\exp\left\{-\lambda(E_1 + \sum_{k=2}^{M} E_k/k)\right\}\tau(\lambda E_1)\right]
\end{aligned}$$

Here the inequality follows from Theorem 61 and the identity uses Fact 59. We want to split the expectation, and to do so, we use Chebyshev's association principle. Observe that $\exp(-\lambda E_1)$ is clearly non-increasing in E_1 and that $\tau(\lambda E_1)$ is clearly non-decreasing in E_1 for $E_1 \geq 0$ ($E_1 > 0$ a.s.). Hence, we can split the expectation to get

$$\operatorname{Ent}[\exp\{-\lambda X_{(1)}\}] \leq \mathbb{E}[\exp\{-\lambda X_{(1)}\}] \times \mathbb{E}[\tau(\lambda E)]$$

The second inequality follows now by multiplying both sides by $\exp(\lambda\mu)$. □

Theorem 63. *Let $X_1, \dots, X_M \sim \operatorname{Exp}(\theta)$ iid and define $Z = \max_i X_i$ then $Z - \mathbb{E}Z$ is sub-exponential with parameters $(4/\theta^2, 2/\theta)$.*

Proof. We use the logarithmic Sobolev inequality, and proceed with Herbst's method. Unfortunately since our inequality is not in the standard form, we must reproduce most of the argument. However, we can unify the two tails by noticing that we currently have for centered Y (e.g., $Y = X_{(1)} - \mathbb{E}X_{(1)}$ or $Y = \mathbb{E}X_{(1)} - X_{(1)}$),

$$\operatorname{Ent}[\exp\{\lambda Y\}] \leq \mathbb{E}[\exp\{\lambda Y\}]f(\lambda) \tag{5.13}$$

for some differentiable function f, which involves either τ or ψ depending on the tail. We will use such an inequality to bound the moment generating function of Y.

For notational convenience, define $\phi(\lambda) = \log \mathbb{E} \exp\{\lambda Y\}$ and observe that

$$\phi'(\lambda) = \frac{1}{\lambda}\left(\frac{\mathrm{Ent}[\exp\{\lambda Y\}]}{\mathbb{E}\exp\{\lambda Y\}} + \log \mathbb{E}\exp\{\lambda Y\}\right)$$

Together with the inequality in Eq. (5.13), this gives

$$\begin{aligned} \lambda\phi'(\lambda) - \phi(\lambda) &= \frac{\mathrm{Ent}[\exp\{\lambda Y\}]}{\mathbb{E}\exp\{\lambda Y\}} \leq f(\lambda) \\ \Leftrightarrow \ \frac{\phi'(\lambda)}{\lambda} - \frac{\phi(\lambda)}{\lambda^2} &\leq f(\lambda)/\lambda^2, \qquad \forall\ \lambda > 0 \end{aligned}$$

Observe now that the left hand side is precisely the derivative of the function $G(\lambda) = \phi(\lambda)/\lambda$. Hence, we can integrate both sides from 0 to λ, we get

$$\frac{\phi(\lambda)}{\lambda} \leq \int_0^\lambda f(t)/t^2 dt.$$

This last step is justified in part by the fact that $\lim_{t\to 0} \phi(t)/t = 0$ by L'Hopital's rule. Thus we have $\log \mathbb{E}\exp\{\lambda Y\} \leq \lambda \int_0^\lambda f(t)/t^2 dt$.

The upper tail: For the upper tail $Z - \mathbb{E}Z$, we have $f(t) = \mathbb{E}\psi(tE)$ where $E \sim \mathrm{Exp}(\theta)$ and $\psi(x) = 1 + (x-1)e^x$. By direct calculation, we have for $t < \theta$

$$\begin{aligned} \mathbb{E}\psi(tE) &= 1 + \mathbb{E}tE\exp(tE) - \mathbb{E}\exp(tE) \\ &= 1 - \frac{\theta}{\theta - t} + t\int_0^\infty x\exp(tx)\theta\exp(-\theta x)dx \\ &= 1 - \frac{\theta}{\theta - t} + \frac{t\theta}{(\theta - t)^2} = \frac{t^2}{(\theta - t)^2}. \end{aligned}$$

Thus, we get

$$\log \mathbb{E}\exp\{\lambda(Z - \mathbb{E}Z)\} \leq \lambda \int_0^\lambda \frac{1}{(\theta - t)^2} dt = \frac{\lambda^2}{\theta(\theta - \lambda)}.$$

If $\lambda \leq \theta/2$, this bound is $2\lambda^2/\theta^2$. Thus, according to definition 3, $Z - \mathbb{E}Z$ is sub-exponential with parameters $(4/\theta^2, 2/\theta)$.

The lower tail: For the lower tail $\mathbb{E}Z - Z$ we need to control $\mathbb{E}\tau(tE)$ where $E \sim \mathrm{Exp}(\theta)$, $\tau(x) = e^x - x - 1$. Direct calculation, using the moment generating function of exponential random variables gives

$$\mathbb{E}\tau(tE) = \frac{t^2}{\theta(\theta - t)}$$

So the integral bound is

$$\begin{aligned}\log \mathbb{E}\exp\{\lambda(\mathbb{E}Z - Z)\} &\leq \lambda \int_0^\lambda \frac{1}{\lambda(\lambda - t)} = \frac{\lambda}{\theta}\log\left(\frac{\theta}{\theta - \lambda}\right) \\ &= \frac{\lambda}{\theta}\left(\sum_{i=1}^{\infty}(\lambda/\theta)^i/i\right) \\ &= \frac{\lambda^2}{\theta^2}\left(\sum_{i=1}^{\infty}(\lambda/\theta)^{i-1}/i\right)\end{aligned}$$

If $\lambda/\theta \leq 1/2$ the series inside the paranthesis is clearly bounded by 2. Thus $\mathbb{E}Z - Z$ is sub-exponential with parameters $(4/\theta^2, 2/\theta)$ as before. □

Now that we have established that the maximum is sub-Exponential, we can bound the number of evaluations for the various methods. This is the main result for this section.

Theorem 64. *Let the time taken X for completing an evaluation to f be a random variable that is $\mathrm{Exp}(\theta)$ distributed. Let $\delta \in (0,1)$ and denote N_{syn} and N_{asy} denote the number of evaluations by synchronous and asynchronous algorithms with time T. Then with probability at least $1-\delta$, for any $\alpha \in (0,1)$ there exists $T_{\alpha,\delta}$ such that for all $T > T_{\alpha,\delta}$,*

$$\begin{aligned}N_{\text{seq}} &\in \left(\frac{T\theta}{(1+\alpha)} - 1, \frac{MT\theta}{(1-\alpha)}\right), \\ N_{\text{syn}} &\in \left(M\left(\frac{T\theta}{h_M(1+\alpha)} - 1\right), \frac{MT\theta}{h_M(1-\alpha)}\right) \\ N_{\text{asy}} &\in \left(M\left(\frac{T\theta}{(1+\alpha)} - 1\right), \frac{MT\theta}{(1-\alpha)}\right)\end{aligned}$$

Proof. In the synchronous setting, the k^{th} batch issues M jobs with lengths $(X_{k1}, \ldots, X_{kM})$ and the batch ends after $Y_k = \max_i X_{ki}$. Since the sequence of random variables $\{Y_k\}_{k\geq 1}$ are all iid and sub-exponential with parameters $(4/\theta^2, 2/\theta)$, in a similar way to the proof of Lemma 57, with $S_n = \sum_{k=1}^n Y_k$ we get that

$$\mathbb{P}\left(\exists n;\ |S_n - \mathbb{E}S_n| \geq \underbrace{\sqrt{8n\theta^{-2}\log(n^2\pi^2/(3\delta))} + \frac{2}{\theta}\log(n^2\pi^2/(3\delta))}_{\triangleq \epsilon_n}\right) \leq \delta$$

This follows from Bernstein's inequality (Proposition 13) and the union bound. As in Lemma 57 this means that:

$$\frac{Nh_M}{\theta} - \epsilon_N \leq S_N \leq T \leq S_{N+1} \leq \frac{(N+1)h_M}{\theta} + \epsilon_{N+1}.$$

Here we also used the fact that $\mathbb{E}Y_k = h_M/\theta$ from Lemma 60. Now assuming there exists $T_{\alpha,\delta}$ such that for all $T \geq T_{\alpha,\delta}$, we have $\epsilon_N \leq Nh_M\alpha/\theta$, we get

$$\frac{Nh_M}{\theta}(1-\alpha) \leq T \leq \frac{(N+1)h_m}{\theta}(1+\alpha).$$

The existence of $T_{\alpha,\delta}$ is based on the same argument as in Lemma 57. Re-arranging these inequalities, which gives a bound on the number of batches completed, leads to the bounds on the number of evaluations for the synchronous case.

Applying the same argument to a single worker on the sequence $\{X_k\}_{k\geq 1}$, we get

$$\frac{N_{\text{seq}}}{\theta}(1-\alpha) \leq T \leq \frac{N_{\text{seq}}+1}{\theta}(1+\alpha).$$

Repeating this argument for all M workers with $\delta \leftarrow \delta/M$ and then taking a union bound yields the result for N_{asy}. □

Putting it altogether

Finally, we put the results in Theorems 49, 53, 58 and 64 together to obtain the following result. This is a formal version of Theorem 48 in Chapter 5.3. Below, Ξ denotes the supremum of a GP of $\mathcal{X}$ which is a well defined and finite value [4].

Theorem 65. *Let $f \sim \mathcal{GP}(\mathbf{0}, \kappa)$ where $\kappa : \mathcal{X}^2 \to \mathbb{R}$ satisfies Assumption 1 and $\kappa(x, x') \leq 1$. Then for all $\alpha > 0$, the Bayes simple regret for* seqTS, synTS *and* asyTS, *satisfies the following for sufficiently large T.*

$$\begin{aligned}
\text{seqTS:} \quad \mathbb{E}[S'_T] &\leq \frac{C'_1}{n_{\text{seq}}} + \sqrt{\frac{C_2\beta_{n_{\text{seq}}}\Psi_{n_{\text{seq}}}}{n_{\text{seq}}}}, \\
\text{synTS:} \quad \mathbb{E}[S'_T] &\leq \frac{C'_1}{n_{\text{syn}}} + \frac{(M-1)\beta_{M-1}^{1/2}}{n_{\text{syn}}} + \sqrt{\frac{C_2\beta_{n_{\text{syn}}+M}\Psi_{n_{\text{syn}}}}{n_{\text{syn}}}}, \\
\text{asyTS:} \quad \mathbb{E}[S'_T] &\leq \frac{C'_1}{n_{\text{asy}}} + \frac{2\,\Xi\,\gamma_M}{n_{\text{asy}}} + \sqrt{\frac{C_2 C_\kappa \beta_{n_{\text{asy}}}\Psi_{n_{\text{asy}}}}{n_{\text{asy}}}}.
\end{aligned}$$

Here, n_{seq}, n_{syn}, and n_{asy} are defined as follows.

$$n_{\text{seq}} = \frac{T}{\theta(1+\alpha)} - 1, \quad n_{\text{syn}} = M\left[\frac{T}{\theta_M(1+\alpha)} - 1\right], \quad n_{\text{asy}} = M\left[\frac{T}{\theta(1+\alpha)} - 1\right].$$

Moreover, θ, θ_M are defined follows for the uniform, half-normal and exponential cases respec-

tively.

$$\begin{aligned}
\text{Unif}(a,b): &\qquad \theta = \frac{a+b}{2}, &\qquad \theta_M = \frac{a+bM}{M+1} \\
\mathcal{HN}(\zeta^2): &\qquad \theta = \frac{\zeta\sqrt{2}}{\sqrt{\pi}}, &\qquad \theta_M \in \zeta \cdot \Theta(\log(M)) \\
\text{Exp}(\lambda): &\qquad \theta = \frac{1}{\lambda}, &\qquad \theta_M = \frac{h_M}{\lambda}
\end{aligned}$$

Further, Ψ_n is the MIG in Definition 1, β_n is as defined in (5.4), Ξ is the expected supremum of the GP from Lemma 6, ξ_M is from (5.3), and $C_1 = \pi^2/6 + \sqrt{2\pi}/12 + 1$, $C_2 = 2/\log(1+\eta^{-2})$ are constants.

Proof. We will prove the result for the asynchronous case and note that the other two results are obta similar argument. Let $\mathcal{V}$ denote the event $N \geq M\left[\frac{T}{\theta(1+\alpha)} - 1\right]$, i.e. the random number of plays has exceeded the given bound above. For any given $\delta \in (0,1)$, Theorems 58 and 64 allow us to control the probability of the event below δ. We will choose $\delta = \frac{1}{2\,\Xi\,n_{\text{asy}}}$ where Ξ is the expected supremum of the GP in Lemma 6. Since the randomness in the evaluation times are independent of the prior, noise and the algorithm, we can decompose $\mathbb{E}[S'_T]$ as follows and use the result in (5.10) for $\mathbb{E}[S_n]$.

$$\begin{aligned}
\mathbb{E}[S'_T] &\leq \mathbb{E}[\mathbb{E}[S_N]|\mathcal{V}]\mathbb{P}(\mathcal{V}) + \mathbb{E}[\mathbb{E}[S_N]|\mathcal{V}^c]\mathbb{P}(\mathcal{V}^c) \\
&\leq \mathbb{E}[S_{n_{\text{asy}}}] \cdot 1 \;+\; 2\,\Xi\,\delta
\end{aligned}$$

Here we have used the definition of the Bayes simple regret with time in (5.2) which guarantees that it is never worse than $\sup_x |f(x_\star) - f(x)| \leq 2\,\Xi$. The theorem follows by plugging in values for $\mathbb{E}[S_n]$ and δ. The "sufficiently large T" requirement is because Theorems 58 and 64 hold only for $T > T_{\alpha,\delta} = T_{\alpha,\frac{1}{2\,\Xi\,n_{\text{asy}}}}$. Since the dependencies of δ on $T_{\alpha,\delta}$ is polylogarithmic, and as n_{asy} is growing linearly with T, the above condition is equivalent to $T \gtrsim \text{polylog}(T)$ which is achievable for large enough T. □

Chapter 6

Bandits on Graph-structured Domains: An Example in Neural Architecture Search

Bayesian optimisation methods have seen tremendous success in several optimisation problems in optimal policy search, industrial design, and scientific experimentation, where function evaluations are typically expensive. That said, the quintessential use case for BO in machine learning is *model selection* [103, 232]. For instance, consider selecting the regularisation parameter λ and kernel bandwidth h for an SVM. We can set this up as a zeroth order optimisation problem where our domain is a two dimensional space of (λ, h) values, and each function evaluation trains the SVM on a training set, and computes the accuracy on a validation set. The goal is to find the model, i.e. hyperparameters, with the highest validation accuracy.

The majority of the bandit and BO literature has focused on settings where the domain $\mathcal{X}$ is either Euclidean or categorical. This suffices for many tasks, such as the SVM example above. However, with recent successes in deep learning, neural networks are increasingly becoming the method of choice for many machine learning applications. A number of recent work have designed novel neural network architectures to significantly outperform the previous state of the art [91, 101, 228, 243]. This motivates studying model selection over the space of neural architectures to optimise for generalisation performance. A critical challenge in this endeavour is that evaluating a network via train and validation procedures is very expensive.

To motivate the ensuing discussion, recall that in its most unadorned form, a BO algorithm operates sequentially, starting at time 0 with a GP prior for f; at time t, it incorporates results of evaluations from $1, \ldots, t-1$ in the form of a posterior for f. It then uses this posterior to construct an acquisition function φ_t, where $\varphi_t(x)$ is a measure of the value of evaluating f at x at time t if our goal is to maximise f. While we have encountered upper confidence bounds and Thompson sampling as possible strategies for constructing φ_t frequently in this thesis, there are several other examples as mentioned in Chapter 2.2. Accordingly, BO chooses to evaluate f at the maximiser of the acquisition, i.e. $x_t = \operatorname{argmax}_{x \in \mathcal{X}} \varphi_t(x)$. There are two key ingredients to realising this plan for GP based BO. First, we need to quantify the similarity between two points x, x' in the domain in the form of a kernel $\kappa(x, x')$. The kernel is needed to define the

GP, which allows us to reason about an unevaluated value $f(x')$ when we have already evaluated $f(x)$. Secondly, we need a method to maximise φ_t.

These two steps are fairly straightforward in conventional domains. For example, in Euclidean spaces, we can use one of many popular kernels such as Gaussian, Laplacian, or Matérn ; we can maximise φ_t via off the shelf branch-and-bound or gradient based methods. However, when each $x \in \mathcal{X}$ is a neural network architecture, this is not the case. Hence, our challenges in this work are two-fold. First, we need to *quantify (dis)similarity between two networks*. Intuitively, in Figure 6.1, network 6.1(a) is more similar to network 6.1(b), than it is to 6.1(c). Secondly, we need to be able to traverse the space of such networks to *optimise the acquisition function*. Our main contributions are as follows.

1. We develop a (pseudo-)distance for neural network architectures called OTMANN (Optimal Transport Metrics for Architectures of Neural Networks) that can be computed efficiently via an optimal transport program.
2. We develop a BO framework for optimising functions on neural network architectures called NASBOT (Neural Architecture Search with Bayesian Optimisation and Optimal Transport). This includes an evolutionary algorithm to optimise the acquisition function.
3. Empirically, we demonstrate that NASBOT outperforms other baselines on model selection tasks for multi-layer perceptrons (MLP) and convolutional neural networks (CNN).

Related Work

Historically, evolutionary (genetic) algorithms (EA) have been the most common method used for designing architectures [58, 138, 156, 173, 205, 237, 273]. EA techniques are popular as they provide a simple mechanism to explore the space of architectures by making a sequence of changes to networks that have already been evaluated. However, as we will discuss shortly, EA algorithms, while conceptually and computationally simple, are typically not best suited for optimising functions that are expensive to evaluate. A related line of work first sets up a search space for architectures via incremental modifications, and then explores this space via random exploration, MCTS, or A* search [46, 155, 178]. Some of the methods above can only optimise among feed forward structures, e.g. Figure 6.1(a), but cannot handle spaces with arbitrarily structured networks, e.g. Figures 6.1(b), 6.1(c).

The most successful recent architecture search methods that can handle arbitrary structures have adopted reinforcement learning (RL) [12, 281, 282, 283]. However, architecture search is in essence an *optimisation* problem – find the network with the highest function value. There is no explicit need to maintain a notion of state and solve the credit assignment problem in RL [240]. Since RL is fundamentally more difficult than optimisation [109], these methods typically need to try a very large number of architectures to find the optimum. This is not desirable, especially in computationally constrained settings.

None of the above methods have been designed with a focus on the expense of evaluating a neural network, with an emphasis on being judicious in selecting which architecture to try next.

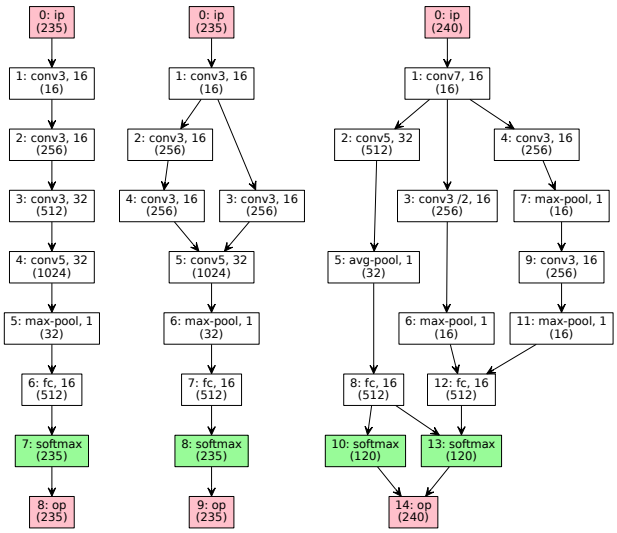
Bayesian optimisation (BO) techniques are well suited for expensive evaluations. They maintain introspective models of the function and determine future points for evaluation via predictions and uncertainty estimates from this model. Hence, BO usually consumes more computation to determine future points than the methods above, but this pays dividends when evaluating f itself is extremely expensive. However, most BO methods are designed for Euclidean or categorical domains. Snoek et al. [232] and Jenatton et al. [108] use BO to design neural architectures, but they can only handle feed forward structures.

One of the key technical contributions of this work is the OTMANN distance for neural architectures, which views a neural network as a graph and compares them via an optimal transport (OT) program [253]. This distance is inspired by Wasserstein (earth mover's) distances which also have an OT formulation. As we will discuss shortly, while OTMANN has similarities to Wasserstein, it is not a Wasserstein distance itself. There have also been prior work defining various distances and kernels on graphs [68, 172, 255, 256]. We cannot directly apply them in our setting because neural networks are more complex objects; in addition to the graphical structure, neural networks are also defined by the type of operations performed at each layer, the number of neurons, etc. Moreover, the computation of the above distances are more expensive than OTMANN.

6.1 Problem Set Up

Our goal is to maximise a function f defined on a space $\mathcal{X}$ of neural network architectures. When we evaluate f at $x \in \mathcal{X}$, we obtain a possibly noisy observation y of $f(x)$. In the context of architecture search, f is the performance on a validation set after x is trained on the training set. If $x_\star = \text{argmax}_{\mathcal{X}} f(x)$ is the optimal architecture, and x_t is the architecture evaluated at time t, we want the simple regret $S_n = f(x_\star) - \max_{t \leq n} f(x_t)$ to vanish fast as the number of evaluations $n \to \infty$.

GP/BO in the context of architecture search: We will begin by contextualising GPs and BOs in our domain $\mathcal{X}$ of neural architectures. Recall that the $\kappa(x, x')$ is a measure of similarity between x and x'. If $\kappa(x, x')$ is large, then $f(x)$ and $f(x')$ are highly correlated. Hence, the GP effectively imposes a smoothness condition on $f : \mathcal{X} \to \mathbb{R}$; i.e. since networks a and b in Figure 6.1 are similar, they are likely to have similar cross validation performance. Moreover, recall that in BO, when selecting the next point, we balance between *exploitation*, choosing points that we believe will have high f value, and *exploration*, choosing points that we do not know much about so that we do not get stuck at a bad optimum. For example, if we have already evaluated $f(\mathrm{a})$, then exploration incentivises us to choose c over b since we can reasonably gauge $f(\mathrm{b})$ from $f(\mathrm{a})$. On the other hand, if $f(\mathrm{a})$ has high value, then exploitation incentivises choosing b, as it is more likely to be the optimum than c.



(a) (b) (c)

Figure 6.1: An illustration of some CNN architectures. In each layer, i: indexes the layer, followed by the label (e.g `conv3`), and then the number of units (e.g. number of filters). The input and output layers are pink while the decision (`softmax`) layers are green.
From Chapter 6.2: The layer mass is denoted in parentheses. The following are the normalised and unnormalised OTMANN distances $d, \bar{d}$. All self distances are 0, i.e. $d(\mathcal{G},\mathcal{G}) = \bar{d}(\mathcal{G},\mathcal{G}) = 0$. The unnormalised distances are, $d(\mathrm{a},\mathrm{b}) = 175.1$, $d(\mathrm{a},\mathrm{c}) = 1479.3$, $d(\mathrm{b},\mathrm{c}) = 1621.4$. The normalised distances are, $\bar{d}(\mathrm{a},\mathrm{b}) = 0.0286$, $\bar{d}(\mathrm{a},\mathrm{c}) = 0.2395$, $\bar{d}(\mathrm{b},\mathrm{c}) = 0.2625$.

6.1.1 A Graph-theoretic Formalism for Neural Networks

Our formalism will view a neural network as a graph whose vertices are the layers of the network. We will use the CNNs in Fig. 6.1 to illustrate the concepts. A neural network $\mathcal{G} = (\mathcal{L}, \mathcal{E})$ is defined by a set of layers $\mathcal{L}$ and directed edges $\mathcal{E}$. An edge $(u, v) \in \mathcal{E}$ is a ordered pair of layers. In Fig. 6.1, the layers are depicted by rectangles and the edges by arrows. A layer $u \in \mathcal{L}$ is equipped with a layer label $\ell\ell(u)$ which denotes the type of operations performed at the layer. For instance, in Fig. 6.1(a), $\ell\ell(1) = \texttt{conv3}$, $\ell\ell(5) = \texttt{max-pool}$ denote a 3×3 convolution and a max-pooling operation. The attribute ℓu denotes the number of computational units in a layer. In Fig. 6.1(b), $\ell u(5) = 32$ and $\ell u(7) = 16$ are the number of convolutional filters and fully connected nodes.

In addition, each network has *decision layers* which are used to obtain the predictions of the network. For a classification task, the decision layers perform `softmax` operations and output the probabilities an input datum belongs to each class. For regression, the decision layers perform `linear` combinations of the outputs of the previous layers and output a single scalar. All networks have at least one decision layer. When a network has multiple decision layers, we average the output of each decision layer to obtain the final output. The decision layers are shown in green in Fig. 6.1. Finally, every network has a unique *input layer* u_{ip} and *output layer* u_{op} with labels $\ell\ell(u_{\text{ip}}) = \texttt{ip}$ and $\ell\ell(u_{\text{op}}) = \texttt{op}$. It is instructive to think of the role of u_{ip} as feeding a data point to the network and the role of u_{op} as averaging the results of the decision layers. The input and output layers are shown in pink in Fig. 6.1. We refer to all layers that are not input, output or decision layers as *processing layers*.

The directed edges are to be interpreted as follows. The output of each layer is fed to each of its children; so both layers 2 and 3 in Fig. 6.1(b) take the output of layer 1 as input. When a layer has multiple parents, the inputs are concatenated; so layer 5 sees an input of $16+16$ filtered channels coming in from layers 3 and 4. Finally, we mention that neural networks are also characterised

by the values of the weights/parameters between layers. In architecture search, we typically do not consider these weights. Instead, an algorithm will (somewhat ideally) assume access to an oracle that can minimise the loss function on the training set and find the optimal weights.

We next describe a distance $d : \mathcal{X}^2 \to \mathbb{R}_+$ for neural architectures. Recall that our eventual goal is a kernel for the GP; given a distance d, we will aim for $\kappa(x, x') = e^{-\beta d(x,x')^p}$, where $\beta, p \in \mathbb{R}_+$, as the kernel. Many popular kernels take this form. For e.g. when $\mathcal{X} \subset \mathbb{R}^n$ and d is the L^2 norm, $p = 1, 2$ correspond to the Laplacian and Gaussian kernels respectively.

6.2 OTMANN: Optimal Transport Metrics for Architectures of Neural Networks

To motivate this distance, note that the performance of a neural network is determined by the amount of computation at each layer, the types of these operations, and how the layers are connected. A meaningful distance should account for these factors. To that end, OTMANN is defined as the minimum of a matching scheme which attempts to match the computation at the layers of one network to the layers of the other. We incur penalties for matching layers with different types of operations or those at structurally different positions. We will find a matching that minimises these penalties, and the total penalty at the minimum will give rise to a distance. We first describe two concepts, layer masses and path lengths, which we will use to define OTMANN.

6.2.1 Preliminaries

Layer masses: The layer masses $\ell m : \mathcal{L} \to \mathbb{R}_+$ will be the quantity that we match between the layers of two networks when comparing them. $\ell m(u)$ quantifies the significance of layer u. For processing layers, $\ell m(u)$ will represent the amount of computation carried out by layer u and is computed via the product of $\ell u(u)$ and the number of incoming units. For example, in Fig. 6.1(b), $\ell m(5) = 32 \times (16 + 16)$ as there are 16 filtered channels each coming from layers 3 and 4 respectively. Assigning mass for the input, output, is not as straightforward since there is no computation at these layers. However, they occupy a significant role in the architecture. Hence, we use $\ell m(u_{\text{ip}}) = \ell m(u_{\text{op}}) = \zeta \sum_{u \in \mathcal{PL}} \ell m(u)$ where $\mathcal{PL}$ denotes the set of processing layers, and $\zeta \in (0, 1)$ is a parameter to be determined. Intuitively, we are using an amount of mass that is proportional to the amount of computation in the processing layers. Similarly, the decision layers occupy a significant role in the architecture as they directly influence the output. While there is computation being performed at these layers, this might be problem dependent – there is more computation performed at the softmax layer in a 10 class classification problem than in a 2 class problem. Following the same intuition as we did for the input/output layers, we assign an amount of mass proportional to the mass in the processing layers. Since the outputs of the decision layers are averaged, we distribute the mass among all decision layers; that is, if $\mathcal{DL}$ are decision layers, $\forall\, u \in \mathcal{DL}, \ell m(u) = \frac{\zeta}{|\mathcal{DL}|} \sum_{u \in \mathcal{PL}} \ell m(u)$. Our motivations for setting the layer masses for the input, output, and decision layers using such heuristics primarily stem from

practical performance considerations – we found that not assigning significant layer masses for these layers tended to equate different architectures and consequently result in poor performance. In all our experiments, we use $\zeta = 0.1$. In Fig. 6.1, the layer masses for each layer are shown in parantheses.

Path lengths from/to u_{ip}/u_{op}: In a neural network $\mathcal{G}$, a path from u to v is a sequence of layers $u_1, \ldots, u_s$ where $u_1 = u$, $u_s = v$ and $(u_i, u_{i+1}) \in \mathcal{E}$ for all $i \leq s-1$. The length of this path is the number of hops from one node to another in order to get from u to v. For example, in Fig. 6.1(c), $(2, 5, 8, 13)$ is a path from layer 2 to 13 of length 3. Let the shortest (longest) path length from u to v be the smallest (largest) number of hops from one node to another among all paths from u to v. Additionally, define the random walk path length as the expected number of hops to get from u to v, if, from any layer we hop to one of its children chosen uniformly at random. For example, in Fig. 6.1(c), the shortest, longest and random walk path lengths from layer 1 to layer 14 are 5, 7, and 5.67 respectively. For any $u \in \mathcal{L}$, let $\delta_{\text{op}}^{\text{sp}}(u), \delta_{\text{op}}^{\text{lp}}(u), \delta_{\text{op}}^{\text{rw}}(u)$ denote the length of the shortest, longest and random walk paths from u to the output u_{op}. Similarly, let $\delta_{\text{ip}}^{\text{sp}}(u), \delta_{\text{ip}}^{\text{lp}}(u), \delta_{\text{ip}}^{\text{rw}}(u)$ denote the corresponding lengths for walks from the input u_{ip} to u. As the layers of a neural network can be topologically ordered[1], the above path lengths are well defined and finite. Further, for any $s \in \{\text{sp,lp,rw}\}$ and $t \in \{\text{ip,op}\}$, $\delta_t^s(u)$ can be computed for all $u \in \mathcal{L}$, in $\mathcal{O}(|\mathcal{E}|)$ time (see Chapter 6.3.3 for details).

6.2.2 Description of OTMANN

We are now ready to describe OTMANN. Given two networks $\mathcal{G}_1 = (\mathcal{L}_1, \mathcal{E}_1), \mathcal{G}_2 = (\mathcal{L}_2, \mathcal{E}_2)$ with n_1, n_2 layers respectively, we will attempt to match the layer masses in both networks. We let $Z \in \mathbb{R}_+^{n_1 \times n_2}$ be such that $Z(i,j)$ denotes the amount of mass matched between layer $i \in \mathcal{G}_1$ and $j \in \mathcal{G}_2$. The OTMANN distance is computed by solving the following optimisation problem.

$$\begin{aligned} &\underset{Z}{\text{minimise}} \quad \phi_{\text{lmm}}(Z) + \phi_{\text{nas}}(Z) + \nu_{\text{str}}\phi_{\text{str}}(Z) \\ &\text{subject to} \ \sum_{j \in \mathcal{L}_2} Z_{ij} \leq \ell m(i), \ \sum_{i \in \mathcal{L}_1} Z_{ij} \leq \ell m(j), \ \forall i, j \end{aligned} \tag{6.1}$$

The label mismatch term ϕ_{lmm}, penalises matching masses that have different labels, while the structural term ϕ_{str} penalises matching masses at structurally different positions with respect to each other. If we choose not to match any mass in either network, we incur a non-assignment penalty ϕ_{nas}. $\nu_{\text{str}} > 0$ determines the trade-off between the structural and other terms. The inequality constraints ensure that we do not over assign the masses in a layer. We now describe $\phi_{\text{lmm}}, \phi_{\text{nas}}$, and ϕ_{str}.

Label mismatch penalty ϕ_{lmm}: We begin with a label penalty matrix $M \in \mathbb{R}^{L \times L}$ where L is the number of all label types and $M(\mathtt{x}, \mathtt{y})$ denotes the penalty for transporting a unit mass from a layer with label $\mathtt{x}$ to a layer with label $\mathtt{y}$. We then construct a matrix $C_{\text{lmm}} \in \mathbb{R}^{n_1 \times n_2}$ with $C_{\text{lmm}}(i,j) = M(\ell\ell(i), \ell\ell(j))$ corresponding to the mislabel cost for matching unit mass from

[1] A topological ordering is an ordering of the layers $u_1, \ldots, u_{|\mathcal{L}|}$ such that u comes before v if $(u, v) \in \mathcal{E}$.

	conv3	conv5	max-pool	avg-pool	fc
conv3	0	0.2	∞	∞	∞
conv5	0.2	0	∞	∞	∞
max-pool	∞	∞	0	0.25	∞
avg-pool	∞	∞	0.25	0	∞
fc	∞	∞	∞	∞	0

Table 6.1: An example label mismatch cost matrix M. There is zero cost for matching identical layers, < 1 cost for similar layers, and infinite cost for disparate layers.

each layer $i \in \mathcal{L}_1$ to each layer $j \in \mathcal{L}_2$. We then set $\phi_{\text{lmm}}(Z) = \langle Z, C_{\text{lmm}} \rangle = \sum_{i \in \mathcal{L}_1, j \in \mathcal{L}_2} Z(i,j)C(i,j)$ to be the sum of all matchings from $\mathcal{L}_1$ to $\mathcal{L}_2$ weighted by the label penalty terms. This matrix M, illustrated in Table 6.1, is a parameter that needs to be specified for OTMANN. They can be specified with an intuitive understanding of the functionality of the layers; e.g. many values in M are ∞, while for similar layers, we choose a value less than 1.

Non-assignment penalty ϕ_{nas}: We set this to be the amount of mass that is unassigned in both networks, i.e. $\phi_{\text{nas}}(Z) = \sum_{i \in \mathcal{L}_1} \left(\ell m(i) - \sum_{j \in \mathcal{L}_2} Z_{ij}\right) + \sum_{j \in \mathcal{L}_2} \left(\ell m(j) - \sum_{i \in \mathcal{L}_1} Z_{ij}\right)$. This essentially implies that the cost for not assigning unit mass is 1. The costs in Table 6.1 are defined relative to this. For similar layers $\mathtt{x}$, $\mathtt{y}$, $M(\mathtt{x},\mathtt{y}) \ll 1$ and for disparate layers $M(\mathtt{x},\mathtt{y}) \gg 1$. That is, we would rather match `conv3` to `conv5` than not assign it, provided the structural penalty for doing so is small; conversely, we would rather not assign a `conv3`, than assign it to `fc`. This also explains why we did not use a trade-off parameter like ν_{str} for ϕ_{lmm} and ϕ_{nas} – it is simple to specify reasonable values for $M(\mathtt{x},\mathtt{y})$ from an understanding of their functionality.

Structural penalty ϕ_{str}: We define a matrix $C_{\text{str}} \in \mathbb{R}^{n_1 \times n_2}$ where $C_{\text{str}}(i,j)$ is small if layers $i \in \mathcal{L}_1$ and $j \in \mathcal{L}_2$ are at structurally similar positions in their respective networks. We then set $\phi_{\text{str}}(Z) = \langle Z, C_{\text{str}} \rangle$. For $i \in \mathcal{L}_1,\ j \in \mathcal{L}_2$, we let $C_{\text{str}}(i,j) = \frac{1}{6} \sum_{s \in \{\text{sp, lp, rw}\}} \sum_{t \in \{\text{ip,op}\}} |\delta_t^s(i) - \delta_t^s(j)|$ be the average of all path length differences, where δ_t^s are the path lengths defined previously. We define ϕ_{str} in terms of the shortest/longest/random-walk path lengths from/to the input/output, because they capture various notions of information flow in a neural network; a layer's input is influenced by the paths the data takes before reaching the layer and its output influences all layers it passes through before reaching the decision layers. If the path lengths are similar for two layers, they are likely to be at similar structural positions. Further, this form allows us to solve (6.1) efficiently via an OT program and prove distance properties about the solution. If we need to compute pairwise distances for several networks, as is the case in BO, the path lengths can be pre-computed in $\mathcal{O}(|\mathcal{E}|)$ time, and used to construct C_{str} for two networks at the moment of computing the distance between them.

This completes the description of our matching program. Shortly, we will see how it can be computed via an optimal transport scheme using efficient solvers [195]. Our theorem below, shows that the solution of (6.1) is a pseudo-distance under some mild regularity conditions on the label penalty matrix M which is under our control and easy to satisfy.

Theorem 66. *Assume that the mislabel cost matrix M satisfies the triangle inequality; i.e. for all labels $\mathtt{x}$, $\mathtt{y}$, $\mathtt{z}$ we have $M(\mathtt{x},\mathtt{z}) \leq M(\mathtt{x},\mathtt{y}) + M(\mathtt{y},\mathtt{z})$. Let $d(\mathcal{G}_1, \mathcal{G}_2)$ be the solution of (6.1) for networks $\mathcal{G}_1, \mathcal{G}_2$. Then $d(\cdot,\cdot)$ is a pseudo-distance. That is, for all networks $\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3$, it satisfies, $d(\mathcal{G}_1, \mathcal{G}_2) > 0$, $d(\mathcal{G}_1, \mathcal{G}_2) = d(\mathcal{G}_2, \mathcal{G}_1)$, $d(\mathcal{G}_1, \mathcal{G}_1) = 0$ and $d(\mathcal{G}_1, \mathcal{G}_3) \leq d(\mathcal{G}_1, \mathcal{G}_2) + d(\mathcal{G}_2, \mathcal{G}_3)$.*

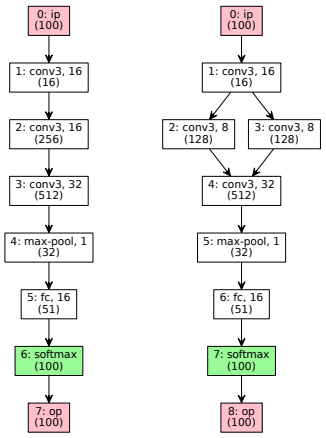


Figure 6.2: An example of 2 CNNs which have $d = \bar{d} = 0$ distance. The OT solution matches the mass in each layer in the network on the left to the layer horizontally opposite to it on the right with 0 cost. For layer 2 on the left, its mass is mapped to layers 2 and 3 on the left. However, while the descriptor of these networks is different, their functional behaviour is the same.

Some remarks are in order. First, observe that while $d(\cdot,\cdot)$ is a pseudo-distance, it is not a distance; i.e. $d(\mathcal{G}_1,\mathcal{G}_2) = 0 \not\Rightarrow \mathcal{G}_1 = \mathcal{G}_2$. For example, while the networks in Figure 6.2 have different descriptors according to our formalism in Chapter 6.1.1, their distance is 0. However, it is not hard to see that their functionality is the same – in both cases, the output of layer 1 is passed through 16 `conv3` filters and then fed to a layer with 32 `conv3` filters – and hence, this property is desirable in this example. It is not yet clear however, if the topology induced by our metric equates two functionally dissimilar networks. We leave it to future work to study equivalence classes induced by the OTMANN distance. Second, despite the OT formulation, this is not a Wasserstein distance. In particular, the supports of the masses and the cost matrices change depending on the two networks being compared.

The normalised OTMANN distance: For what follows, define $\bar{d}(\mathcal{G}_1,\mathcal{G}_2) = d(\mathcal{G}_1,\mathcal{G}_2)/(tm(\mathcal{G}_1)+tm(\mathcal{G}_2))$ where $tm(\mathcal{G}_i) = \sum_{u\in\mathcal{L}_i} \ell m(u)$ is the total mass of a network. Note that $\bar{d} \leq 1$. While $\bar{d}$ does not satisfy the triangle inequality, it provides a useful measure of dissimilarity normalised by the amount of computation. Our experience suggests that d puts more emphasis on the amount of computation at the layers over structure and vice versa for $\bar{d}$. Therefore, it is prudent to combine both quantities in any downstream application. The caption in Fig. 6.1 gives $d, \bar{d}$ values for the examples in that figure when $\nu_{\text{str}} = 0.5$.

Some Remarks

We conclude this section with a couple of remarks.

Masses on the decision & input/output layers: It is natural to ask why one needs to model the mass in the decision and input/output layers. For example, a seemingly natural choice is to use 0 for these layers. Using 0 mass, is a reasonable strategy if we were to allow only one decision layer. However, when there are multiple decision layers, consider comparing the following two networks: the first has a feed forward MLP with non-linear layers, the second is the same network but with an additional linear decision layer u, with one edge from u_{ip} to u and an edge from u to u_{op}. This latter models the function as a linear + non-linear term which might be suitable for some problems unlike modeling it only as a non-linear term. If we do not add layer masses for

the input/output/decision layers, then the distance between both networks would be 0 - as there will be equal mass in the FF part for both networks and they can be matched with 0 cost.

Connections to other distances: OTMANN shares similarities with Wasserstein (earth mover's) distances which also have an OT formulation. However, it is not a Wasserstein distance itself—in particular, the supports of the masses and the cost matrices change depending on the two networks being compared. Second, while there has been prior work for defining various distances and kernels on graphs, we cannot use them in BO because neural networks have additional complex properties in addition to graphical structure, such as the type of operations performed at each layer, the number of neurons, etc. The above work either define the distance/kernel between vertices or assume the same vertex (layer) set [68, 141, 172, 230, 256], none of which apply in our setting. While some methods do allow different vertex sets [255], they cannot handle layer masses and layer similarities. Moreover, the computation of the above distances are more expensive than OTMANN. Hence, these methods cannot be directly plugged into BO framework for architecture search. We next describe the optimal transport reformulation of OTMANN.

6.2.3 Optimal Transport Reformulation

We begin with a review optimal transport. Throughout this section, $\langle \cdot, \cdot \rangle$ denotes the Frobenius dot product. $\mathbf{1}_n, \mathbf{0}_n \in \mathbb{R}^n$ denote a vector of ones and zeros respectively.

A review of Optimal Transport [253]: Let $y_1 \in \mathbb{R}_+^{n_1}, y_2 \in \mathbb{R}_+^{n_2}$ be such that $\mathbf{1}_{n_1}^\top y_1 = \mathbf{1}_{n_2}^\top y_2$. Let $C \in \mathbb{R}_+^{n_1 \times n_2}$. The following optimisation problem,

$$\begin{aligned} \underset{Z}{\text{minimise}} \quad & \langle Z, C \rangle \\ \text{subject to} \quad & Z > 0, \;\; Z\mathbf{1}_{n_2} = y_1, \;\; Z^\top \mathbf{1}_{n_1} = y_2. \end{aligned} \tag{6.2}$$

is called an *optimal transport* program. One interpretation of this set up is that y_1 denotes the supplies at n_1 warehouses, y_2 denotes the demands at n_2 retail stores, C_{ij} denotes the cost of transporting a unit mass of supplies from warehouse i to store j and Z_{ij} denotes the mass of material transported from i to j. The program attempts to find transportation plan which minimises the total cost of transportation $\langle Z, C \rangle$.

OT formulation of (6.1): In addition to providing an efficient way to solve (6.1), the OT formulation will allow us to prove the metric properties of the solution (Theorem 66). When computing the distance between $\mathcal{G}_1, \mathcal{G}_2$, for $i = 1, 2$, let $tm(\mathcal{G}_i) = \sum_{u \in \mathcal{L}_i} \ell m(u)$ denote the total mass in $\mathcal{G}_i$, and $\bar{n}_i = n_i + 1$ where $n_i = |\mathcal{L}_i|$. $y_1 = [\{\ell m(u)\}_{u \in \mathcal{L}_1}, tm(\mathcal{G}_2)] \in \mathbb{R}^{\bar{n}_1}$ will be the supplies in our OT problem, and $y_2 = [\{\ell m(u)\}_{u \in \mathcal{L}_2}, tm(\mathcal{G}_1)] \in \mathbb{R}^{\bar{n}_2}$ will be the demands. To define the cost matrix, we augment the mislabel and structural penalty matrices $C_{\text{lmm}}, C_{\text{str}}$ with an additional row and column of zeros; i.e. $C'_{\text{lmm}} = [C_{\text{lmm}} \, \mathbf{0}_{n_1}; \mathbf{0}_{\bar{n}_2}^\top] \in \mathbb{R}^{\bar{n}_1 \times \bar{n}_2}$; C'_{str} is defined similarly. Let $C'_{\text{nas}} = [\mathbf{0}_{n_1, n_2} \, \mathbf{1}_{n_1}; \mathbf{1}_{n_2}^\top \, 0] \in \mathbb{R}^{\bar{n}_1 \times \bar{n}_2}$. Now define $C' = C'_{\text{lmm}} + C'_{\text{str}} + C'_{\text{nas}}$. We show that (6.1) is equivalent to the following OT program.

$$\begin{aligned} \underset{Z'}{\text{minimise}} \quad & \langle Z', C' \rangle \\ \text{subject to} \quad & Z'\mathbf{1}_{\bar{n}_2} = y_1, \quad Z'^\top \mathbf{1}_{\bar{n}_1} = y_2. \end{aligned} \tag{6.3}$$

One interpretation of (6.3) is that the last row/column appended to the cost matrices serve as a non-assignment layer and that the cost for transporting unit mass to this layer from all other layers is 1. The costs for mislabelling was defined relative to this non-assignment cost. The costs for similar layers is much smaller than 1; therefore, the optimiser is incentivised to transport mass among similar layers rather than not assign it provided that the structural penalty is not too large. Correspondingly, the cost for very disparate layers is much larger so that we would never match, say, a convolutional layer with a pooling layer. In fact, the ∞'s in Table 6.1 can be replaced by any value larger than 2 and the solution will be the same. The following theorem shows that (6.1) and (6.3) are equivalent.

Theorem 67. *Problems* (6.1) *and* (6.3) *are equivalent, in that they both have the same minimum and we can recover the solution of one from the other.*

6.2.4 Some Illustrations of the OTMANN distance

We illustrate that OTMANN computes reasonable distances on neural network architectures via a two-dimensional t-SNE visualisation [164] of the architectures. Given a distance matrix between m objects, t-SNE embeds them in a d dimensional space so that objects with small distances are placed closer to each other. Figure 6.3 shows the t-SNE embedding using the OTMANN distance and its noramlised version. We have indexed 13 networks in both figures in a-n and displayed their architectures in Figure 6.4. Similar networks are placed close to each other indicating that OTMANN induces a meaningful topology among neural network architectures.

Next, we show that the distances induced by OTMANN are correlated with validation error performance. In Figure 6.5 we provide the following scatter plot for networks trained for three different datasets, indoor location [247], naval propulsion [45], and slice localisation [84] datasets. Each point in the figure is for pair of networks. The x-axis is the OTMANN distance between the pair and the y-axis is the difference in the validation error on the dataset. In each figure we used 300 networks giving rise to $45K$ pairwise points in each scatter plot. As the figure indicates, when the distance is small the difference in performance is close to 0. However, as the distance increases, the points are more scattered. Intuitively, one should expect that while networks that are far apart could perform similarly or differently, similar networks should perform similarly. Hence, OTMANN induces a useful topology in the space of architectures that is smooth for validaiton performance on real world datasets. This demonstrates that it can be incorporated in a BO framework to optimise a network based on its validation error.

6.3 NASBOT: Neural Architecture Search with Bayesian Optimisation & Optimal Transport

We now describe NASBOT, our BO algorithm for neural architecture search. Recall that in order to realise the BO scheme outlined above, we need to specify (a) a kernel κ for neural architectures

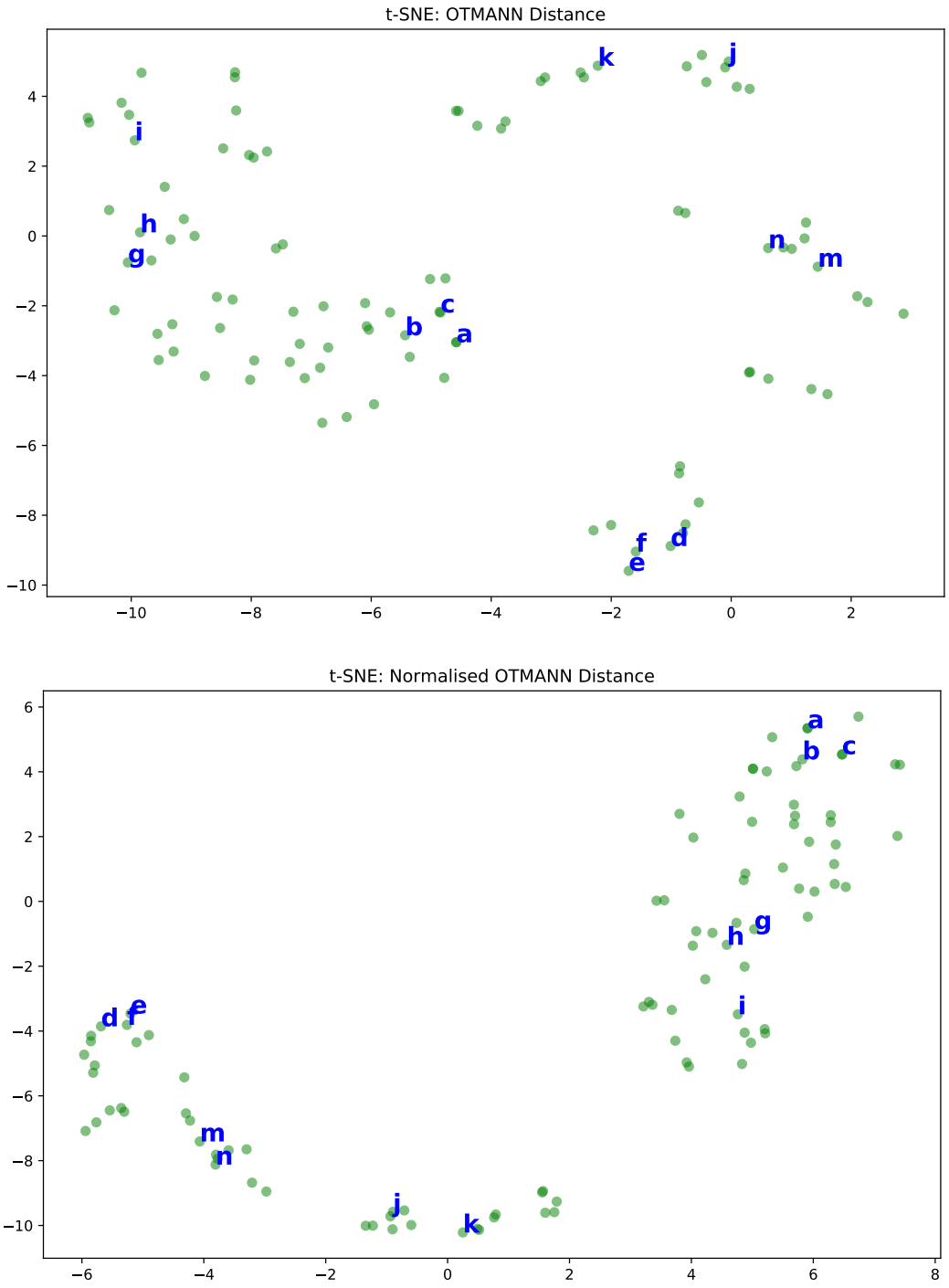


Figure 6.3: Two dimensional t-SNE embeddings of 100 randomly generated CNN architectures based on the OTMANN distance (top) and its normalised version (bottom). Some networks have been indexed a-n in the figures; these network architectures are illustrated in Figure 6.4. Networks that are similar are embedded close to each other indicating that the OTMANN induces a meaningful topology among neural network architectures.

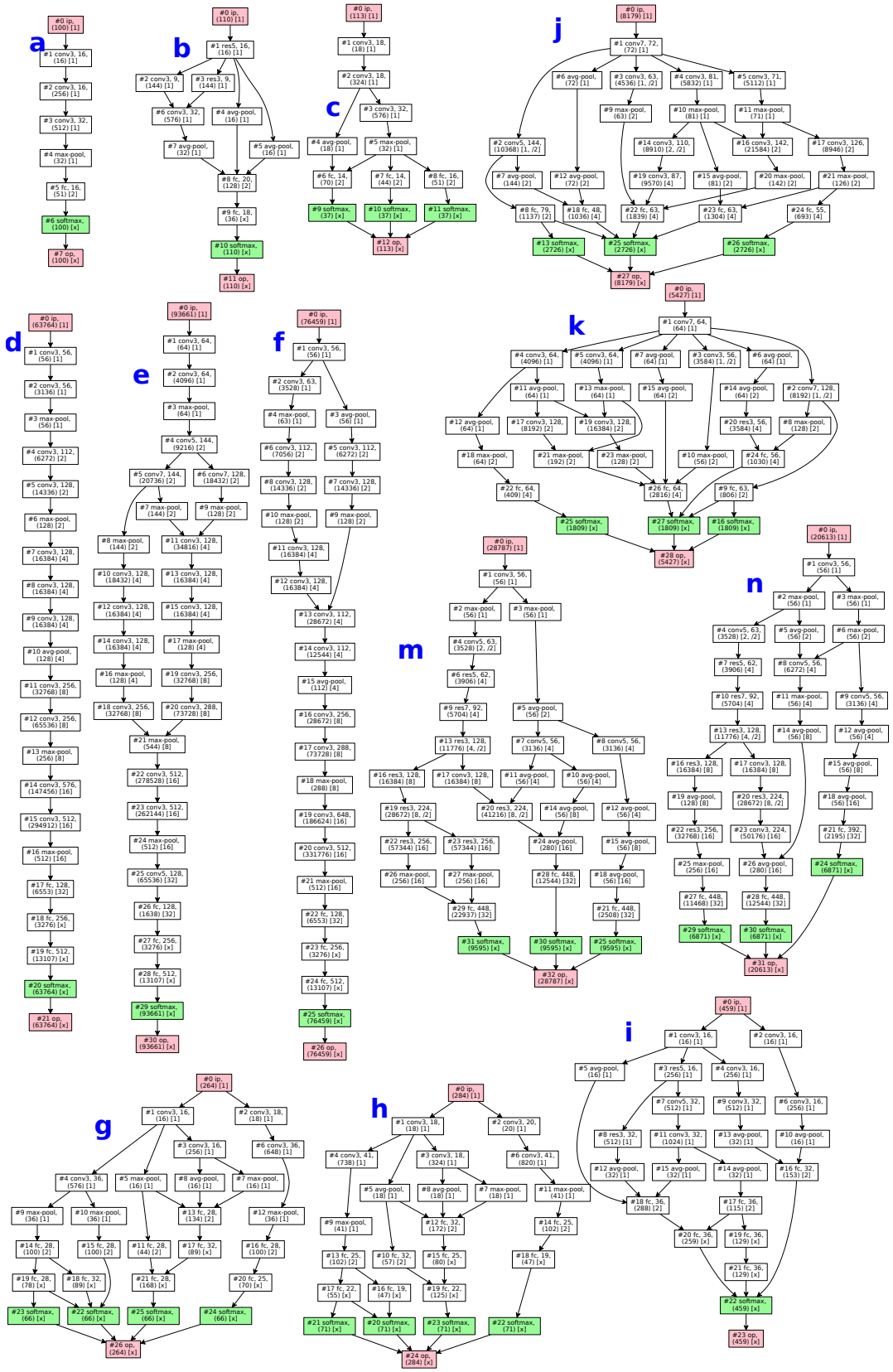


Figure 6.4: Illustrations of the nextworks indexed a-n in Figure 6.3.

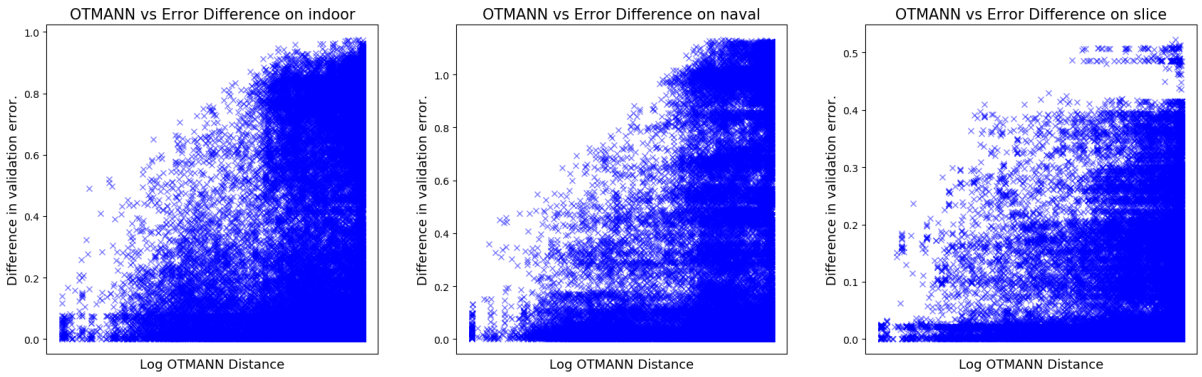


Figure 6.5: Each point in the scatter plot indicates the log distance between two architectures (x axis) and the difference in the validation error (y axis), on the Indoor, Naval and Slice datasets. We used 300 networks, giving rise to $\sim 45K$ pairwise points. On all datasets, when the distance is small, so is the difference in the validation error. As the distance increases, there is more variance in the validation error difference. Intuitively, one should expect that while networks that are far apart could perform similarly or differently, networks with small distance should perform similarly.

and (b) a method to optimise the acquisition φ_t over these architectures.

6.3.1 The Kernel

As described in the main text, we use a negative exponentiated distance as our kernel. Precisely, we use,

$$\kappa(\cdot,\cdot) = \alpha e^{-\sum_i \beta_i d_i^p(\cdot,\cdot)} + \bar{\alpha} e^{-\sum_i \bar{\beta}_i \bar{d}_i^{\bar{p}}(\cdot,\cdot)}. \tag{6.4}$$

Here, $d_i, \bar{d}_i$, are the OTMANN distance and its normalised counterpart developed in Chapter 6.2, computed with different values for $\nu_{\text{str}} \in \{\nu_{\text{str},i}\}_i$. $\beta_i, \bar{\beta}_i$ manage the relative contributions of $d_i, \bar{d}_i$, while $(\alpha, \bar{\alpha})$ manage the contributions of each kernel in the sum. An ensemble approach of the above form, instead of trying to pick a single best value, ensures that NASBOT accounts for the different topologies induced by the different distances $d_i, \bar{d}_i$. In the experiments we report, we used $\{\nu_{\text{str},i}\}_i = \{0.1, 0.2, 0.4, 0.8\}$, $p = 1$ and $\bar{p} = 2$. Our experience suggests that NASBOT was not particularly sensitive to these choices expect when we used only very large or only very small values in $\{\nu_{\text{str},i}\}_i$.

6.3.2 Optimising the Acquisition

We use a evolutionary algorithm (EA) approach to optimise the acquisition function. We begin with an initial pool of networks and evaluate the acquisition φ_t on those networks. Then we generate a set of N_{mut} mutations of this pool as follows. First, we stochastically select N_{mut}

candidates from the set of networks already evaluated such that those with higher φ_t values are more likely to be selected than those with lower values. Then we apply a mutation operator to each candidate, to produce a modified architecture. Finally, we evaluate the acquisition on this N_{mut} mutations, add it to the initial pool, and repeat for the prescribed number of steps.

Mutation Operator: To describe the mutation operator, we will first define a library of modifications to a neural network. These modifications, described in Table 6.2, might change the architecture either by increasing or decreasing the number of computational units in a layer, by adding or deleting layers, or by changing the connectivity of existing layers. They provide a simple mechanism to explore the space of architectures that are close to a given architecture. The *one-step mutation operator* takes a given network and applies one of the modifications in Table 6.2 picked at random to produce a new network. The *k-step mutation operator* takes a given network, and repeatedly applies the one-step operator k times – the new network will have undergone k changes from the original one. One can also define a compound operator, which picks the number of steps probabilistically. In our implementation of NASBOT, we used such a compound operator with probabilities $(0.5, 0.25, 0.125, 0.075, 0.05)$; i.e. it chooses a one-step operator with probability 0.5, a 4-step operator with probability 0.075, etc. Typical implementations of EA in Euclidean spaces define the mutation operator via a Gaussian (or other) perturbation of a chosen candidate. It is instructive to think of the probabilities for each step in our scheme above as being analogous to the width of the Gaussian chosen for perturbation.

Sampling strategy: The sampling strategy for EA is as follows. Let $\{z_i\}_i$, where $z_i \in \mathcal{X}$ be the points evaluated so far. We sample N_{mut} new points from a distribution π where $\pi(z_i) \propto \exp(g(z_i)/\sigma)$. Here g is the function to be optimised (for NASBOT, φ_t at time t). σ is the standard deviation of all previous evaluations. As the probability for large g values is higher, they are more likely to get selected. σ provides normalisation to account for different ranges of function values.

Since our candidate selection scheme at each step favours networks that have high acquisition value, our EA scheme is more likely to search at regions that are known to have high acquisition. The stochasticity in this selection scheme and the fact that we could take multiple steps in the mutation operation ensures that we still sufficiently explore the space. Since an evaluation of φ_t is cheap, we can use many EA steps to explore several architectures and optimise φ_t.

Considerations when performing modifications: The modifications in Table 6.2 is straightforward in MLPs. But in CNNs one needs to ensure that the image sizes are the same when concatenating them as an input to a layer. This is because strides can shrink the size of the image. When we perform a modification we check if this condition is violated and if so, disallow that modification. When a skip modifier attempts to add a connection from a layer with a large image size to one with a smaller one, we add `avg-pool` layers at stride 2 so that the connection can be made (this can be seen, for e.g. in the second network in Fig. 6.10).

6.3.3 Implementation Details

We describe some implementation details for OTMANN and NASBOT.

Operation	Description
dec_single	Pick a layer at random and decrease the number of units by $1/8$.
dec_en_masse	First topologically order the networks, randomly pick $1/8$ of the layers (in order) and decrease the number of units by $1/8$. For networks with eight layers or fewer pick a $1/4$ of the layers (instead of 1/8) and for those with four layers or fewer pick $1/2$.
inc_single	Pick a layer at random and increase the number of units by $1/8$.
inc_en_masse	Choose a large sub set of layers, as for dec_en_masse, and increase the number of units by $1/8$.
dup_path	This modifier duplicates a random path in the network. Randomly pick a node u_1 and then pick one of its children u_2 randomly. Keep repeating to generate a path $u_1, u_2, \ldots, u_{k-1}, u_k$ until you decide to stop randomly. Create duplicate layers $\tilde{u}_2, \ldots, \tilde{u}_{k-1}$ where $\tilde{u}_i = u_i$ for $i = 2, \ldots, k-1$. Add these layers along with new edges $(u_1, \tilde{u}_2)$, $(\tilde{u}_{k-1}, u_k)$, and $(\tilde{u}_j, \tilde{u}_{j+1})$ for $j = 2, \ldots, k-2$.
remove_layer	Picks a layer at random and removes it. If this layer was the only child (parent) of any of its parents (children) u, then adds an edge from u (one of its parents) to one of its children (u).
skip	Randomly picks layers u, v where u is topologically before v and $(u, v) \notin \mathcal{E}$. Add (u, v) to $\mathcal{E}$.
swap_label	Randomly pick a layer and change its label.
wedge_layer	Randomly pick any edge $(u, v) \in \mathcal{E}$. Create a new layer w with a random label $\ell\ell(w)$. Remove (u, v) from $\mathcal{E}$ and add $(u, w), (w, v)$. If applicable, set the number of units $\ell u(w)$ to be $(\ell u(u) + \ell u(v))/2$.

Table 6.2: Descriptions of modifiers to transform one network to another. The first four change the number of units in the layers but do not change the architecture, while the last five change the architecture.

	c3	c5	c7	mp	ap	fc	sm
c3	0	0.2	0.3				
c5	0.2	0	0.2				
c7	0.3	0.2	0				
mp				0	0.25		
ap				0.25	0		
fc						0	
sm							0

Table 6.3: The label mismatch cost matrix M we used in our CNN experiments. $M(\texttt{x}, \texttt{y})$ denotes the penalty for transporting a unit mass from a layer with label x to a layer with label y. The labels abbreviated are conv3, conv5, conv7, max-pool, avg-pool, fc, and softmax in order. A blank indicates ∞ cost. We have not shown the ip and op layers, but they are similar to the fc column, 0 in the diagonal and ∞ elsewhere.

Implementation Details for OTMANN

Computing path lengths δ_s^t: Algorithm 9 computes all path lengths in $O(|\mathcal{E}|)$ time. Note that topological sort of a connected digraph also takes $O(|\mathcal{E}|)$ time. The topological sorting ensures that $\delta_{\text{op}}^{\text{rw}}$ is always computed for the children in step 4. For $\delta_{\text{op}}^{\text{sp}}, \delta_{\text{op}}^{\text{lp}}$ we would replace the averaging of Δ in step 5 with the minimum and maximum of Δ respectively.

Algorithm 9 Compute $\delta_{\text{op}}^{\text{rw}}(u)$ for all $u \in \mathcal{L}$ in OTMANN, from Kandasamy et al. [130]

Require: $\mathcal{G} = (\mathcal{L}, \mathcal{E})$, $\mathcal{L}$ is topologically sorted in S.
1: $\delta_{\text{op}}^{\text{rw}}(u_{\text{op}}) = 0$, $\delta_{\text{op}}^{\text{rw}}(u) = \texttt{nan}\ \ \forall u \neq u_{\text{op}}$.
2: **while** S is not empty **do**
3: $\quad u \leftarrow \text{pop\_last}(S)$
4: $\quad \Delta \leftarrow \{\delta_{\text{op}}^{\text{rw}}(c) : c \in \text{children}(u)\}$
5: $\quad \delta_{\text{op}}^{\text{rw}}(u) \leftarrow 1 + \text{average}(\Delta)$
6: **end while**
7: **Return** $\delta_{\text{op}}^{\text{rw}}$.

For $\delta_{\text{ip}}^{\text{rw}}$ we make the following changes to Algorithm 9. In step 1, we set $\delta_{\text{ip}}^{\text{rw}}(u_{\text{ip}}) = 0$, in step 3, we pop_first and Δ in step 4 is computed using the parents. $\delta_{\text{ip}}^{\text{sp}}, \delta_{\text{ip}}^{\text{lp}}$ are computed with the same procedure but by replacing the averaging with minimum or maximum as above.

Label Penalty Matrices: The label penalty matrices used in our NASBOT implementation, described below, satisfy the triangle inequality condition in Theorem 66.

CNNs: Table 6.3 shows the label penalty matrix M for used in our CNN experiments with labels conv3, conv5, conv7, max-pool, avg-pool, softmax, ip, op. conv$k$ denotes a $k \times k$ convolution while avg-pool and max-pool are pooling operations. In addition, we also use res3, res5, res7 layers which are inspired by ResNets. A res$k$ uses 2 concatenated conv$k$ layers but the input to the first layer is added to the output of the second layer before the relu activation – See Figure 2 in He et al. [91]. The layer mass for res$k$ layers is twice that of a conv$k$ layer. The costs for the res in the label penalty matrix is the same as the conv block. The cost between a res$k$ and conv$j$ is $M(\texttt{res}k, \texttt{conv}j) = 0.9 \times M(\texttt{conv}k, \texttt{conv}j) + 0.1 \times 1$; i.e. we are using a convex combination of the conv costs and the non-assignment cost. The intuition is that a res$k$ is similar to conv$k$ block except for the residual addition.

	re	cr	<rec>	lg	ta	lin
re	0	.1	.1	.25	.25	
cr	.1	0	.1	.25	.25	
<rec>	.1	.1	0	.25	.25	
lg	.25	.25	.25	0	.1	
ta	.25	.25	.25	.1	0	
lin						0

Table 6.4: The label mismatch cost matrix M we used in our MLP experiments. The labels abbreviated are `relu`, `crelu`, `<rec>`, `logistic`, `tanh`, and `linear` in order. `<rec>` is place-holder for any other rectifier such as `leaky-relu`, `softplus`, `elu`. A blank indicates ∞ cost. The design here was simple. Each label gets 0 cost with itself. A rectifier gets 0.1 cost with another rectifier and 0.25 with a sigmoid; vice versa for all sigmoids. The rest of the costs are infinity. We have not shown the `ip` and `op`, but they are similar to the `lin` column, 0 in the diagonal and ∞ elsewhere.

MLPs: Table 6.4 shows the label penalty matrix M for used in our MLP experiments with labels `relu`, `crelu`, `leaky-relu`, `softplus`, `elu`, `logistic`, `tanh`, `linear`, `ip`, `op`. Here the first seven are common non-linear activations; `relu`, `crelu`, `leaky-relu`, `softplus`, `elu` rectifiers while `logistic` and `tanh` are sigmoidal activations.

Other details: Our implementation of OTMANN differs from what is described in the main text in two ways. First, in our CNN experiments, for a `fc` layer u, we use $0.1 \times \ell m(u) \times \langle$#-incoming-channels$\rangle$ as the mass, i.e. we multiply it by 0.1 from what is described in the main text. This is because, in the convolutional and pooling channels, each unit is an image where as in the `fc` layers each unit is a scalar. One could, in principle, account for the image sizes at the various layers when computing the layer masses, but this also has the added complication of depending on the size of the input image which varies from problem to problem. Our approach is simpler and yields reasonable results.

Secondly, we use a slightly different form for C_{str}. First, for $i \in \mathcal{L}_1,\ j \in \mathcal{L}_2$, we let $C^{\text{all}}_{\text{str}}(i,j) = \frac{1}{6}\sum_{s\in\{\text{sp, lp, rw}\}}\sum_{t\in\{\text{ip,op}\}} |\delta^s_t(i) - \delta^s_t(j)|$ be the average of *all* path length differences; i.e. $C^{\text{all}}_{\text{str}}$ captures the path length differences when considering all layers. For CNNs, we similarly construct matrices $C^{\text{conv}}_{\text{str}}, C^{\text{pool}}_{\text{str}}, C^{\text{fc}}_{\text{str}}$, except they only consider the convolutional, pooling and fully connected layers respectively in the path lengths. For $C^{\text{conv}}_{\text{str}}$, the distances to the output (from the input) can be computed by zeroing outgoing (incoming) edges to layers that are not convolutional. We can similarly construct $C^{\text{pool}}_{\text{str}}$ and $C^{\text{fc}}_{\text{str}}$ only counting the pooling and fully connected layers. Our final cost matrix for the structural penalty is the average of these four matrices, $C_{\text{str}} = (C^{\text{all}}_{\text{str}} + C^{\text{conv}}_{\text{str}} + C^{\text{pool}}_{\text{str}} + C^{\text{fc}}_{\text{str}})/4$. For MLPs, we adopt a similar strategy by computing matrices $C^{\text{all}}_{\text{str}}, C^{\text{rec}}_{\text{str}}, C^{\text{sig}}_{\text{str}}$ with all layers, only rectifiers, and only sigmoidal layers and let $C_{\text{str}} = (C^{\text{all}}_{\text{str}} + C^{\text{rec}}_{\text{str}} + C^{\text{sig}}_{\text{str}})/3$. The intuition is that by considering certain types of layers, we are accounting for different types of information flow due to different operations.

Implementation Details for NASBOT

Tuning GP Hyperparameters in BO: NASBOT, as described above has 11 hyperparameters of its own; $\alpha, \bar{\alpha}, \{(\beta_i, \bar{\beta}_i)\}_{i=1}^4$ and the GP noise variance η^2. While maximising the GP marginal likelihood is a common approach to pick hyperparameters, this might cause over-fitting when there are many of them. Further, as training large neural networks is typically expensive, we have to content with few observations for the GP in practical settings. Our solution is to start with a (uniform) prior over these hyperparameters and sample hyperparameter values from the posterior under the GP likelihood [232], which we found to be robust. While it is possible to treat ν_{str} itself as a hyperparameter of the kernel, this will require us to re-compute all pairwise distances of networks that have already been evaluated each time we change the hyperparameters. On the other hand, with the above approach, we can compute and store distances for different $\nu_{\text{str},i}$ values whenever a new network is evaluated, and then compute the kernel cheaply for different values of $\alpha, \bar{\alpha}, \{(\beta_i, \bar{\beta}_i)\}_i$.

Initialisation: We initialise NASBOT (and other methods) with an initial pool of 10 networks. These networks are illustrated in Fig. 6.6 for CNNs and Fig. 6.7 for MLPs. All initial networks have feed forward structure. For the CNNs, the first 3 networks have structure similar to the VGG nets [228] and the remaining have blocked feed forward structures as in He et al. [91]. We also use blocked structures for the MLPs with the layer labels decided arbitrarily.

Domain: For NASBOT, and other methods, we impose the following constraints on the search space. If the EA modifier (explained below) generates a network that violates these constraints, we simply skip it.

- Maximum number of layers: 60
- Maximum mass: 10^8
- Maximum in/out degree: 5
- Maximum number of edges: 200
- Maximum number of units per layer: 1024
- Minimum number of units per layer: 8

Layer types: We use the layer types detailed in Chapter 6.3.3 for both CNNs and MLPs. For CNNs, all pooling operations are done at stride 2. For convolutional layers, we use either stride 1 or 2 (specified in the illustrations). For all layers in a CNN we use `relu` activations.

Parallel BO: We use a parallelised experimental set up where multiple models can be evaluated in parallel. We handle parallel BO via the hallucination technique in Ginsbourger et al. [73].

Acquisition: We use the GP-EI acquisition function [113].

Other details on the EA procedure: The EA procedure is also initialised with the same initial pool of feedforward networks in Figures 6.6, 6.7. In our NASBOT implementation, we increase the total number of EA evaluations n_{EA} at rate $\mathcal{O}(\sqrt{t})$ where t is the current time step in NASBOT. We set N_{mut} to be $\mathcal{O}(\sqrt{n_{\text{EA}}})$. Hence, initially we are only considering a small

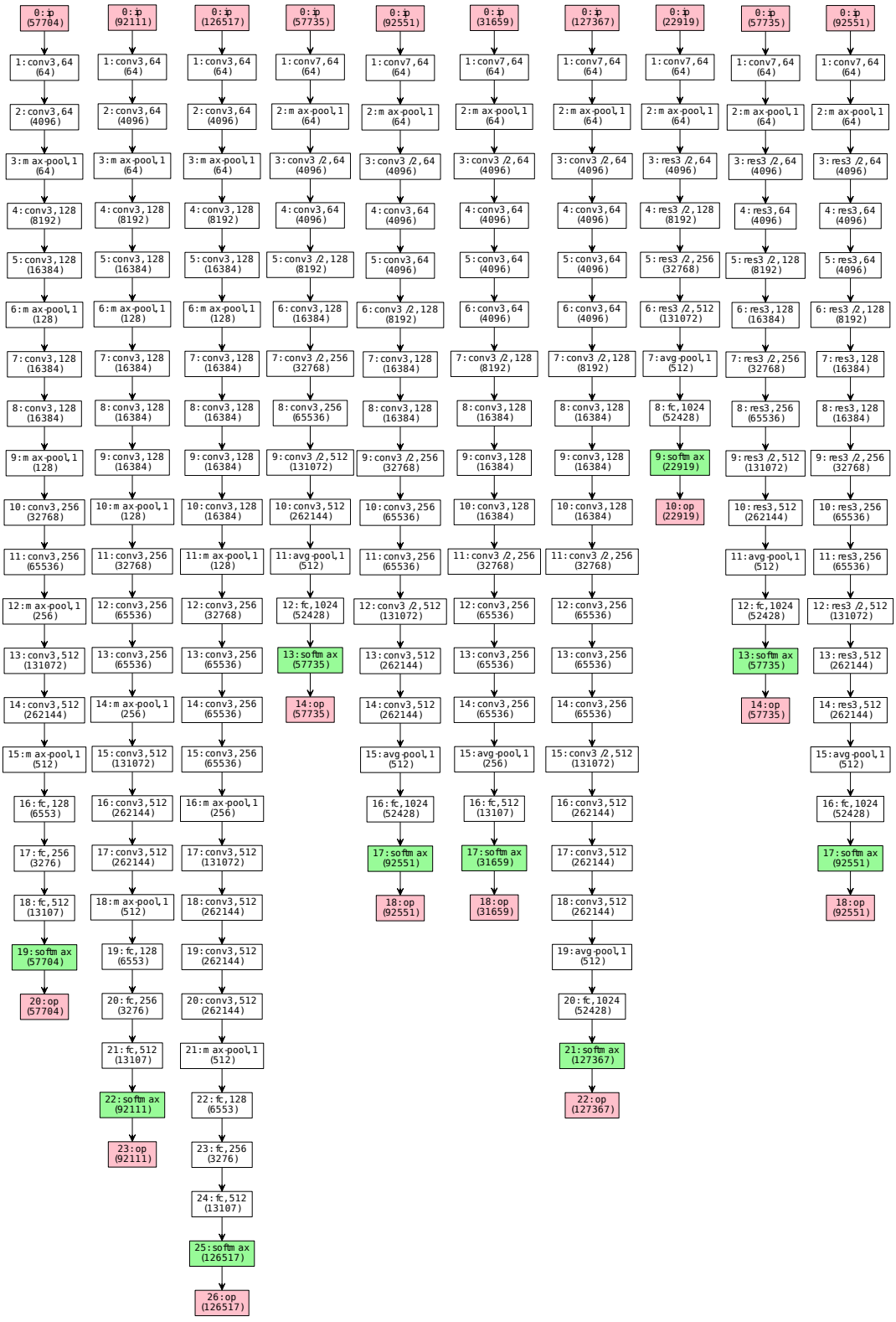


Figure 6.6: Initial pool of CNN network architectures. The first 3 networks have structure similar to the VGG nets [228] and the remaining have blocked feed forward structures as in He et al. [91].

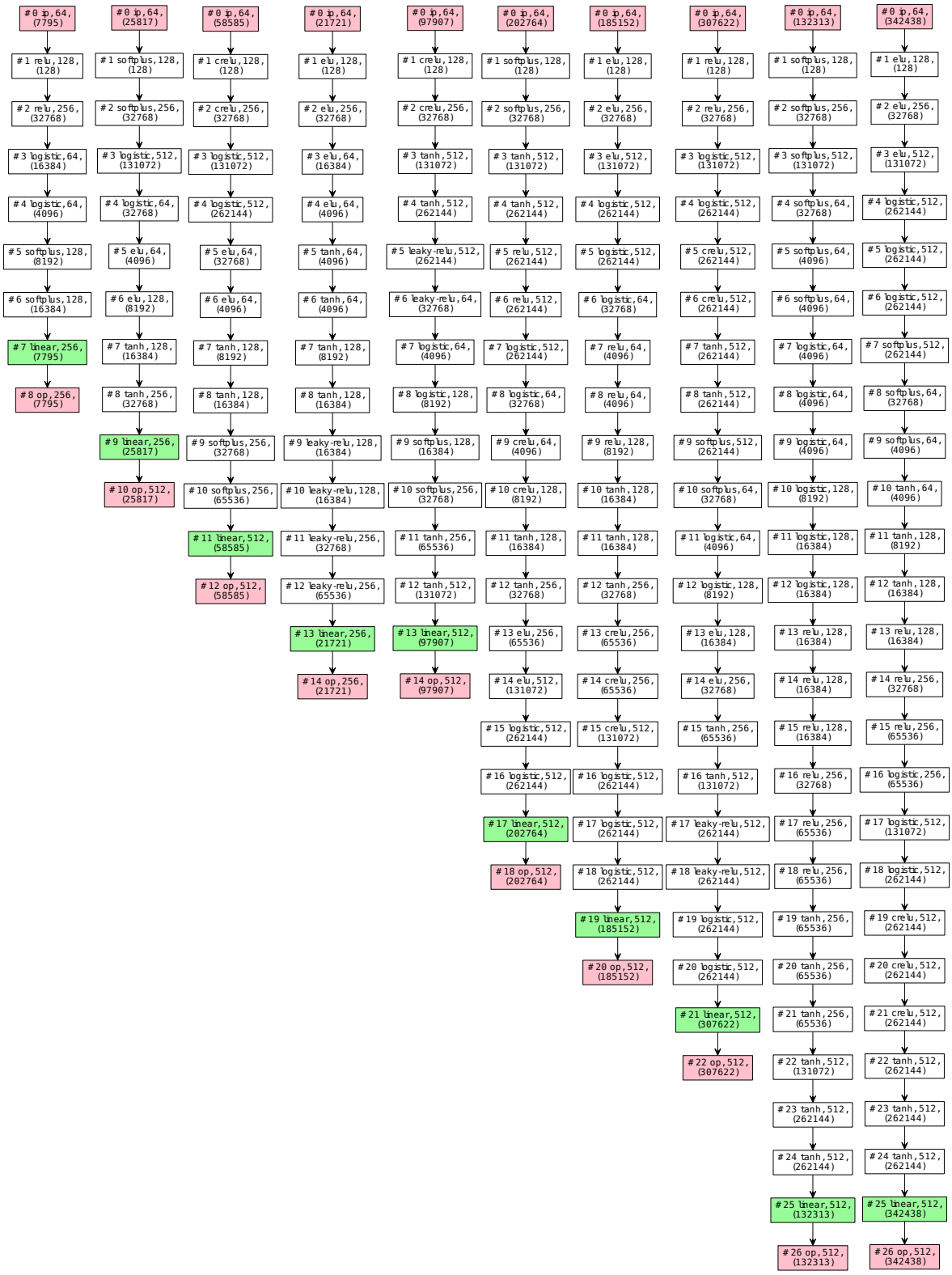


Figure 6.7: Initial pool of MLP network architectures.

neighborhood around the initial pool, but as we proceed along BO, we expand to a larger region, and also spend more effort to optimise φ_t.

6.4 Experiments

We now describe our experiments on NASBOT on several model selection tasks. Additionally, we provide ablation studies on the NASBOT framework on synthetic examples.

Baselines: We compare NASBOT to the following baselines. RAND: random search; EA (Evolutionary algorithm): the same EA procedure described above. TreeBO [108]: a BO method which only searches over feed forward structures. Random search is a natural baseline to compare optimisation methods. However, unlike in Euclidean spaces, there is no natural way to randomly explore the space of architectures. Our RAND implementation, operates in exactly the same way as NASBOT, except that the EA procedure is fed a random sample from Unif$(0, 1)$ instead of the GP acquisition each time it evaluates an architecture. Hence, RAND is effectively picking a random network from the same space explored by NASBOT; neither method has an unfair advantage because it considers a different space. While there are other methods for architecture search, their implementations are highly nontrivial and are not made available. We have described some implementation details for these methods at the end of these section.

Datasets: We use the following datasets: blog feedback [29], indoor location [247], slice localisation [84], naval propulsion [45], protein tertiary structure [202], news popularity [57], Cifar10 [149]. The first six are regression problems for which we use MLPs. The last is a classification task on images for which we use CNNs. Table 6.5 gives the size and dimensionality of each dataset. For the first 6 datasets, we use a $0.6 - 0.2 - 0.2$ train-validation-test split and normalised the input and output to have zero mean and unit variance. Hence, a constant predictor will have a mean squared error of approximately 1. For Cifar10 we use $40K$ for training and $10K$ each for validation and testing.

Experimental Set up: Each method is executed in an asynchronously parallel set up of 2-4 GPUs, That is, it can evaluate multiple models in parallel, with each model on a single GPU. When the evaluation of one model finishes, the methods can incorporate the result and immediately re-deploy the next job without waiting for the others to finish. For the blog, indoor, slice, naval and protein datasets we use 2 GeForce GTX 970 (4GB) GPUs and a computational budget of 8 hours for each method. For the news popularity dataset we use 4 GeForce GTX 980 (6GB) GPUs with a budget of 6 hours and for Cifar10 we use 4 K80 (12GB) GPUs with a budget of 10 hours. For the regression datasets, we train each model with stochastic gradient descent (SGD) with a fixed step size of 10^{-5}, a batch size of 256 for 20K batch iterations. For Cifar10, we start with a step size of 10^{-2}, and reduce it gradually. We train in batches of 32 images for 60K batch iterations. The methods evaluate between 70-120 networks depending on the size of the networks chosen and the number of GPUs. We have described details on the training procedures at the end of this section.

Results: Fig. 6.8 plots the best validation score for each method against time. In Table 6.5, we

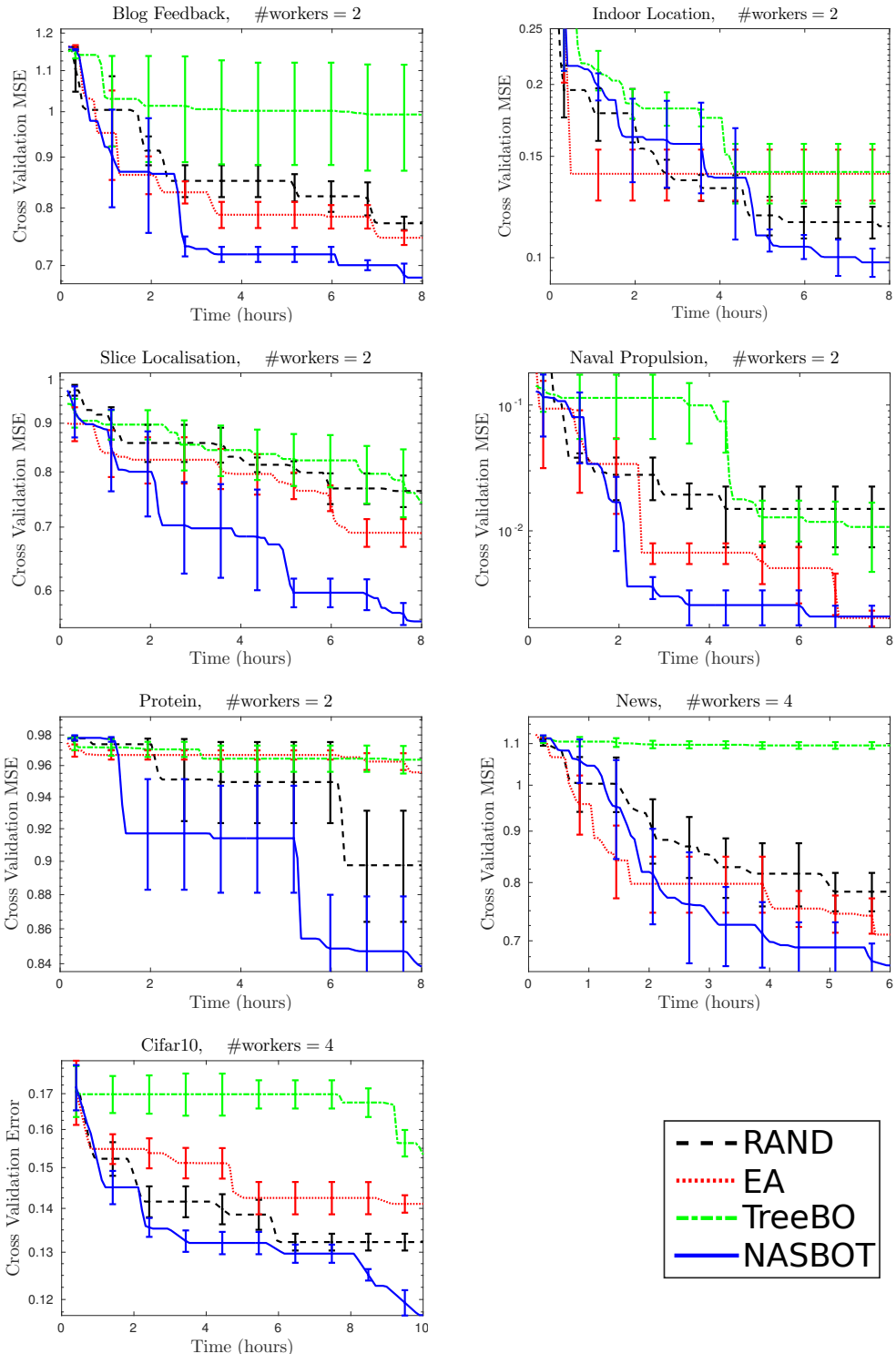


Figure 6.8: Cross validation results for neural architecture search. In all figures, the x axis is time. The y axis is the mean squared error (MSE) in the first 6 figures and the classification error in the last. Lower is better in all cases. The title of each figure states the dataset and the number of parallel workers (GPUs). All figures were averaged over at least 5 independent runs of each method. Error bars indicate one standard error.

Method	Blog $(60K, 281)$	Indoor $(21K, 529)$	Slice $(54K, 385)$	Naval $(12K, 17)$	Protein $(46K, 9)$	News $(40K, 61)$	Cifar10 $(60K, 1K)$	Cifar10 $150K$ iters
RAND	0.780 $\pm$ 0.034	**0.115 $\pm$ 0.023**	0.758 $\pm$ 0.041	0.0103 $\pm$ 0.002	0.948 $\pm$ 0.024	**0.762 $\pm$ 0.013**	0.1342 $\pm$ 0.002	0.0914 $\pm$ 0.008
EA	0.806 $\pm$ 0.040	0.147 $\pm$ 0.010	0.733 $\pm$ 0.041	**0.0079 $\pm$ 0.004**	1.010 $\pm$ 0.038	**0.758 $\pm$ 0.038**	0.1411 $\pm$ 0.002	0.0915 $\pm$ 0.010
TreeBO	0.928 $\pm$ 0.053	0.168 $\pm$ 0.023	0.759 $\pm$ 0.079	0.0102 $\pm$ 0.002	0.998 $\pm$ 0.007	0.866 $\pm$ 0.085	0.1533 $\pm$ 0.004	0.1121 $\pm$ 0.004
NASBOT	**0.731 $\pm$ 0.029**	**0.117 $\pm$ 0.008**	**0.615 $\pm$ 0.044**	**0.0075 $\pm$ 0.002**	**0.902 $\pm$ 0.033**	**0.752 $\pm$ 0.024**	**0.1209 $\pm$ 0.003**	**0.0869 $\pm$ 0.004**

Table 6.5: The first row gives the number of samples N and the dimensionality D of each dataset in the form (N, D). The subsequent rows show the regression MSE or classification error (lower is better) on the *test set* for each method. The last column is for Cifar10 where we took the best models found by each method in 24K iterations and trained it for $120K$ iterations. When we trained the VGG-19 architecture using our training procedure, we got test errors 0.1718 (60K iterations) and 0.1018 (150K iterations).

present the results on the test set with the best model chosen on the basis of validation set performance. On the Cifar10 dataset, we also trained the best models for longer ($150K$ iterations). These results are in the last column of Table 6.5. We see that NASBOT is the most consistent of all methods. The average time taken by NASBOT to determine the next architecture to evaluate was 46.13s. For RAND, EA, and TreeBO this was 26.43s, 0.19s, and 7.83s respectively. The time taken to train and validate models was on the order of 10-40 minutes depending on the model size. Fig. 6.8 includes this time taken to determine the next point. Like many BO algorithms, while NASBOT's selection criterion is time consuming, it pays off when evaluations are expensive.

Optimal Network Architectures We illustrate and compare the optimal neural network architectures found by different methods. In Figures 6.10-6.13, we show some optimal network architectures found on the Cifar10 data by NASBOT, EA, RAND, and TreeBO, respectively. We also show some optimal network architectures found for these four methods on the Indoor data, in Figures 6.14-6.17, and on the Slice data, in Figures 6.18-6.21. A common feature among all optimal architectures found by NASBOT was the presence of long skip connections and multiple decision layers. In Figure 6.7, we show the initial pool of MLP network architectures, and in Figure 6.6, we show the initial pool of CNN network architectures.

Finally, we note that while our Cifar10 experiments fall short of the current state of the art [155, 156, 282], the amount of computation in these work is several orders of magnitude more than ours (both the computation invested to train a single model and the number of models trained). Further, they use constrained spaces specialised for CNNs, while NASBOT is deployed in a very general model space. We believe that our results can also be improved by employing enhanced training techniques such as image whitening, image flipping, drop out, etc. For example, using our training procedure on the VGG-19 architecture [228] yielded a test set error of 0.1018 after $150K$ iterations. However, VGG-19 is known to do significantly better on Cifar10. That said,

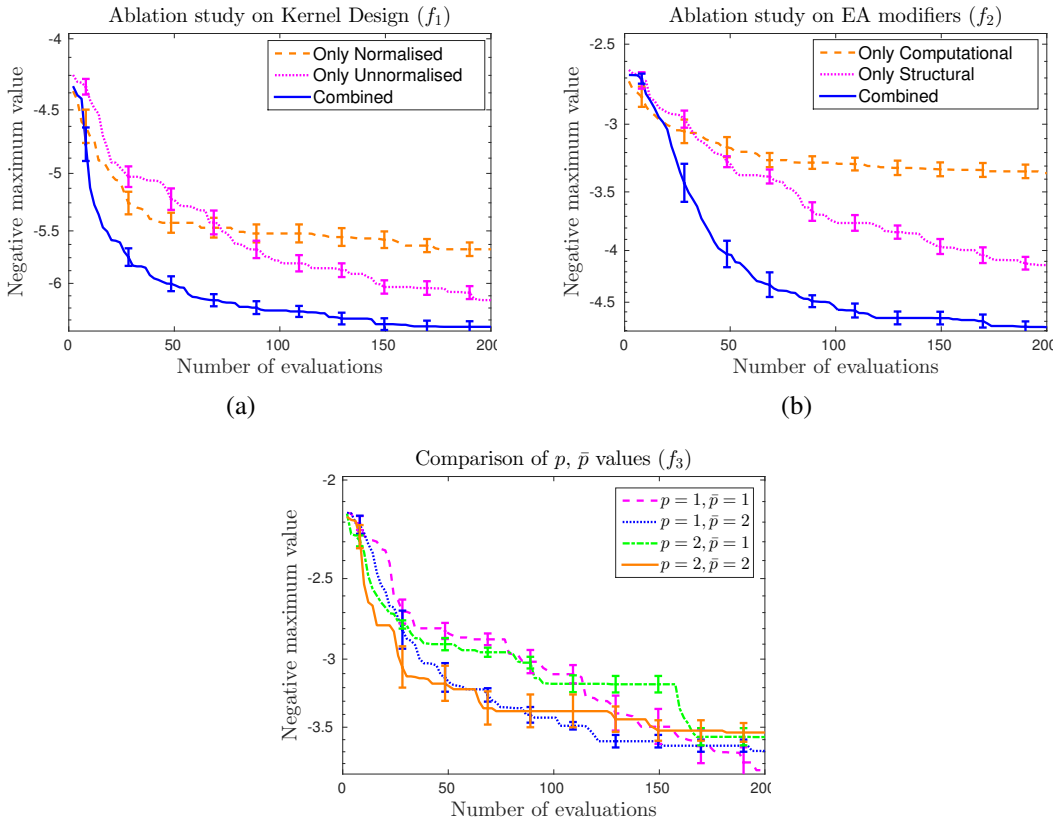


Figure 6.9: We compare NASBOT for different design choices in our framework. (a): Comparison of NASBOT using only the normalised distance $e^{-\beta\bar{d}}$, only the unnormalised distance $d^{-\beta d}$, and the combination $e^{-\beta d} + e^{-\beta\bar{d}}$. (b): Comparison of NASBOT using only the EA modifiers which change the computational units (top 4 in Table 6.2), modifiers which only change the structure of the networks (bottom 5 in Table 6.2), and all 9 modifiers. (c): Comparison of NASBOT with different choices for p and $\bar{p}$. In all figures, the x axis is the number of evaluations and the y axis is the negative maximum value (lower is better). All figures were produced by averaging over at least 10 runs.

we believe our results are encouraging and lay out the premise for BO for neural architectures.

Ablation Studies and Design Choices

We conduct experiments comparing the various design choices in NASBOT. Due to computational constraints, we carry them out on synthetic functions.

Combined distances: In Figure 6.9(a), we compare NASBOT using only the normalised distance, only the unnormalised distance, and the combined kernel as in (6.4). While the individual distances performs well, the combined form outperforms both.

On the EA procedure: Next, we modify our EA procedure to optimise the acquisition. We

execute NASBOT using only the EA modifiers which change the computational units (first four modifiers in Table 6.2), then using the modifiers which only change the structure of the networks (bottom 5 in Table 6.2), and finally using all 9 modifiers, as used in all our experiments. The combined version outperforms the first two.

Choices for $p, \bar{p}$: Finally, we experiment with different choices for p and $\bar{p}$ in (6.4). As the figures indicate, the performance was not particularly sensitive to these choices.

Synthetic Functions: Below we describe the three synthetic functions f_1, f_2, f_3 used in our synthetic experiments. f_3 applies for CNNs while f_1, f_2 apply for MLPs. Here am denotes the average mass per layer, $\mathrm{deg_i}$ is the average in degree the layers, $\mathrm{deg_o}$ is the average out degree, δ is the shortest distance from u_{ip} to u_{op}, str is the average stride in CNNS, $\mathrm{frac\_conv3}$ is the fraction of layers that are `conv3`, $\mathrm{frac\_sigmoid}$ is the fraction of layers that are sigmoidal.

$$
\begin{aligned}
f_0 &= \exp(-0.001 * |\mathrm{am} - 1000|) + \exp(-0.5 * |\mathrm{deg_i} - 5|) + \exp(-0.5 * |\mathrm{deg_o} - 5|) + \\
&\quad \exp(-0.1 * |\delta - 5|) + \exp(-0.1 * ||\mathcal{L}| - 30|) + \exp(-0.05 * ||\mathcal{E}| - 100|) \\
f_1 &= f_0 + \exp(-3 * |\mathrm{str} - 1.5|) + \exp(-0.3 * ||\mathcal{L}| - 50|) + \\
&\quad \exp(-0.001 * |\mathrm{am} - 500|) + \mathrm{frac\_conv3} \\
f_2 &= f_0 + \exp(-0.001 * |\mathrm{am} - 2000|) + \exp(-0.1 * ||\mathcal{E}| - 50|) + \mathrm{frac\_sigmoid} \\
f_3 &= f_0 + \mathrm{frac\_sigmoid}
\end{aligned}
$$

Additional Details on the Experiments

Baselines

RAND: Our RAND implementation, operates in exactly the same way as NASBOT, except that the EA procedure is fed a random sample from $\mathrm{Unif}(0, 1)$ instead of the GP acquisition each time it evaluates an architecture. That is, we follow the same schedule for n_{EA} and N_{mut} as we did for NASBOT . Hence RAND has the opportunity to explore the same space as NASBOT, but picks the next evaluation randomly from this space.

EA: This is as described before except that we fix $N_{\text{mut}} = 10$ all the time. In our experiments where we used a budget based on time, it was difficult to predict the total number of evaluations so as to set N_{mut} in perhaps a more intelligent way.

TreeBO: As the implementation from Jenatton et al. [108] was not made available, we wrote our own. It differs from the version described in the paper in a few ways. We do not tune for a regularisation penalty and step size as they do to keep it line with the rest of our experimental set up. We set the depth of the network to 60 as we allowed 60 layers for the other methods. We also check for the other constraints given in above before evaluating a network. The original paper uses a tree structured kernel which can allow for efficient inference with a large number of samples. For simplicity, we construct the entire kernel matrix and perform standard GP inference. The result of the inference is the same, and the number of GP samples was always below 120 in our experiments so a sophisticated procedure was not necessary.

Details on Training

Training: In all methods, for each proposed network architecture, we trained the network on the train data set, and periodically evaluated its performance on the validation data set. For MLP experiments, we optimised network parameters using stochastic gradient descent with a fixed step size of 10^{-5} and a batch size of 256 for 20,000 iterations. We computed the validation set MSE every 100 iterations; from this we returned the minimum MSE that was achieved. For CNN experiments, we optimised network parameters using stochastic gradient descent with a batch size of 32. We started with a learning rate of 0.01 and reduced it gradually. We also used batch normalisation and trained the model for $60,000$ batch iterations. We computed the validation set classification error every 4000 iterations; from this we returned the minimum classification error that was achieved.

Validation: After each method returned an optimal neural network architecture, we again trained each optimal network architecture on the train data set, periodically evaluated its performance on the validation data set, and finally computed the MSE or classification error on the test data set. For MLP experiments, we used the same optimisation procedure as above; we then computed the test set MSE at the iteration where the network achieved the minimum validation set MSE. For CNN experiments, we used the same optimisation procedure as above, except here the optimal network architecture was trained for 120,000 iterations; we then computed the test set classification error at the iteration where the network achieved the minimum validation error.

6.5 Proofs of Theoretical Results

We will first prove Theorem 67 as we will need the result for the proof of the distance properties.

Proof of Theorem 67

Proof. We will show that there exists a bijection between feasible points in both problems with the same value for the objective. First let $Z \in \mathbb{R}^{n_1 \times n_2}$ be a feasible point for (6.1). Let $Z' \in \mathbb{R}^{\bar{n}_1 \times \bar{n}_2}$ be such that its first $n_1 \times n_2$ block is Z and, $Z_{\bar{n}_1 j} = \sum_{i=1}^{n_1} Z_{ij}$, $Z_{i\bar{n}_2} = \sum_{j=1}^{n_2} Z_{ij}$, and $Z_{\bar{n}_1,\bar{n}_2} = \sum_{ij} Z_{ij}$. Then, for all $i \leq n_1$, $\sum_j Z'_{ij} = \ell m(j)$ and $\sum_j Z'_{\bar{n}_1 j} Z'_{ij} = \sum_j \ell m(j) - \sum_{ij} Z_{ij} + Z_{\bar{n}_1,\bar{n}_2} = tm(\mathcal{G}_2)$. We then have, $Z'\mathbf{1}_{\bar{n}_2} = y_1$ Similarly, we can show $Z'^{\top}\mathbf{1}_{\bar{n}_1} = y_2$. Therefore, Z' is feasible for (6.3). We see that the objectives are equal via simple calculations,

$$\begin{aligned}
\langle Z', C' \rangle &= \langle Z', C'_{\text{lmm}} + C'_{\text{str}} \rangle + \langle Z', C'_{\text{nas}} \rangle \qquad (6.5)\\
&= \langle Z, C_{\text{lmm}} + C_{\text{str}} \rangle + \sum_{j=1}^{n_2} Z'_{ij} + \sum_{i=1}^{n_1} Z'_{ij} \\
&= \langle Z, C_{\text{lmm}} \rangle + \langle Z, C_{\text{str}} \rangle + \sum_{i \in \mathcal{L}_1} \big(\ell m(i) - \sum_{j \in \mathcal{L}_2} Z_{ij}\big) + \sum_{j \in \mathcal{L}_2} \big(\ell m(j) - \sum_{i \in \mathcal{L}_1} Z_{ij}\big).
\end{aligned}$$

The converse also follows via a straightforward argument. For given Z' that is feasible for (6.3), we let Z be the first $n_1 \times n_2$ block. By the equality constraints and non-negativity of Z', Z is feasible for (6.1). By reversing the argument in (6.5) we see that the objectives are also equal. □

Proof of Theorem 66

We are now ready to prove the pseudo-distance properties of OTMANN.

Proof. We will use the OT formulation (6.3) in this proof. The first three properties are straightforward. Non-negativity follows from non-negativity of Z', C' in (6.3). It is symmetric since the cost matrix for $d(\mathcal{G}_2, \mathcal{G}_1)$ is $C'^\top$ if the cost matrix for $d(\mathcal{G}_1, \mathcal{G}_2)$ is C and $\langle Z', C' \rangle = \langle Z'^\top, C'^\top \rangle$ for all Z'. We also have $d(\mathcal{G}_1, \mathcal{G}_1) = 0$ since, then, C' has a zero diagonal.

To prove the triangle inequality, we will use a gluing lemma, similar to what is used in the proof of Wasserstein distances [195]. Let $\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3$ be given and m_1, m_2, m_3 be their total masses. Let the solutions to $d(\mathcal{G}_1, \mathcal{G}_2)$ and $d(\mathcal{G}_2, \mathcal{G}_3)$ be $P \in \mathbb{R}^{\bar{n}_1 \times \bar{n}_2}$ and $Q \in \mathbb{R}^{\bar{n}_2 \times \bar{n}_3}$ respectively. When solving (6.3), we see that adding extra mass to the non-assignment layers does not change the objective, as an optimiser can transport mass between the two layers with 0 cost. Hence, we can assume w.l.o.g that (6.3) was solved with $y_i = \left[\{\ell m(u)\}_{u \in \mathcal{L}_i}, \left(\sum_{j \in \{1,2,3\}} tm(\mathcal{G}_j) - tm(\mathcal{G}_i)\right)\right] \in \mathbb{R}^{\bar{n}_i}$ for $i = 1, 2, 3$, when computing the distances $d(\mathcal{G}_1, \mathcal{G}_2)$, $d(\mathcal{G}_1, \mathcal{G}_3)$, $d(\mathcal{G}_2, \mathcal{G}_3)$; i.e. the total mass was $m_1 + m_2 + m_3$ for all three pairs. We can similarly assume that P, Q account for this extra mass, i.e. $P_{\bar{n}_1 \bar{n}_2}$ and $Q_{\bar{n}_2 \bar{n}_3}$ have been increased by m_3 and m_1 respectively from their solutions in (6.3).

To apply the gluing lemma, let $S = P\mathrm{diag}(1/y_2)Q \in \mathbb{R}^{\bar{n}_1 \times \bar{n}_3}$, where $\mathrm{diag}(1/y_2)$ is a diagonal matrix whose $(j,j)^{\text{th}}$ element is $1/(y_2)_j$ (note $y_2 > 0$). We see that S is feasible for (6.3) when computing $d(\mathcal{G}_1, \mathcal{G}_3)$,

$$R\mathbf{1}_{\bar{n}_3} = P\mathrm{diag}(1/y_2)Q\mathbf{1}_{\bar{n}_3} = P\mathrm{diag}(1/y_2)y_2 = P\mathbf{1}_{\bar{n}_2} = y_1.$$

Similarly, $R^\top \mathbf{1}_{\bar{n}_1} = y_3$. Now, let U', V', W' be the cost matrices C' in (6.3) when computing $d(\mathcal{G}_1, \mathcal{G}_2)$, $d(\mathcal{G}_2, \mathcal{G}_3)$, and $d(\mathcal{G}_1, \mathcal{G}_3)$ respectively. We will use the following technical lemma whose proof is given below.

Lemma 68. *For all $i \in \mathcal{L}_1$, $j \in \mathcal{L}_2$, $k \in \mathcal{L}_3$, we have $W'_{ik} \leq U'_{ij} + V'_{jk}$.*

Applying Lemma 68 yields the triangle inequality.

$$\begin{aligned} d(\mathcal{G}_1, \mathcal{G}_3) \leq \langle R, W' \rangle &= \sum_{i \in \mathcal{L}_1, k \in \mathcal{L}_3} W'_{ik} \sum_{j \in \mathcal{L}_2} \frac{P_{ij} Q_{jk}}{(y_2)_j} \leq \sum_{i,j,k} (U'_{ij} + V'_{jk}) \frac{P_{ij} Q_{jk}}{(y_2)_j} \\ &= \sum_{ij} \frac{U'_{ij} P_{ij}}{(y_2)_j} \sum_k Q_{jk} + \sum_{jk} \frac{V'_{jk} Q_{jk}}{(y_2)_j} \sum_k P_{ij} \\ &= \sum_{ij} U'_{ij} P_{ij} + \sum_{jk} V'_{jk} Q_{jk} = d(\mathcal{G}_1, \mathcal{G}_2) + d(\mathcal{G}_2, \mathcal{G}_3) \end{aligned}$$

The first step uses the fact that $d(\mathcal{G}_1, \mathcal{G}_3)$ is the minimum of all feasible solutions and the third step uses Lemma 68. The fourth step rearranges terms and the fifth step uses $P^\top \mathbf{1}_{\bar{n}_1} = Q\mathbf{1}_{\bar{n}_3} = y_2$. □

Proof of Lemma 68. Let $W' = W'_{\text{lmm}} + W'_{\text{str}} + W'_{\text{nas}}$ be the decomposition into the label mismatch, structural and non-assignment parts of the cost matrices; define similar quantities $U'_{\text{lmm}}, U'_{\text{str}}, U'_{\text{nas}}$, $V'_{\text{lmm}}, V'_{\text{str}}, V'_{\text{nas}}$ for U', V'. Noting $a \leq b + c$ and $d \leq e + f$ implies $a + d \leq b + e + c + f$, it is sufficient to show the triangle inquality for each component individually. For the label mismatch term, $(W'_{\text{lmm}})_{ik} \leq (U'_{\text{lmm}})_{ij} + (V'_{\text{lmm}})_{jk}$ follows directly from the conditions on M by setting $\mathtt{x} = \ell\ell(i)$, $\mathtt{y} = \ell\ell(j)$, $\mathtt{z} = \ell\ell(k)$, where i, j, k are indexing in $\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3$ respectively.

For the non-assignment terms, when $(W'_{\text{nas}})_{ik} = 0$ the claim is true trivially. $(W'_{\text{nas}})_{ik} = 1$, either when $(i = \bar{n}_1, k \leq n_3)$ or $(i \leq n_1, k = \bar{n}_3)$. In the former case, when $j \leq n_2$, $(U'_{\text{nas}})_{jk} = 1$ and when $j = \bar{n}_2$, $(V'_{\text{nas}})_{\bar{n}_2} = 1$ as $k \leq n_3$. We therefore have, $(W'_{\text{nas}})_{ik} = (U'_{\text{nas}})_{ij} + (V'_{\text{nas}})_{jk} = 1$. A similar argument shows equality for the $(i \leq n_1, k = \bar{n}_3)$ case as well.

Finally, for the structural terms we note that W'_{str} can be written as $W'_{\text{str}} = \sum_t W'^{(t)}$ as can $U'^{(t)}, T'^{(t)}$. Here t indexes over the choices for the types of distances considered, i.e. $t \in \{\text{sp, lp, rw}\} \times \{\text{ip, op}\}$. It is sufficient to show $(W'^{(t)})_{ik} \leq (U'^{(t)})_{ij} + (T'^{(t)})_{jk}$. This inequality takes the form,

$$|\delta^{(t)}_{1i} - \delta^{(t)}_{3k}| \leq |\delta^{(t)}_{1i} - \delta^{(t)}_{2j}| + |\delta^{(t)}_{2j} - \delta^{(t)}_{3k}|.$$

Where $\delta^{(t)}_{g\ell}$ refers to distance type t in network g for layer s. The above is simply the triangle inequality for real numbers. This concludes the proof of Lemma 68. □

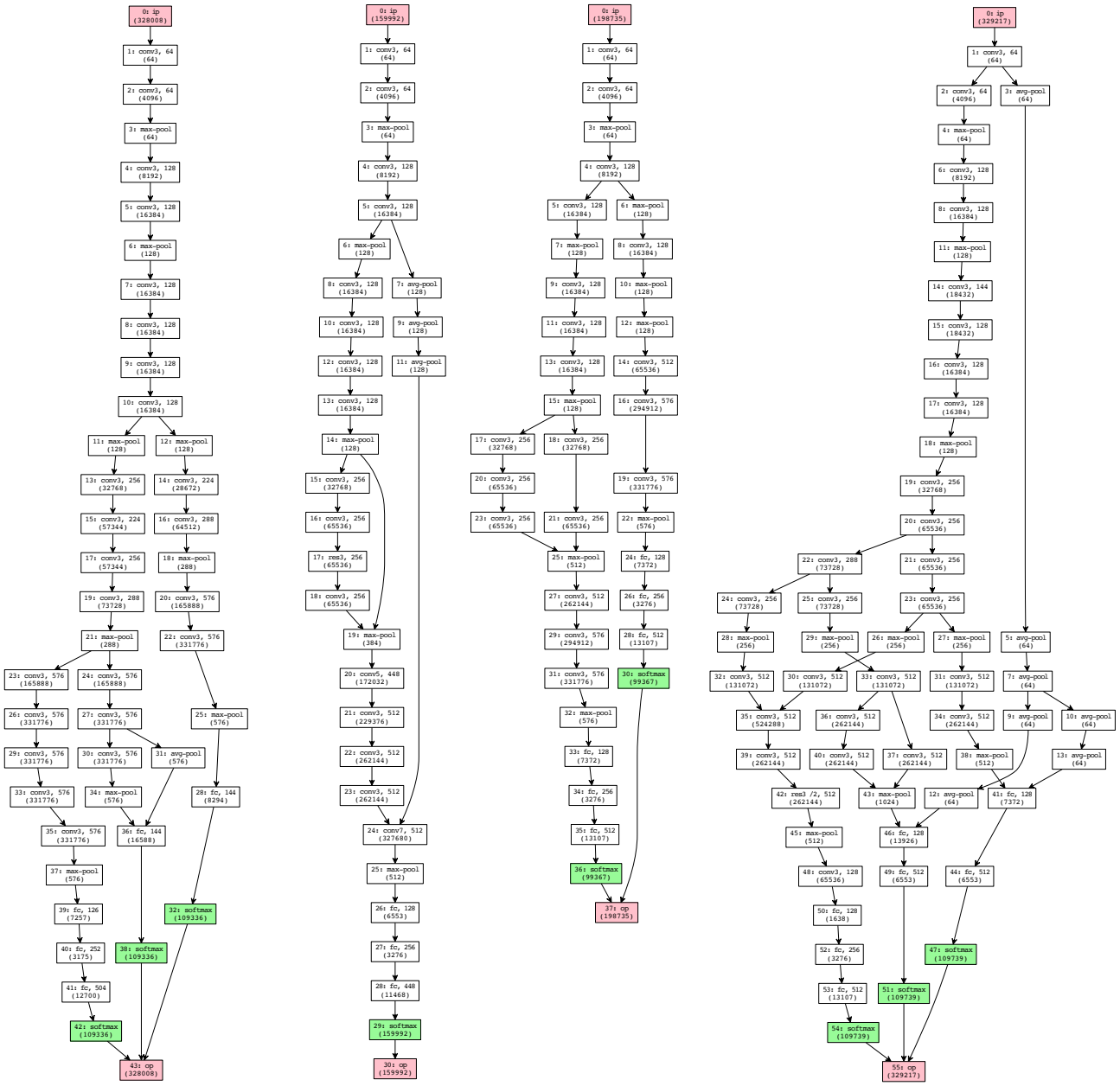


Figure 6.10: Optimal network architectures found with NASBOT on the Cifar10 dataset.

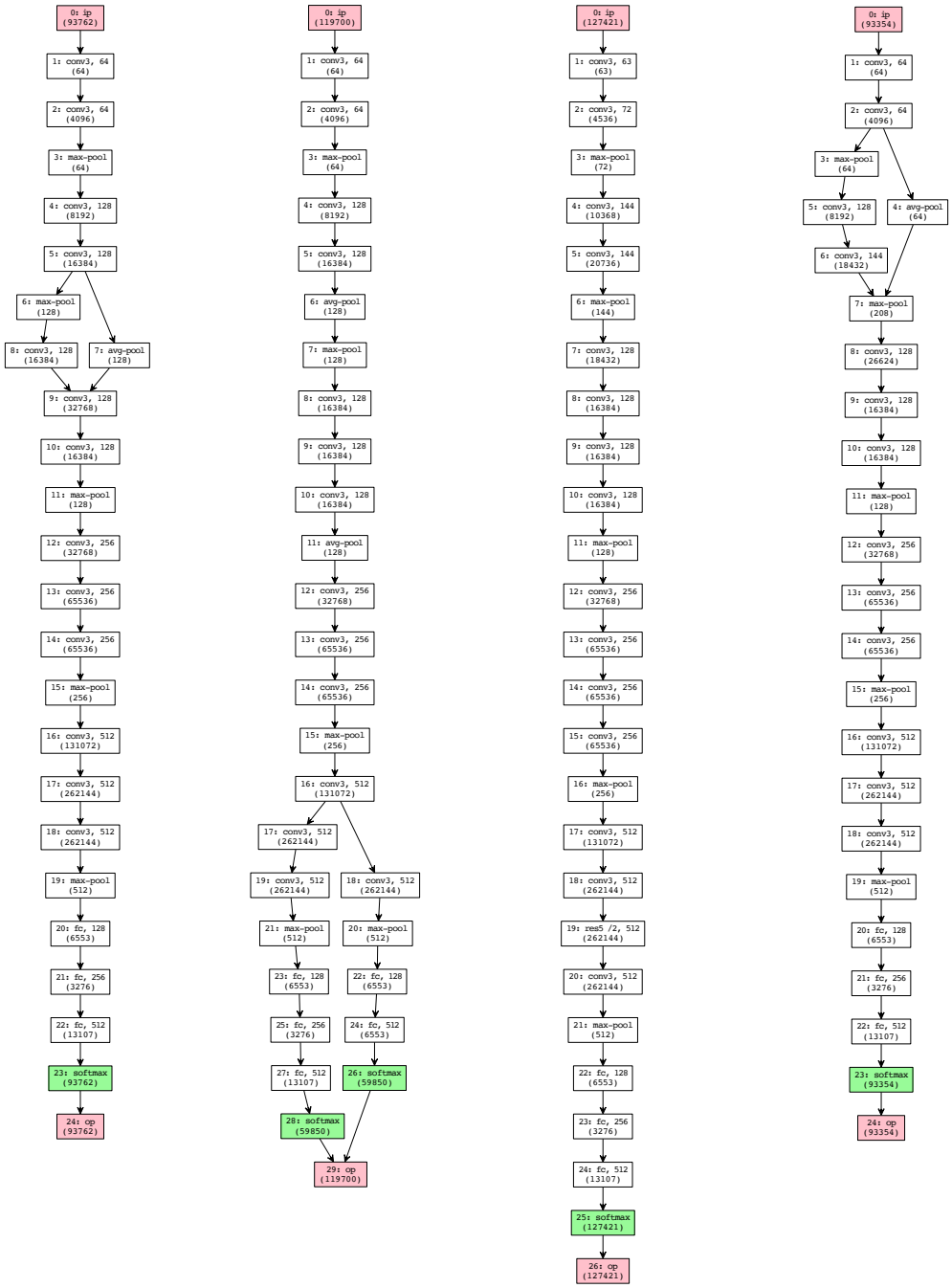


Figure 6.11: Optimal network architectures found with EA on the Cifar10 dataset.

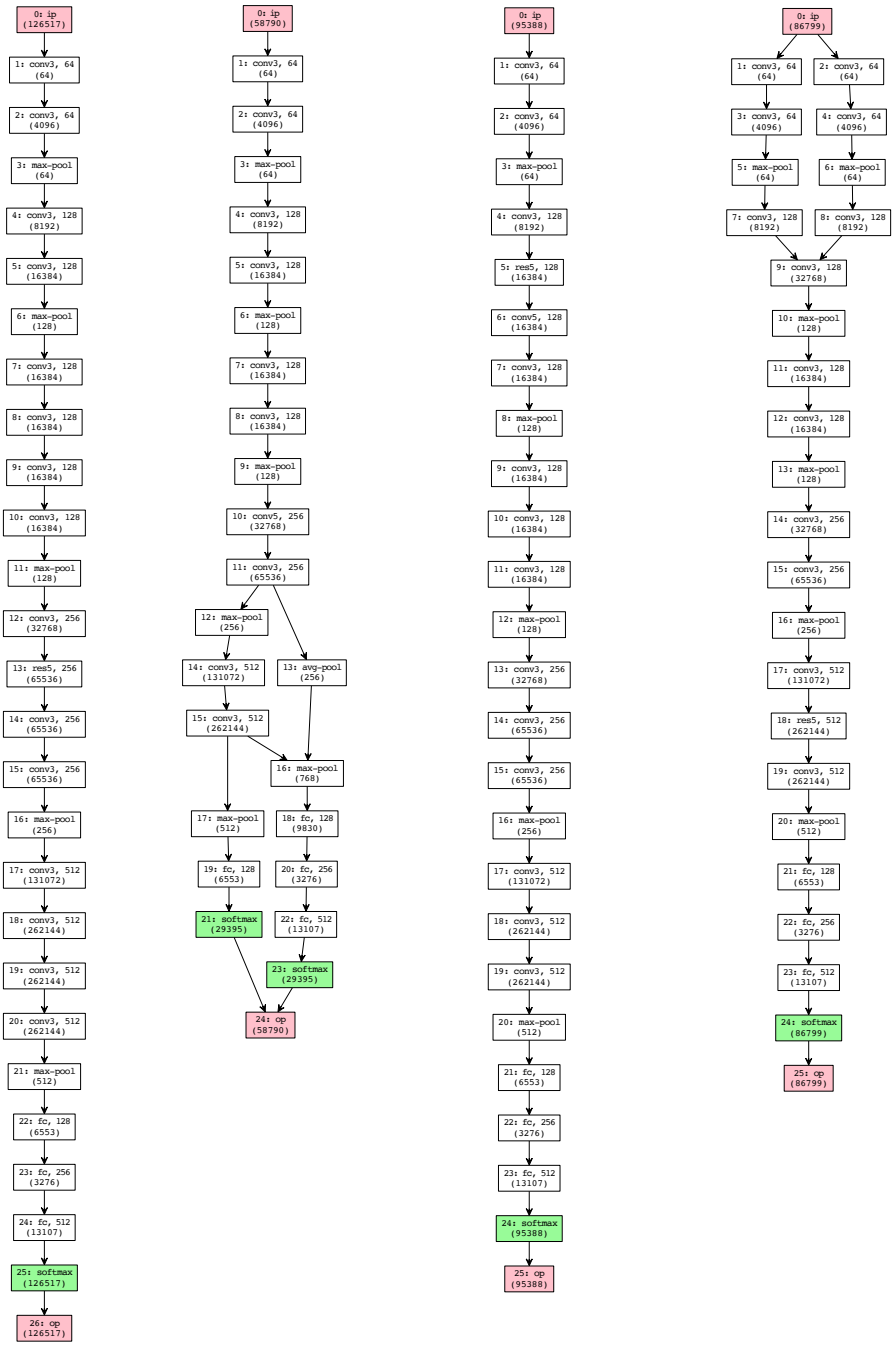


Figure 6.12: Optimal network architectures found with RAND on the Cifar10 dataset.

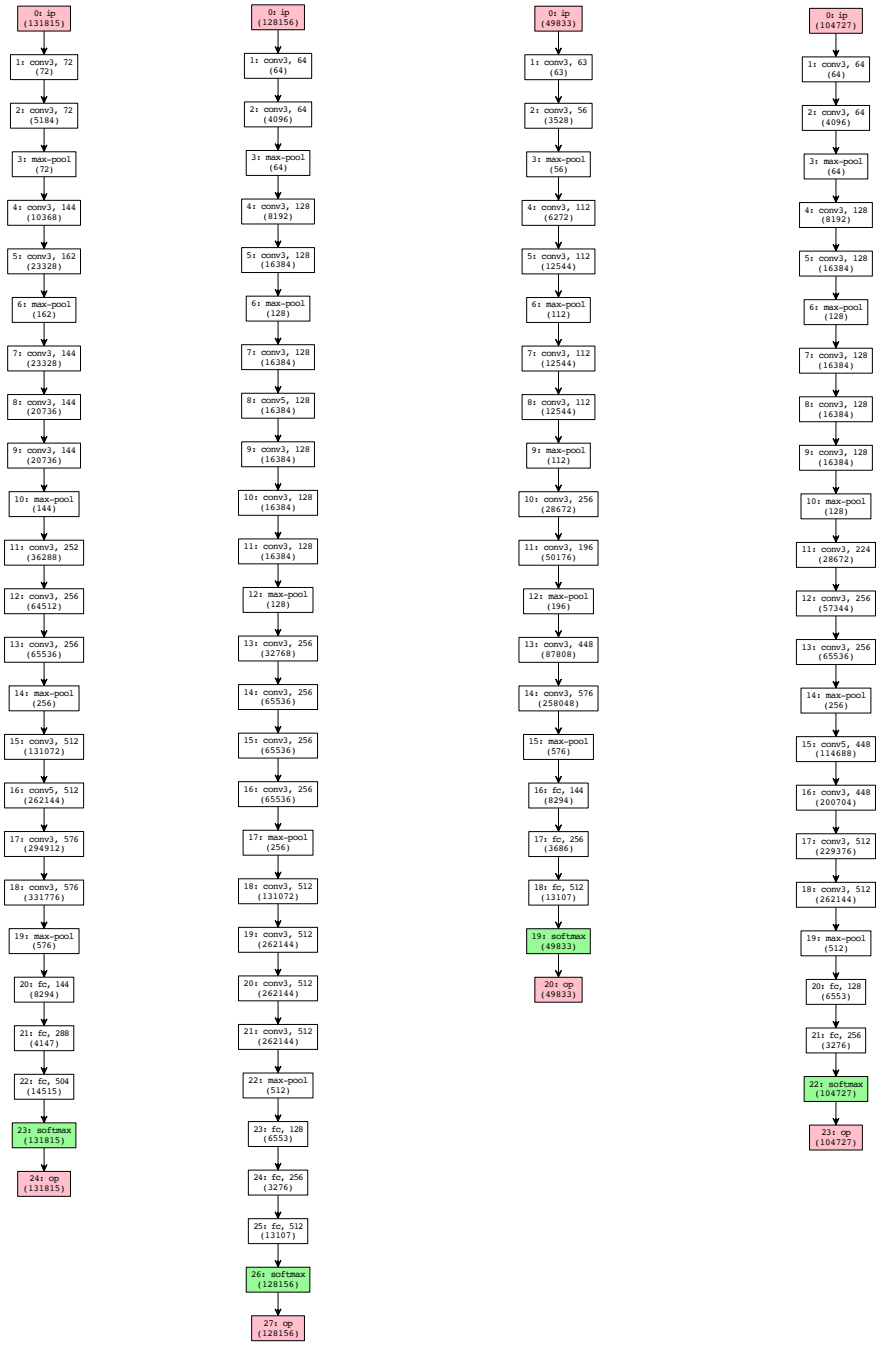


Figure 6.13: Optimal network architectures found with TreeBO on the Cifar10 dataset.

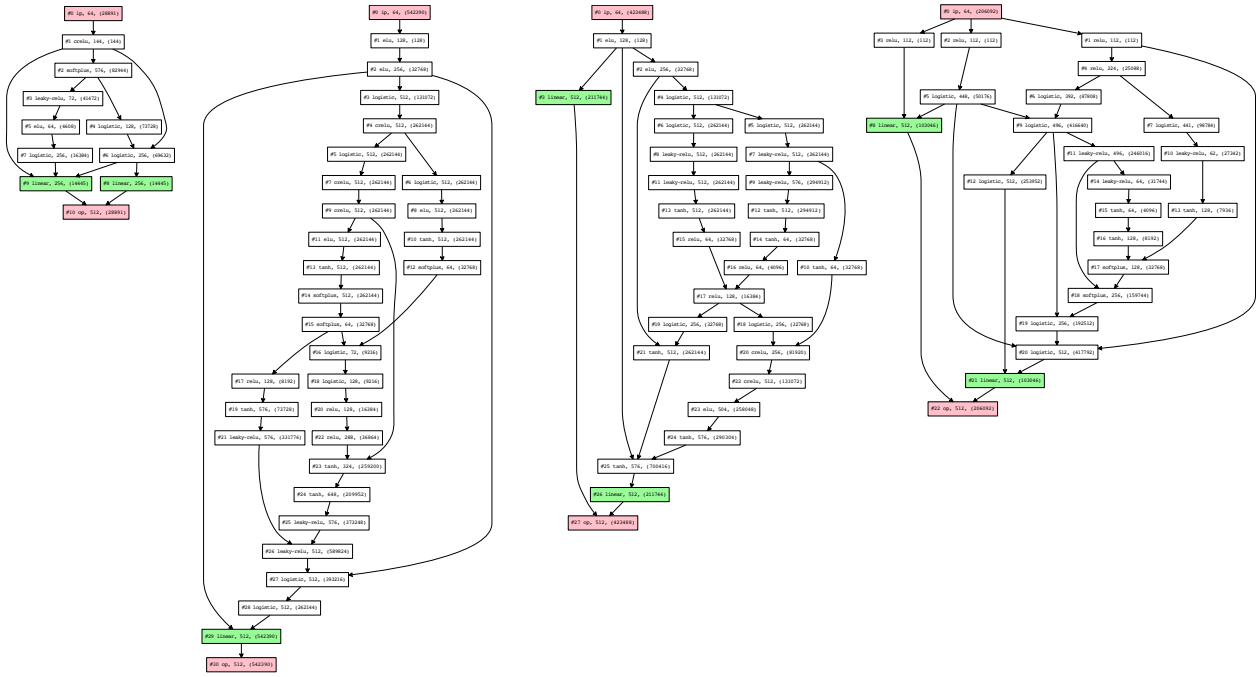


Figure 6.14: Optimal network architectures found with NASBOTon the indoor location dataset.

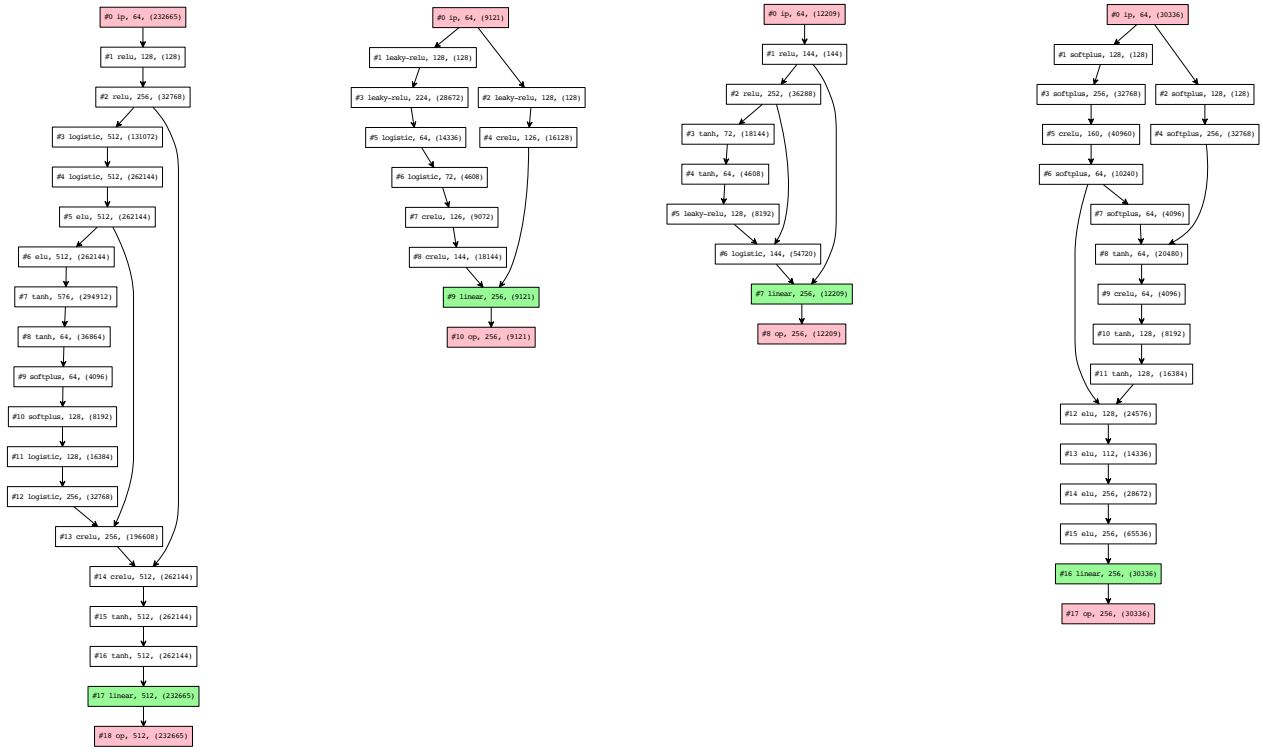


Figure 6.15: Optimal network architectures found with EA on the indoor location dataset.

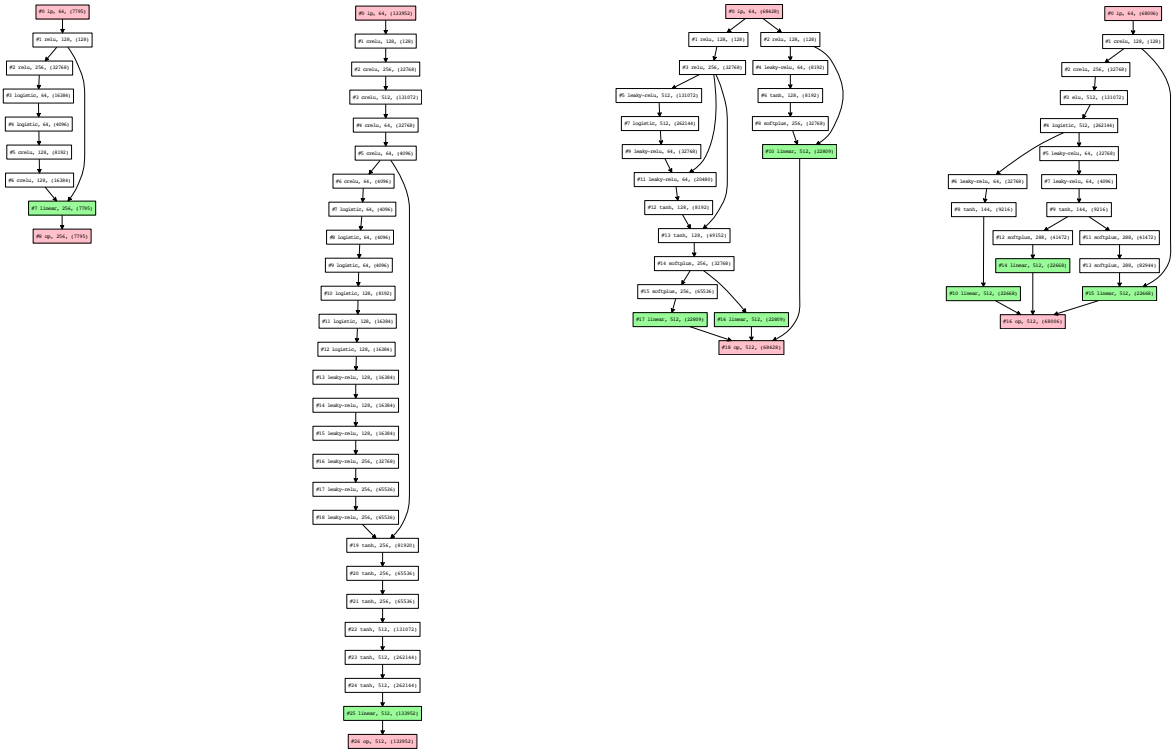


Figure 6.16: Optimal network architectures found with RAND on the indoor location dataset.

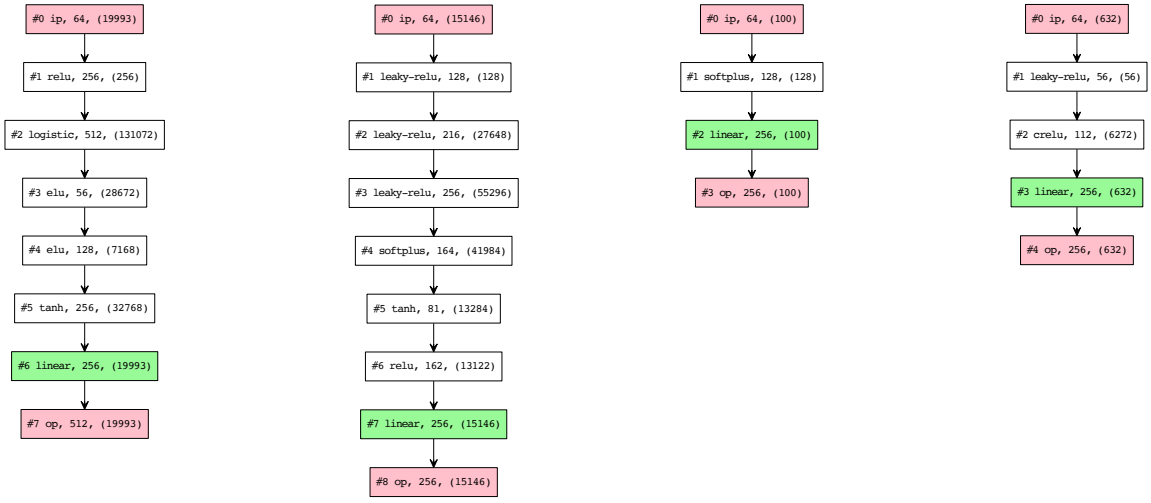


Figure 6.17: Optimal network architectures found with TreeBO on the indoor location dataset.

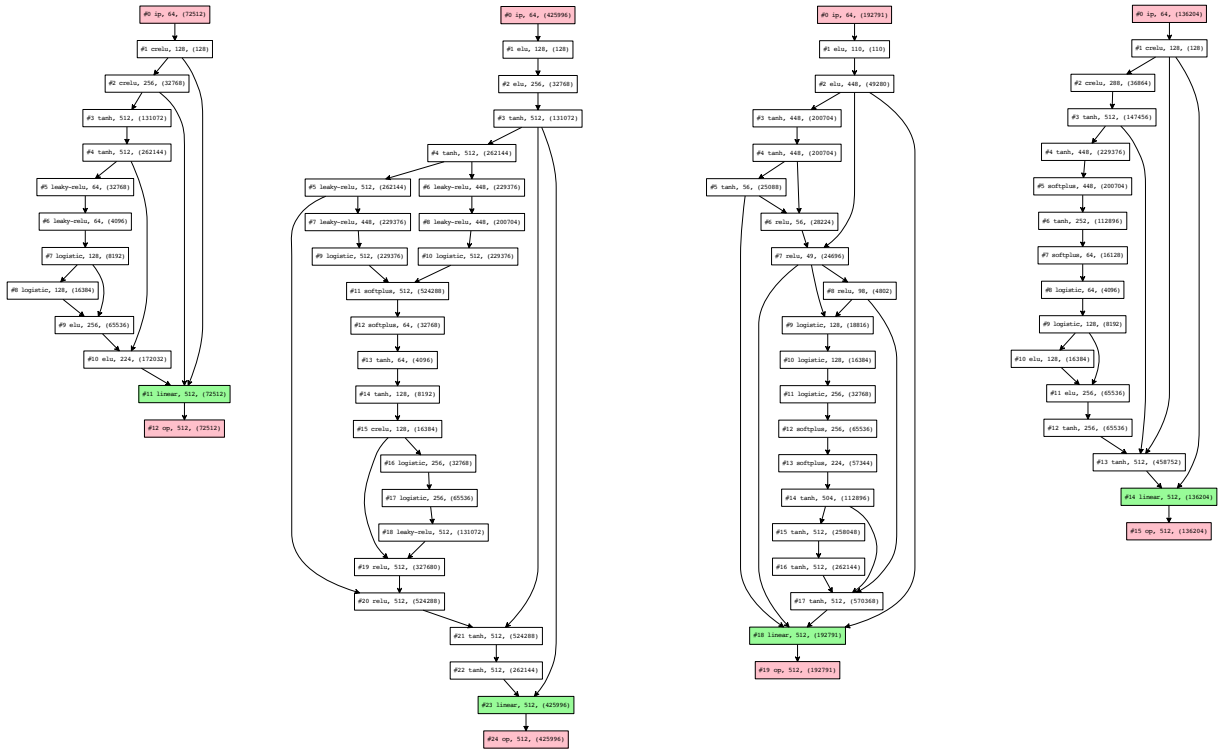


Figure 6.18: Optimal network architectures found with NASBOT on the slice localisation dataset.

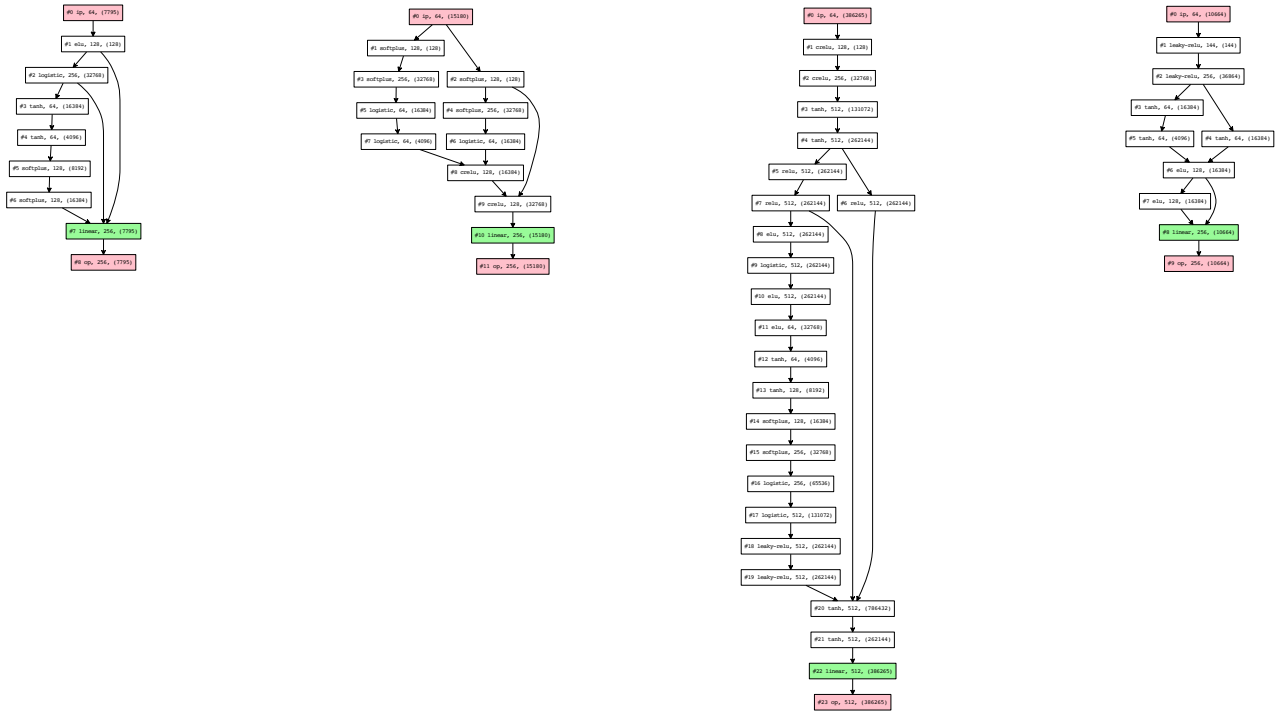


Figure 6.19: Optimal network architectures found with EA on the slice localisation dataset.

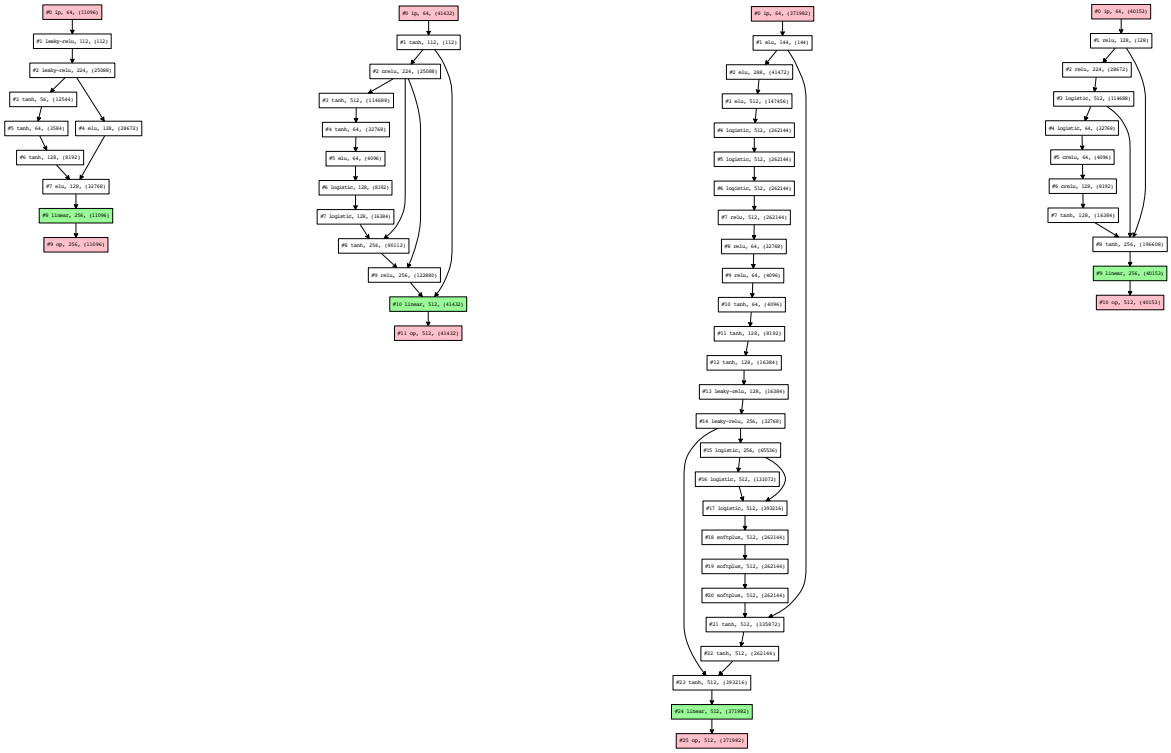


Figure 6.20: Optimal network architectures found with RAND on the slice localisation dataset.

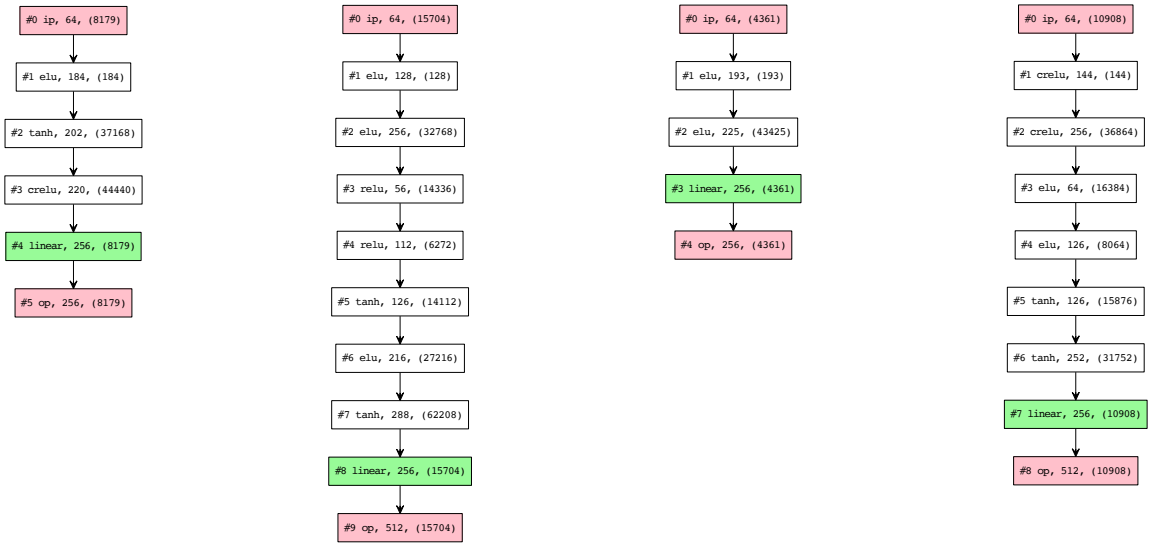


Figure 6.21: Optimal network architectures found with TreeBO on the slice localisation dataset.

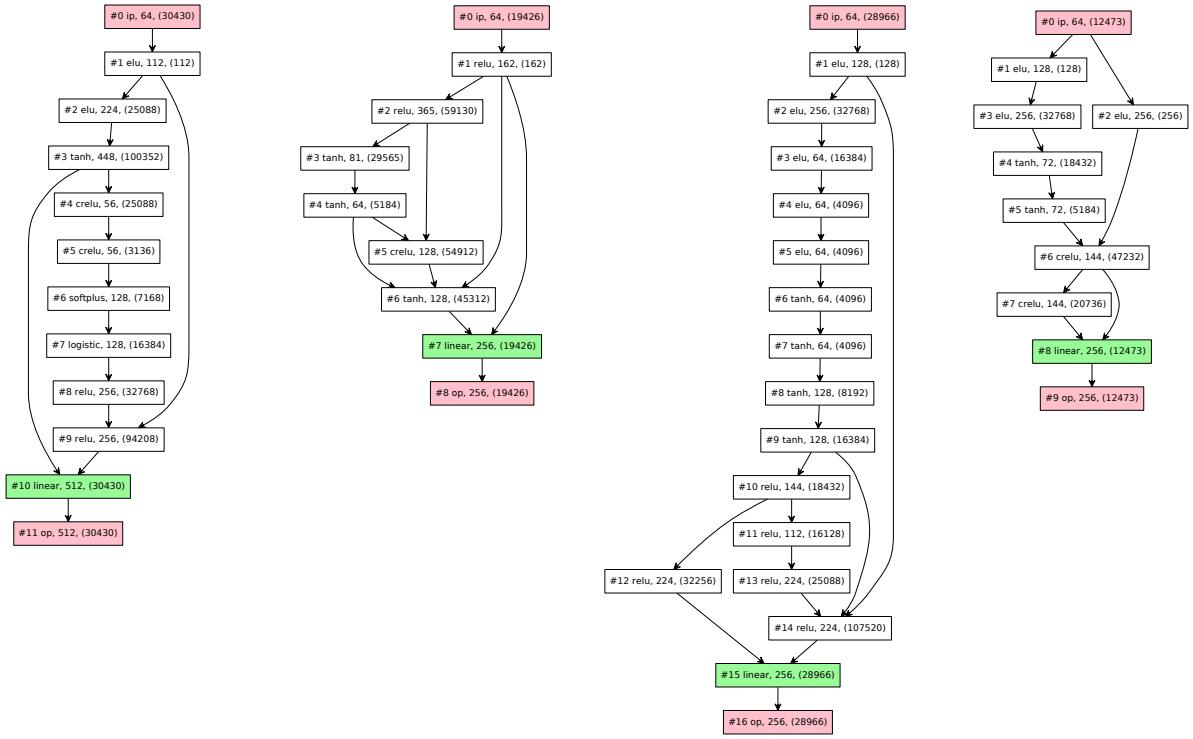


Figure 6.22: Optimal network architectures found with NASBOTon the naval propulsion dataset.

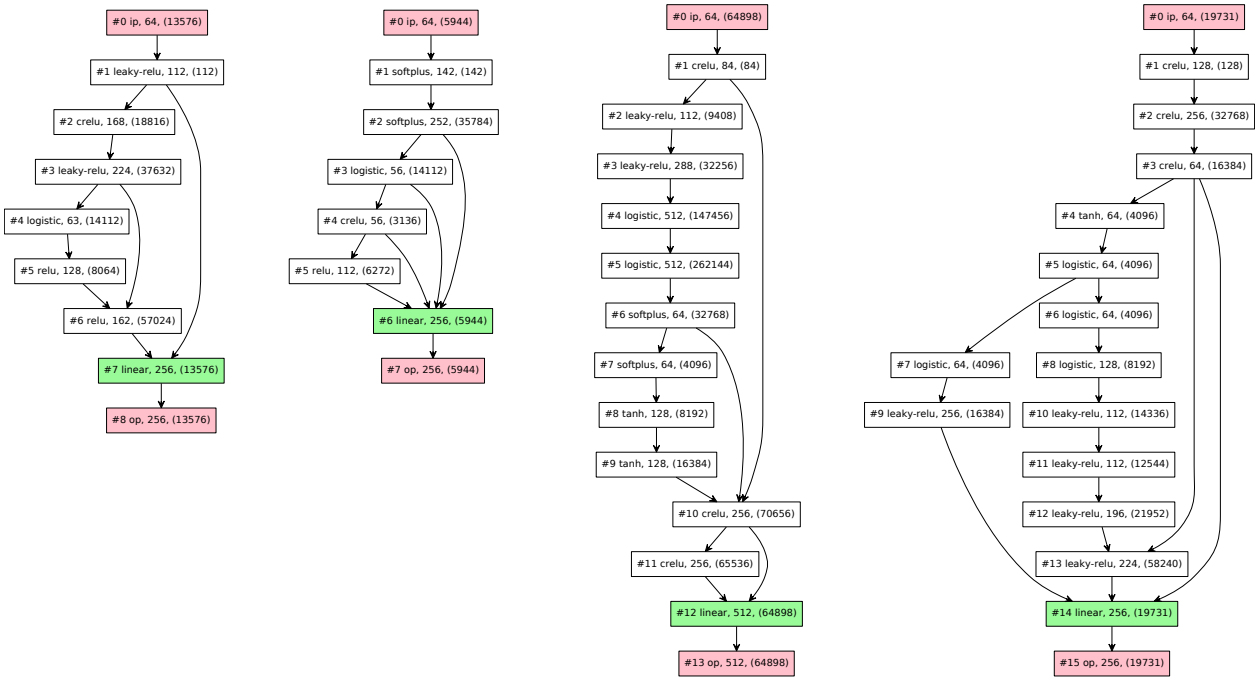


Figure 6.23: Optimal network architectures found with NASBOTon the news dataset.

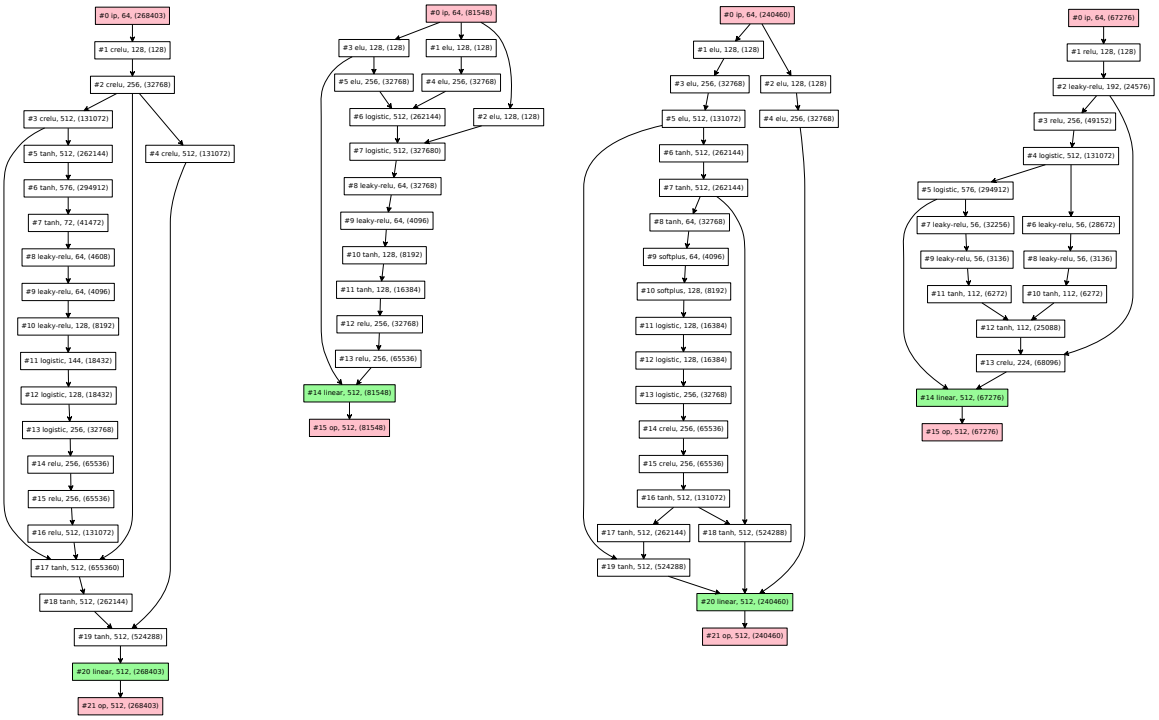


Figure 6.24: Optimal network architectures found with NASBOTon the protein structure prediction dataset.

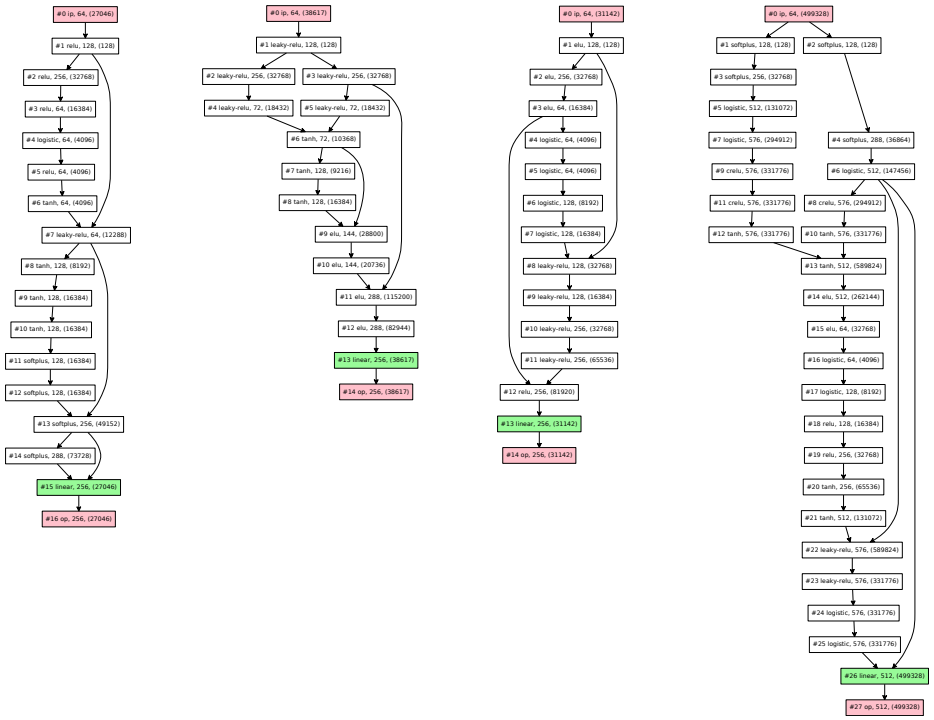


Figure 6.25: Optimal network architectures found with NASBOTon the blog dataset.

Chapter 7

Dragonfly: An Open Source Library for Robust & Scalable Bandit Optimisation

Model-based Bayesian methods for bandits, such as Gaussian process bandits, provide a powerful framework for optimising expensive black-box functions. They use introspective Bayesian models of the function to reason about where to perform the next evaluation, and are consequently more sample efficient than other methods for black-box optimisation. However, today there is a large disconnect between theoretical and methodological developments and the availability of practical tools that can be deployed in real world problems.

In this chapter, which describes the work in Kandasamy et al. [133], we develop Dragonfly, an open source Python platform for Bayesian optimisation. Our contributions in this chapter are as follows.

1. We integrate our individual efforts on scalability in the previous chapters into one framework which enables us to leverage opportunities and address challenges unique to modern large scale problems.
2. We develop new ensemble methods for choosing between multiple acquisitions and multiple models. We demonstrate that the ensemble methods are more robust, working consistenty better on a wider range of problems, than approaches which aim to make a single best choice over the entire optimisation process.
3. We develop a series of methodological improvements that enable the application of Bayesian optimisation in complex domains, including Euclidean, integral, discrete, discrete numeric spaces and combinatorial spaces such as neural networks.
4. We compare Dragonfly to several other packages and algorithms for black-box optimisation and demonstrate that we perform better or competitively in a variety of synthetic benchmarks and real world tasks in computational astrophysics and model selection. Crucially, Dragonfly is able to consistently perform well across a wide array of problems.

Dragonfly is released open source at `dragonfly.github.io`. It is compatible with Python 2

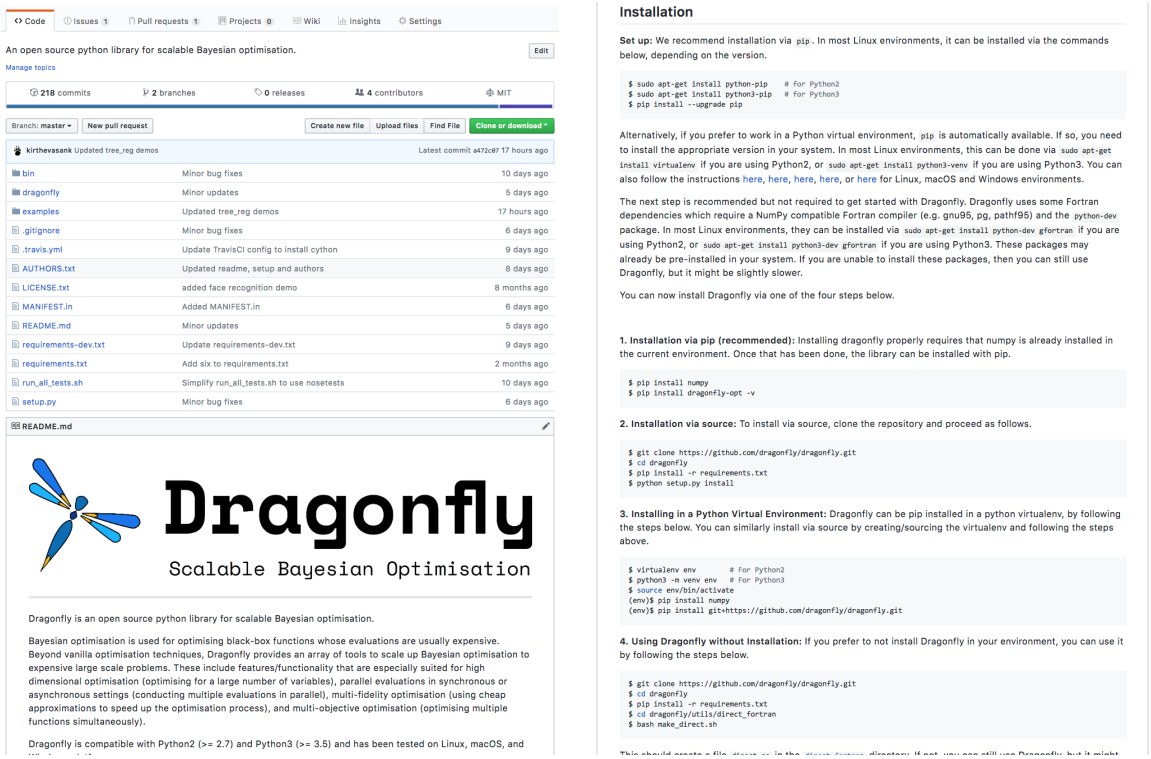


Figure 7.1: A screenshot of the Dragonfly repository on Github, availabe at `dragonfly.github.io`.

and Python 3 and has been tested on Linux, Mac OS, and Windows platforms. We have released Dragonfly under an MIT license which we hope will foster future research in bandit methods.

This chapter is organised as follows. Chapter 7.1 describes our ensemble methods for robust Bayesian optimisation and Chapter 7.2 explains how we combine our different scalability approaches. Chapter 7.3 describes the implementation of various parts of the BO pipeline in Dragonfly. Chapter 7.4 contains the experiments while Chapter 7.5 provides details on the user API for Dragonfly.

7.1 Robust Bayesian Optimisation in Dragonfly

We begin by describing our methods for robust GP based Bayesian optimisation in Dragonfly. We favour randomised ensemble approaches which use multiple acquisitions and GP hyperparameter values, which we found to be more robust as compared to making a single best choice.

7.1.1 Choice and Optimisation of Acquisition

Dragonfly implements several common acquisitions for BO such as GP-UCB, GP-EI, TTEI [199], TS, Add-GP-UCB, and PI [150]. The general practice in the BO literature has been for a practi-

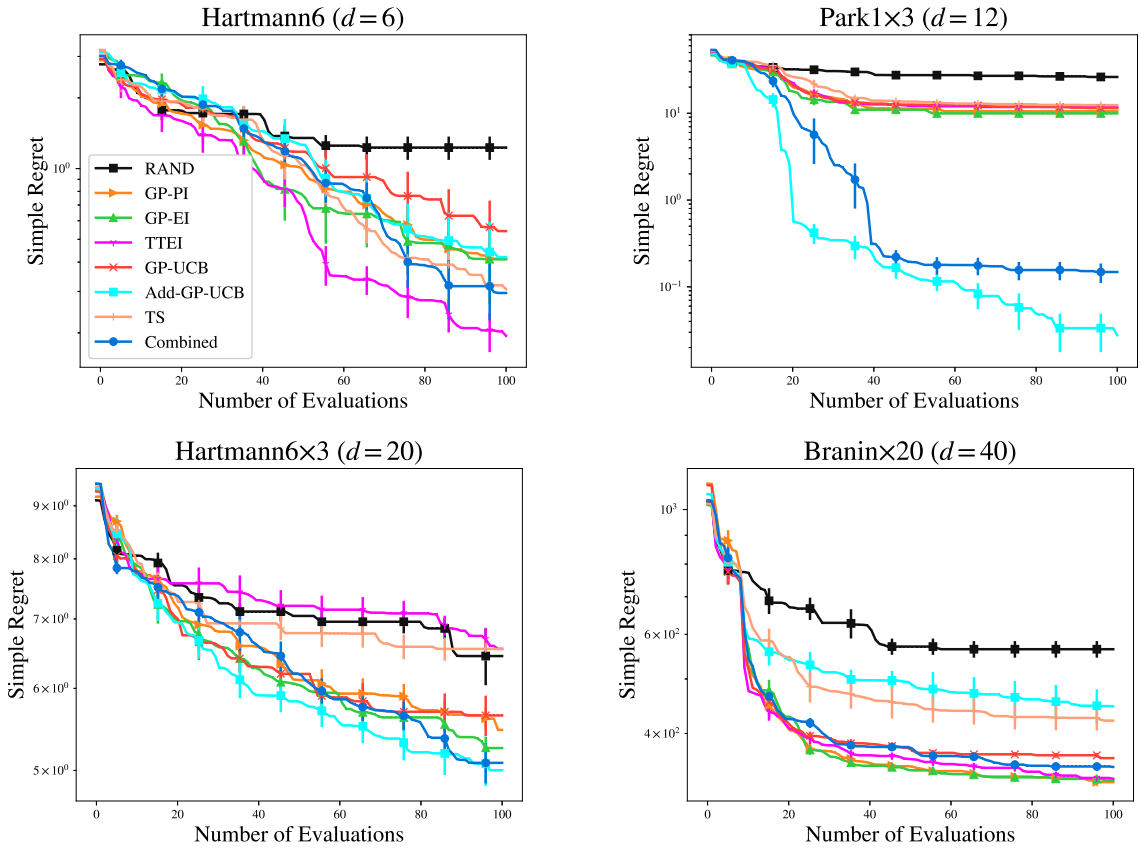


Figure 7.2: Comparison of using individual acquisitions such as GP-UCB, GP-EI, TTEI, TS, PI, and Add-GP-UCB versus the ensemble method as described in Chapter 7.1.1. We have also shown random sampling (RAND) for comparison. The ensemble approach is typically able to perform almost as well as the single best acquisition on each individual problem. We plot the simple regret (1.1), so lower is better. Error bars indicate one standard error. All curves were produced by averaging over 10 independent runs.

tioner to pick their favourite acquisition, and use it for the entire optimisation process. However, the performance of each acquisition can be very problem dependent, as demonstrated in Figure 7.2. Therefore, in Dragonfly, we adopt a randomised ensemble approach which chooses different acquisitions at different iterations instead of attempting to pick a single best one.

Our ensemble routine maintains a list of m acquisitions ℓ^{acq} along with a weight vector $w_t^{\text{acq}} = \{w_t^{\text{acq}}[\alpha]\}_{\alpha \in \ell^{\text{acq}}} \in \mathbb{R}^m$. We set $w_0^{\text{acq}}[\alpha] = \omega_0^{\text{acq}}$ for all $\alpha \in \ell^{\text{acq}}$. Suppose at time step t, we chose acquisition θ and found a higher f value than the current best value. We then update $w_{t+1}^{\text{acq}}[\alpha] \leftarrow w_t^{\text{acq}}[\alpha] + \mathbb{1}(\theta = \alpha)$; otherwise, $w_{t+1}^{\text{acq}}[\alpha] \leftarrow w_t^{\text{acq}}[\alpha]$. At time t, we choose acquisition $\theta \in \ell^{\text{acq}}$ with probability $w_t^{\text{acq}}[\theta] / \sum_\alpha w_t^{\text{acq}}[\alpha]$. This strategy initially samples all acquisitions with equal probability, but progressively favours those that perform better on the problem.

By default, we set $\ell^{\text{acq}} = \{\text{GP-UCB, GP-EI, TS, TTEI}\}$; for entirely Euclidean domains, we also include Add-GP-UCB. We do not incorporate PI since it consistently underperformed other acquisitions in our experiments. As Figure 7.2 indicates, the ensemble approach is robust across

different problems, either performing almost as good as the best acquisition on the given problem or outperforming all of them. Shahriari et al. [222] use an entropy based approach to select among multiple acquisitions. This, however requires computing/optimising all of them which can be computationally expensive. We found that our approach, while heuristic in nature, performed well in our experiments.

Finally, we note that we do not implement entropy-search based acquisitions [94, 95, 259, 260] since their computation, can, in general, be quite expensive.

7.1.2 GP Hyperparameters

One of the main challenges in GP based BO is that the selection of the GP hyperparameters[1] themselves could be notoriously difficult. While a common approach is to choose them by maximising the marginal likelihood, in some cases, this could also cause overfitting in the GP, especially in the early iterations [232]. The most common strategy to overcome this issue is to maintain a prior on the hyperparameters and integrate over the posterior [98, 167, 232]. However, this can be very computationally burdensome, and hence prohibitive in applications where function evaluations are only moderately expensive. Instead, in this work, we focus on a different approach that uses posterior sampling. Precisely, at each iteration, one may sample a set of GP hyperparameters from the posterior conditioned on the data, and us them for the GP at that iteration. Intuitively, this is similar to a Thompson sampling procedure where the prior on the hyperparameters specifies a prior on a meta-model, and once we sample the hyperparameters, we use an acquisition of our choice. When this acquisition is TS, this procedure is exactly Thompson sampling using the meta-prior.

Our experience suggested that maximising the marginal likelihood (ML) generally worked well in settings where the function was smooth; for less smooth functions, sampling from the posterior (PS) tended to work better. We speculate that this is because, with smooth functions, a few points are sufficient to estimate the landscape of the function, and hence maximum likelihood does not overfit; since it has already estimated the GP hyperparameters well, it does better than PS. On the other hand, while ML is prone to overfit for non-smooth functions, the randomness in PS prevents us from getting stuck at bad GP hyperparameters. As we demonstrate in Figure 7.3, either of these approaches may perform better than the other depending on the problem.

Therefore, similar to how we handled the acquisitions, we adopt a randomised ensemble approach where we choose either maximum likelihood or sampling from the posterior at every iteration. Precisely, our GP hyperparameter tuning strategy proceeds as follows. After every n_{cyc} evaluations of f, we fit a single GP to it via maximum likelihood, and also sample n_{cyc} hyperparameter values from the posterior. At every iteration, the algorithm chooses either maximum likelihood or sampling from the posterior in a randomised fashion. If it chooses the former, it uses the single best GP, and if it chooses the latter, it uses one of the sampled values. For this, we let $w_t^{\text{hp}} = \{w_t^{\text{hp}}[h]\}_{h \in \ell^{\text{hp}}} \in \mathbb{R}^2$ where $\ell^{\text{hp}} = \{\texttt{ML}, \texttt{PS}\}$ and choose strategy $h \in \ell^{\text{hp}}$ with probability

[1] "hyperparameters" here refer to those of the GP, and should not be conflated with the title of this thesis, where "hyperparameter tuning" refers to the general practice of optimising a system's performance.

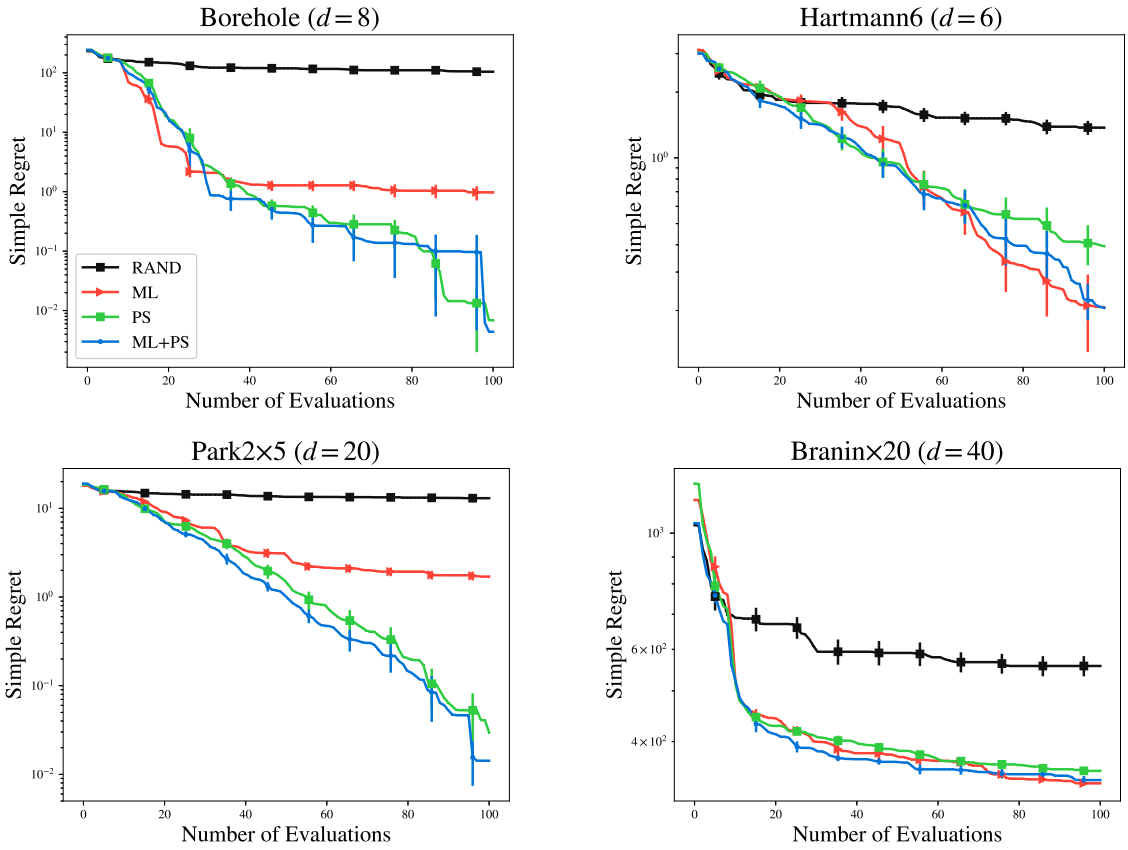


Figure 7.3: Comparison of using only maximum likelihood (ML), only posterior sampling (PS) and the ensemble method (ML+PS) as described in Chapter 7.1.2. We have also shown random sampling (RAND) for comparison. The combined ensemble approach is able to perform as well as or better than the best choice for the given problem. We plot the simple regret (1.1), so lower is better. Error bars indicate one standard error. All curves were produced by averaging over 10 independent runs.

$w_t^{\text{hp}}[h]/(w_t^{\text{hp}}[\texttt{ML}] + w_t^{\text{hp}}[\texttt{PS}])$. We update w_t^{hp} in a manner similar to w_t^{acq}. Figure 7.3 demonstrates that this strategy is usually able to do as well as if not better than the best of ML and PS on the given problem. We also note that we fit GP hyperparameters only once every n_{cyc} evaluations due to computational reasons; regardless of the chosen hyperparameters, the GP posterior (2.1), is computed using all the data collected up until the current iteration.

For maximum likelihood of continuous hyperparameters, we use either DiRect or PDOO; if discrete hyperparameters are also present, we optimise the continuous parameters for all choices of discrete values; this was feasible since there were only a handful of values for the discrete GP hyperparameters we encountered. For posterior sampling of the GP hyperparameters, we impose a uniform prior and use the following Gibbs sampling procedure. At every iteration, we iterate through each of the hyperparameters in a randomised order; on each visit to a hyperparameter, we fix the values of the rest of the hyperparameters, and sample a value of the current hyperparameter conditioned on the fixed values. For continuous hyperparameters, we do so via slice sampling [177] and for discrete hyperparameters we do so via Metropolis-Hastings [89]. We

use a burn-in of 1000 samples and collect a sample every 100 samples from thereon to avoid correlation. Our implementations of slice sampling and Metropolis-Hastings are taken from the PyMC3 library [215], and adapted to suit our setting. We also experimented with NUTS [97] for continuous hyperparameters, and found it to be significantly more computationally expensive but with no significant gains in performance over slice sampling.

7.2 Scalable Bayesian Optimisation in Dragonfly

In addition to the above techniques for robust BO, Dragonfly integrates the techniques developed in earlier chapters of this thesis for scalable Bayesian optimisation. We describe them below, along with modifications to facilitate the integration.

High Dimensions

In our original work [120], we used Add-GP-UCB as a single acquisition throughout the entire BO process. Here, as described above in Chapter 7.1, we treat it as one of many acquisitions that can be chosen from in the ensemble approach. Moreover, while the original version of Add-GP-UCB used a fixed group dimensionality, in Dragonfly we treat the maximum group dimensionality and the decomposition of the dimensions as GP kernel hyperparameters, that need to be chosen. This can pose some computational concerns since the number of possible decompositions grows combinatorially with dimension. We address these challenges, both for maximising the marginal likelihood (`ML`) and sampling from the posterior (`PS`) as follows.

For the former, since a complete maximisation can be computationally challenging, we perform a partial maximisation by first choosing an upper bound p_{max} for the maximum group dimensionality. For each group dimensionality $p \in \{1, \ldots, p_{\text{max}}\}$, we select k different decompositions chosen at random. For each such decomposition, we optimise the marginal likelihood over the remaining hyperparameters, and choose the decomposition with the highest likelihood.

For sampling from the posterior, it is sufficient to specify a prior distribution to sample from for our Gibbs sampling procedure. We use a uniform prior over the integral domain $\{1, \ldots, p_{\text{max}}\}$ for the maximum group dimensionality p. We then randomly shuffle the coordinates $\{1, \ldots, d\}$ to produce an ordering. Given a maximum group dimensionality p and an ordering, one can uniquely identify a decomposition by iterating through the ordering and grouping them in groups of size at most p. For example, in a $d = 7$ dimensional problem, $p = 3$ and the ordering $4, 7, 3, 6, 1, 5, 2$ yields the decomposition $\{(3, 4, 7), (1, 5, 6), (2)\}$ of three groups having group dimensionalities 3, 3, and 1 respectively. It is straightforward to efficiently sample from this distribution and is used as the prior in our slice sampling procedure.

Since the set of permutations forms a very large combinatorial space, a partial maximisation for `ML` may not recover the true maximiser of the marginal likelihood. Similarly, for posterior sampling, we might not be able to search this large combinatorial space properly with a small

number of samples. However, in this work we adopt a pragmatic view of the additive model (3.1) and treat it as a simpler statistical model to model f in the small sample regime, as opposed to truly believing that f is additive. Under this view, with a partial maximisation we can hope to capture some existing marginal structure in f and obtain a reasonable model for the data; at the same time, even an exhaustive maximisation will not do much better than a partial maximisation if there is no additive structure. Likewise, a few samples would serve the purpose of finding a suitable additive model when sampling from the posterior. By default, Dragonfly uses $p_{\max} = 6$ and $k = 25$ when using the Add-GP-UCB acquisition. Shortly, we will describe some additional details when combining an additive model with multi-fidelity optimisation.

Other Additive Models in Dragonfly

Note that the additive model of (3.1) can be used with other acquisitions such as GP-UCB and GP-EI; even though the acquisition might be difficult to maximise, one can still expect to have statistical gains due to the simpler model. In instances where evaluation of f is very expensive, maximising φ_t may not be a bottleneck allowing the use of other acquisitions. Another form of p^{th} order additive model commonly used in the literature takes the form $f(x) = \sum_{1 \leq i_1 < i_2 < \cdots < i_p \leq d} f^{(j)}(x_{i_1}, x_{i_2}, \ldots, x_{i_p})$, where the summation is over all $\binom{d}{p}$ combinations of p coordinates [54, 117, 224]. While there is a combinatorial number of terms, an additive kernel over this model can be computed using the Girard-Newton formula for elementary symmetric polynomials [165]. Dragonfly has functionality to use group-additive and ESP style kernels in the GP if necessary. However, they are not used by default.

Multi-fidelity Optimisation

Dragonfly implements functionality for multi-fidelity Bayesian optimisation, specifically the BOCA framework of Chapter 4.3. We opt for BOCA over MF-GP-UCB since it can handle more general approximations, and has fewer parameters to tune. As we did before, we assume the existence of a function $g : \mathcal{Z} \times \mathcal{X} \to \mathbb{R}$ where $\mathcal{X}$ is the domain and $\mathcal{Z}$ is the fidelity space. We let $g \sim \mathcal{GP}(\mathbf{0}, \kappa)$ and upon querying at (z, x) we observe $y = g(z, x) + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \eta^2)$. $\kappa : (\mathcal{Z} \times \mathcal{X})^2 \to \mathbb{R}$ is the prior covariance defined on the product space, which, as before, is taken to be of the form. $\kappa([z, x], [z', x']) = \kappa_0 \, \kappa_{\mathcal{Z}}(z, z') \, \kappa_{\mathcal{X}}(x, x')$. Here, $\kappa_0 \in \mathbb{R}_+$ is the scale of the kernel and $\kappa_{\mathcal{Z}}, \kappa_{\mathcal{X}}$ are kernels defined on $\mathcal{Z}, \mathcal{X}$ such that $\|\kappa_{\mathcal{Z}}\|_\infty = \|\kappa_{\mathcal{Z}}\|_\infty = 1$. At time step t, a multi-fidelity algorithm would choose a fidelity $z_t \in \mathcal{Z}$ and a point $x_t \in \mathcal{X}$ in the domain to evaluate based on its previous fidelity, domain point, observation triples $\{(z_i, x_i, y_i)\}_{i=1}^{t-1}$.

A key property about the two-step fidelity selection criterion in BOCA (see Chapter 4.3.2 and equation (4.24)) that we prove in Kandasamy et al. [127] is that is chooses a fidelity z_t with good cost to information trade-off, *given* that we are going to evaluate g at x_t. In particular, it applies to x_t chosen in any arbitrary fashion, and not necessarily via an upper confidence bound. Therefore, while our theoretical results in Chapters 4.3 and 4.6 hold only for the selection of x_t

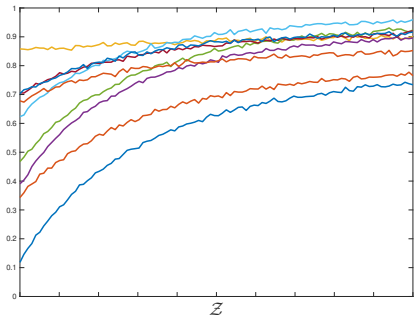


Figure 7.4: An illustration of GP sample paths drawn from the exponential decay kernel [242] conditioned on being positive. They are suitable for representing the validation accuracy along a fidelity dimension in machine learning applications where, for e.g. validation accuracy tends to increase as we use more data and/or train for more iterations.

via (4.23), in Dragonfly we adopt the two step procedure described above, but allow x_t to be chosen also via other common acquisitions.

In Kandasamy et al. [127], we choose the fidelity kernel $\kappa_{\mathcal{Z}}$ to be a radial kernel. This typically induces smoothness in g acros $\mathcal{Z}$, which can be useful in many applications. However, in many model selection problems we see in statistics and machine learning, the approximations are obtained either by using less data and or less iterations in an iterative training procedure. In such cases, as we move to the expensive fidelities, the validation performance tends to be monotonic – for example, when the size of the training set increases, one expects the validation accuracy to keep improving. Swersky et al. [242] demonstrated that an exponential decay kernel $\kappa_{\text{ed}} : \mathbb{R}_+ \times \mathbb{R}_+ \to \mathbb{R}$ of the following form can strongly support such sample paths,

$$\kappa_{\text{ed}}(u, u') = \frac{1}{(u + u' + 1)^{\alpha}}.$$

We have illustrated such sample paths in Figure 7.4. In a p dimensional fidelity space, one can use $\kappa_{\mathcal{Z}}(z, z') = \prod_{i=1}^{p} \kappa_{\text{ed}}(z_i, z_i')$ as the kernel for the fidelity space if all fidelity dimensions exhibit such behaviour. Unfortunately, the information gain $\xi(z)$ is not defined for non-radial kernels. In Dragonfly, we use $\xi(z) = \|z - z_\bullet\|$ which is similar to the approximation of the information gain for SE kernels. Intuitively, as z moves away from $z_\bullet$, the information gap increases as $g(z, \cdot)$ provides less information about $g(z_\bullet, \cdot)$. The parameters of $\kappa_{\mathcal{Z}}$ are treated as hyperparameters of the GP and are chosen in the same way described above.

Combining multi-fidelity with Add-GP-UCB: When using an additive kernel $\kappa_{\mathcal{X}} = \sum_j \kappa^{(j)}$ for the domain $\mathcal{X}$ in multi-fidelity settings, the resulting product kernel also takes an additive form, $\kappa([z, x], [z', x']) = \sum_j \kappa_{\mathcal{Z}}(z, z')\kappa(x^{(j)}, x^{(j)\prime})$. When using Add-GP-UCB, in the first step we choose $x_t^{(j)} = \text{argmax}_{x^{(j)} \in \mathcal{X}^{(j)}} \nu_{t-1}^{(j)}(z_\bullet, x^{(j)}) + \beta_t^{1/2} \tau_{t-1}^{(j)}(z_\bullet, x^{(j)})$ for all j to obtain the next evaluation x_t. Here $\nu_{t-1}^{(j)}, \tau_{t-1}^{(j)}$ are the posterior GP mean and standard deviation of the j^{th} function in the above decomposition. Then we choose the fidelity z_t as specified in Chapter 4.3.2.

Parallelisation

BO, as described in Chapter 2.2, is a sequential algorithm which determines the next query after completing the previous queries. However, in many applications, we may have access to multiple workers and hence carry out several evaluations simultaneously. As we demonstrated

theoretically in Chapter 5, when there is high variability in evaluation times, it is prudent for the algorithm to operate in the *asynchronous* setting, where a worker is re-deployed immediately with a new evaluation once it completes its an evaluation. In contrast, if all evaluations take roughly the same amount of time, it is meaningful to wait for all workers to finish, and incorporate all their feedback before issuing out the next set of queries in batches.

In our implementations of all acquisitions except TS, we handle parallelisation using the hallucination technique of Desautels et al. [52], Ginsbourger et al. [73]. Here, we pick the next point exactly like in the sequential setting, but the posterior is based on $\mathcal{D}_t \cup \{(x, \mu_{t-1}(x))\}_{x \in F_t}$, where F_t are the points in evaluation by other workers at step t and μ_{t-1} is the posterior mean conditioned on just $\mathcal{D}_t$; this preserves the mean of the GP, but shrinks the variance around the points in F_t. The hallucination technique explicitly discourages the algorithm from picking points close to those that are in evaluation. However, in the case of Thompson sampling, as we demonstrated in Chapter 5, the inherent randomness ensures that the points chosen for parallel evaluation are sufficiently different [129]. Therefore, when using Thompson sampling, Dragonfly does not use hallucinations. Finally, we note that while there are other techniques for parallelising BO, they either require choosing additional parameters and/or are computationally expensive [78, 134, 221, 262].

Neural Architecture Search

Dragonfly implements the NASBOT algorithm of Chapter 6 for neural architecture search. Our implementation follows the description of Chapter 6 but integrates the robustness and multi-fidelity ideas into the framework. Specifically, we use a uniform prior on the 11 GP hyperparameters and sample them using slice sampling as described in Chapter 7.1.2. Moreover, we combine the TTEI, GP-EI, TS, and UCB acquisitions as described in 7.1.1.

Multi-fidelity Neural Architecture Search: Like other model selection problems, in architecture search, the cross validation performance can be approximated by training the model for a fewer iterations. For this, we define a one dimensional fidelity space which is the number of training iterations and use the exponential decay kernel on this space. We use the two-step BOCA procedure as described in Chapter 7.2: first choose a network using a sampled acquisition, then, choose the fidelity as described in Chapter 4.3.2.

7.3 Bayesian Optimisation Implementation in Dragonfly

We now describe our BO implementation in Dragonfly. These include some modest methodological contributions in optimising the acquisition and defining kernels, in order to be able to handle domains with different variable types and arbitrary constraints.

7.3.1 Domains and Fidelity Spaces

We begin by describing the various variable types that can be used in Dragonfly. Next, we describe how one may specify constraints on these variables. The variable types and the constraints will allow us to define domains and fidelity spaces over which we may optimise some function[2].

- **Euclidean domain:** A d dimensional rectangular Euclidean domain is specified by parameters $\{(a_i, b_i)\}_{i=1}^d$ where $\{a_i\}_{i=1}^d$ are the lower bounds and $\{b_i\}_{i=1}^d$ the upper bounds of each dimension. Here, $a_i, b_i \in \mathbb{R}$ for all i. $x \in \mathbb{R}^d$ is in this domain if $x_i \in [a_i, b_i]$ for all i.
- **Integral domain:** A d dimensional rectangular Integral domain is specified by parameters $\{(a_i, b_i)\}_{i=1}^d$ where $\{a_i\}_{i=1}^d$ are the lower bounds and $\{b_i\}_{i=1}^d$ the upper bounds of each dimension. Here, $a_i, b_i \in \mathbb{Z}$ for all i. $x \in \mathbb{Z}^d$ is in this domain if $x_i \in [a_i, b_i]$ for all i.
- **Discrete domain:** A discrete domain is specified by a set of items A. We similarly define a product discrete domain to be a set of set of items $\times_{i=1}^d A_i$ and any d-tuple x is in this domain if $x_i \in A_i$ for all i. A special type of Discrete domain is the Boolean domain where each $A_i = \{0, 1\}$ is used to indicate if a variable of interest is either off (0) or on (1).
- **Discrete numeric domain:** Defined similarly as discrete domain and product discrete domain except each item in the A_i's is a real number.
- **Discrete Euclidean Domains:** This is similar to a discrete domain, but each element is a Euclidean vector and all Euclidean vectors in the domain have the same dimensionality.
- **Neural network domain:** A neural network domain is used to define spaces over neural network architectures for neural architecture search. Such a domain is specified by constraints on the following parameters.
  - `max_num_layers`, `min_num_layers`: The maximum and minimum number of layers in the network.
  - `max_num_units_per_layer`, `min_num_units_per_layer`: The maximum and minimum number of computational units in a layer.
  - `max_num_edges`: The maximum number of directed edges in the network, where an edge is a connection from one layer to another.
  - `max_in_degree`: The maximum number of incoming edges into a layer.
  - `max_out_degree`: The maximum number of outgoing edges from a layer.
  - `max_mass`, `min_mass`: The maximum and minimum mass of a network, as defined by the OTMANN distance [130].
- **Cartesian product domain:** This domain is used to specify Cartesian products of the above domains. Specifically, a k-tuple x is said to be in a Cartesian product domain $\times_{i=1}^k A_i$ if $x_i \in A_i$ for all $i \in \{1, \ldots, k\}$. Here, each A_i could be of one of the above domain types or a Cartesian product domain. In particular, a Cartesian product domain allows us to combine different domains in order to obtain complex domain that we can optimise over.

[2]We will overload the term domain to mean i) the domain $\mathcal{X}$ over which a function may be optimised or ii) a set of items which could either be a domain $\mathcal{X}$ or a fidelity space $\mathcal{Z}$.

```
{
"name": "domain_1",

"domain":{
  "x0":{
    "name":"x0",
    "type":"int",
    "min":0,
    "max":14
  },
  "x1":{
    "name":"x1",
    "type":"discrete",
    "items":"foo-bar"
  },
  "x2":{
    "name":"x2",
    "type":"discrete_numeric",
    "items":"4-10-23-45-78-87.1-91.8-99-75.7-28.1-3.141593"
  }
 }

}
```

Figure 7.5: An example domain specification in Dragonfly in JSON format. The domain includes an integral variable, a discrete variable, and a discrete numeric variable.

Domain Constraints

In addition to specifying a variable, one might wish to impose constraints on the allowable values of these variables. These constraints can be specified in Dragonfly via an expression or a function that returns a Boolean. For example, if we have a Euclidean variable $a \in [-1, 1]^2$, a constraint $\|a\| \leq 1$ ensures that function evaluations are constrained to inside the unit ball.

Specifying Domains in Dragonfly

The most straightforward way to specify a domain in Dragonfly is via a JSON file. We have demonstrated them via two examples. First, Figure 7.5 presents a simple use case specifying a domain having three variables. The first `x0` is an integer taking values in $[0, 14]$, the second `x1` is a discrete variable which can be one of `foo` or `bar`, and the third `x2` is a discrete numeric variable taking numeric but a discrete set of values specified by the `items` field.

Next, Figure 7.6 describes the optimisation space for a real world electrolyte design task where one wishes to optimise for a variety of properties such as bulk conductivity and viscosity (see Chapter 7.4. Here, we have a library of 5 lithium salts, $LiPF_6$, $LiCoO_2$, $LiMnO_2$, $LiNiO_2$, $LiAlO_2$, (we will refer to the last four as $LiXO_2$), of which we can use a maximum of 3 in each design. The `xxx_present` Boolean variables indicating whether or not each salt is present. We have used a Boolean variable for $LiPF_6$ and a Boolean array of size 4 for the $LiXO_2$ salts. The `max_num_salts` constraint ensures that there are at most 3 salts. The `xxx_mol` variables indicate the molarity of each salt. The experimental apparatus can only add them in increments of 0.05 and hence they are added as discrete numeric variables with different ranges.

```
{
"name": "electrolyte",

"domain" : {
  "LiPF6_present" : {
    "name":"LiPF6_present",
    "type":"boolean"
  },
  "LiXO2_salts_present" : {
    "name":"LiXO2_salts_present",
    "type":"boolean",
    "dim":4
  },
  "LiPF6_mol" : {
    "name":"LiPF6_mol",
    "type":"discrete_numeric",
    "items":"0.0:0.05:3.5"
  },
  "LiXO2_salts_mol" : {
    "name":"LiXO2_salts_mol",
    "type":"discrete_numeric",
    "items":"0.0:0.05:3.0",
    "dim":4
  },
  "solvent_fractions" : {
    "name":"solvent_fractions",
    "type":"float",
    "min":0,
    "max":1,
    "dim":3
  }
 },

"domain_constraints" : {
  "constraint_1" : {
    "name" : "max_num_salts",
    "constraint": "LiPF6_present +  sum(LiXO2_salts_present) <= 4"
  },
  "constraint_2" : {
    "name" : "max_molarity",
    "constraint": "LiPF6_present * LiPF6_mol + sum([a * b for (a, b) in zip(LiXO2
        _salts_present, LiXO2_salts_mol)]) <= 7.8"
  },
  "constraint_3" : {
    "name" : "solvent_fraction_constraint",
    "constraint": "solvent_fraction_constraint.py"
  }
 }

"fidel_space" : {
  "mixing_time": {
    "name":"mixing_time",
    "type":"int",
    "min":60,
    "max":300
  }
 },

"fidel_to_opt" : [300]

}
```

Figure 7.6: An example domain specification in Dragonfly in JSON format. The domain includes several discrete and discrete numeric variables while the fidelity space consists of a Euclidean variable.

The total salt concentration should be less than 7.8 to avoid crash out and this is indicated in the `max_molarity` constraint. The solvent can be composed of 4 ingredients: water, ethyl carbonate, ethyl methyl carbonate, and dimethyl carbonate. We represent this space via the 3 dimensional `solvent_fractions` variable which represent the fractions of the last 3 ingredients. It is then assumed that 1 minus their sum is the fraction of water present. The solvent fractions need to satisfy a variety of constraints which cannot be specified via a string. They are specified via a python function defined in `solvent_fraction_constraint.py`. Finally, the fidelity space consists of one variable, the mixing time, which can be varied between 60 seconds and 300 seconds. The longer the mixing, the more reliable the experimental readings, but they take more time to complete. We wish to optimise for the desired properties when the solution is well mixed, i.e. 300 seconds, and this is specified by the `fidel_to_opt` field.

7.3.2 Kernels

Dragonfly implements the following kernels which enables defining GPs over the above domains.

- **Squared Exponential (SE) Kernel:** Let $x, x' \in \mathbb{R}^d$. The SE kernel, parametrised by a bandwidth vector $h \in \mathbb{R}^d$, and a scale parameter $\sigma \in \mathbb{R}_+$, takes the following form.

  $$\kappa_{\sigma,h}(x, x') = \sigma \exp\left(-\sum_{i=1}^{d} \frac{(x_i - x_i')^2}{2h_i^2}\right)$$

  The SE kernel can be used on Euclidean, integral, discrete Euclidean, and discrete numeric domains.

- **Matérn Kernel:** Let $x, x' \in \mathbb{R}^d$. The Matérn kernel, parametrised by a bandwidth vector $h \in \mathbb{R}^d$, a smoothness parameter $\nu \in \mathbb{R}$, and a scale parameter $\sigma \in \mathbb{R}_+$, takes the following form.

  $$\kappa_{\nu,h}(x, x') = \sigma \frac{2^{1-\nu}}{\Gamma(\nu)} (\sqrt{2\nu} z)^{\nu} B_{\nu}(\sqrt{2\nu} z), \quad z^2 = \sum_{i=1}^{d} \frac{(x_i - x_i')^2}{2h_i^2}$$

  Here B_ν is the modified Bessel function. The SE kernel can be used on Euclidean, integral, discrete Euclidean and discrete numeric domains. Following recommendations in Snoek et al. [232], we use the Matérn kernel with parameter $\nu = 2.5$ as the default kernel for the above domains.

- **Hamming Kernel:** Given x, x' in some product of discrete domains $\times_{i=1}^{k} A_i$, the Hamming distance is a useful measure of similarity between x and x'. We implement the following form of the Hamming kernel, parametrised by a scale parameter $\sigma \in \mathbb{R}_+$ and a weight wector $\alpha \in \mathbb{R}_+^k$, such that $\sum_i \alpha_i = 1$.

  $$\kappa_{\sigma,\alpha}(x, x') = \sigma \sum_{i=1}^{k} \alpha_i \mathbb{1}(x_i = x_i').$$

  The Hamming kernel is used as the default kernel for discrete domains.

- **OTMANN Kernel:** The OTMANN distance and the derivative kernel, as described in [130], is used as the default kernel for neural network domains.
- **Additive Kernels:** As described in Chapter 7.2, we implement a few additive kernels such as the group additive kernel [120] and the ESP kernel [54, 117, 224].
- **Exponential Decay Kernel:** As described in Chapter 7.2, we implement the exponential decay kernel which is useful for multi-fidelity optimisation for settings where we may expect monotonic behaviour across the fidelity space, such as in model selection.

7.3.3 Optimising the Acquisition

To maximise the acquisition φ_t in purely Euclidean spaces with no constraints, we use the dividing rectangles algorithm [112] or the PDOO algorithm [86] by default, depending on the dimensionality. In all other cases, we use an evolutionary algorithm. For this, we begin with an initial pool of randomly chosen points in the domain and evaluate the acquisition at those points. Then we generate a set of N_{mut} mutations of this pool as follows. First, we stochastically select N_{mut} candidates from this set such that those with higher φ_t values are more likely to be selected than those with lower values. Then we apply a mutation operator to each candidate. Finally, we evaluate the acquisition on this N_{mut} mutations, add it to the initial pool, and repeat for the prescribed number of steps.

We apply the following mutation operators to each individual variable of a given candidate point to obtain the final mutation. For Euclidean variables, we sample from a Gaussian centred at the current point. For integral, discrete numeric, and discrete Euclidean variables, we assign probabilities to all points in the domain based on how far they are from the candidate – we then sample using these probabilities. For discrete variables, we choose the same candidate with probability 0.5 or randomly choose another element in the domain with probability 0.5. In addition to this, each time we generate a new candidate we test if they satisfy the constraints specified. If they do not, we reject that sample and keep sampling until all constraints are satisfied. One disadvantage to this rejection sampling procedure is that if the constrains only permit a small subset of the entire domain, it could significantly slow down the optimisation of the acquisition.

7.3.4 Initialisation

Following recommendations in a line of BO work [23, 171], we bootstrap BO with n_{init} evaluations. Assume that the domain to maximise f over is a product of K domains. We generate n_{init} points from each of the individual domains and concatenate them to produce the n_{init} initialisation points. For individual Euclidean domains, these points were chosen via Latin hypercube sampling; for neural network domains, we do so using n_{init} feed forward architectures; for remaining domain types, we choose them via uniformly random sampling. Moreover, as we did in the optimisation of the acquisition, we use rejection sampling to test if the constraints on the

domain are satisfied. In our implementation, by default, n_{init} is set to $5d$ where d is the total dimensionality of the domain but is capped off at 0.075 of the total evaluation budget.

We have summarised the resulting procedure for Bayesian optimisation in Dragonfly in Algorithm 10. q denotes the next query. In usual BO settings, it refers simply to the next point $x \in \mathcal{X}$ in the domain, i.e. $q = (x)$; in multi-fidelity settings it can also refer to a fidelity parameter z, i.e. $q = (z, x)$. $\text{acq}(q), \text{hp}(q)$ refer to the acquisition and the choice of $\{\texttt{ML}, \texttt{PS}\}$ for GP hyperparameter selection. multinomial_sample(ℓ, w) samples an element from a list ℓ from the multinomial distribution $\{w_i / \sum_j w_j\}_{i=1}^{|\ell|}$.

Algorithm 10 BO in Dragonfly with M asynchronous workers, from Kandasamy et al. [133]

Require: n_{init}, n_{cyc}, ℓ^{acq}, ℓ^{hp}.

$\mathcal{D}_0 \leftarrow$ Evaluate f at n_{init} points.
$y_{\max} \leftarrow$ maximum y value in $\mathcal{D}_0$.
$w^{\text{acq}} = \omega_0^{\text{acq}} \mathbf{1}_{|\ell^{\text{acq}}|}$.
$w^{\text{hp}} = \omega_0^{\text{hp}} \mathbf{1}_2$.
for $j = 0, 1, 2 \ldots$ **do**
  Wait for a worker to finish.
  $\mathcal{D}_j \leftarrow \mathcal{D}_{j-1} \cup \{(q, y)\}$ where (q, y) are the worker's previous query and observation.
  if $\mod (t, n_{\text{cyc}}) = 0$, **then** `# updates for GP hyperparameters`
    $\Theta^{\texttt{PS}} \leftarrow$ sample n_{cyc} GP hyperparameter values.
    $\theta^{\texttt{ML}} \leftarrow$ maximise GP marginal likelihood to find best GP hyperparameter values.
  end if
  if $y > y_{\max}$, **then** `# update weights if new max-value was found`
    $w^{\text{acq}}[\text{acq}(q)] = w^{\text{acq}}[\text{acq}(q)] + 1$.
    $w^{\text{hp}}[\text{hp}(q)] = w^{\text{hp}}[\text{hp}(q)] + 1$.
  end if
  $\theta \leftarrow$ multinomail_sample$([\text{pop}(\Theta), \theta^{\texttt{ML}}], w^{\text{hp}})$. `# choose GP hyperparameters`
  $\alpha \leftarrow$ multinomial_sample$(\ell^{\text{acq}}, w^{\text{acq}})$. `# choose acquisition`
  $\mu_{t-1} \leftarrow$ Compute posterior GP mean given $\mathcal{D}_j$ using θ.
  Compute hallucinated posterior $\mathcal{GP}_{t-1} \leftarrow \mathcal{GP}(\mu_{t-1}, \kappa_{t-1}; \mathcal{D}_j \cup \{(x, \mu_{t-1}(x))\}_{x \in F_t}, \theta)$.
  $q_t \leftarrow$ Determine next query for evaluation using acquisition α and GP $\mathcal{GP}_{t-1}$.
  Re-deploy worker with an evaluation at q_t.
end for

7.3.5 Multi-objective Optimisation

This sub-chapter describes work with Biswajit Paria on multi-objective optimisation, who was primiarily responsible for the ideas, algorithms, and analysis. The methods were intergrated into Dragonfly, where we incorporate many techniques, such as the ensemble approaches to GP hyperparameter selection and acquisition, parallelisation, and high dimensions into the multi-objective framework as well.

In many practical applications, we are required to maximise multiple objectives, which typically tend to be competing in nature. For instance, in drug discovery, a chemist wishes to find a drug that has high solubility, high potency, and low toxicity. Drugs that are very potent are also likely to be toxic, so these two objectives are typically competing. Formally, in multi-objective optimisation (MOO) [51], we wish to optimise K functions $f_1, \ldots, f_K$ where $f_k : \mathcal{X} \to \mathbb{R}$ for all $k \in \{1, \ldots, K\}$, where, it is generally not possible to optimise all of them simultaneously. Hence, most MOO algorithms, aim to recover a set of Pareto optimal points (see Chapter 2, [21]). A point $x \in \mathcal{X}$ is said to be Pareto optimal if no other point dominates x on all objectives, i.e. for all $x' \in \mathcal{X}/x$, there exists k such that $f_k(x) > f_k(x')$. Traditional algorithms for MOO (e.g. [55, 93, 140, 196, 198, 245, 278, 280]) aim to recover the entire Pareto front of points in $\mathcal{X}$. In many applications however, it is not necessary, and often wasteful to recover the entire Pareto front, especially when each experiment (evaluation of all functions) can be expensive. For example, in the above drug discovery example, such an algorithm might still aim to find drugs that are Pareto optimal, but are too toxic to administer to a patient. A chemist might be more interested in finding Pareto optimal drugs, but with more acceptable toxicity levels.

In Paria et al. [190], we developed MOORS (Multi-objective Optimisation with Random Scalarisations), a randomised Bayesian multi-objective optimisation approach. When compared to other approaches for MOO, it enjoys the following advantages.

1. *Flexibility*: MOORS allows a practitioner to flexibly explore Pareto optimal regions of interest, and change the region during the course of experimentation if necessary. The approach is flexible enough to recover the entire Pareto front when necessary.
2. *Theoretical guarantees*: Our approach seamlessly lends itself to analysis using a suitable notion of regret, and achieves sub-linear regret bounds.
3. *Computational simplicity*: The computational complexity of our approach scales linearly with the number of objectives K. This is in contrast to several other methods whose complexity scales exponentially.

We now briefly describe MOORS. We begin by defining two notions of scalarisations for the K objectives, linear and Tchebyhev. Let $\lambda \in \mathbb{R}^K_+$. Then, they are respectively defined as,

$$g_{\text{lin}}(\lambda, x) = \sum_{k=1}^{K} \lambda_k f_k, \qquad g_{\text{tch}}(\lambda, x) = \min_{k \in \{1,\ldots,K\}} \frac{1}{\lambda_k} \sigma_k(f_k(x)). \tag{7.1}$$

Here, $\sigma_k : \mathbb{R} \to \mathbb{R}_+$ is a monotone function. Given a desired notion of scalarisation above, at each iteration, MOORS first samples λ from a given distribution p_λ, and then queries at some $x \in \mathcal{X}$ so as to maximised the linearisation for the sampled λ. The specific strategy for choosing x, given λ follows by applying an acquisition such as UCB or TS on the joint posterior of all objectives. The flexibility of the approach comes from our ability to specify p_λ, which can also be changed during the optimisation procedure.

7.4 Experiments

We now present our experimental results. We compare Dragonfly to the following algorithms and packages for zeroth order optimisation. RAND: uniform random search of the domain; EA: evolutionary algorithm; PDOO: Parallel Deterministic Optimistic Optimisation from Grill et al. [86]; HyperOpt: hyperparameter optimisation package from Bergstra et al. [13]; SMAC: Sequential Model based Algorithm Configuration from Hutter et al. [103]; Spearmint: a BO package from Snoek et al. [232]; GPyOpt: a GP based BO package [11]. Of these PDOO is a deterministic non-Bayesian algorithm specifically designed for Euclidean domains. SMAC, Spearmint, and GPyOpt are model based Bayesian optimisation procedures, where SMAC uses random forests as its main model, while Spearmint and GPyOpt use GPs. For EA, we use the same procedure used to optimise the acquisition in Chapter 7.3.

7.4.1 Experiments on Synthetic Benchmarks

We begin with some experiments comparing Dragonfly to the above methods in a suit of standard benchmarks for zeroth order optimisation. All functions are available in the Dragonfly repository at `dragonfly.github.io`.

Euclidean Domains

Our first set of experiments are on a series of synthetic benchmarks in Euclidean domains. We use the Branin ($d = 2$), Hartmann3 ($d = 3$), Park1 ($d = 4$), Park2 ($d = 4$), Hartmann6 ($d = 6$), and Borehole ($d = 8$) benchmarks, and compare all methods on their performance over 200 evaluations. The results are given in Figure 7.7, where we plot the simple regret (1.1) against the number of evaluations (lower is better). We do not compare EA in these experiments to avoid clutter in the figures, since its performance was typically worse than all other BO methods.

Next, we construct high dimensional versions of the above benchmarks and compare all methods in Figure 7.8. The high dimensional forms were obtained via an additive model $f(x) = f'(x^{(1)}) + f'(x^{(2)}) \dots$ where f' is a lower dimensional function and the $x^{(i)}$'s are coordinates forming a low dimensional subspace. For example, in the Hartmann3x6 problem, we have an 18 dimensional function obtained by considering six Hartmann3 functions along coordinate groups $\{1, 2, 3\}, \{4, 5, 6\}, \dots, \{16, 17, 18\}$. Spearmint is not shown on most of the problems since it was too slow in high dimensional settings. SMAC's initialisation procedure threw an error in dimensions larger than 40 and is hence not shown in the corresponding experiments.

Next, we use the same test functions as above, but add noise to the function evaluations. As before, we evaluate all methods on the simple regret $\operatorname{argmax} f - \max_{x_t} f(x_t)$, which is now more challenging since the true function value is never observed. For the Branin, Hartmann3, Harmann6, Park1, and Hartmann6 functions we add Gaussian noise with standard deviation 0.1, for the Park2 function, we add Gaussian noise with standard deviation 0.5, and for the Borehole

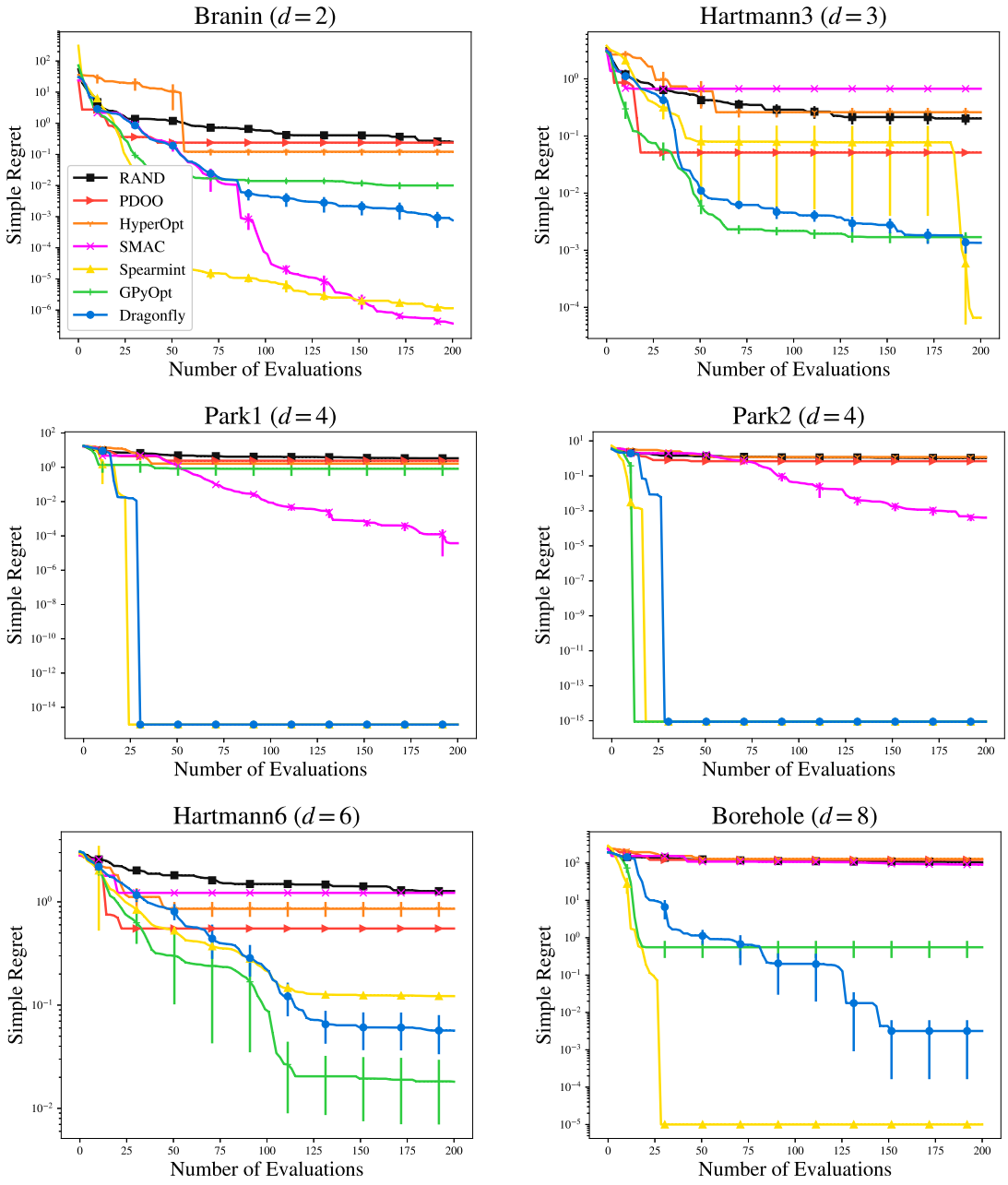


Figure 7.7: Comparison of Dragonfly with other algorithms and BO packages on functions with *noiseless* evaluations defined on Euclidean domains. We plot the simple regret (1.1) so lower is better. The title states the name of the function, and its dimensionality. All curves were produced by averaging over 20 independent runs. Error bars indicate one standard error. The legend for all curves is available in the first figure.

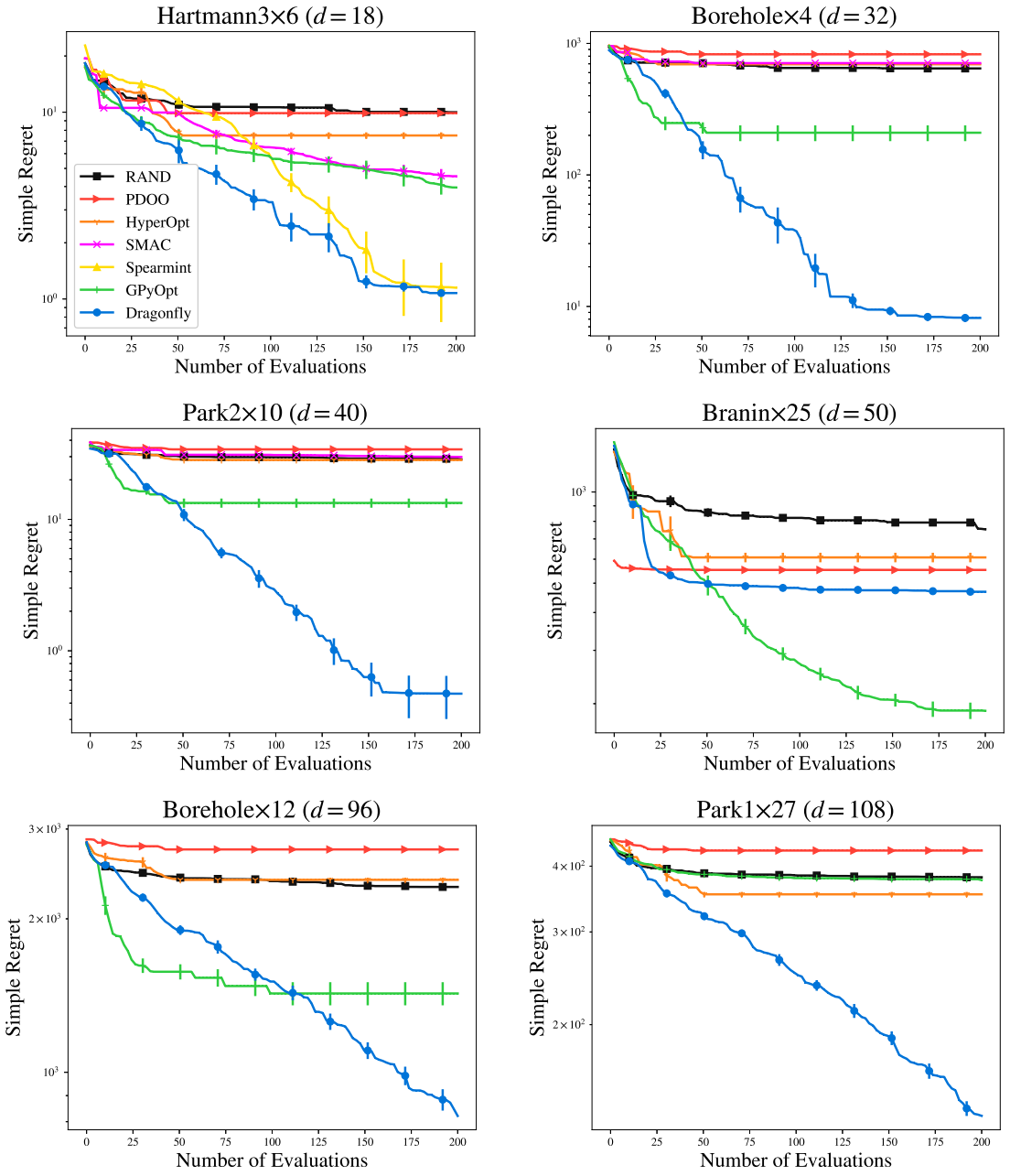


Figure 7.8: Comparison of Dragonfly with other algorithms and BO packages on functions with *noiseless* evaluations defined on high dimensional Euclidean domains. We plot the simple regret (1.1) so lower is better. SMAC's initialisation procedure did not work in dimensions larger than 40 so it is not shown in the respective figures. Spearmint is not shown on all figures since it was too slow to run on high dimensional problems. See caption under Figure 7.7 for more details.

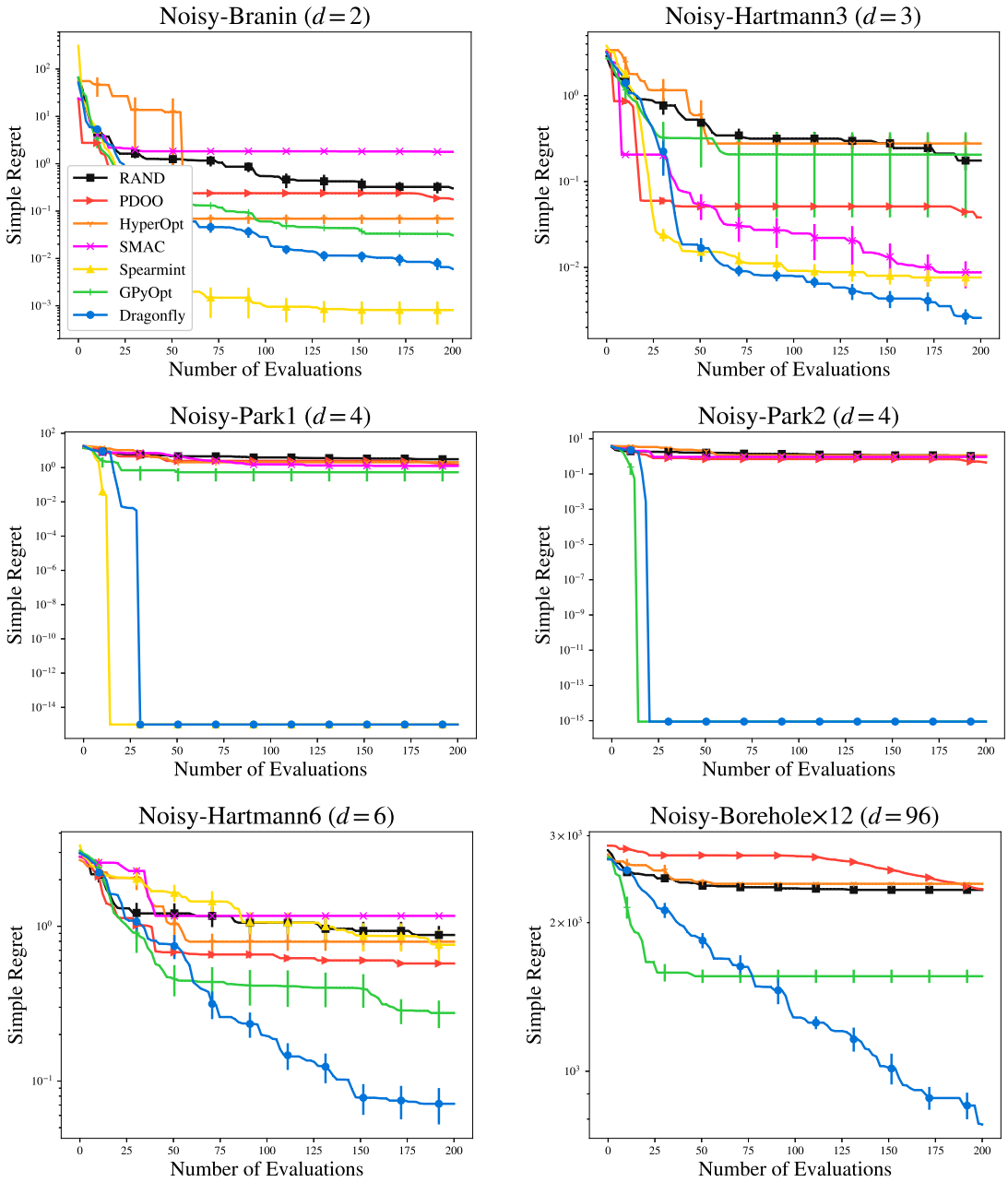


Figure 7.9: Comparison of Dragonfly with other algorithms and BO packages on functions with *noisy* evaluations defined on Euclidean domains. We plot the simple regret (1.1) so lower is better. The title states the name of the function, and its dimensionality. See caption under Figure 7.7 for more details.

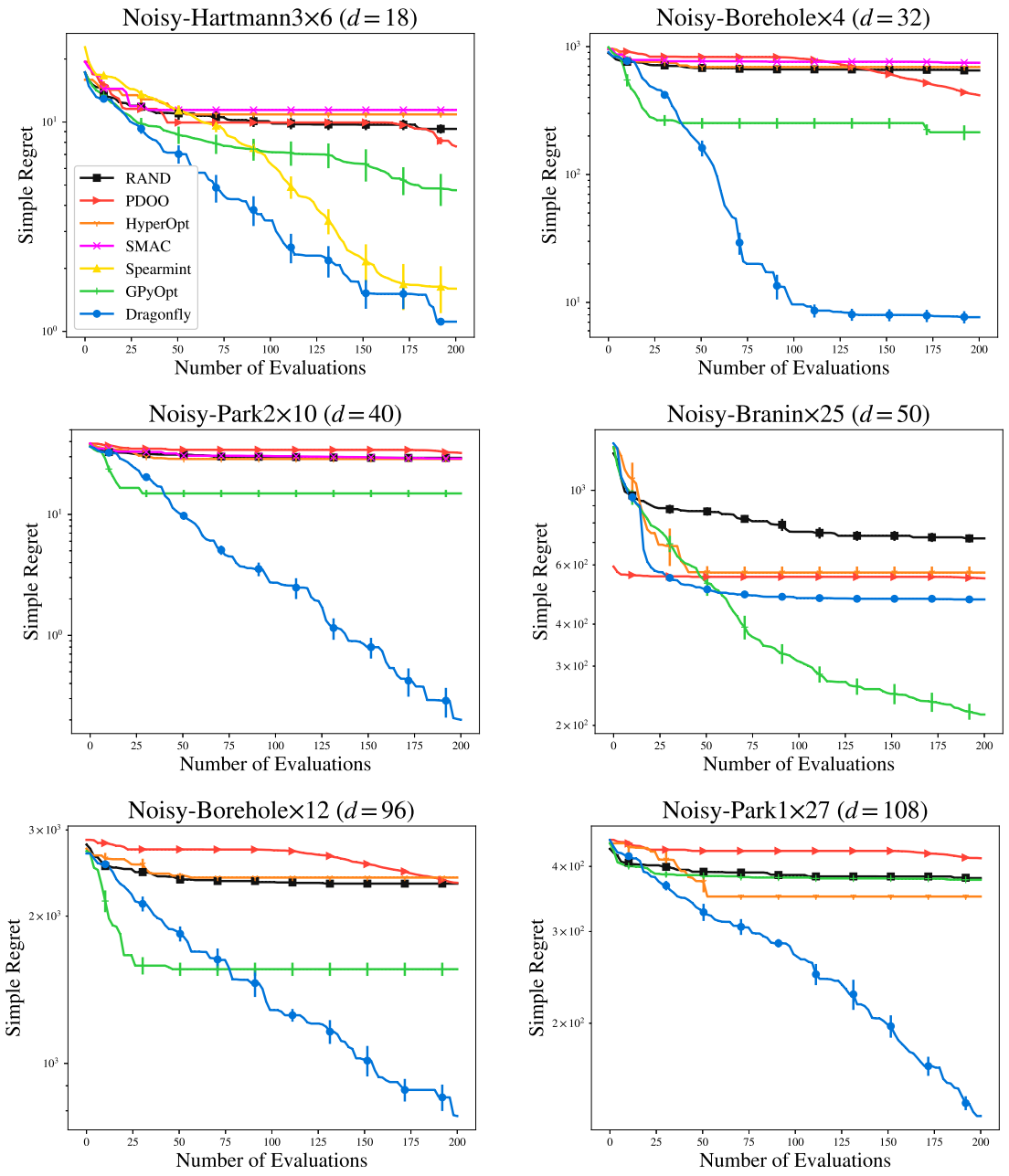


Figure 7.10: Comparison of Dragonfly with other algorithms and BO packages on functions with *noisy* evaluations defined on Euclidean domains. We plot the simple regret (1.1) so lower is better. SMAC's initialisation procedure did not work in dimensions larger than 40 so it is not shown in the respective figures. Spearmint is not shown on all figures since it was too slow to run on high dimensional problems. See caption under Figure 7.7 for more details.

function, we add Gaussian noise with standard deviation 5; the noise scales were determined based on the range of the function. The results are given in Figure 7.9 for the low dimensional benchmarks and in Figure 7.10 for the high dimensional benchmarks.

Take-aways: Spearmint and Dragonfly perform consistently well across all problems. Spearmint perhaps performs slightly better on the low dimensional tasks, but is prohibitively expensive beyond 20 dimensions. Dragonfly outperforms other methods significantly in the higher dimensional problems, except for the 50 dimensional Branin$\times 25$ function where GPyOpt does better. Dragonfly is also fairly robust to noise in the evaluations unlike some other methods whose performances appear to deteriorate under noise.

Non-Euclidean Domains

Next, we compare Dragonfly to the above baselines on non-Euclidean domains. For this, we modify the above Euclidean benchmarks so that they can take non-Euclidean arguments. Specifically, we use modified versions of the Borehole, Hartmann6, Park1 and Park2 functions. Additionally, we also construct a synthetic function defined on neural network domains. The results are given in Figure 7.13. Since the true maximum of these functions are not known, we simply plot the maximum value found against the number of evaluations (higher is better). We also construct variations of these function which can take a fidelity argument. Hence, a strategy may use these approximations to speed up the optimisation process. The x-axis in all cases refers to the expended capital, which was chosen so that a single fidelity algorithm would perform exactly 200 evaluations. We compare a multi-fidelity version of Dragonfly, which uses the BOCA strategy [127], to choose the fidelities and points for evaluation.

Next, as before, we consider noisy versions of the above functions. As before, we add Gaussian noise whose scale is determined by the range of the function. We wish to remind the reader that these functions and the approximations are defined in the Dragonfly repository.

Take-aways: Dragonfly performs best on two out of the five cases in which we compare all the methods, but more importantly is able to do consistently well across all problems. GPyOpt and Dragonfly perform very well on some problems, but also perform poorly on others. In addition, Dragonfly is also able to outperform the (fairly weak) EA and RAND baselines on the synthetic neural architecture search task. It is interesting to note that the improvements due to multi-fidelity optimisation are modest in some cases, and in the case of the Park1_3 function, it performs worse than the single fidelity version. We believe this is due to two factors. First, the multi-fidelity methods spends an initial fraction of its capital at the lower fidelities, and the simple regret is ∞ until it queries the highest fidelity. Second, there is an additional statistical difficulty in estimating, what is now a more complicated GP across the domain and fidelity space. The combination of these factors undermines the advantages the cheaper approximations may provide. However, the multi-fidelity version is able to do better in most cases. For example, on the neural architecture search task where we have a complex domain, the multi-fidelity version is able to do significantly better.

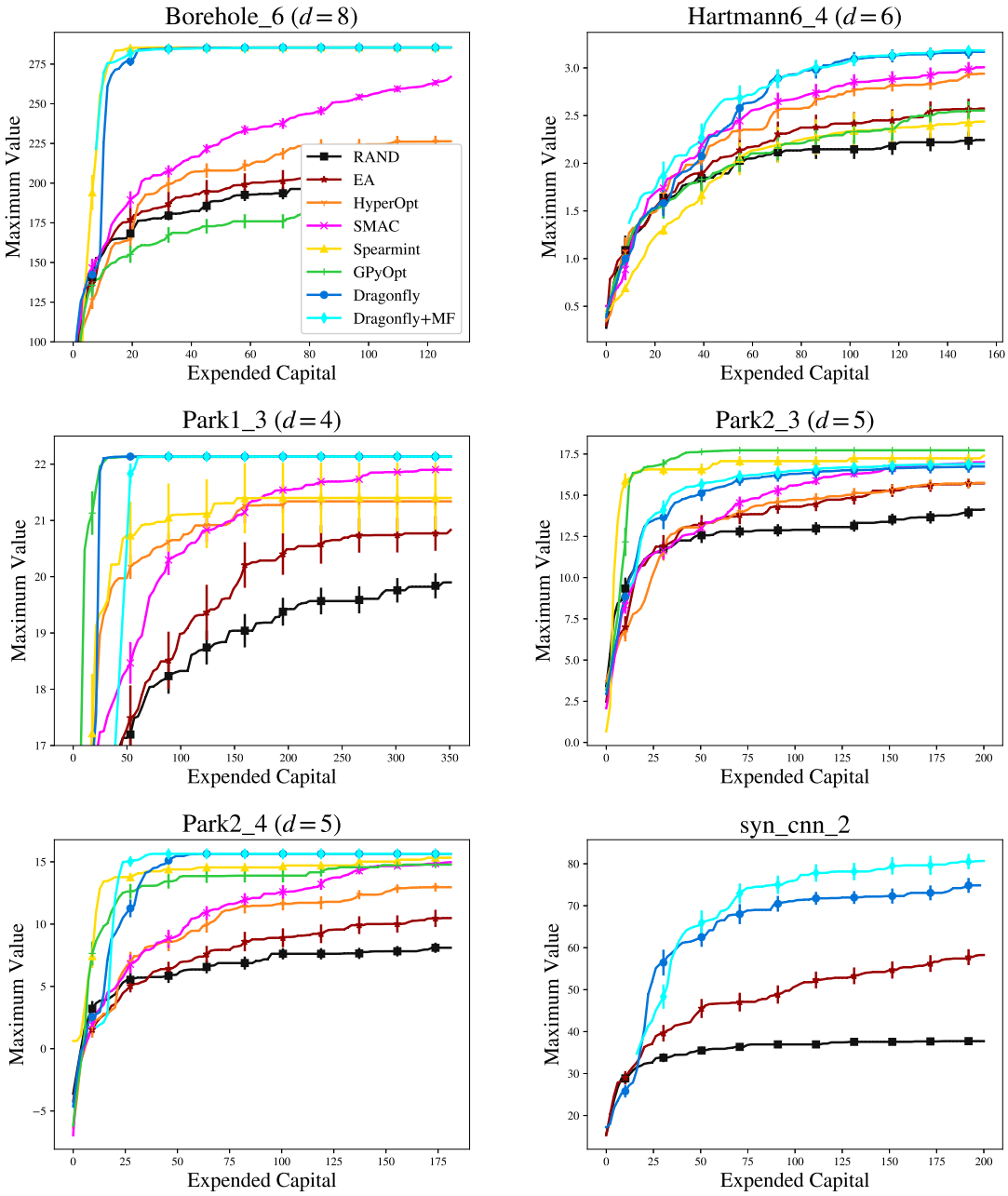


Figure 7.11: Comparison of Dragonfly with other algorithms and BO packages on synthetic functions with *noiseless* evaluations defined on non-Euclidean domains. We plot the maximum value, so higher is better. The x-axis shows the expended capital, which was chosen so that a single fidelity method would perform exactly 200 evaluations. The title states the name of the function, and its dimensionality (number of variables). We do not state the dimensionality for the synthetic CNN function since the dimensionality of a space of CNN architectures is not defined. See `dragonfly.github.io` for a description of these functions and the approximations for the multi-fidelity curves. All curves were produced by averaging over 20 independent runs. Error bars indicate one standard error. The legend for all curves is available in the first figure. We do not compare Spearmint, HyperOpt, SMAC, and GPyOpt on the synthetic CNN functions since they do not support optimising over neural architectures.

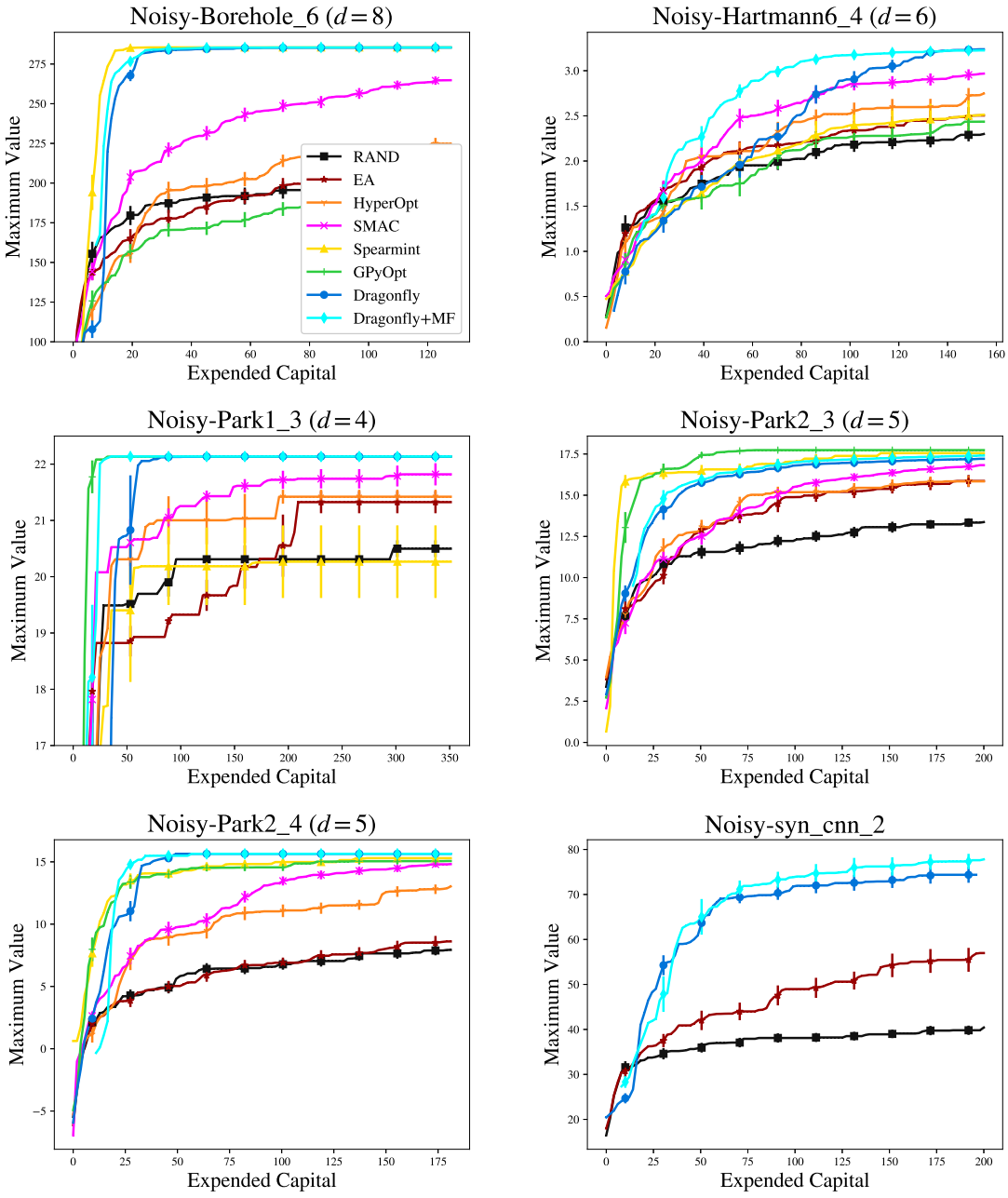


Figure 7.12: Comparison of Dragonfly with other algorithms and BO packages on synthetic functions with *noisy* evaluations defined on non-Euclidean domains. We plot the maximum value, so higher is better. See caption under Figure 7.13 for more information on the figures.

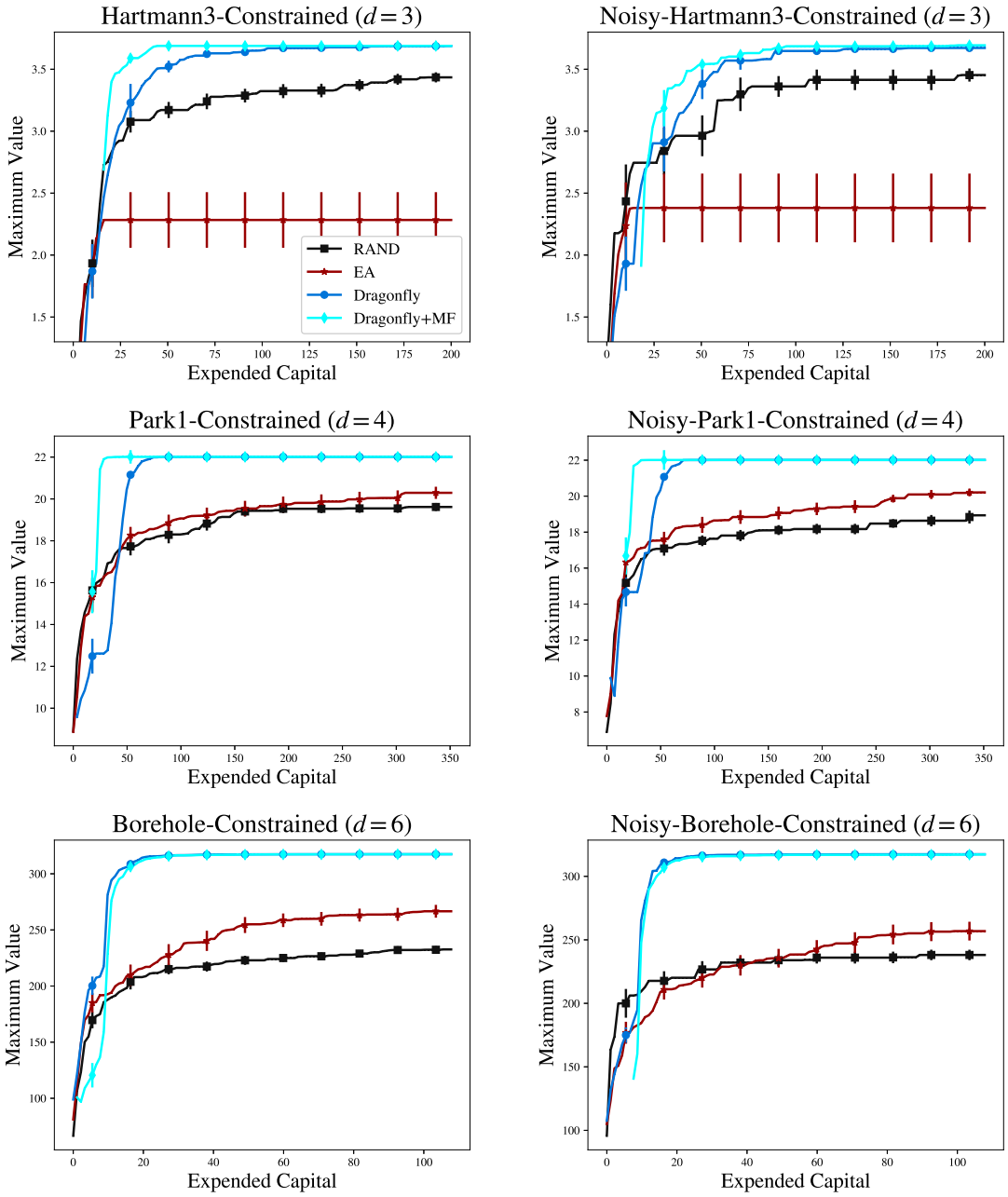


Figure 7.13: Comparison of Dragonfly with RAND and EA on synthetic functions with constraints on the domain. We plot the maximum value, so higher is better. We perform experiments on three different synthetic functions where the left column is when the function evaluations are noiseless, and the right column is when noise is added to the evaluations. The title states the name of the function, and its dimensionality (number of variables). See `github.com/dragonfly/dragonfly/tree/master/demos_synthetic` for a description of these functions and the approximations for the multi-fidelity curves. All curves were produced by averaging over 20 independent runs. Error bars indicate one standard error.

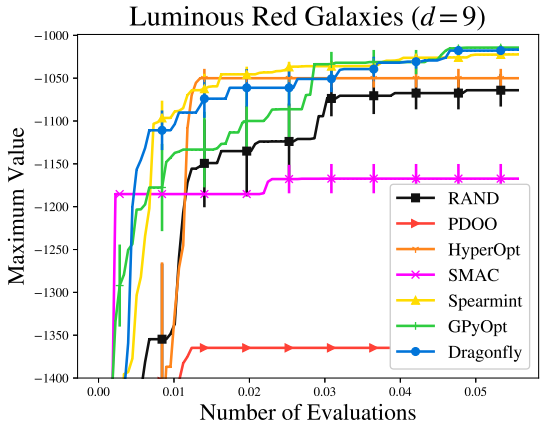


Figure 7.14: Results on the maximum likelihood estimation problem on the luminous red galaxies dataset [244]. The x-axis is the number of evaluations and the y-axis is the highest likelihood found so far (higher is better). All curves were produced by averaging over 10 independent runs. Error bars indicate one standard error.

Domains with Constraints

For the final set of experiments, we consider three optimisation tasks where we impose additional constraints on the domain variables. Specifically, we consider versions of the Hartmann3, Park1 and Borehole functions. As an example, the Hartmann3 function is usually defined on the domain $[0,1]^3$; however, we consider a constrained domain $\mathcal{X} = \left\{x \in [0,1]^3 \,;\, x_1^2 + x_2^2 \leq 1/2\right\}$. Descriptions of the other functions and domains are available in the Dragonfly repository. In Figure 7.13, we compare Dragonfly to RAND and EA . We do not compare to other methods and packages, since, to our knowledge, they cannot handle arbitrary constraints of the above form. Figure 7.13 also presents results when function evaluations are noisy.

7.4.2 Experiments on Astrophysical Maximum Likelihood Problems

In this section, we consider two maximum likelihood problems in computational Astrophysics.

Luminous Red Galaxies: Here we used data on Luminous Red Galaxies (LRGs) for maximum likelihood inference on 9 cosmological parameters, spatial curvature $\Omega_k \in (-0.01, 0.03)$, dark energy fraction $\Omega_\Lambda \in (0.7, 0.8)$, cold dark matter density $\omega_c \in (0.1, 0.105)$, baryonic density $\omega_B \in (0.02, 0.3)$, scalar spectral index $n_s \in (0.5, 1.7)$, scalar fluctuation amplitude $A_s \in (0.65, 0.75)$, running of spectral index $\alpha \in (-0.02, 0.01)$, galaxy bias $b \in (1.0, 2.0)$, and a nonlinear correction factor $Q_{\text{nl}} \in (30, 31)$. The likelihood is computed via the galaxy power spectrum which measures the distribution of temperature fluctuations as a function of scale. Software and data were taken from Kandasamy et al. [121], Tegmark et al [244]. Each evaluation here is relatively cheap, and hence we compare all methods on the number of evaluations in Figure 7.14. We do not compare a multi-fidelity version since cheap approximations were not available for this problem. Spearmint, GPyOpt, and Dragonfly do well on this task.

Type Ia Supernova: We use data on TypeIa supernova for maximum likelihood inference on 3 cosmological parameters, the Hubble constant $H_0 \in (60, 80)$, the dark matter fraction $\Omega_M \in (0, 1)$ and dark energy fraction $\Omega_\Lambda \in (0, 1)$. We use data from Davis et al [50] which has data on 192 supernovae, and the likelihood is computed using the method described in [225]. This

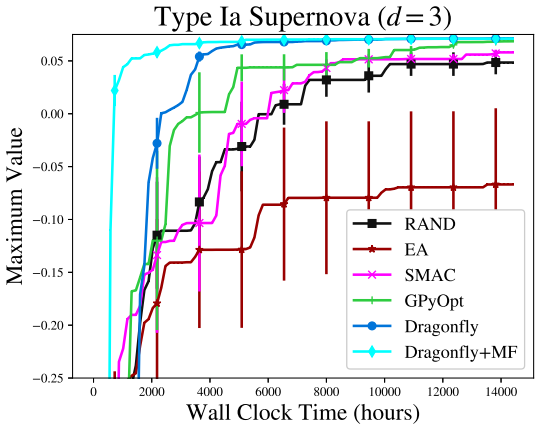


Figure 7.15: Results on the maximum likelihood estimation problem on the Type Ia supernova dataset [50]. The x-axis is time and the y-axis is the highest likelihood found so far (higher is better). We do not compare PDOO, HyperOpt and Spearmint because they do not provide an API for optimising over time. All curves were produced by averaging over 10 independent runs. Error bars indicate one standard error.

requires a one dimensional numerical integration for each point in the dataset. We construct a $p = 2$ dimensional multi-fidelity problem where we can choose between data set size $N \in [50, 192]$ and perform the integration on grids of size $G \in [10^2, 10^6]$ via the trapezoidal rule. As the cost function for fidelity selection, we used $\lambda(N, G) = NG$ as the computation time is linear in both parameters. Our goal is to maximise the average log likelihood at $z_\bullet = [192, 10^6]$. Each method was given a budget of 4 hours on a 3.3 GHz Intel Xeon processor with 512GB memory. The results are given in Figure 7.15 where we plot the maximum average log likelihood (higher is better) against wall clock time. The plot includes the time taken by each method to determine the next point for evaluation. We do not compare Spearmint and HyperOpt since they do not provide an API for optimisation on a time budget.

7.4.3 Experiments on Model Selection Problems

We next provide experiments on three regression problems. In all cases, we plot the regression error (lower is better) against wall clock time. Moreover, we set up a one dimensional fidelity space where a multi-fidelity algorithm may choose to use a subset of the dataset to approximate the performance when training with the entire training set. We do not compare Spearmint and HyperOpt since they do not provide an API for optimisation on a time budget.

SALSA, Energy Appliances: We tune 30 parameters of the SALSA regression method [117] on the energy appliances dataset [30]. These parameters are the additive order (integer), the type of kernel (discrete), the kernel scale (float), and the bandwidths for the 27 dimensions (float). The goal was to train the method on a training set of 8000 points and find the configuration with the lowest error on a validation set of 2000 points. A multi-fidelity algorithm could approximate the training procedure using a subset of the training set of size $z \in (2000, 8000)$. As the cost function, we use $\lambda(z) = z^3$, since training time is cubic in the training set size. Each method was given a budget of 8 hours on a 2.6 GHz Intel Xeon processor with 384GB memory. The results are presented in Figure 7.16. In this example, SMAC does very well because its initial value luckily landed at a good configuration.

Random forest regression, News popularity: We tune random forest regression (RFR) on the

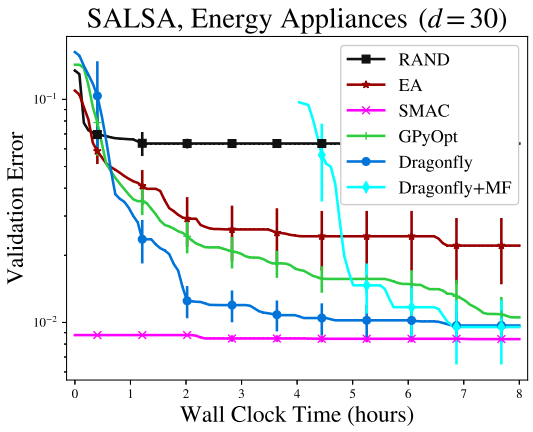


Figure 7.16: Results on the SALSA model selection problem comparing Dragonfly to other packages.The x-axis is wall clock time and the y-axis is the regression error (lower is better). We do not compare HyperOpt and Spearmint because they do not provide an API for optimising over time. All curves were produced by averaging over 10 independent runs. Error bars indicate one standard error.

news popularity dataset [57]. We tune 6 parameters available in the Scikit-Learn implementation of RFR which include the number of estimators (integer), criterion (discrete), tree depth (integer), min_samples_split (float), min_samples_leaf, (float), and max_features (float) parameters. The goal was to train the method on a training set of 20000 points and find the configuration with the lowest error on a validation set of 10000 points. A multi-fidelity algorithm could approximate the training procedure using a subset of the training set of size $z \in (5000, 20000)$. As the cost function, we use $\lambda(z) = z$, since training time is linear in the training set size. Each method was given a budget of 6 hours on a 3.3 GHz Intel Xeon processor with 512GB memory. The results are presented in the left column of Figure 7.17.

Gradient Boosted Regression, Naval Propulsion: We tune 7 parameters of gradient boosted regression (GBR) on the naval propulsion dataset [45]. The parameters in the Scikit-Learn implementation of GBR include the number of estimators (integer), criterion (discrete), learning rate (float), tree depth (integer), subsample fraction (float), min_samples_split (float), and min_samples_leaf (float). We used a training set of 9000 points, and a validation set of 2000 points. A multi-fidelity algorithm could approximate the training procedure using a subset of the training set of size $z \in (2000, 9000)$. As the cost function, we use $\lambda(z) = z$. Each method was given a budget of 3 hours on a 2.6 GHz Intel Xeon processor with 384GB memory. The results are presented in the right column of Figure 7.17.

Table 7.1 compares the final error achieved by all methods on the above three datasets at the end of the respective optimisation budgets. In addition to the methods in Figure 7.17, we also show the results for BO (with Dragonfly) at the lowest fidelity, random search at the lowest fidelity and Hyperband [154], which is a multi-fidelity method which uses random search and successive halving. For example, for BO and random search at the lowest fidelity on the RFR problem, we performed the same procedure as RAND and Dragonfly, but only using 5000 points at each evaluation. Interestingly, we see that for the GBR experiment, BO using only a fraction of the training data, outperforms BO using the entire training set. We speculate that this is because, in this problem, one can get good predictions even with 2000 points, and the lower fidelity versions are able to do better since they are able to perform more evaluations within the specified time budget than the versions which query the higher fidelity.

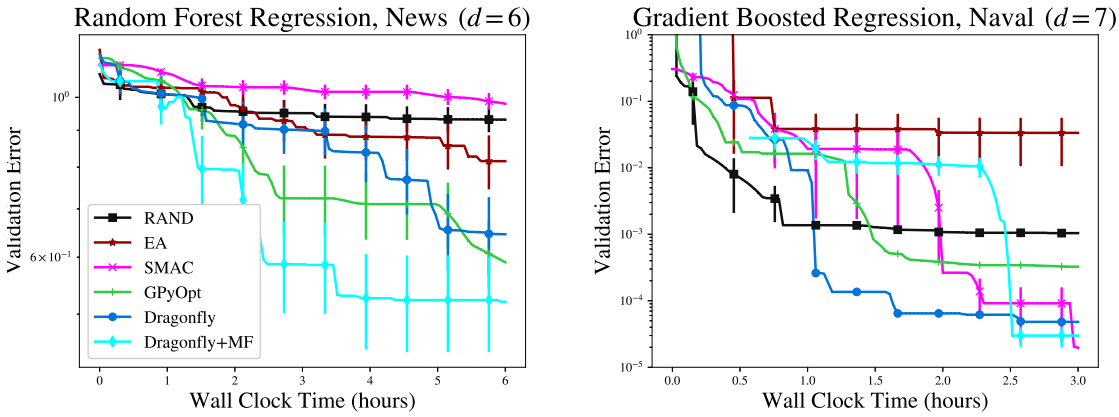


Figure 7.17: Results on the random forest regression and gradient boosted regression problems comparing Dragonfly to other packages.The title states the method, the data set used and the dimensionality of the problem. The x-axis is wall clock time and the y-axis is the regression error (lower is better). We do not compare HyperOpt and Spearmint because they do not provide an API for optimising over time. All curves were produced by averaging over 10 independent runs. Error bars indicate one standard error.

Dataset	Dragonfly	Dragonfly (lowest)	Dragonfly +MF	GPyOpt	SMAC	RAND	RAND (lowest)	Hyperband
RFR (News)	0.6456 ±0.08044	.6631 ±0.0110	**0.5203 ±0.0805**	0.5904 ±0.0635	0.9802 ±0.0291	0.9314 ±0.0371	0.8501 ±0.0563	0.6812 ±0.0412
GBR (Naval)	5.82e−5 ±1.52e−5	**1.14e−5 ±7.3e−7**	3.00e−5 ±1.00e−5	3.26e−4 ±1.46e−5	1.97e−5 ±5.27e−6	1.04e−3 ±1.10e−5	1.31e−5 ±2.34e−6	**1.13e−5 ±6.4e−7**
SALSA (Energy)	0.0097 ±0.0015	0.9950 ±0.0033	0.0095 ±0.0031	0.0105 ±0.0022	**0.0084 ±0.0003**	0.0634 ±0.0019	0.9974 ±0.0032	0.3217 ±0.1032

Table 7.1: Final least squared errors in the regression problems of Chapter 7.4.3. In addition to the methods in Figure 7.17, we compare Dragonfly and random search at the lowest fidelity, as well as Hyperband.

Neural Architecture Search

In our final set of experiments, we compare Dragonfly to NASBOT on some neural architecture search problems. Here, NASBOT is the vanilla implementation in Chapter 6, while the Dragonfly version uses multi-fidelity approximations and also tunes for the learning rate. Each function evaluation, trains an architecture $x \in \mathcal{X}$ with stochastic gradient descent (SGD) with a fixed batch size of 256. We used the number of batch iterations in a one dimensional fidelity space, i.e. $\mathcal{Z} = [4000, 20000]$ for Dragonfly while NASBOT always queried with $z_\bullet = 20,000$ iterations. As the cost function, we use $\lambda(z) = z$, since training time is linear in the number of iterations.

We test both methods in an asynchronously parallel set up of two GeForce GTX 970 (4GB) GPU workers with a computational budget of 8 hours. On each run, both methods were initialised with 10 feed-forward architectures, all evaluated at the highest fidelity; these initial networks are illustrated in Figure 17 of Kandasamy et al. [130]. Additionally, we also impose the following constraints on the space of architectures.

- Maximum number of layers: 60

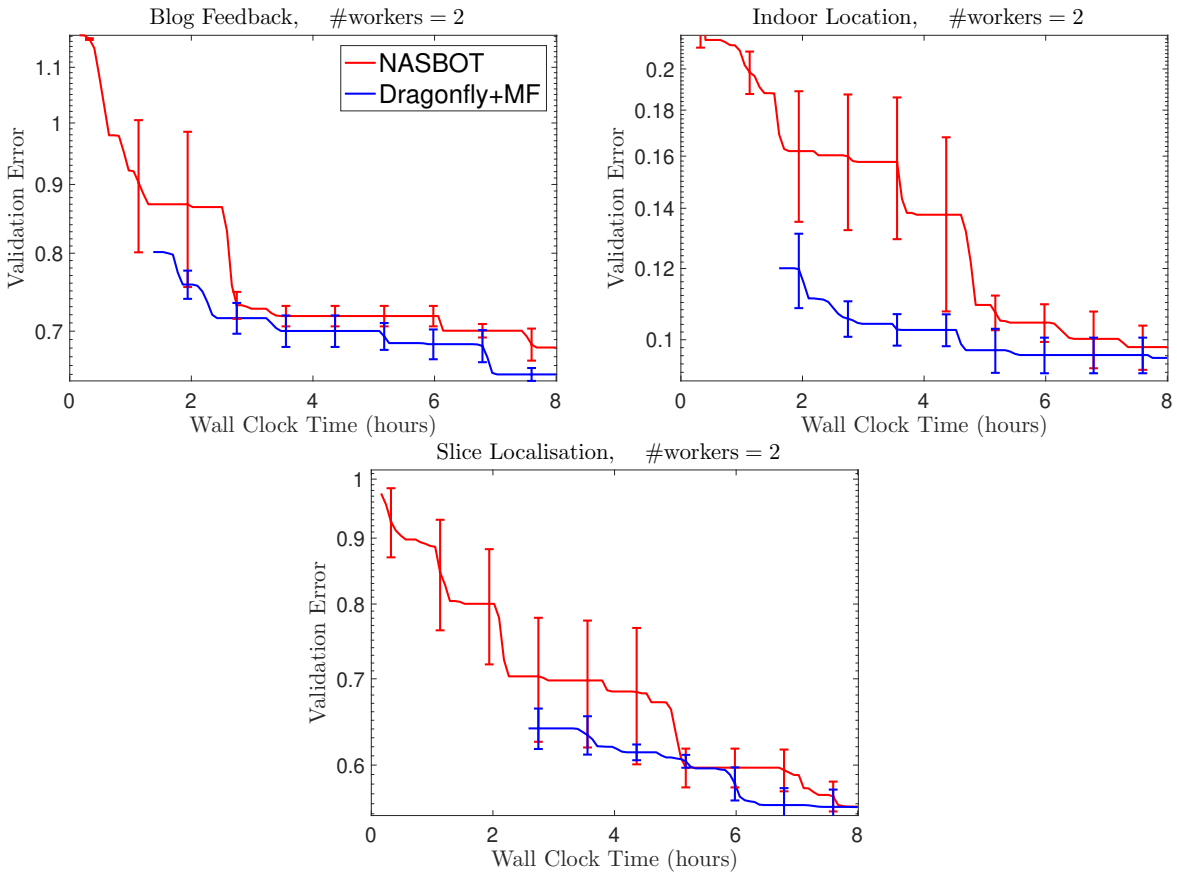


Figure 7.18: Results on the neural architecture search experiments. In all figures, the x-axis is time. The y axis is the mean squared validation error (lower is better). The title of each figure states the dataset. In all cases, we used a parallel set up of two asynchronous workers, where each worker is a single GPU training a single model. We used a one dimensional fidelity space where we chose the number of batch iterations from 4000 to 20,000 ($z_\bullet = 20,000$). All figures were averaged over 5 independent runs. Error bars indicate one standard error.

- Maximum mass: 10^8
- Maximum in/out degree: 5
- Maximum number of edges: 200
- Maximum number of units per layer: 1024
- Minimum number of units per layer: 8

We present the results of our experiments on the blog feedback [29], indoor location [247], and slice localisation [84], datasets in Figure 7.18. Dragonfly outperforms NASBOT primarily because it is able to use cheaper evaluations to approximate fully trained models, and additionally since it tunes the learning rate and uses more robust techniques for selecting the acquisition and GP hyperparameters.

7.4.4 Dragonfly in Use Elsewhere

We conclude this section by stating some examples where Dragonfly has been used for various optimisation tasks in different applications. They are presented separately as I was not primarily responsible for these results.

Electrolyte Design

Chemists designing electrolytes for lithium-ion batteries, are interested in finding a suitable design for the use case. This is typically characterised by requiring that the bulk conductivity of the electrolyte be maximised, while ensuring that the voltage stability and viscosity are at acceptable levels. When designing this electrolyte, the chemist can choose to include a few of several salts and solvents from a large library. Once these salts and solvents are chosen, they also need to choose the molarities of each salt and the amount of each solvent included in the design. Moreover, there are other additional design constraints—for instance, the total salt concentration should not exceed a pre-specified level in order to avoid saturation.

Figure 7.19 shows an automated experimental apparatus for measuring various electrolyte properties, used at the Scott Institute for Energy at Carnegie Mellon University. This apparatus was interfaced with Dragonfly in order to measure various electrolyte properties, and design new electrolytes in a feedback driven manner. The design space was similar to the example in Figure 7.6. It included various Na and Li salts such as $\mathrm{LiPF_6}$, $\mathrm{Na_2SO_4}$, and $\mathrm{Na_2NO_3}$, and solvents such as Ethyl Methyl Carbonate, Ethylene Carbonate, DiMethyl Carbonate, and water.

Optimising Computing Infrastructure

Computing infrastructure has become increasingly complex over the last few years, with several hundreds or thousands of configuration parameters. For example, modern real time streaming systems can have more than hundred knobs that need to be configured. Practitioners are interested in optimising for several criteria such as latency, CPU/memory footprints, and cost when managing complex infrastructure systems, depending on the application at hand. These systems need to be optimised via trial and error, and moreover, since each experiment can be quite expensive, we need to find optimal parameters in as few experiments as possible.

In Figure 7.20, we compare Dragonfly, RAND, and EA for optimising the latency of a real time streaming system involving Spark, Kafka, and Redis database components. All methods were deployed in a parallel set up of 10 workers, where each worker measured the latency on the Yahoo streaming benchmark [1]. Each evaluation was executed on a `t2.xlarge` instance on AWS and took approximately 10 minutes. The Figure shows the results for two fixed througput values, and we see that Bayesian optimisation using Dragonfly is able to outperform other methods. This work, led by Hai Pham, is part of a larger project which uses bandit methods to optimise computing infrastructure systems.

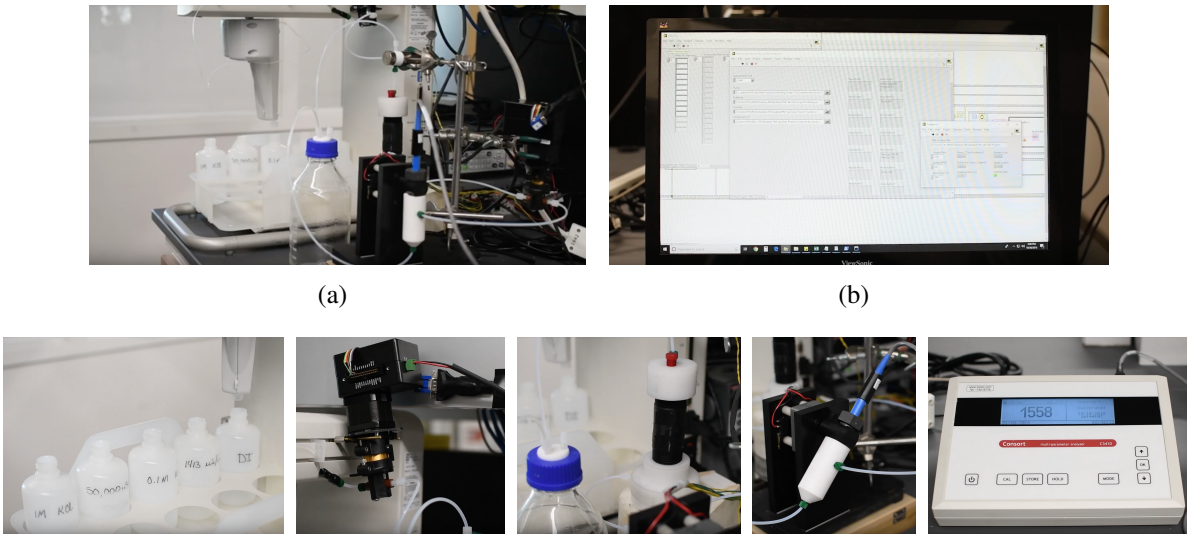


Figure 7.19: Visuals of the apparatus used in the electrolyte design task. (a): A picture of the experimental set up. (b): Dragonfly is interfaced with the LabVIEW software to communicate experimental configurations and measurements. (c): Ingredients are chosen according to specifications from Dragonfly. (d)- (f): Various stages of taking a measurement. (g): The conductivity (and other measurements) are measured and fed back to Dragonfly. The entire video can be viewed at `https://youtu.be/XUPWv1J_DX4`. These experiments were conducted by researchers at the Scott Institutute for Energy at CMU. Willie Neiswanger put together the video.

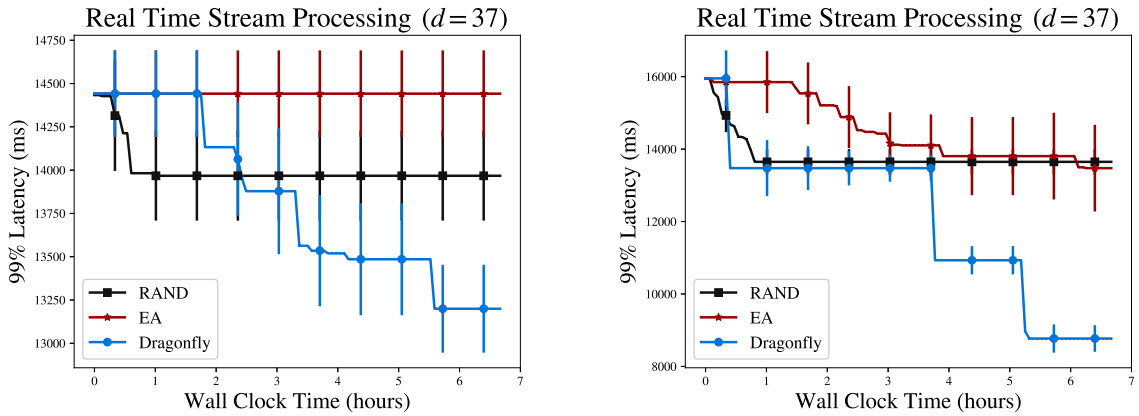


Figure 7.20: Experimental results comparing Dragonfly to EA and RAND for optimising a real time streaming system where data was streamed through a Spark/Kafka pipeline and written to a Redis database. The left figure shows when the best latency values found by each method (lower is better) when the throughput was fixed at 20K while the right figure shows the same when the throughput was fixed to 10K. Hai Pham was primarily responsible for these experiments.

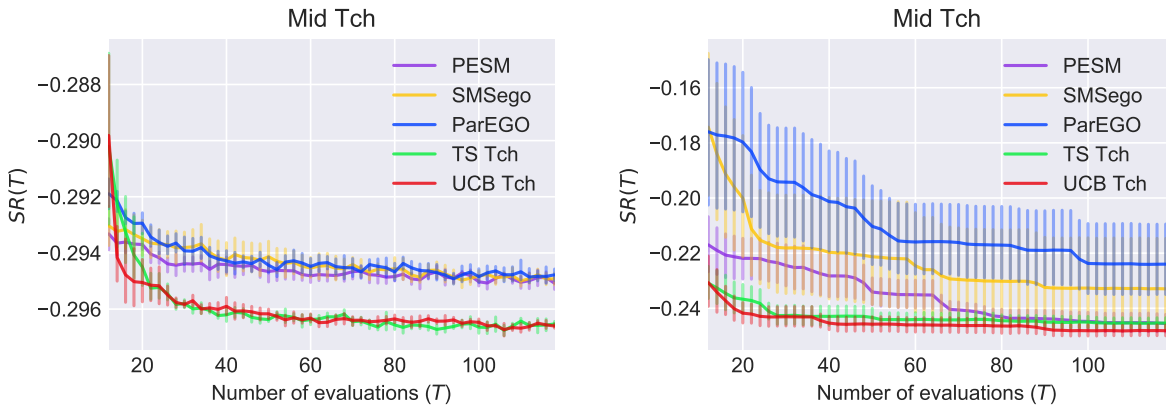


Figure 7.21: Comparison of multi-objective optimisation methods implemented in Dragonfly (UCB Tch and TS Tch) with other methods for MOO on the Locality Sensitive Hashing and Viola and Jones problems described in Chapter 7.4. The y-axis is the Tsebychev simple regret for MOO (see (7.1) and Paria et al. [190]) and the x-axis is the number of evaluations. Biswajit Paria was responsible for these experiments and these results are taken directly from Paria et al. [190].

Multi-objective Optimisation

As stated in Chapter 7.3.5, Dragonfly implements Bayesian optimisation methods for multi-objective optimisation. In this section, we compare the methods for the Tsebychev criterion (7.1) using upper confidence bounds (UCB Tch) and Thompson sampling (TS Tch) implemented in Dragonfly against the following methods for multi-objective optimisation: PESM [93], SMSego [198], and ParEGO [140].

Locality Sensitive Hashing: Our first experiment is on Locality Sensitive Hashing (LSH) [8] which is a randomised algorithm for computing k-nearest neighbours. We tune a number of parameters in LSH, including the number of hash tables, the number of hash bits, and the number of probes to make for each query. We wish to optimise three different performance criteria for LSH: the average query time, precision and memory usage, which can be competing. For instance, while increasing the number of hash tables results in smaller query times, it leads to an increase in the memory footprint. Similarly, while increasing the number of probes leads to a higher precision, it increases the query time. We run LSH on Glove word embeddings [192], using the Glove Wikipedia-Gigaword dataset trained on 6B tokens with a vocabulary size of 400K and 300-d embeddings. Figure 7.21 compares UCB Tch and TS Tch to the other methods listed above for optimising the three criteria listed above.

Viola Jones: The Viola Jones algorithm [254] is a fast stagewise face detection algorithm. At each stage a simple feature detector is run over the image producing a real value. If the value is smaller than a threshold the algorithm exits with a negative decision (i.e. "no face"), otherwise the image is processed by the next stage and so on. The Viola Jones pipeline has 27 tunable

thresholds. We treat these thresholds as inputs and optimize for sensitivity, specificity, and the time per query. The results are presented in Figure 7.21.

7.5 User Interface and APIs

We conclude this chapter with some details on installing and using Dragonfly. A full documentation and tutorials to get started are available at `dragonfly.github.io`.

Installation: The most straightforward way to install Dragonfly is via `pip`.

```
$ pip install numpy
$ pip install dragonfly-opt
```

Dragonfly can also be installed via source, after installing the required dependencies.

```
$ pip install numpy scipy future six
$ git clone https://github.com/dragonfly/dragonfly
$ cd dragonfly
$ python setup.py install
```

Command line usage: To use Dragonfly via the command line, we need to specify the optimisation problem (i.e. the function to be maximised and the domain) and the optimisation parameters. We have demonstrated these on the Branin function in the `examples` directory. The domain, fidelity space (if applicable) and the function should be specified in JSON or protocol buffer format, as illustrated in Figures 7.5 and 7.6. In these figures, the `name` field specifies the name of the python file which contains the objective and it should be in the same directory as the JSON file. The following command, executes Dragonfly on the branin function.

```
$ cd dragonfly/examples
$ dragonfly-script.py --config synthetic/branin/config.json --
   options options_files/options_example.txt
```

Here, the `options` flag points to a text file which specifies the parameters for optimisation. We have illustrated one such options file in Figure 7.22. The multi-fidelity version of this demo can be run via the following command.

```
$ dragonfly-script.py --config synthetic/branin/config_mf.json
   --options options_files/options_example.txt
```

Using Dragonfly in the Python shell: Using Dragonfly in the Python shell is similar to using other libraries. Below, is one example.

```
# Acquisition function type
--acq ttei-ei-ucb

# No. of initial samples to ignore during sampling
--gpb_post_hp_tune_burn 170

# The budget of evaluations
--budget 100

# For mimisiation, set the max_or_min flag to min
--max_or_min max
```

Figure 7.22: An example options file in Dragonfly. In this file, `--acq` specifies the list of acquisitions to use in the ensemble method, and `--budget` indicates the budget of evaluations.

```
$ python
>>> from dragonfly import minimise_function
>>> min_val, min_pt, history = minimise_function(lambda x: x ** 4
   - x**2 + 0.1 * x, [[-10, 10]], 100);
...
>>> min_val, min_pt
(-0.32122746026750953, array([-0.7129672]))
```

Using Dragonfly in Python code: Dragonfly can be imported and used in Python as shown below. We have given a bare bones example, but in general, `func` is the function to be maximised, `domain` is a domain object, which can be loaded from a configuration file as specified above, and `budget` indicates the budget available for optimisation.

```
from dragonfly import minimise_function, maximise_function
from dragonfly.exd.domains import EuclideanDomain
func = lambda x: lambda x: x ** 4 - x**2 + 0.1 * x
domain = EuclideanDomain([[-10, 10]])
max_capital = 100

# Maximise the function
min_val, min_pt, history = minimise_function(func, domain, max_capital)
print(min_val, min_pt)

# Minimise the negative of the function
min_val, min_pt, history = maximise_function(lambda x: -func(x),
                                             domain, max_capital)
```

Chapter 8

Beyond Bandits: Adaptive Decision Making in Stateless Environments

In this chapter, we take a step back from bandits and analyse more general settings for sequential decision making under certainty. While bandit methods have seen great success in several hyperparameter tuning and optimisation problems, real world data collection tasks are more broad and complex, In such problems, like in the bandit setting, an agent takes an action, obtains an observation, and proceeds sequentially to choose the next action in order to achieve an application specific-goal. The agent should use the results of the previous actions to plan actions so as to achieve the desired goal in as few actions as possible. However, the problems we study in this setting differ from the bandit framework in a crucial way: the reward of the action is not directly observed by the agent. In typical applications, the reward of each action could depend on past actions and observations, and moreover, on uknown system characteristics. This makes is substantially more challenging than the bandit framework.

In Chapter 8.1, we first study a general design of experiments (DOE) framework for Bayesian decision making under uncertainty in stateless settings, and propose a general-purpose algorithm for such problems. Next, in Chapter 8.2, we analyse active posterior estimation, a special class of DOE tasks and develop specialised algorithms for that setting.

8.1 A General Framework for Adaptive Goal Oriented Design of Experiments

Many real world problems fall into the DOE framework, where one wishes to design a sequence of experiments and collect data so as to achieve a desired goal. For example, in electrolyte design for batteries, a chemist would like to conduct experiments that measure battery conductivity in order to identify an electrolyte that maximises the conductivity. On a different day, she would like to conduct experiments with different electrolyte designs to learn how the viscosity of the

electrolyte changes with design. These two tasks, black-box optimisation and active learning, fall under the umbrella of DOE and are pervasive in industrial and scientific applications.

While several methods exist for specific DOE tasks, real world problems are broad and complex, and specialised approaches have limited applicability. Continuing with the electrolyte design example, the chemist can typically measure both conductivity and viscosity with a single experiment [71]. Since such experiments are expensive, it is wasteful to first perform a set of experiments to optimise conductivity and then a fresh set to learn viscosity. It is preferable to design a single set of experiments that simultaneously achieves both goals. Another example is metallurgy, where one wishes to conduct experiments to identify phase transitions in an alloy as the composition of metals changes [28]. Here and elsewhere, both the model and the goal of the experimenter are very application specific and cannot be simply shoe-horned into formulations like black-box optimisation or active learning. In addition, domain knowledge about the problem may need to be considered in selecting experiments, as it may significantly reduce the number of experiments needed to achieve the desired goal.

To address these desiderata, we develop a general and flexible framework for *goal oriented* DOE, where a practitioner may specify her desired goal via a reward function λ. λ can depend on the data collected during the DOE process and unknown system characteristics, and hence cannot be directly computed by a decision-maker. We then develop an *adaptive* myopic strategy for DOE, inspired by posterior (Thompson) sampling for multi-armed bandits [246], which uses results from past experiments to plan future experiments and achieve the goal, i.e. maximise λ. Our approach has two key advantages. First, our Bayesian formulation allows one to straightforwardly specify domain expertise. Moreover, modern tools for probabilistic programming enable pratitioners to apply a Bayesian algorithm such as ours in a fairly straightforward manner. Second, our myopic strategy is simple and computationally attractive in comparison with policies that engage in long-term planning. Nevertheless, borrowing ideas from submodular optimisation and reinforcement learning, we derive natural conditions under which our myopic policy is competitive with the globally optimal one. Our contributions in this chapter are:

1. We propose a flexible framework for DOE that allows a practitioner to describe their system (via a probabilistic model) and specify their goal (via a reward function). We also derive an algorithm, Myopic Posterior Sampling (MPS), for this setting.

2. In our theoretical analysis, we explore conditions under which MPS, which learns about the system over time, is competitive with myopic and globally optimal strategies that have full knowledge of the system. For this, we leverage ideas from Thompson sampling, reinforcement learning, and adaptive submodularity.

3. Empirically, we demonstrate MPS performs favourably in a variety of synthetic and real world DOE problems. Despite our general formulation, MPS is competitive with specialised methods designed for particular problems. More importantly, it enables DOE in non-standard application-specific settings. Our Python implementation and experiments are available at `github.com/kirthevasank/mps`.

Related Work

The term DOE has been used to mean different things in different settings. Classically, the focus has been on "learning" an unknown system, and as such, the objective has been framed as maximising some notion of information gathered about the system. We will refer to these tasks as L-DOE problems in order to differentiate it from our setting, which subsumes L-DOE. Classical L-DOE focuses on discrete settings [41, 206] or linear models [7, 56]. Recently, there have been several Bayesian approaches for L-DOE that adopt probabilistic programming in more complex models [188, 201]. However, it is not clear if L-DOE approaches are efficient or appropriate for maximising an arbitrary user-specified reward λ. A second difference is that many of these approaches are non-adaptive, in that they aim to find an optimal batch of experiments beforehand without incorporating feedback from completed experiments. While some do explore adaptive approaches for L-DOE, they aim for globally optimal policies (e.g. Rainforth [201]), which can be computationally prohibitive, except in the most trivial cases.

We focus on posterior sampling (PS) [246] as the bandit algorithm, since it has proven to be quite general and admits a clean Bayesian analysis [212]. PS has been studied in a number of bandit settings [80, 129, 136], and some episodic RL problems [79, 184, 185], where the agent is allowed to restart. In contrast, here we study PS on a single long trajectory with no restarts.

Myopic/greedy policies, while computationally simple, are known to be near-optimal for sequential decision making problems with *adaptive submodularity* [74], which generalises submodularity and formalizes a diminishing returns property. Adaptive submodularity has been used for several adaptive DOE setups [38, 39, 40, 75]. However, in these work, the reward only depends on the data collected and can be directly computed by the decision-maker. As we will see shortly, in our setting, this translates to the agent knowing the system characteristics. As such, these results are complementary to ours: adaptive submodularity controls the approximation error (the difference between myopic- and globally-optimal strategies, both of which know the system), while we control the estimation error (how close our policy which needs to learn about the system is to the myopic optimal policy that knows the system). As we show in Theorem 73, with adaptive submodularity, MPS can also compete with the globally optimal policy. In a similar vein, Frazier et al. [62], Wang and Powell [258] use knowledge gradient approaches for information collection tasks which are framed as myopic adaptive submodular set maximisation problems; but as before, the system is known to the decision-maker. Prior results for learning in submodular environments are episodic and allow restarts [66, 67], which is unnatural in the DOE setup. In addition to the above, several papers have developed Bayesian methods for specific DOE applications such as black-box optimisation [61], active search [110], level set estimation [81] and many more [121, 163, 187], some of which adopt myopic strategies.

Our theoretical analysis leverages ideas from reinforcement learning (RL) since at each round the agent makes a decision (what experiment to perform) with the goal of maximising a long-term reward. In that light, one goal of our work is to understand when myopic "bandit-like" strategies perform well in RL environments with long-term temporal dependencies. There are two main differences with prior work [106, 137, 158, 184, 238]: first, we make no explicit assumptions about the complexity of the state and action space, instead placing assumptions on the reward

structure and optimal policy, which is a better fit for our applications. More importantly, in our setup, the true reward is never revealed to the agent, and instead it receives side-observations that provide information about an underlying parameter governing the environment. Secondly, our focus is on understanding when myopic strategies have reasonable performance rather than on achieving global optimality.

8.1.1 Formalism

Let Θ denote a parameter space, $\mathcal{X}$ an action space, and $\mathcal{Y}$ an outcome space. We consider a Bayesian setting where a *true parameter* $\theta_\star \in \Theta$ is drawn from a prior distribution ρ_0. A decision maker repeatedly chooses an action $X \in \mathcal{X}$, conducts an experiment at X, and observes the outcome $Y_X \in \mathcal{Y}$. We assume Y_X is drawn from a *likelihood* $\mathbb{P}(\cdot|X, \theta_\star)$, with known distributional form. This process proceeds for n rounds, resulting in a *data sequence* $D_n = \{(X_j, Y_{X_j})\}_{j=1}^n$, which is an ordered multiset of action-observation pairs. Unlike, classical formalisms for DOE, we study a setting where we intend to achieve a desired goal, specified via a *reward function* $\lambda : \Theta \times \mathcal{D} \to \mathbb{R}$, that we wish to maximise. Here, $\mathcal{D}$ denotes the set of all possible data sequences. In particular, after n rounds, we focus on the following two criteria, depending on the application:

$$(\mathbf{a})\ \ \Lambda(\theta_\star, D_n) = \sum_{t=1}^{n} \lambda(\theta_\star, D_t) \qquad (\mathbf{b})\ \ \lambda(\theta_\star, D_n), \tag{8.1}$$

Here, $D_t = \{(X_j, Y_{X_j})\}_{j=1}^t$ denotes the *prefix* of length t of the data sequence D_n collected by the decision maker. The former notion is the cumulative sum of all rewards, while the latter corresponds to the reward once all experiments are complete. Since λ depends on the unknown true parameter $\theta_\star$, the decision maker cannot compute the reward during the data collection process, and instead must infer the reward from observations in order to maximise it. This is a key distinction from existing work on reinforcement learning and sequential optimisation, and one of the new challenges in our setting.

Example 1. *A motivating example is* Bayesian active learning *[40]. Here, actions X correspond to data points while Y_X is the label and $\mathbb{P}(y|x, \theta)$ specifies an assumed discriminative model. We may set $\lambda(\theta, D_n) = -\|\beta(\theta) - \hat{\beta}(D_n)\|_2^2$ where β is a parameter of interest and $\hat{\beta}$ is a predetermined estimator (e.g. via maximum likelihood). The true reward $\lambda(\theta_\star, D_n)$ is not available to the decision maker since it requires knowing $\beta(\theta_\star)$.*

Notation: For each $t \in \mathbb{N}$, let $\mathcal{D}_t = \{\{(X_j, Y_{X_j})\}_{j=1}^t : X_j \in \mathcal{X}, Y_{X_j} \in \mathcal{Y}\}$ denote the set of all data sequences of length t, so that $\mathcal{D} = \bigcup_{t\in\mathbb{N}} \mathcal{D}_t$. Let $D \uplus D'$ denote the concatenation of two sequences. $D \prec D'$ and $D' \succ D$ both equivalently denote that D is a prefix of D'. Given a data sequence D_t, we use $D_{t'}$ for $t' < t$ to denote the prefix of the first t' action-observation pairs.

A *policy* for experiment design chooses a sequence of actions $\{X_j\}_{j\in\mathbb{N}}$ based on past actions and observations. In particular, for a *randomised* policy $\pi = \{\pi_j\}_{j\in\mathbb{N}}$, at time t, an action is drawn from $\pi_t(D_{t-1}) = \mathbb{P}(X_t \in \cdot|D_{t-1})$. Two policies that will appear frequently in the sequel

are $\pi^\star_{\mathrm{M}}$ and $\pi^\star_{\mathrm{G}}$, both of which operate with knowledge of $\theta_\star$. $\pi^\star_{\mathrm{M}}$ is the myopic optimal policy, which, from every data sequence D_t chooses the action X maximising the expected reward at the next step: $\mathbb{E}[\lambda(\theta_\star, D_t \uplus \{(X, Y_X)\})|\theta_\star, D_t]$. On the other hand $\pi^\star_{\mathrm{G}}$ is the non-myopic, globally optimal adaptive policy, which in state D_t with $n-t$ steps to go chooses the action to maximise the expected long-term reward: $\mathbb{E}[\lambda(\theta_\star, D_t \uplus \{(X, Y_X)\} \uplus D_{t+2:n}) \mid \pi^\star_{\mathrm{G}}, \theta_\star, D_t]$. $\pi^\star_{\mathrm{G}}$ may depend on the time horizon n while $\pi^\star_{\mathrm{M}}$ does not.

8.1.2 Myopic Posterior Sampling (MPS) for Design of Experiments

We present a simple and intuitive myopic strategy that aims to maximise λ based on the posterior of the data collected so far. For this, first define the expected look-ahead reward $\lambda^+ : \Theta \times \mathcal{D} \times \mathcal{X} \to [0, 1]$, such that $\lambda^+(\theta, D, x)$ is the expected reward at the next time step if $\theta \in \Theta$ were the true parameter, D was the current data sequence collected, and we were to take action $x \in \mathcal{X}$. Precisely,

$$\lambda^+(\theta, D, x) = \mathbb{E}_{Y_x \sim \mathbb{P}(Y|x,\theta)} \Big[\lambda\big(\theta,\ D \uplus \{(x, Y_x)\}\big) \Big]. \tag{8.2}$$

The proposed policy, presented in Algorithm 11, is called MPS (Myopic Posterior Sampling) and is denoted $\pi^{\mathrm{PS}}_{\mathrm{M}}$. At time step t, it first samples a parameter value θ from the posterior for $\theta_\star$ conditioned on the data, i.e. $\theta \sim \mathbb{P}(\theta_\star|D_{t-1})$. Then, it chooses the action X_t that is expected to maximise the reward λ by pretending that θ was the true parameter. It performs the experiment at X_t, collects the observation Y_{X_t}, and proceeds to the next time step.

Algorithm 11 MPS ($\pi^{\mathrm{PS}}_{\mathrm{M}}$) from Kandasamy et al. [131, 132]

Require: Prior ρ_0 for $\theta_\star$, Conditional $\mathbb{P}(Y|X, \theta)$.

1: $D_0 \leftarrow \varnothing$.
2: **for** $t = 1, 2, \ldots$ **do**
3: Sample $\theta \sim \rho_{t-1} \equiv \mathbb{P}(\theta_\star|D_{t-1})$.
4: Choose $X_t = \mathrm{argmax}_{x \in \mathcal{X}} \lambda^+_{t-1}(\theta, D_{t-1}, x)$.
5: $Y_{X_t} \leftarrow$ conduct experiment at X_t.
6: Set $D_t \leftarrow D_{t-1} \cup \{(X_t, Y_{X_t})\}$.
7: **end for**

A natural quesion that may arise is the need to sample from the posterior ρ_{t-1} for $\theta_\star$, instead of taking an expectation of λ^+ over ρ_{t-1}. In fact, many policies for non-adaptive L-DOE take an expectation over the posterior [188]. However, in adaptive settings where $\theta_\star$ is unknown, taking the expecation may fail as it may not explore sufficiently. For example, in bandit problems, which is a special case of our setting (Chapter 8.1.4), this amounts to choosing the maximum of the posterior mean of the payoff function, which is known to fail spectacularly.

Computational considerations: It is worth pointing out some computational considerations in Algorithm 11. First, sampling from the posterior in step 3 might be difficult, especially in complex Bayesian models. Fortunately however, the field of Bayesian inference has made great

strides in the recent past with the development of fast techniques for approximate inference methods such as MCMC or variational inference [92, 181]. Moreover, today we have efficient probabilistic programming tools [16, 31, 248] that allow a practitioner to intuitively incorporate domain expertise via a prior and obtain the posterior given data. Secondly, the maximisation of the look ahead reward in step 4 can also be non-trivial, especially since it might involve empirically computing the expectation in (8.2). This is similar to existing work in Bayesian optimisation which assume access to such an optimisation oracle [27, 235]. That said, in many practical settings where experiments can cost significant time, money, and resources, these considerations are less critical.

Despite these concerns, it is worth mentioning that myopic strategies are still computationally far more attractive than policies which try to behave globally optimally. For example, extending MPS to a k step look-ahead might involve an optimisation over $\mathcal{X}^k$ in step 4 of Algorithm 11 which might be impractical for large values of k. Moreover, in many problems where system characteristics $\theta_\star$ are known to the decision maker, myopic policies can be competitive with globally optimal policies [74, 183, 266]. In Chapter 8.1.3, we identify conditions where $\pi_{\mathrm{M}}^{\mathrm{PS}}$ can be competitive with the globally optimal policy $\pi_{\mathrm{G}}^\star$ which knows $\theta_\star$.

8.1.3 Theoretical Analysis

In this section we derive theoretical guarantees for $\pi_{\mathrm{M}}^{\mathrm{PS}}$. Our emphasis is on understanding conditions under which myopic algorithms which need to learn $\theta_\star$ can perform competitively with the myopic optimal and the globally optimal oracles $\pi_{\mathrm{M}}^\star, \pi_{\mathrm{G}}^\star$ which know $\theta_\star$ (see Chapter 8.1.2). Going forward, to simplify the exposition, we will assume that λ is bounded, i.e. $\lambda : \Theta \times \mathcal{X} \to [0, 1]$. Moreover, w.l.o.g, we will assume for all $\theta \in \Theta$, $\sup_{D \in \mathcal{D}} \lambda(\theta, D) = 1$. This condition is mild since for any other bounded reward λ', we can set $\lambda(\theta, D) \triangleq 1 + \lambda'(\theta, D) - \sup_{D \in \mathcal{D}} \lambda'(\theta, D)$.

For criterion $(\mathbf{a})$, we are interested in upper bounding $\mathbb{E}[\Lambda(\theta_\star, D_n)|D_n \sim \pi_{\mathrm{M}}^{\mathrm{PS}}]$ in terms of $\mathbb{E}[\Lambda(\theta_\star, D_n)|D_n \sim \pi_{\mathrm{M}}^\star]$, which yields a *cumulative regret bound*, and for criterion $(\mathbf{b})$, we wish to bound $\mathbb{E}[\lambda(\theta_\star, D_n) \mid D_n \sim \pi_{\mathrm{M}}^{\mathrm{PS}}]$ in terms of the analogous quantities for $\pi_{\mathrm{M}}^\star, \pi_{\mathrm{G}}^\star$, which serves as a *final regret bound*. Note that a comparison with $\pi_{\mathrm{G}}^\star$ on $(\mathbf{a})$ is meaningless since it might take low reward actions in the early stages in order to do well in the long run. In fact, our bounds for $(\mathbf{a})$ will hold when $\lambda(\theta_\star, D)$ is an ordered multi-set function in D, but for $(\mathbf{b})$ when $\lambda(\theta_\star, D)$ is a multi-set function, i.e. the ordering does not matter. We will aim to provide Bayesian regret bounds, which hold in expectation over $\theta_\star \sim \rho_0$.

The following proposition shows that without further assumptions, a non-trivial regret bound is impossible. Such results are common in the RL literature, and necessitate several structural assumptions [49, 106, 137]. Its proof is given in Chapter 8.3.

Proposition 69. *For all policies π which do not know $\theta_\star$, there exists a DOE problem where $\mathbb{E}_{\theta_\star \sim \rho_0}[\lambda(\theta_\star, D_n^\star) - \lambda(\theta_\star, D_n)|D_n^\star \sim \pi_{\mathrm{M}}^\star, D_n \sim \pi] \geq 1/2$ for all $n \geq 1$.*

Motivated by this lower bound, we impose the following condition on the parameter space and reward structure, under which a policy can achieve sub-linear regret. For this, first note we can

assume that, at all time steps, the observations $Y \sim \mathbb{P}(\cdot|x, \theta_\star)$ are generated for all $x \in \mathcal{X}$, but we only observe those for the chosen X_t. With this in consideration, let $\mathbb{E}_{Y,t+1:|\theta}$ denote expecation over all observations generated from time $t+1$ onwards when $\theta_\star = \theta$.

Condition 70. *Let θ, θ' denote parameter values in Θ and $\pi^\theta_{\mathrm{M}}, \pi^{\theta'}_{\mathrm{M}}$ be the myopically optimal policies when $\theta_\star = \theta$, and $\theta_\star = \theta'$ respectively. Let H denote a data sequence and D_n, D'_n be the data sequences collected by π^θ_{M} and $\pi^{\theta'}_{\mathrm{M}}$ respectively when starting from H. Then, there exists sequences $\{\epsilon_n\}_{n\geq 1}$, $\{\tau_n\}_{n\geq 1}$ such that the following hold.*

1. *π^θ_{M} achieves asymptotically similar reward $\forall\ \theta \in \Theta$,*

$$\sup_{\theta,\theta'\in\Theta} \sup_{H\in\mathcal{D}} \left\{ \mathbb{E}_{Y,|H|+1:|\theta} \lambda(\theta, H \uplus D_n) - \mathbb{E}_{Y,|H|+1:|\theta'} \lambda(\theta', H \uplus D'_n) \right\} \ \leq\ \epsilon_n.$$

2. *The rate of convergence is better than $\mathcal{O}(1/\sqrt{n})$. That is, letting $\sqrt{\tau_n} = 1 + \sum_{j=1}^n \epsilon_j$, we have $\tau_n \in o(n)$.*

The condition states that when we execute $\pi^\star_{\mathrm{M}}$, the myopically optimal policy which knows and depends on the value of $\theta_\star$, from any prefix H, it achieves asymptotically similar λ for all values of $\theta_\star$. It is worth emphasising that the condition involves executing π^θ_{M} in the environment where the true parameter $\theta_\star$ is θ. A condition of the above form seems necessary for any myopic algorithm π that does not know $\theta_\star$ for the following reason. Assume that the myopic $\pi^\star_{\mathrm{M}}$ can quickly achieve large λ value when $\theta_\star \in \Theta_g$ but is slow when $\theta_\star \in \Theta_b$. Since π does not know $\theta_\star$ it needs to hedge against the "bad" situation, i.e. $\theta_\star \in \Theta_b$. However, in doing so, it will necessarily perform poorly against $\pi^\star_{\mathrm{M}}$ when $\theta_\star \in \Theta_g$ as $\pi^\star_{\mathrm{M}}$ can quickly achieve large λ. Condition 70 prevents such situations. As we will see shortly, the regret for $\pi^{\mathrm{PS}}_{\mathrm{M}}$ will depend on τ_n which dictates how differently $\pi^\star_{\mathrm{M}}$ can behave for different values of $\theta_\star$. In particular, sublinearity of τ_n is necessary for sublinear regret with $\pi^\star_{\mathrm{M}}$.

In Chapter 8.3.3 we provide a more interpretable sufficient condition which implies Condition 70, and demonstrate that it is satisfied with $\tau_n \in \mathcal{O}(1)$ for bandit and black-box optimisation problems and $\tau_n \in \mathcal{O}(\log n)$ for an active learning problem. We also consider a setting where λ has "state-like" structure; under assumptions similar to standard assumptions in reinforcement learning with ergodic Markov decision processes, we are able to show that Condition 70 holds. Finally we mention that if Condition 70 holds for two reward functions λ_1, λ_2, it is also true for the sum, $\lambda_1 + \lambda_2$ and the product, $\lambda_1 \cdot \lambda_2$, and can thus be applied to combined objective settings such as, the electrolyte design example, we will see shortly in Chapter 8.1.4.

Before stating the main theorem, we introduce the maximum information gain, Ψ_n, which captures the statistical difficulty of the learning problem.

$$\Psi_n = \max_{D_n \subset \mathcal{D}_n} \mathrm{I}(\theta_\star; D_n). \tag{8.3}$$

Here $\mathrm{I}(\cdot;\cdot)$ is the Shannon mutual information. Ψ_n measures the maximum information a set of n action-observation pairs can tell us about the true parameter $\theta_\star$. The quantity appears as a statistical complexity measure in many Bayesian adaptive data analysis settings [82, 162, 235]. Below, we list some examples of common models which demonstrate that Ψ_n is typically sublinear in n.

Example 2. *We have the following bounds on Ψ_n for common models [235]:*

1. ***Finite sets:*** *If Θ is finite, $\Psi_n \leq \log(|\Theta|)$ for all n.*
2. ***Linear models:*** *Let $\mathcal{X} \subset \mathbb{R}^d$, $\theta \in \mathbb{R}^d$, and $Y_x|x, \theta \sim \mathcal{N}(\theta^\top x, \eta^2)$. For a multi-variate Gaussian prior on $\theta_\star$, $\Psi_n \in \mathcal{O}(d \log(n))$.*
3. ***Gaussian process:*** *For a Gaussian process prior with RBF kernel over a compact domain $\mathcal{X} \subset \mathbb{R}^d$, and with Gaussian likelihood, we have $\Psi_n \in \mathcal{O}(\log(n)^{d+1})$.*

We now state our main theorem for finite action spaces $\mathcal{X}$.

Theorem 71. *Let $\mathcal{X}$ be finite and assume Condition 70 holds. Let τ_n be as defined in Condition 70. Then,*

$$\mathbb{E}[\Lambda(\theta_\star, \pi^\star_{\mathrm{M}}) - \Lambda(\theta_\star, \pi^{\mathrm{PS}}_{\mathrm{M}})] \leq \sqrt{\frac{|\mathcal{X}| n \tau_n \Psi_n}{2}}.$$

Theorem 71 establishes a sublinear regret bound for $\pi^{\mathrm{PS}}_{\mathrm{M}}$ against $\pi^\star_{\mathrm{M}}$ when $\tau_n \Psi_n \in o(n)$. The $|\mathcal{X}|$ term captures the complexity of our action space, Ψ_n captures the complexity of the prior on $\theta_\star$. The $\sqrt{n}$ dependence is in agreement with prior results for Thompson sampling [135, 213]. Thus, under Condition 70, $\pi^{\mathrm{PS}}_{\mathrm{M}}$ is competitive with the myopic optimal policy $\pi^\star_{\mathrm{M}}$, with average regret tending to 0.

We now compare $\pi^{\mathrm{PS}}_{\mathrm{M}}$ to the globally optimal policy $\pi^\star_{\mathrm{G}}$, when λ is a multi-set function, i.e. the ordering in D_n does not matter. For this, we first introduce the notions of *monotonicity* and *adaptive submodularity*.

Condition 72. *(Monotonicity and Adaptive Submodularity [74]) Let $\mathbb{E}_{Y_x}$ denote the expectation over the likelihood $Y_x \sim \mathbb{P}(\cdot|x, \theta_\star)$. The following two statements are true for all $\theta \in \Theta$, $D, D' \in \mathcal{D}$, $D \prec D'$, and $x \in \mathcal{X}$. λ is a monotone, meaning that $\mathbb{E}_{Y_x}[\lambda(\theta, D \uplus \{(x, Y_x)\})] \geq \lambda(\theta, D)$. Moreover, λ is adaptive submodular, meaning that,*

$$\mathbb{E}_{Y_x}[\lambda(\theta, D \uplus \{(x, Y_x)\})] - \lambda(\theta_\star, D) \;\geq\; \mathbb{E}_{Y_x}[\lambda(\theta, D' \uplus \{(x, Y_x)\})] - \lambda(\theta_\star, D').$$

Monotonicity states that adding more data increases the reward in expectation, while adaptive submodularity formalises a notion of diminishing returns. That is, performing the same action is more beneficial when we have less data. It is easy to see that some assumption is needed here, since even in simple problems $\pi^\star_{\mathrm{M}}$ can be arbitrarily worse than $\pi^\star_{\mathrm{G}}$. We now state the second main result of this paper.

Theorem 73. *Assume that λ satisfies conditions 70 and 72. Let τ_n be as defined in Theorem 71. Then, for all $\gamma < 1$, we have*

$$\mathbb{E}[\lambda(\theta_\star, D_n)|D_n \sim \pi^{\mathrm{PS}}_{\mathrm{M}}] \;\geq\; (1-\gamma)\mathbb{E}[\lambda(\theta_\star, D^\star_{\gamma n})|D^\star_{\gamma n} \sim \pi^\star_{\mathrm{G}}] - \sqrt{\frac{|\mathcal{X}| \tau_n \Psi_n}{2n}}.$$

The theorem states that $\pi^{\mathrm{PS}}_{\mathrm{M}}$ in n steps is guaranteed to perform up to a $1-\gamma$ factor as well as $\pi^\star_{\mathrm{G}}$ executed for $\gamma n < n$ steps, up to an additive $\sqrt{\tau_n \Psi_n / n}$ term. The result captures both approximation and estimation errors, in the sense that we are using a myopic policy to approximate a

globally optimal one, and we are learning a good myopic policy from data. In comparison, prior works on adaptive submodular optimisation focus on approximation errors and typically achieve $1 - 1/e$ approximation ratios against the n steps of $\pi^\star_{\mathrm{G}}$. Our bound is quantitatively worse, but focusing on a much more difficult task, and we view the results as complementary. Observe that an analogous bound holds against $\pi^\star_{\mathrm{M}}$, since it is necessarily worse that $\pi^\star_{\mathrm{G}}$.

8.1.4 Examples & Experiments

In this section, we give some concrete examples of DOE problems that can be specified by a reward function λ and present experimental results for these settings. We compare $\pi^{\mathrm{PS}}_{\mathrm{M}}$ to random sampling (RAND), the myopically optimal policy $\pi^\star_{\mathrm{M}}$ which assumes access to $\theta_\star$, and to specialised methods developed for the particular problem, when available. In the interest of aligning our experiments with our theoretical analysis, we compare methods on both criteria in (8.1), although in these applications, the final reward $\lambda(\theta_\star, D_n)$ is more relevant than the cumulative one $\Lambda(\theta_\star, D_n)$. To better visualise the differences between the respective methods, we plot the negative reward in a semilog plot. Our Python implementation and experimental set up will be made available open source.

High-level Takeaways: Despite being a quite general, $\pi^{\mathrm{PS}}_{\mathrm{M}}$ outperforms, or performs as well as, specialised methods. $\pi^{\mathrm{PS}}_{\mathrm{M}}$ is competitive, but typically worse than the non-realisable $\pi^\star_{\mathrm{M}}$. Finally $\pi^{\mathrm{PS}}_{\mathrm{M}}$ enables effective DOE in complex settings where no prior methods seem applicable.

Active Learning

Problem: As described previously, we wish to learn some parameter $\beta_\star = \beta(\theta_\star)$ which is a function of the true parameter $\theta_\star$. Each time we query some $X \in \mathcal{X}$, we observe a label $Y \sim \mathbb{P}(Y|X, \theta_\star)$. We conduct two synthetic experiments in this setting. We use $\lambda(\theta_\star) \stackrel{\Delta}{=} -\|\beta_\star - \hat{\beta}(D_n)\|_2^2$ as the reward where $\hat{\beta}$ is a regularised maximum likelihood estimator. In addition to RAND and $\pi^\star_{\mathrm{M}}$, we compare $\pi^{\mathrm{PS}}_{\mathrm{M}}$ to ActiveSetSelect of Chaudhuri et al. [36].

Experiment 1: We use the following logistic regression model: $Y_x|x, \theta \sim \mathcal{N}(f_\theta(x), \eta^2)$ where $f_\theta(x) = \frac{a}{1+e^{b(x-c)}}$. The true parameter is $\theta_\star = (a, b, c, \eta^2)$ and our goal is to estimate $\beta_\star = (a, b, c)$. The MLE is computed via gradient ascent on the log likelihood. In our experiments, we used $a = 2.1, b = 7, c = 6$ and $\eta^2 = 0.01$ as $\theta_\star$. We used normal priors $\mathcal{N}(2, 1)$, $\mathcal{N}(5, 3)$ and $\mathcal{N}(5, 3)$ for a, b, c respectively and an inverse gamma $\mathrm{IG}(20, 1)$ prior for η^2. As the action space, we used $\mathcal{X} = [0, 10]$. For variational inference, we used a normal approximation for the posterior for a, b, c and an inverse gamma approximation for η^2. The results are given in the first row of Figure 8.1.

Experiment 2: In the second example, we use the following linear regression model: $Y_x|x, \theta \sim \mathcal{N}(f_\theta(x), 0.01)$ where $f_\theta(x) = \sum_{i=1}^{16} \theta_{*i}\phi(x - c_i)$. Here, $\phi(v) = \frac{1}{\sqrt{0.2\pi}} e^{-5\|v\|_2^2}$ and the points $c_1, \ldots, c_{16}$ were arranged in a 4×4 grid within $[0, 1]^2$. We set $\theta_{*i} = g(c_i)$, with $g(v) =$

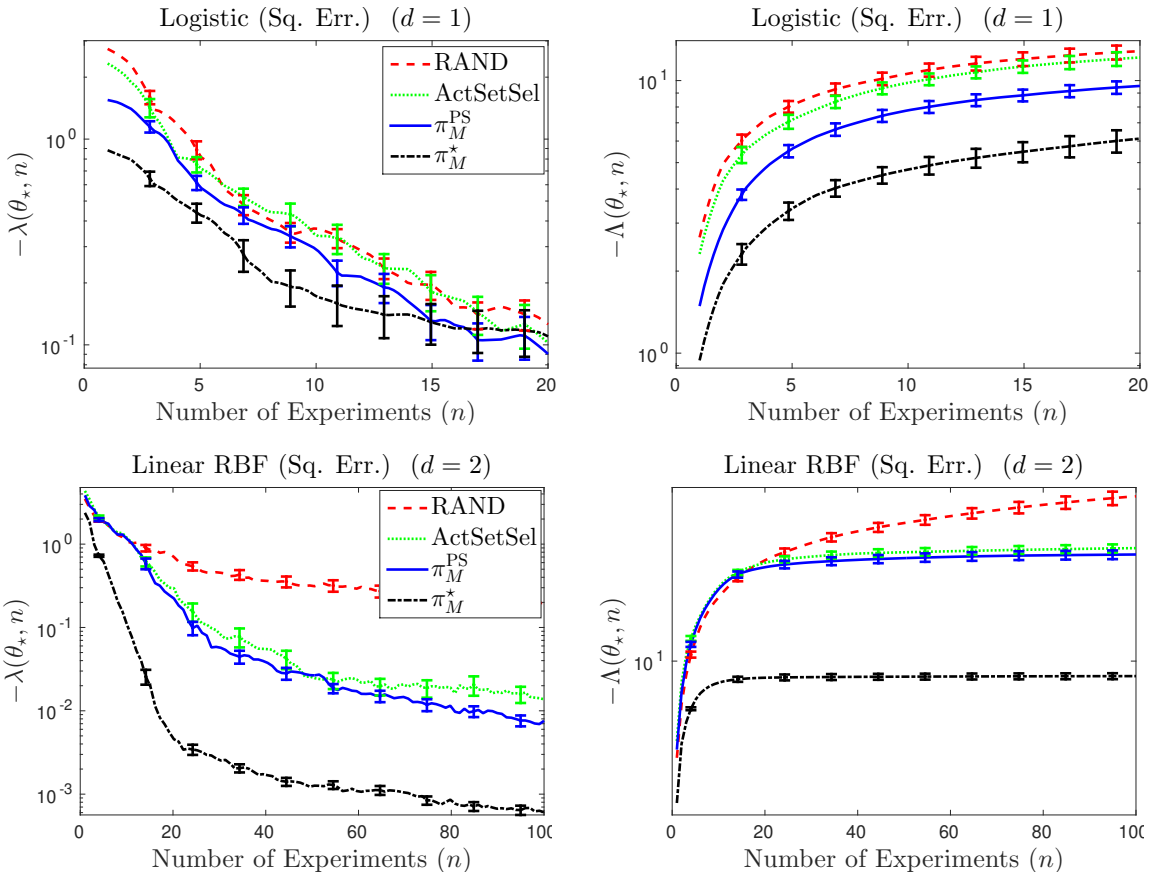


Figure 8.1: Results on the synthetic active learning experiments in Chapter 8.1.4 comparing all methods on the squared error reward. The title states the model and the dimensionality. In all figures, the x axis is the number of experiments n. In the left figures, the y axis is the final negative reward $-\lambda(\theta_\star, n)$ at the n^{th} iteration. In the right figures, it is the corresponding negative cumulative reward $-\Lambda(\theta_\star, n)$. Lower is better in both cases. The legend is given in the left figures. All curves were averaged over 20 runs, and error bars indicate one standard error.

$\sin(3.9\pi((v_1 - 0.1)^2 + v_2 + 0.1))$. Our goal is to estimate $\beta_\star = \theta_\star$. As the action space, we used $\mathcal{X} = [0, 1]^2$. The posterior for $\theta_\star$ was calculated in closed form using a normal distribution $\mathcal{N}(0, I_{16})$ as the prior. The results are given in the second row of Figure 8.1.

Alternative Problem Formalism: A common formalism for parameter estimation in discriminative models [36, 63] is to maximise the expected likelihood of the data for a given sampling distribution Γ on $\mathcal{X}$. Here, one wishes to maximise $\lambda(\theta_\star, D_n) \triangleq \mathbb{E}_{X\sim\Gamma, Y\sim\mathbb{P}(Y|X,\theta_\star)}[\log \mathbb{P}(Y|X, \hat{\theta})]$, where $\hat{\theta}$ is an estimator for θ obtained from D_n.

Experiments 3 & 4: We use the same models as in Experiment 1 & 2 but with the above reward function. We let Γ be the uniform distribution on the respective domains and $\hat{\theta}$ be the maximum likelihood estimator for θ. The results are given in Figure 8.2. In these experiments, in order to compute the reward λ, we evaluate the expecation over $X \sim \Gamma, Y \sim \mathbb{P}(\cdot|X, \theta)$ empirically by drawing 1000 (x, y) pairs; we first sample 1000 x values uniformly at random and then draw y

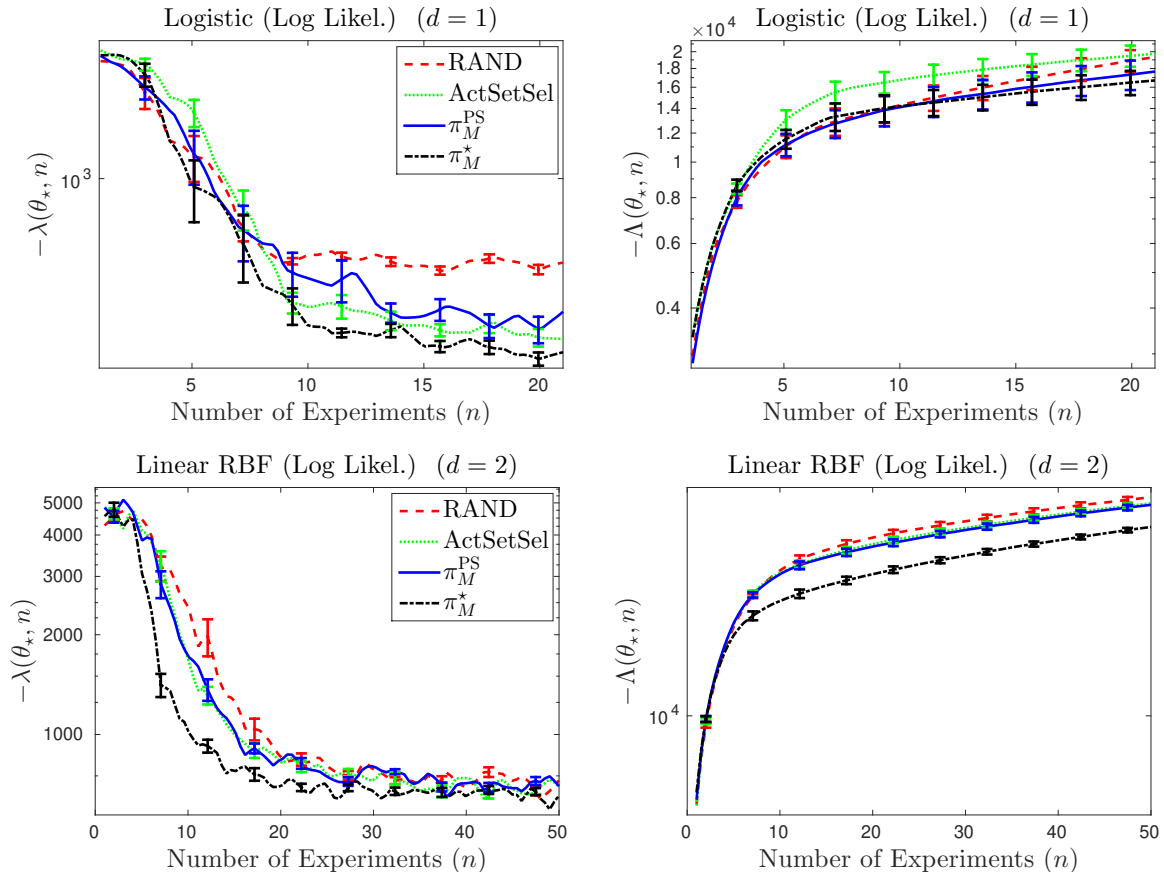


Figure 8.2: Results on the synthetic active learning experiments in Chapter 8.1.4 comparing all methods on the log likelihood reward. In all figures, the x axis is the number of experiments n. In the left figures, the y axis is the final negative reward $-\lambda(\theta_\star, n)$ at the n^{th} iteration. In the right figures, it is the corresponding negative cumulative reward $-\Lambda(\theta_\star, n)$. See caption under Figure 8.1 for more details.

from the likelihood for the given θ value.

Posterior Estimation & Active Regression

Problem: Consider estimating a non-parametric function $f_{\theta_\star}$, which is known to be uniformly smooth. An action $x \in \mathcal{X}$ queries $f_{\theta_\star}$, upon which we observe $Y_x = f_{\theta_\star}(x) + \epsilon$, where $\mathbb{E}[\epsilon] = 0$. If the goal is to learn $f_{\theta_\star}$ uniformly well in L^2 error, i.e. with reward $-\|f_{\theta_\star} - \hat{f}(D_n)\|^2$, adaptive techniques may not perform significantly better than non-adaptive ones [267]. However, if our reward was $\lambda(\theta_\star, D_n) \stackrel{\Delta}{=} -\|\sigma(f_{\theta_\star}) - \sigma(\hat{f}(D_n))\|^2$ for some monotone superlinear function σ, then adaptive techniques may do better by requesting more evaluations at regions with high $f_{\theta_\star}$ value. This is because, $\lambda(\theta_\star, D_n)$ is more sensitive to such regions due to the transformation σ.

A particularly pertinent instance of this formulation arises in astrophysical applications where one wishes to estimate the posterior distribution of cosmological parameters, given some astro-

nomical data Q [191]. Here, an astrophysicist specifies a prior Ξ over the cosmological parameters $Z \in \mathcal{X}$, and the likelihood of the data for a given choice of the cosmological parameters $x \in \mathcal{X}$ is computed via an expensive astrophysical simulation. The prior and the likelihood gives rise to an unknown log joint density[1] $f_{\theta_\star}$ defined on $\mathcal{X}$, and the goal is to estimate the the joint density $p(Z = x, Q) = \exp(f_{\theta_\star}(x))$ so that we can perform posterior inference. Adopting assumptions from Kandasamy et al. [121], we model $f_{\theta_\star}$ as a Gaussian process, which is reasonable since we expect a log density to be smoother than the density itself. As we wish to estimate the joint density, λ takes the above form with $\sigma = \exp$.

Experiment 5: We use data on Type I-a supernova from Davis et al [50]. We wish to estimate the posterior over the Hubble constant $H \in (60, 80)$, the dark matter fraction $\Omega_M \in (0, 1)$ and the dark energy fraction $\Omega_E \in (0, 1)$, which constitute our three dimensional action space $\mathcal{X}$. The likelihood is computed via the Robertson-Walker metric. In addition to $\pi^\star_{\mathrm{M}}$ and RAND, we compare $\pi^{\mathrm{PS}}_{\mathrm{M}}$ to Gaussian process based exponentiated variance reduction (GP-EVR) [121] which was specifically designed for this setting. We evaluate the reward via numerical integration. The results are presented in the first row of Figure 8.3.

Level Set Estimation

Problem: In active level set estimation (LSE), one wishes to determine which regions of a space $\mathcal{X}$ fall above or below a given level set of an expensive to evaluate function $f_{\theta_\star}$. An experiment evaluates this function and returns $Y_x = f_{\theta_\star}(x) + \epsilon$, where $\mathbb{E}[\epsilon] = 0$. We adopt the setting of Gotovos et al. [81], where a method for LSE returns its predictions for being above/below the threshold on a pre-specified set of discrete points $\mathcal{X}' \subset \mathcal{X}$. The reward function λ is set to be average prediction accuracy.

Experiment 6: Here we used data on Luminous Red Galaxies (LRGs) to compute the galaxy power spectrum of 9 cosmological parameters: spatial curvature $\Omega_k \in (-1, 0.9)$, dark energy fraction $\Omega_\Lambda \in (0, 1)$, cold dark matter density $\omega_c \in (0, 1.2)$, baryonic density $\omega_B \in (0.001, 0.25)$, scalar spectral index $n_s \in (0.5, 1.7)$, scalar fluctuation amplitude $A_s \in (0.65, 0.75)$, running of spectral index $\alpha \in (-0.1, 0.1)$ and galaxy bias $b \in (0, 3)$. Software and data were taken from Kandasamy et al. [121], Tegmark et al [244]. We wish to find regions of the cosmological parameter space, where the power spectrum is larger than a pre-specified threshold. Following Gotovos et al. [81], we model the function as a Gaussian process. The function values vary from approximately -1×10^{18} and -1×10^{15}. We set the threshold to -3×10^{16} which is approximately the 75$^{\text{th}}$ percentile when we randomly sampled the function value at several thousand points. We compare $\pi^{\mathrm{PS}}_{\mathrm{M}}$ to random search, $\pi^\star_{\mathrm{M}}$, and the Gaussian process based level set estimation (GP-LSE) method of Gotovos et al. [81]. Following Gotovos et al. [81], we model the power spectrum as a GP, and define the reward function as described above where $\mathcal{X}'$ is a set of $\sim 20K$ points. The results are presented in the second row of Figure 8.3.

[1] One should not conflate the prior over $\mathcal{X}$ specified with the astrophysics model, with prior over Θ assumed in our set up.

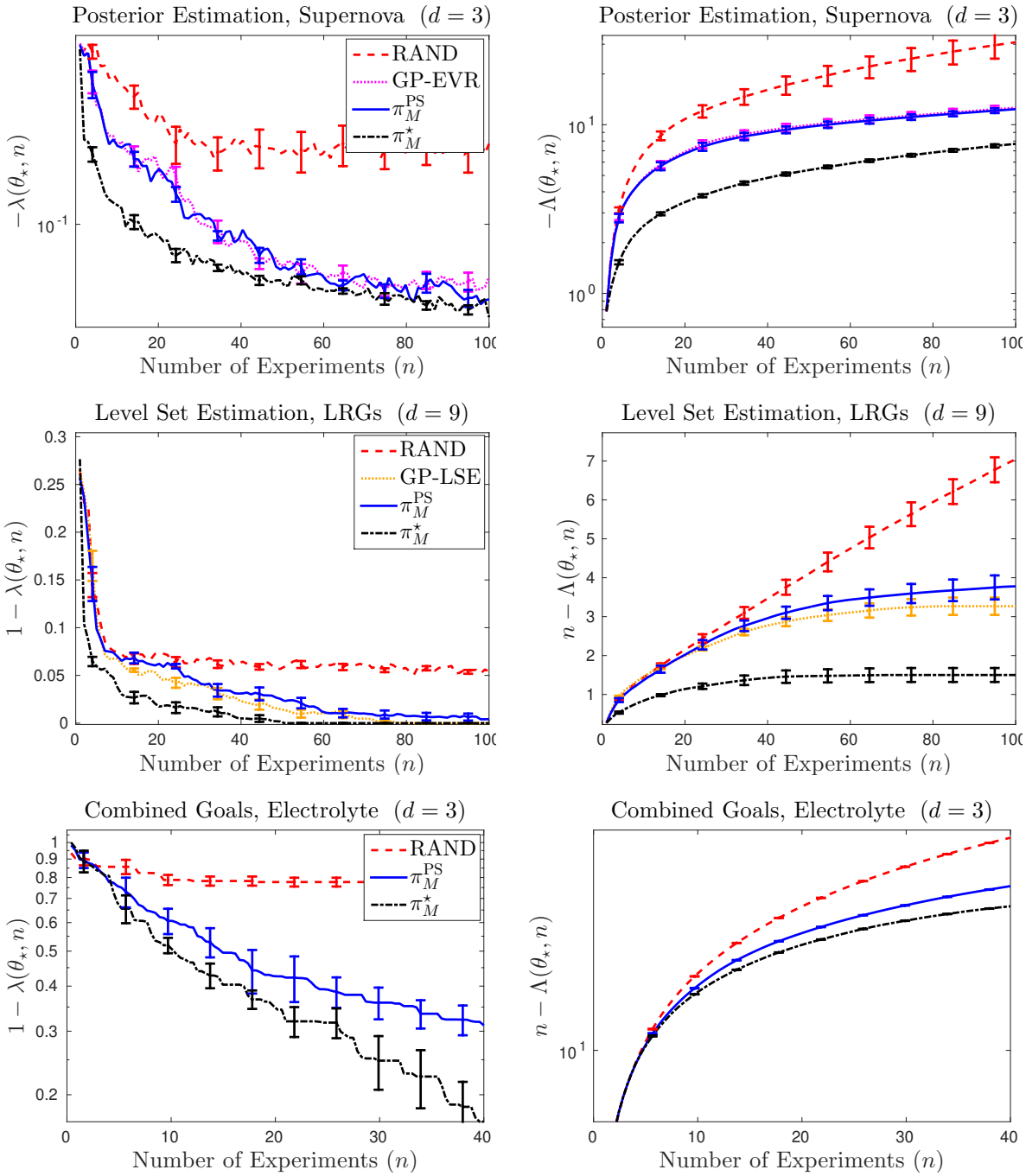


Figure 8.3: Results on the real experiments. The first row is for the posterior estimation problem, the second row is for the level set estimation problem, and the third row is for the combined objective problem, all of which are described in Chapter 8.1.4. In the left figures, the y axis is the negative reward $-\lambda(\theta_\star, D_n)$ and in the right figures, it is the negative cumulative reward $-\Lambda(\theta_\star, D_n)$ for the corresponding experiment. The legend is given in the left figures. See caption under Figure 8.1 for more details.

Combined and Customised Objectives

Problem: In many real world problems, one needs to design experiments with multiple goals. For example, an experiment might evaluate multiple objectives, and the task might be to optimise some of them, while learning the parameters for another. Classical methods specifically designed for active learning or optimisation may not be suitable in such settings. One advantage to the proposed framework is that it allows us to combine multiple goals in the form of a reward function. For instance, if an experiment measures two functions $f_{\theta_\star,1}, f_{\theta_\star,2}$ and we wish to learn f_1 while optimising f_2, we can define the reward as $\lambda(\theta_\star, D_n) = -\|f_{\theta_\star,1} - \hat{f}_1(D_n)\|^2 + \max_{X_t, t\leq n} \big(f_{\theta_\star,2}(X_t) - \max_x f_{\theta_\star,2}(x)\big)$. Here $\hat{f}_1$ is an estimate for $f_{\theta_\star,1}$ obtained from the data, $\|\cdot\|$ is the L_2 norm and $\max_{X_t,t\leq n} f_{\theta_\star,2}(X_t)$ is the maximum point of $f_{\theta_\star,2}$ we have evaluated so far. Below, we demonstrate one such application.

Experiment 7: In battery electrolyte design, one tests an electrolyte composition under various physical conditions. On an experiment at $x \in \mathcal{X}$, we obtain measurements $Y_x = (Y_{x,\text{sol}}, Y_{x,\text{vis}}, Y_{x,\text{con}})$ which are noisy measurements of the solvation energy f_{dissol}, the viscosity f_{vis} and the specific conductivity f_{con}. Our goal is to estimate f_{dissol} and f_{vis} while optimising f_{con}. Hence,

$$\lambda(\theta_\star, D_n) \;=\; \alpha\big(\max_{X_t,t\leq n} f_{\text{con}}(X_t) - \max_{x\in\mathcal{X}} f_{\text{con}}(x)\big) - \beta\|f_{\text{dissol}} - \hat{f}_{\text{dissol}}(D_n)\|^2 - \gamma\|f_{\text{vis}} - \hat{f}_{\text{vis}}(D_n)\|^2,$$

where, the parameters α, β, γ were chosen so as to scale each objective and ensure that none of them dominate the reward. In our experiment, we use the dataset from Gering [71]. Our action space $\mathcal{X}$ is parametrised by the following three variables: $Q \in (0,1)$ measures the proportion of two solvents EC and EMC in the electrolye, $S \in (0, 3.5)$ is the molarity of the salt $LiPF_6$ and $T \in (-20, 50)$ is the temperature in Celsius. We use the following prior which is based off a physical understanding of the interaction of these variables. $f_{\text{con}} : \mathcal{X} \to \mathbb{R}$ is sampled from a Gaussian process (GP), $f_{\text{vis}}(Q,S,T) = \exp(-aT)g_{\text{vis}}(Q,S)$ where g_{vis} is sampled from a GP, and $f_{\text{dissol}}(Q,S,T) = b + \exp(cQ - dS - eT)$. We use inverse gamma priors for a, b, d, e and a normal prior for c. For variational inference, we used inverse gamma approximations for a, b, d, e, a normal approximation for c, and GP approximations for f_{con} and g_{vis}. We use the posterior mean of f_{dissol} and f_{vis} under this prior as the estimates $\hat{f}_{\text{dissol}}, \hat{f}_{\text{vis}}$. We present the results in the third row of Figure 8.3 where we compare RAND, $\pi^{\text{PS}}_{\text{M}}$ and $\pi^{\star}_{\text{M}}$. This is an example of a customised DOE problem for which no prior method seems directly applicable.

Bandits & Bayesian Optimisation

Bandits and Bayesian optimisation are self-evident special cases of our formulation. Here, $\theta_\star$ specifies a function $f_{\theta_\star} : \mathcal{X} \to \mathbb{R}$. When we choose a point $X \in \mathcal{X}$ to evaluate the function, we observe $Y_X = f_{\theta_\star}(X) + \epsilon$ where $\mathbb{E}[\epsilon] = 0$. In the bandit framework, the reward is the instantaneous regret $\lambda(\theta_\star, D_n) = f_{\theta_\star}(X_n) - \max_{x\in\mathcal{X}} f_{\theta_\star}(x)$. In Bayesian optimisation, one is interested in simply finding a single value close to the optimum and hence $\lambda(\theta_\star, D_n) = \max_{t\leq n} f_{\theta_\star}(X_t) - \max_{x\in\mathcal{X}} f_{\theta_\star}(x)$. In either case, $\pi^{\text{PS}}_{\text{M}}$ reduces to the Thompson

sampling procedure as $\text{argmax}_{x \in \mathcal{X}} \lambda^+(\theta_\star, D_{t-1}, x) = \text{argmax}_{x \in \mathcal{X}} f_\theta(x)$, where f_θ is a random function drawn from the posterior. Since prior work has demonstrated that TS performs empirically well in several real world optimisation tasks [35, 96, 129], we omit experimental results for this example. One can also cast other variants of Bayesian optimisation, including multi-objective optimisation [93, 190] and constrained optimization [70], in our general formulation.

Some Experimental Details

Specification of the prior: In our experiments, we use a fixed prior in all our applications. In real world applications, the prior could be specified by a domain expert with knowledge of the given DOE problem. In some instances, the expert may only be able to specify the relations between the various variables involved. In such cases, one can specify the parametric form for the prior, and learn the parameters of the prior in an adaptive data dependent fashion using maximum likelihood and/or maximum a posteriori techniques [232].

Computing the posterior: Experiments 2 and 4 which use a Bayesian linear regression model admit analytical computation of the posterior. So do experiments 5 and 6 which use a Gaussian process model. For experiments 1, 3, and 7 we use the Edward probabilistic programming framework [248] for a variational approximation of the posterior. The sample in step 3 is drawn from this approximation.

Optimising λ^+: In all our experiments, the look-ahead reward (8.2) is computed empirically by drawing 50 samples from $Y|X, \theta$ for the sampled θ and a given $x \in \mathcal{X}$. For experiments 1 and 3 which are one dimensional, we maximise λ^+ by evaluating it on a fine grid of size 100 and choosing the maximum. Similarly, for experiments 2 and 4 which have two dimensional domains, we use a grid of size 2500 and for experiments 5 and 7 which are three dimensional, we use a grid of size 8000. Since experiment 6 is in nine dimensions, on each iteration, we sample 4000 points randomly from the domain and choose the maximum.

8.2 Active Posterior Estimation

In this sub-chapter, we study the problem of *active posterior estimation*, when the likelihood of a Bayesian model is expensive to compute. Computing the posterior distribution of parameters given observations is a central problem in Bayesian statistics. We use the posterior distribution to make inferences about likely parameter values and estimate functionals we are interested in. For simple parametric models with conjugate priors we may obtain the posterior in analytic form. In more complex models where the posterior is analytically intractable, we have to resort to approximation techniques. In some cases, we only have access to a black box which computes the likelihood for a given value of the parameters.

Our goal is an efficient way to estimate posterior densities when calls to this black box are expensive. This work is motivated by applications in computational physics and cosmology. Several cosmological phenomena are characterised by the cosmological parameters (e.g. Hubble

constant, dark energy fraction). Given observations, we wish to make inferences about the parameters. Physicists have developed simulation-based probability models of the Universe which can be used to compute the likelihood of cosmological parameters for a given observation. Many problems in scientific computing have a similar flavour. Expensive simulators in molecular mechanics, computational biology and neuroscience are used to model many scientific processes. Hence this work finds relevance in these fields as well.

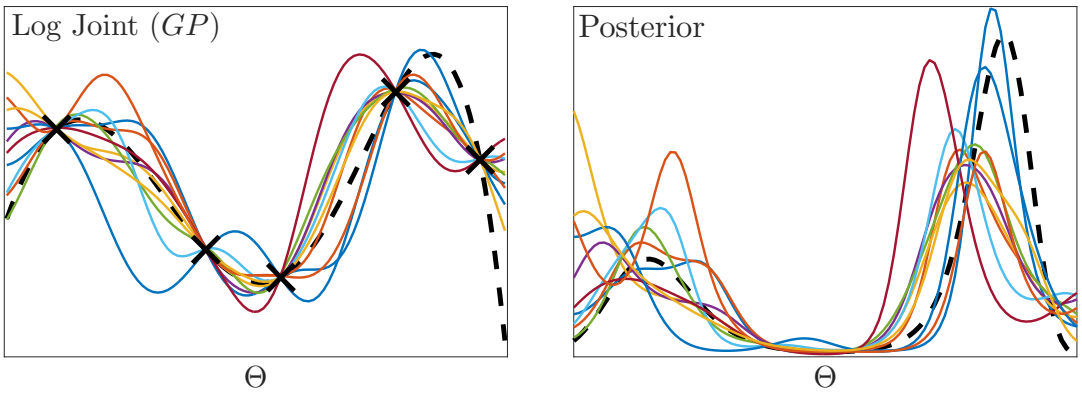
Related Work

Practitioners have conventionally used sampling schemes [166] to approximate the posterior distributions. Rejection sampling and various MCMC methods are common choices. The advantage of MCMC approaches is their theoretical guarantees with large sample sets [208] and thus they are a good choice when likelihood evaluations are cheap. However, none of them is intended to be query efficient when evaluations are expensive. Some methods spend most of their computation evaluating point likelihoods and then discard the likelihood values after doing an acceptance test. This gives insight into the potential gains possible by retaining those likelihoods for use in regression. Despite such deficiencies, MCMC remains one of the most popular techniques for posterior estimation in experimental physics [59, 152, 191] and the other fields [157].

Approximate Bayesian computation (ABC) [168, 169] is a method of last resort for estimating posteriors when a likelihood can not be computed. Unfortunately, it still requires the generation of simulated data, which is expensive in our setup, and it does not address efficient selection of parameter values to be tested at all. Nested sampling [229] is a technique commonly used is Astrostatistics. Kernel Bayes' Rule [65] is a non-parametric method of computing a posterior based on the embedding of probabilities in an RKHS. Both these methods require sampling from a distribution and do not address the question of which samples to choose if generating them is expensive. The work in Bryan et al. [24] actively learns level sets of an expensive function and derives confidence sets from the results. Gotovos et al. [81] also actively learn level sets via a classification approach. Our work is more general since we estimate the entire posterior.

Our methods draw inspiration from Gaussian Process (GP) based active learning methods such as Bayesian optimisation (BO) [175], Bayesian quadrature (BQ) [160], active GP Regression (AGPR) [219] and several others [120, 121, 142, 161, 235]. These methods have a common modus operandi to determining the experiment θ_t at time step t: construct a utility function φ_t based on the posterior GP conditioned on the queries so far; then maximise φ_t to determine θ_t. $\varphi_t(\theta)$ captures the value of performing an experiment at point θ. The key difference in such methods is essentially in the specification of φ_t to determine the next experiment. In our work, we adopt this strategy. The contributions of this sub-chapter are as follows.

1. We study posterior estimation in an active regression framework. We develop two query strategies that can efficiently compute the posterior of an expensive likelihood model and implement them using Gaussian processes.
2. Empirically, we demonstrate that our methods significantly outperform a suite of classical methods and other heuristics for posterior estimation.



(a) (b)

Figure 8.4: (a) depicts the uncertainty for the log joint probability via samples g drawn from the GP. (b) illustrates the induced uncertainty model $F_{\theta|\mathbf{X_{obs}}}$ for the posterior via the exponentiated and normalised samples $f = \exp g / \int \exp g$.

8.2.1 Bayesian Active Posterior Estimation (BAPE)

Problem Setting: We formally define our posterior distribution estimation problem in a Bayesian framework. Our notation in this sub-chapter will be different from Chapter 8.1. We have a bounded continuous parameter space Θ for the unknown parameters (e.g. cosmological constants). Let $\mathbf{X_{obs}}$ denote our observations (e.g. signals from telescopes). For each $\theta \in \Theta$ we have the ability to query an oracle for the value of the likelihood $\mathcal{L}(\theta) = P(\mathbf{X_{obs}}|\theta)$. These queries are expensive and possibly noisy. Assuming a prior $P_\theta(\theta)$ on Θ, we have the posterior $P_{\theta|\mathbf{X_{obs}}}$.

$$P_{\theta|\mathbf{X_{obs}}}(\theta|\mathbf{X_{obs}}) = \frac{\mathcal{L}(\theta)P_\theta(\theta)}{\int_\Theta \mathcal{L}(\theta)P_\theta(\theta)} = \frac{\mathcal{L}(\theta)P_\theta(\theta)}{P(\mathbf{X_{obs}})} \tag{8.4}$$

Our goal is to obtain an estimate $\widehat{P}_{\theta|\mathbf{X_{obs}}}$ of $P_{\theta|\mathbf{X_{obs}}}$ using as few queries to the oracle.

Some smoothness assumptions on the problem are warranted to make the problem tractable. A standard in the Bayesian nonparametrics literature is to assume that the function of interest is a sample from a Gaussian Process. In what follows we shall model the log joint probability of the cosmological parameters and the observations via a GP[2]. This is keeping in line with Adams et al. [3] who use a similar prior for GP density sampling.

Uncertainty for the posterior via uncertainty for the log joint: Assume that we have queried the likelihood oracle at n points, and for each query point θ_i the oracle provided us with $\mathcal{L}_i \approx P(\mathbf{X_{obs}}|\theta_i)$ answers. Let $A_n = \{\theta_i, \mathcal{L}_i\}_{i=1}^n$ denote the set of these query value pairs. We build our GP using $B_n = \{\theta_i, \log(\mathcal{L}_i P_\theta(\theta_i))\}_{i=1}^n$ as the input output pairs. The GP automatically

[2]We work in the log joint probability space since log smoothes out a function and is more conducive to be modeled as a GP. We also avoid issues such as non-negativity of $\widehat{P}_m^A(\theta|\mathbf{X_{obs}})$. M. Osborne and D. Duvenaud and R. Garnett and C. Rasmussen and S. Roberts and Z. Ghahramani [160] also use a similar log-transform before applying a GP.

induces uncertainty on the log joint probability; let us denote the distribution of $\log P(\mathbf{X_{obs}}, \theta)$ values at any $\theta \in \Theta$ by $\tilde{\mathcal{L}}(\theta)$. Moreover, if g is a sample from this GP, then $f = \exp g / \int \exp g$ denotes a sample from the induced uncertainty model $F_{\theta|\mathbf{X_{obs}}}$ for the posterior $P_{\theta|\mathbf{X_{obs}}}$. A sample from $F_{\theta|\mathbf{X_{obs}}}$ is a distribution over Θ. This is illustrated in Figure 8.4.

Finally, given any estimate $\widehat{\mathcal{P}}_n^A(\mathbf{X_{obs}}, \theta)$ of the log joint probability constructed using a set A_n of n parameter-likelihood pairs, the estimate of the posterior distribution is

$$\widehat{P}_n^A(\theta|\mathbf{X_{obs}}) = \frac{\exp \widehat{\mathcal{P}}_n^A(\mathbf{X_{obs}}, \theta)}{\int_\Theta \exp \widehat{\mathcal{P}}_n^A(\mathbf{X_{obs}}, \theta)}. \tag{8.5}$$

Bayesian Active Posterior Estimation: We now describe the procedure to determine the point at which we should query the likelihood. At time step t, we have already queried at $t-1$ points and have the set A_{t-1} of query value pairs. Our goal is to select the point θ_t for the next experiment to evaluate the likelihood. We adopt a myopic strategy which picks the point that maximizes a utility function φ_t. φ_t needs to capture a measure of divergence $D(\cdot\|\cdot)$ between the true and estimated distributions. A reasonable strategy would be to select θ_t to satisfy

$$\theta_t = \operatorname*{argmin}_{\theta_+ \in \Theta} D(\, P_{\theta|\mathbf{X_{obs}}} \,\|\, \widehat{P}^{A_{t-1} \cup \{(\theta_+, \mathcal{L}(\theta_+))\}} \,) \tag{8.6}$$

where $\widehat{P}^{A_{t-1} \cup \{(\theta_+, \mathcal{L}(\theta_+))\}}$ is our estimate of the posterior using $A_{t-1} \cup \{(\theta_+, \mathcal{L}(\theta_+))\}$. Obviously, this objective is not accessible in practice, since we know neither $P_{\theta|\mathbf{X_{obs}}}$ nor $\mathcal{L}(\theta_+)$. As surrogates to this ideal objective in Equation (8.6), in the following subsections we propose two utility functions for determining the next point: Negative Expected Divergence (NED) and Exponentiated Variance (EV). The first, NED adopts a Bayesian decision theoretic approach akin to the expected error reduction criterion used in active learning [220]. Here, we choose the point in Θ that yields the minimum expected divergence for the next estimate over the uncertainty model. Unfortunately, as we will see in Section 8.2.1, the NED utility is computationally expensive. Therefore, we propose a cheaper alternative, EV. In our experiments we found that both strategies performed equally well – so EV is computationally attractive. That said, some cosmological simulations are very expensive (taking several hours to a day) so NED is justified in such situations. We present our framework for BAPE using an appropriate utility function φ_t in Algorithm 12.

Algorithm 12 BAPE from Kandasamy et al. [121, 128]

Given: Input space Θ, GP prior μ_0, κ_0.

For $t = 1, 2, \ldots$ **do**

1. $\theta_t = \operatorname{argmax}_{\theta_t \in \Theta} \varphi_t(\theta)$
2. $\mathcal{L}_t \leftarrow$ Query oracle at θ_t.
3. Obtain posterior conditioned on $(\theta_i, \mathcal{L}_i P_\theta(\theta_i))_{i=1}^t$

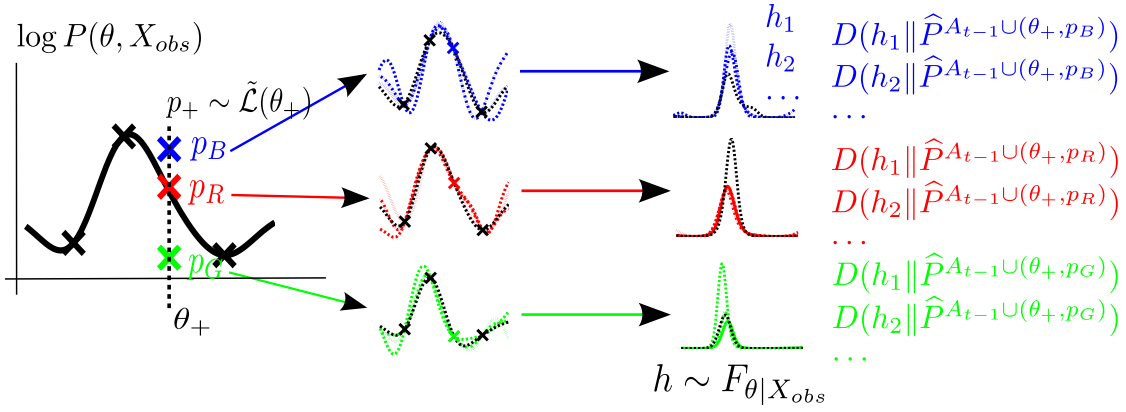


Figure 8.5: An illustration of the NED utility. θ_+ is a candidate for the next evaluation. p_B, p_R, p_G denote values for $p_+ = \log P(\theta_+, \mathbf{X_{obs}})$ sampled from the GP. We add them as hallucinated points and rebuild our GP and generate samples (second step). These samples are exponentiated and normalised (third step) and then its KL divergence with the estimate is computed.

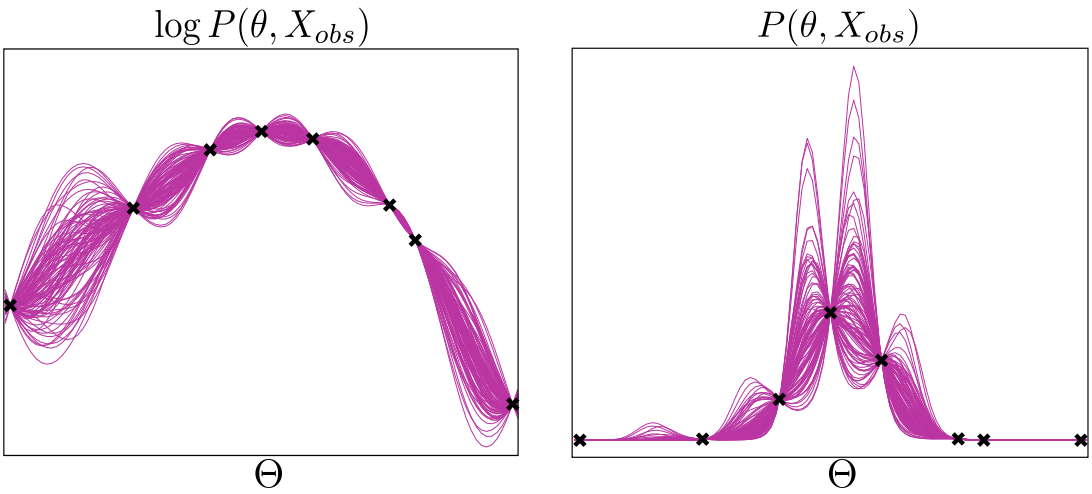
Negative Expected Divergence (NED)

Equation (8.6) says that we should choose the point that results in the highest reduction in divergence *if we knew the likelihood and the true posterior at that point*. In NED, we choose the point with the highest expected reduction in divergence. For the next evaluation we choose the point that minimizes the expected divergence between these models and the next estimate. Precisely,

$$u_t^{\text{NED}}(\theta_+) = -\mathbb{E}_{p_+ \sim \tilde{\mathcal{L}}(\theta_+)} \mathbb{E}_{h \sim F_{\theta|\mathbf{X_{obs}}}} D(\, h \,\|\, \widehat{P}_{m+1}^{A \cup \{(\theta_+, p_+)\}}\,). \tag{8.7}$$

Here $p_+ \in \mathbb{R}$ is sampled from $\tilde{\mathcal{L}}(\theta_+)$, the uncertainty of the log joing probability at θ_+. The *density* h is sampled from $F_{\theta|\mathbf{X_{obs}}}$, the uncertainty model of the posterior obtained by adding (θ_+, p_+) to the set of already available points A_{t-1}. Both $\tilde{\mathcal{L}}(\theta_+)$ and $F_{\theta|\mathbf{X_{obs}}}$ are induced from the log joint GP as explained before. $\widehat{P}_{m+1}^{A \cup \{(\theta_+, p_+)\}}$ denotes the estimate of the posterior obtained by retraining the GP with (θ_+, p_+) as the t^{th} point along with the t points already available. The first expectation above captures our uncertainty over $\log P(\theta_+, \mathbf{X_{obs}})$ while the second captures our remaining uncertainty over $P_{\theta|\mathbf{X_{obs}}}$ after observing $\mathcal{L}(\theta_+)$. Equation (8.7) says that you should minimize the expected divergence by looking one step ahead.

We have illustrated NED in Figure 8.5. Assume we are considering the point θ_+ for the next evaluation. Our GP over the log joint probability gives us uncertainty for $\log P(\theta_+, \mathbf{X_{obs}})$ – depicted by p_B, p_R, p_G in blue, green and red respectively. For p_B, we add (θ_+, p_B) as a hallucinated point to the $t-1$ points we already have and obtain an estimate of the posterior $\widehat{P}^{A_{t-1} \cup (\theta_+, p_B)}$. Next, we rebuild our GP using these t points. We draw samples from the new GP and exponentiate and normalise them to obtain samples h_i from the uncertainty model for the posterior $F_{\theta|\mathbf{X_{obs}}}$. Then we compute the divergence between h_i and $\widehat{P}^{A_{t-1} \cup (\theta_+, p_B)}$. We repeat this for the blue and green points and average all the divergences. The next evaluation point will be that with the lowest expected one step ahead divergence.



(a) (b)

Figure 8.6: (a): Samples drawn from the GP in the log joint probability space. (b): The same samples after exponentiation. High variance in the low likelihood regions are squashed and low variances in the high likelihood regions are blown up. This is the key insight that inspires our methods and the EV utility in particular.

The expectations in the NED utility above are computationally intractable. They can only be approximated empirically by drawing samples and require numerical integration (as Figure 8.5 suggests). For these reasons we propose an alternate utility function below. In our experiments we found that both EV and NED performed equally well.

Exponentiated Variance (EV)

A common active learning heuristic is to choose the point that you are most uncertain about for the next experiment. As before we use a GP on the log joint probability. At any given point in this GP we have an associated posterior variance of the GP. However, this variance corresponds to the uncertainty of the *log* joint probability whereas our objective is in learning the joint probability – which is a multiplicative factor away from the posterior. See Figure 8.6. Therefore, unlike in usual GP active learning methods [219], the variance of interest here is in the exponentiated GP. By observing that an exponentiated Gaussian is a log Normal distribution, the EV utility function is given by

$$u_t^{\text{EV}}(\theta_+) = \exp(2\mu_t(\theta_+) + \sigma_t^2(\theta_+))(\exp(\sigma_t^2(\theta_+)) - 1) \tag{8.8}$$

Here μ_t, σ_t^2 are the posterior mean and variances of the GP at time t. The $\exp(2\mu(\theta_+))$ will squash high variances in the low likelihood regions and amplify low variances in the high likelihood regions. The expression for $u_t^{\text{EV}}(\theta_+)$ corresponds precisely to the variance of $P(\theta, \mathbf{X_{obs}})$ according to the uncertainty model induced by the GP. We choose the point *maximising* the above variance to determine the next query location.

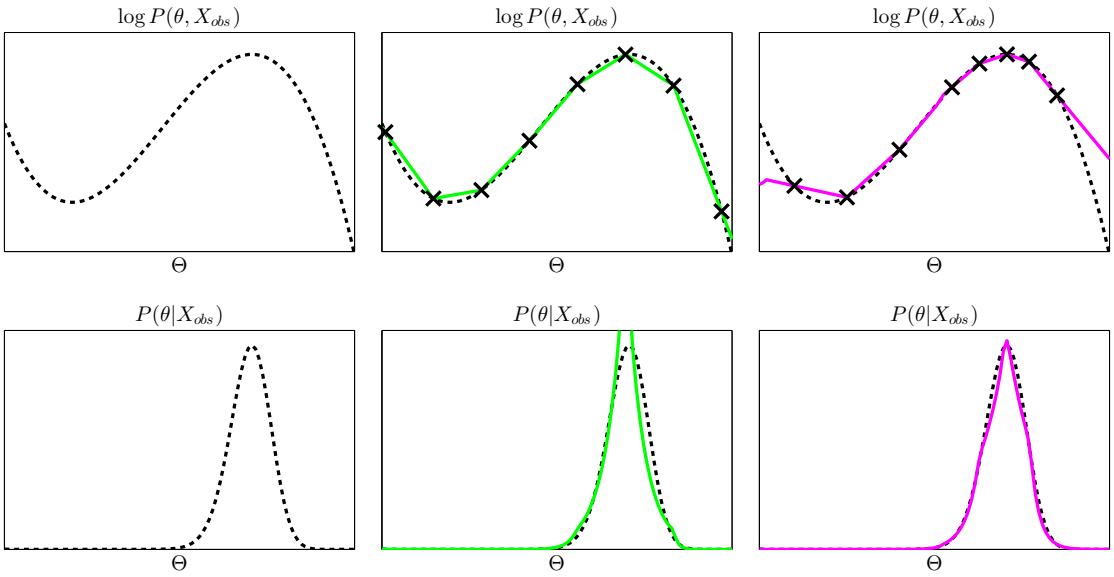


Figure 8.7: The first column shows the log joint probability and the corresponding posterior. In the second column we have estimates of the log joint and the posterior for uniformly spaced points. In the third column we have the same except that more points were chosen in high likelihood regions.

Discussion

We first argue that an active, i.e. an adaptive sequential, strategy will be useful for posterior estimation. In particular, the work of Castro et al. [33] demonstrates that active learning does not perform significantly better than passive strategies to estimate a uniformly smooth function uniformly well. However, in our case we wish to learn the function well at high log probabilty regions as they predominantly determine the shape of the posterior. To illustrate this we have shown a synthetically created log joint probability and the corresponding posterior in the first column of Figure 8.7. The high likelihood regions largely affect the shape of the posterior since the variations in the low likelihood region are squashed after exponentiation. In the second column we queried the likelihood at uniformly spaced points and obtained estimates of the log joint probability and the posterior in green. In the third we have the same estimates in magenta, except that we used more queries at at high likelihood regions. While the green estimate for the *log joint* may be uniformly better than the magenta estimate, it is the opposite for *posterior*. This is because after exponentation, the small errors in the high log probability regions have been inflated after exponentiation for the green estimate, whereas for the magenta estimate the large errors in the low probability regions have been diminished. Hence an active strategy, which uses more queries at the high log probability regions is likely to do better than a passive strategy. NED and EV do precisely this by attaching more emphasis to the uncertainties in the high log probability regions.

Next, it is important to distinguish our objective in this work from similar active learning literature in the GP framework. In BO, the objective is to find the maximum of a function. This means that once the active learner realises that it has found the mode of a function it has less incentive to

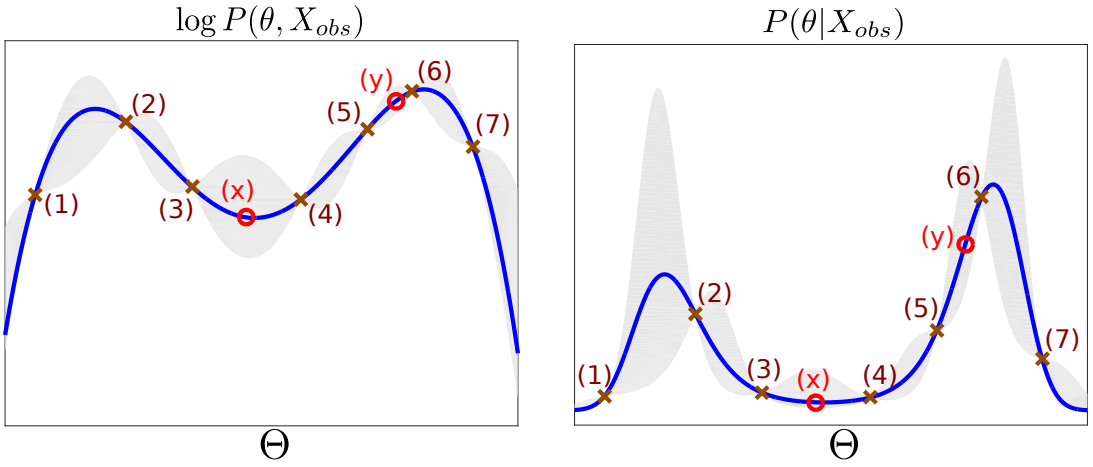


Figure 8.8: (a) and (b) are the true log joint probability and joint probability in blue. Assume that we have already queried at the brown crosses and let the red circles (x) and (y) be candidates for the next query. In BAPE we would be interested in querying (y) but not (x). In AGPR we would be interested in both (x) and (y) whereas in BO we would be keen in neither.

explore around as it would not improve the current maximum values. For instance, consider the log joint probability in Figure 8.8(a) and the joint probability in Figure 8.8(b). We have shown the points where we have already queried at as brown crosses and the red circles (x) and (y) show possible candidates for the next query. The shaded regions represent the uncertainty due to three standard deviations in the GP. In BO, the active learner would not be interested in (y) as, by virtue of points (5), (6) and (7) it knows that (y) is not likely to be higher than (6). On the other hand, in BAPE we are keen on (y) as knowing it with precision will significantly affect our estimate of the posterior (Fig 8.8(b)). In particular to know the posterior well we will need to query at the neighborhood of modes and the heavy tails of a distribution. A BO utility is not interested in such queries. On the other extreme, in AGPR the objective is to learn the function uniformly well. This means in the same figures, AGPR will query point (x). However, given sufficient smoothness, we know that the joint probability will be very low there after exponentiation due to points (3) and (4). Therefore, the BAPE active learner will not be as interested in (x) as AGPR. Observe that the uncerainty at (x) is large in the log joint probability space in comparison to the uncertainty elsewhere; however, in the probability space this is smaller than the uncertainty at the high probability regions. As Figure 8.7 indicates, while we model the log joint probability as a GP we are more interested in the uncertainty model of the posterior/joint probability. Finally, as a special case for BQ, [160] consider evaluating the model evidence–i.e. the integral under the conditional. Their utility function uses approximations tailored to estimating the integral well. Note that our goal of estimating the posterior well is more difficult than estimating an integral under the conditional as the former implies the latter but not vice versa.

8.2.2 Experiments

We perform experiments on a series of low and high dimensional synthetic and real astrophysical experiments. In our experiments the NED, EV utilities were maximised by evaluating them on a grid of size $10^3 - 10^8$ depending on the dimensionality and then choosing the point with the maximum value. For numerical integration in NED, we use the trapezoidal rule. Further, since the inner expectation in Equation (8.7) is expensive we approximate it using a one sample estimate. NED is only tested on low dimensional problems since empirical approximation and numerical integration is computationally expensive in high dimensions. In our experiments, EV slightly outperforms NED probably since the EV utility can be evaluated exactly while NED can only approximated.

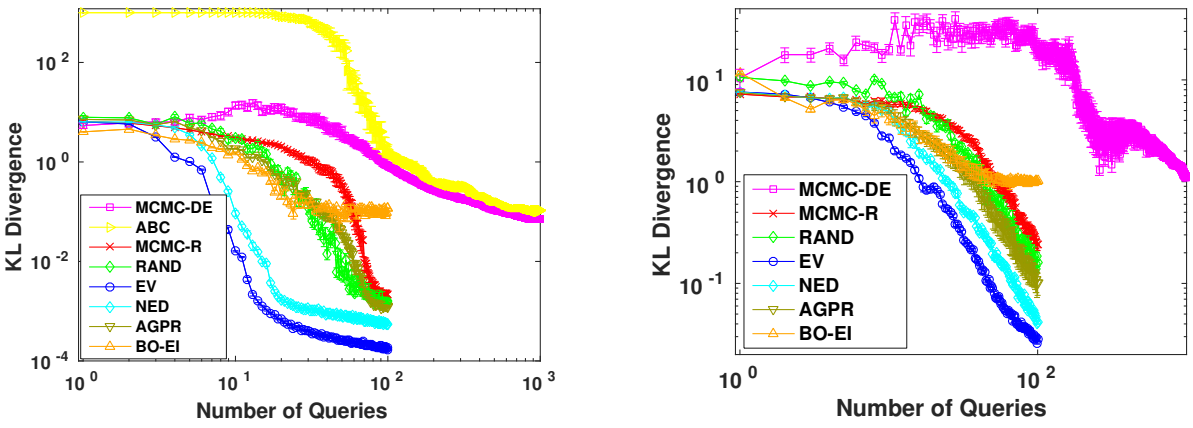
We use a squared exponential kernel in all our experiments. The bandwidth for the kernel was set to be $5n^{-1/d}$ where n is the total number of queries and d is the dimension. This was following several kernel methods (such as kernel regression) which use a bandwidth on the order $O(n^{\frac{-c_1}{c_2+d}})$ [87]. The constant 5 was hand tuned by experimenting with a series of independent synthetic experiments. The other GP hyperparameters, σ_f^2 and σ_n^2 were set via cross validation every 20 iterations. When we tried setting the bandwidth via cross validation too we found that it had a tendency too choose a larger than required bandwidth in the early iterations and then get stuck without decreasing. The consequence of this behaviour is that our method might not sufficiently explore the space and hence miss out on certain regions of the likelihood. Such a phenomenon has also been observed in Bayesian Optimisation and hence the bandwidth is decreased artificially as a precautionary measure against insufficient exploration [264].

Baselines

We first list and describe some potential alternatives for posterior estimation which we use in our empirical evaluation.

1. MCMC - Density Estimation (MCMC-DE): We implement MCMC with a Metropolis Hastings (MH) chain and use kernel density estimation (KDE) on the accepted points to estimate the posterior. When comparing MCMC against NED/EV we consider *the total number of queries* and not just those accepted. There are several variants of the MH proposal scheme and several tuning parameters. Comparing to all of them is nontrivial. We use MH in its basic form using a fixed Gaussian proposal distribution. Practitioners usually adjust the proposal based on the acceptance rate. Here, we chose the proposal manually by trying different values and picking the one that performed best within the queries used. Note that this comparison is advantageous to MCMC. In one experiment we test with Emcee [59], a popular package for Affine Invariant MCMC which automatically fine tunes the proposal bandwidth based on acceptance rate [59].

2. MCMC - Regression (MCMC-R): Here, as in MCMC-DE we use a MH Chain to generate the samples. However, this time we regress on the queries (not samples) to estimate the posterior. We include this procedure since MCMC can be viewed as a heuristic to explore the parameter



(a) (b)

Figure 8.9: (a), (b): Comparison of NED/EV against MCMC-DE, ABC, MCMC-R and RAND for the 1D and 2D synthetic experiments respectively. The x-axis is the number of queries and the y-axis is the KL divergence between the truth and the estimate. All figures were obtained by averaging over 60 trials.

space in high likelihood regions. We show that a principled query strategy outperforms this heuristic.

3. Approximate Bayesian Computing (ABC): There are several variants of ABC [169, 194]. We compare with a basic form given in [168]. At each iteration, we randomly sample θ from the prior and then sample an observation $\mathbf{X}_{\mathbf{sim}}$ from the likelihood. If $d(\mathbf{X}_{\mathbf{sim}}, \mathbf{X}_{\mathbf{obs}}) < \epsilon$ we add θ to our collection. Here $d(\cdot, \cdot)$ is some metric on a sufficient statistic of the observation and $\epsilon > 0$ is a prespecified threshold. We perform a KDE on the collected samples to estimate the posterior. The performance of ABC depends on ϵ: As for MCMC-DE we choose ϵ by experimenting with different values and choosing the value which gives the best performance within the queries used. We compare with total number of parameter values proposed and not just those retained. We compare with ABC only in experiments where it is possible to sample from the likelihood (in addition to evaluating the likelihood).

4. Uniform Random Samples (RAND): Here, we evaluate the likelihood at points chosen uniformly on Θ and then regress on these points.

5. Active Gaussian Process Regression (AGPR): & 6. Bayesian Optimisation (BO-EI): On our synthetic problems we also compare with the GP based active learning methods discussed in Chapter 8.2.1. We choose points using the above criterion and then regress on these points.

Low Dimensional Synthetic Experiments

To illustrate our methods we have two simple yet instructive experiments. In the first, the parameters space is $\Theta = (0, 1)$ equipped with a Beta$(1.2, 1)$ prior. We draw θ from the prior, and then draw 500 samples from a Bernoulli $(\theta^2 + (1-\theta)^2)$ distribution: i.e. $\mathbf{X}_{\mathbf{obs}} \in \{0, 1\}^{500}$. The

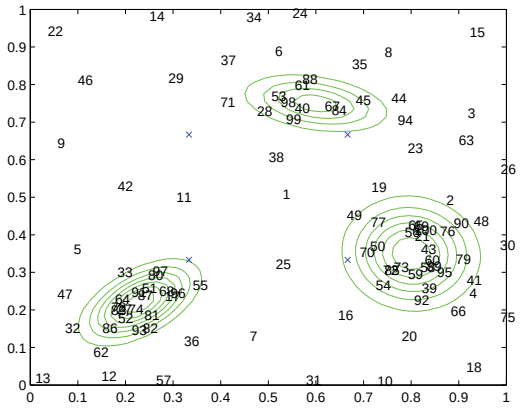


Figure 8.10: The 100 points chosen in order by NED for the 2D experiment. The green contours are the true posterior. Initially the algorithm explores the space before focusing on high probability regions.

ambiguity on the true value of θ creates a bimodal posterior. Figure 8.9(a) compares NED/EV against the other methods as a function of the number of queries. For ABC, we rejected if $\left(\sum_i \mathbf{X_{obs}}^{(i)} - \sum_i \mathbf{X_{sim}}^{(i)}\right) / \sum_i \mathbf{X_{obs}}^{(i)} > 0.02$.

The second experiment is a 2D problem with $\Theta = (0, 1)^2$. We artificially created a 3-modal log-joint posterior shown by green contours in Figure 8.10. Figure 8.9(b) compares all methods. Figure 8.10 shows the points chosen by the NED query strategy in order. We have learned the high log joint probability regions well at the expense of being uncertain at low log joint probability areas. However, this does not affect the posterior significantly as they are very small after exponentiation. ABC does not apply here since we artificially constructed the log posterior and cannot sample from the likelihood.

Our methods outperform existing methods and other heuristics by orders of magnitude on these simple experiments. Both MCMC-DE and ABC require a large number of samples before being competitive with the methods using regression. This corroborates an earlier remark that using the evaluated likelihood values in the estimate can be useful when the queries are expensive. Note that the KL divergence for BO gets stuck without decreasing further. This is because after a certain stage, most evaluations are centred near the maximum. As a consequence, the heavy tails and other modes are not explored properly.

Higher Dimensional Synthetic Experiments

We test in $d = 5$ and 15 dimensions. We construct an artificial log likelihood so that the resulting posterior is mixture of 2 Gaussians centred at $\mathbf{0}$ and $\mathbf{1}$. Both Gaussians had covariance $\sigma^2 I_d$ where $\sigma = 0.5\sqrt{d}$. We evaluate performance by the ability to estimate certain linear functionals. The exact value of these functionals can be evaluated analytically since we know the true posterior. We use a uniform prior. Our log-likelihood, functionals and their true values are

$$\ell(\theta) = \log\left(0.5\,\mathcal{N}\Big(\theta;\ \mathbf{0},\ \frac{d}{4}I_d\Big) + 0.5\,\mathcal{N}\Big(\theta;\ \mathbf{1},\ \frac{d}{4}I_d\Big)\right)$$

$$T_1 = \mathbb{E}\left[\sum_{i=1}^{d} X_i\right] = \frac{d}{2}, \qquad T_3 = \mathbb{E}\left[\sum_{i=1}^{d-2} X_i^2 X_{i+1}\right] = \frac{d-1}{2}(1+\sigma^2),$$
$$T_2 = \mathbb{E}\left[\sum_{i=1}^{d} X_i^2\right] = \frac{d}{2}(1+2\sigma^2), \qquad T_4 = \mathbb{E}\left[\sum_{i=1}^{d-2} X_i X_{i+1} X_{i+2}\right] = \frac{d-2}{2}$$

For MCMC-DE, we draw samples $Z_1, Z_2, \ldots$ from the true likelihood. To estimate $T_i = \mathbb{E}[\phi_i(X)]$ we use the empirical estimator $\widehat{T}_i = 1/N \sum_k \phi_i(Z_k)$. Here $\phi_1 = \sum_{i=1}^{d} X_i$ for T_1 etc. For MCMC-DE we experimented with Gaussian proposal distributions with For EV , MCMC-R and RAND we first use the queried points to obtain an estimate of the log-likelihood by regressing on the likelihood values as explained before. Then we run an MCMC chain on this *estimate* to collect samples and use the empirical estimator for the functionals. Note that evaluating the estimate, unlike the likelihood, is cheap. We did not try NED since numerical integration is intractable in high dimensions. ABC does not apply in this experiment as we cannot sample from the log likelihood. For the proposal distributions for MCMC-DE and MCMC-R methods we used a Gaussian with standard deviations $\{0.25\sigma, 0.5\sigma, \sigma, 2\sigma, 4\sigma\}$ and report the one that performed best within the alotted queries. When applying MCMC on the regression estimates in EV, MCMC-R and RAND we used a Gaussian proposal with standard deviation σ. The results are shown in Figure 8.11. They demonstrate the superiority of our query strategy over the alternatives.

Type Ia Supernovae

We use supernovae data for inference on 3 cosmological parameters: Hubble Constant ($H_0 \in (60, 80)$, Dark Matter Fraction $\Omega_M \in (0, 1)$ and Dark Energy Fraction $\Omega_\Lambda \in (0, 1)$. The likelihood for the experiment is given by the Robertson–Walker metric which models the distance to a supernova given the parameters and the observed red-shift. The dataset is taken from [50]. The parameter space is taken to be $\Theta = (0, 1)^3$ (For H_0 we map it to $(60, 80)$ using an appropriate linear transform). We test NED/EV against MCMC-DE, ABC, MCMC-R, RAND and Emcee, a popular python package for affine invariant MCMC. For ABC, sampling from the likelihood is as expensive as computing the likelihood. Figure 8.12(a) compares all methods. Figure 8.12(b) shows the points queried by EV and the marginals of the true posterior. As expected, most of EV's queries are concentrated around the modes and heavy tails of the posterior. The KL for RAND decreases slowly since it accumulates points at the high likelihood region very slowly. MCMC-R performs poorly since it has only explored part of the high likelihood region. For NED/EV after an initial exploration phase, the error shoots down.

The likelihood evaluations in this experiment are quite cheap, taking only a fraction of a second for each query. Determining the next point is more expensive in EV than methods such as MCMC, ABC and RAND due to matrix inversion in GPs. However, we found that the latter methods required up to 20-30 times the number of likelihood evaluations to be competitive with EV in this experiment. Therefore, despite the fact that the EV query strategy is expensive it performs better than other methods on wall clock time. This illustrates that principled adaptive query strategies can reap great dividends in posterior estimation.

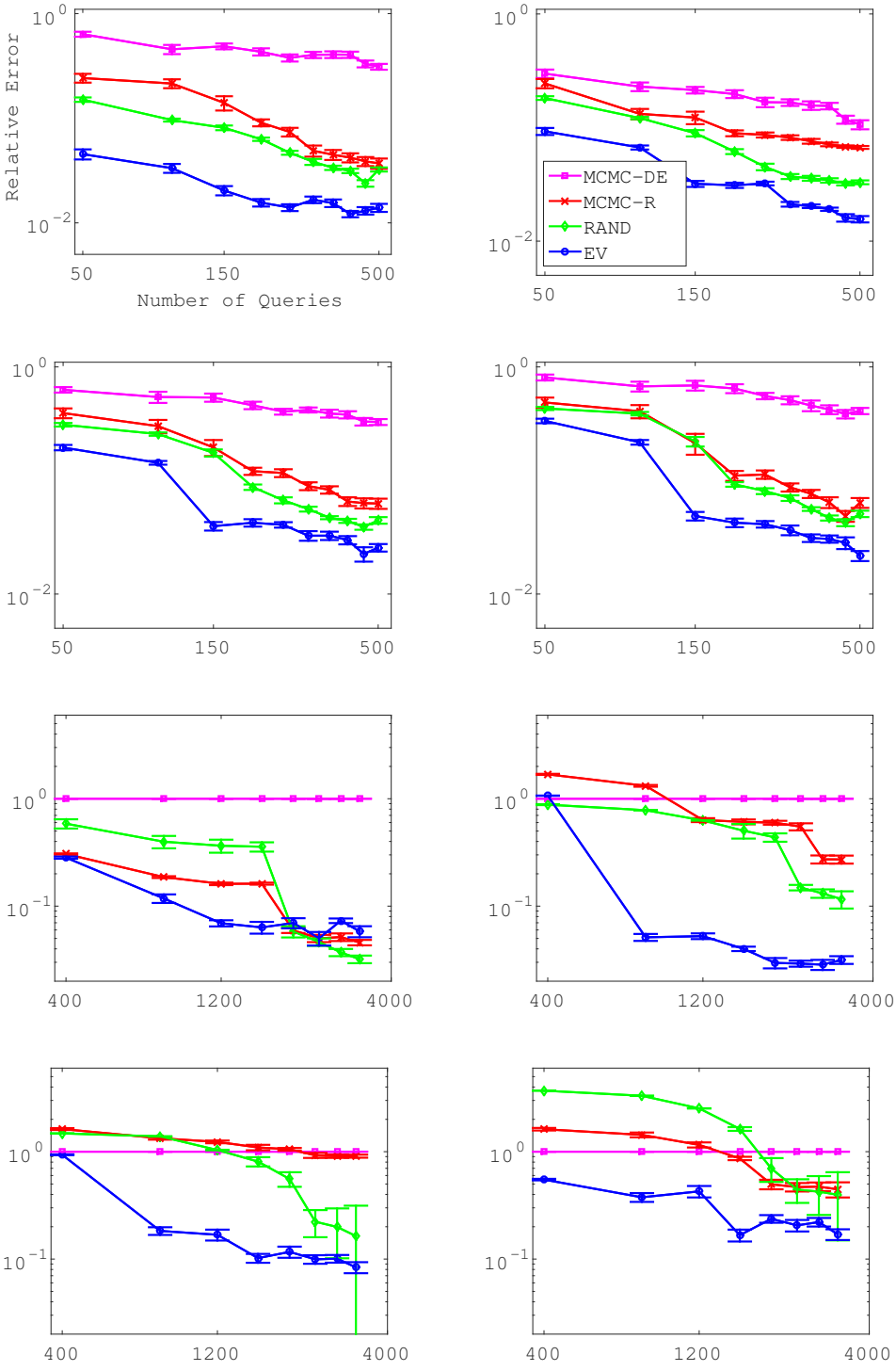
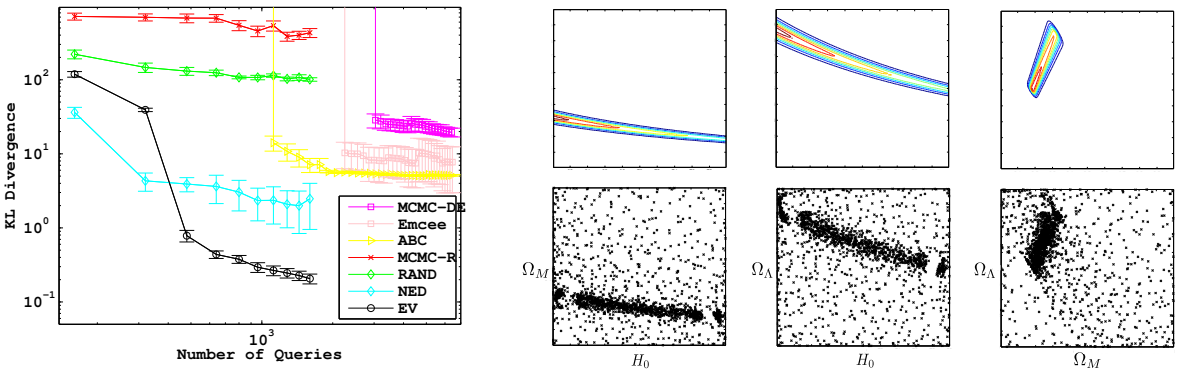


Figure 8.11: The first row is for the functionals T_1, T_2 in $d = 5$ dimensions and the second for is for the functionals T_3, T_4. The last twoape rows are the same four functionals for $d = 15$. The x-axis is the number of queries and the y-axis is $|\widehat{T}_i - T_i|/|T_i|$. We use 500 queries for $d = 5$ and 3200 queries for $d = 15$. All figures were obtained by averaging over 30 trials.



(a) (b)

Figure 8.12: (a): Comparison of NED/EV against MCMC-DE, ABC, Emcee, MCMC-R and RAND on the Type Ia Supernovae dataset. For all regression methods we show results for up to 1600 queries and up to 4 times as many for MCMC and ABC. For evaluation, KL was approximated via numeric integration on a $(100)^3$ grid. Note that MCMC and ABC require several queries before a nontrivial KL with the truth is obtained. All curves were obtained by averaging over 30 runs. (b): Projections of the points selected by EV (bottom row) and the marginal distributions (top row).

Luminous Red Galaxies

Here we used data on Luminous Red Galaxies (LRGs) for inference on 8 cosmological parameters: spatial curvature $\Omega_k \in (-1, 0.9)$, dark energy fraction $\Omega_\Lambda \in (0, 1)$, cold dark matter density $\omega_c \in (0, 1.2)$, baryonic density $\omega_B \in (0.001, 0.25)$, scalar spectral index $n_s \in (0.5, 1.7)$, scalar fluctuation amplitude $A_s \in (0.65, 0.75)$, running of spectral index $\alpha \in (-0.1, 0.1)$ and galaxy bias $b \in (0, 3)$. The likelihood is obtained via the Galaxy Power spectrum which measures the distribution of temperature fluctuations as a function of scale. We use software and data from [244]. Our parameter space is taken to be $(0, 1)^8$ by appropriately linear transforming the range of the variables. Each query takes about 4-5 seconds. In EV, determining the next point takes about 0.5-1 seconds with ≈ 2000 points and about 10-15 seconds with ≈ 10000 points. In this regime, where the cost of the likelihood evaluation is more expensive or comparable to the cost of determining the next point in EV, we significantly outperforms other methods on wall clock time. We do not compare with NED due to the difficulty of high dimensional numerical integration. We do not compare with ABC since the software only permits evaluation of the likelihood and not sampling.

Figure 8.13 shows points queried by MCMC, RAND and EV projected on the first 2 dimensions. MCMC has several high likelihood points but its queries are focused on a small region of the space. RAND does not have many points at high likelihood regions. EV has explored the space fairly well and at the same time has several queries at high likelihood regions. As numerical integration in 8 dimensions is difficult, we cannot obtain ground truth for this experiment. Therefore, we perform the following simple test. We queried $250,000$ points uniformly at random from the parameter space to form a test set. We then run EV, MCMC-R and RAND for up

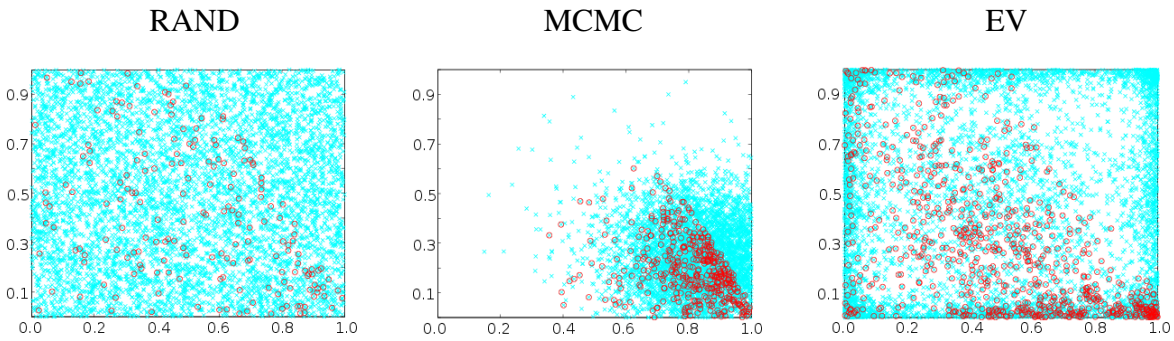


Figure 8.13: The projections of the first 6000 points queried by RAND MCMC, and EV respectively on to the first 2 dimensions in cyan. The points shown in red are queries at high likelihood ($\log P > -50$) points.

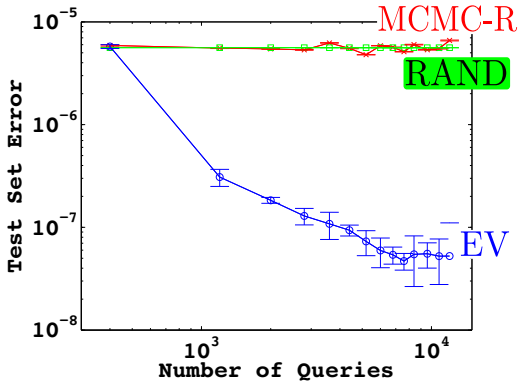


Figure 8.14: Comparison of EV against MCMC-R and RAND. We use up to 12000 queries for all methods. The y-axis is the mean squared reconstruction error. The curves were obtained by averaging over 16 runs.

to $12,000$ queries to collect points and estimate the posterior. Performance is evaluated by the mean squared reconstruction error of the *exponentiated* log joint probabilities (joint probabilities). Figure 8.14 shows the results. The error for RAND and MCMC-R stay the same throughout since the problem is difficult and they did not have sufficient number of high likelihood points throughout the space.

8.3 Proofs of Theoretical Results in Chapter 8.1

We will begin with a proof of the lower bound in Proposition 69.

Proof of Lower Bound (Proposition 69). Consider a setting with uniform prior over two parameters θ_0, θ_1 with two actions X_0, X_1. Set $\lambda(\theta_i, D) = \mathbf{1}\{X_i \notin D\}$. If $\theta_\star = \theta_0$, then $\pi^\star_{\mathrm{M}}$ will repeatedly choose X_1 and achieve reward 1 on every time step, and similarly when $\theta_\star = \theta_1$. On the other hand, conditioned on any randomness of the decision maker (which is external to the randomness of the prior and the observations), the first decision for the decision maker must be the same for both choices of $\theta_\star$. Hence, for one of the two choices for $\theta_\star$, $\lambda(\theta_\star, D_n) = 0$ for all n. Since the prior is equal on both θ_0, θ_1, the average instantaneous regret is at least $1/2$. □ □

8.3.1 Notation and Set up

In this subsection, we will introduce some notation, prove some basic lemmas, and in general, lay the groundwork for the proofs of Theorems 71 and 73. $\mathbb{P}, \mathbb{E}$ denote probabilities and expectations. $\mathbb{P}_t, \mathbb{E}_t$ denote probabilities and expectations when conditioned on the actions and observations up to and including time t, e.g. for any event E, $\mathbb{P}_t(E) = \mathbb{P}(E|D_t)$. For two data sequences A, B, $A \uplus B$ denotes the concatenation of the two sequences. When $x \in \mathcal{X}$, Y_x will denote the random observation from $\mathbb{P}(Y|x, \theta)$.

Let $J_n(\theta_\star, \pi)$ denote the expected sum of cumulative rewards for fixed policy π after n rounds under $\theta_\star$, i.e. $J_n(\theta_\star, \pi) = \mathbb{E}[\Lambda(\theta_\star, D_n)|\theta_\star, D_n \sim \pi]$ (Recall (8.1)). Let $D_t \in \mathcal{D}_t$ be a data sequence of length t. Then, $Q^\pi(D_t, x, y)$ will denote the expected sum of future rewards when, having collected the data sequence D_t, we take action $x \in \mathcal{X}$, observe $y \in \mathcal{Y}$ and then execute policy π for the remaining $n - t - 1$ steps. That is,

$$Q^\pi(D_t, x, y) = \lambda(\theta_\star, D_j \uplus \{(x, y)\}) + \mathbb{E}_{F_{t+2:n}}\left[\sum_{j=t+2}^{n} \lambda(\theta_\star, D_j \uplus \{(x, y)\} \uplus F_{t+2:j})\right]. \tag{8.9}$$

Here, the action-observation pairs collected by π from steps $t+2$ to n are $F_{t+2:n}$. The expectation is over the observations and any randomness in π. While we have omitted for conciseness, Q^π is a function of the true parameter $\theta_\star$. Let d_π^t denote the distribution of D_t when following a policy π for the first t steps. We then have, for all $t \leq n$,

$$J_n(\theta_\star, \pi) = \mathbb{E}_{D_t \sim d_\pi^t}\left[\sum_{j=1}^{t} \lambda(\theta_\star, D_j)\right] + \mathbb{E}_{D_t \sim d_\pi^t}\left[\mathbb{E}_{X \sim \pi(D_t)}[Q^\pi(D_t, X, Y_X)]\right], \tag{8.10}$$

where, recall, Y_X is drawn from $\mathbb{P}(Y|X, \theta_\star)$. The following Lemma decomposes the regret $J_n(\theta_\star, \pi_{\mathrm{M}}^\star) - J_n(\theta_\star, \pi)$ as a sum of terms which are convenient to analyse. The proof is adapted from Lemma 4.3 in Ross and Bagnell [210].

Lemma 74. *For any two policies π_1, π_2,*

$$J_n(\theta_\star, \pi_1) - J_n(\theta_\star, \pi_2) = \sum_{t=1}^{n} \mathbb{E}_{D_{t-1} \sim d_{\pi_1}^{t-1}}\left[\mathbb{E}_{X \sim \pi_1(D_{t-1})}\left[Q^{\pi_2}(D_{t-1}, X, Y_X)\right] - \mathbb{E}_{X \sim \pi_2(D_{t-1})}\left[Q^{\pi_2}(D_{t-1}, X, Y_X)\right]\right]$$

Proof. Let π^t be the policy that follows π_1 from time step 1 to t, and then executes policy π_2 from $t+1$ to n. Hence, by (8.10),

$$J_n(\theta_\star, \pi^t) = \mathbb{E}_{D_{t-1} \sim d_\pi^{t-1}}\left[\sum_{j=1}^{t-1} \lambda(\theta_\star, D_j)\right] + \mathbb{E}_{D_{t-1} \sim d_{\pi_1}^{t-1}}\left[\mathbb{E}_{X \sim \pi_1(D_{t-1})}[Q^{\pi_2}(D_{t-1}, X, Y_X)]\right],$$

$$J_n(\theta_\star, \pi^{t-1}) = \mathbb{E}_{D_{t-1} \sim d_\pi^{t-1}}\left[\sum_{j=1}^{t-1} \lambda(\theta_\star, D_j)\right] + \mathbb{E}_{D_{t-1} \sim d_{\pi_1}^{t-1}}\left[\mathbb{E}_{X \sim \pi_2(D_{t-1})}[Q^{\pi_2}(D_{t-1}, X, Y_X)]\right].$$

The claim follows from the observation, $J(\theta_\star, \pi_1) - J(\theta_\star, \pi_2) = J(\theta_\star, \pi^n) - J(\theta_\star, \pi^0) = \sum_{t=1}^n J(\theta_\star, \pi^t) - J(\theta_\star, \pi^{t-1})$. □

We will use Lemma 74 with π_1 as the policy $\pi^\star_{\mathrm{M}}$ which knows $\theta_\star$ and with π_2 as the policy π whose regret we wish to bound. For this, denote the action chosen by π when it has seen data D_{t-1} as X_t and that taken by $\pi^\star_{\mathrm{M}}$ as X'_t. By Lemma 74 and equation (8.9) we have,

$$\begin{aligned}\mathbb{E}_{\theta_\star}[J_n(\theta_\star, \pi^\star_{\mathrm{M}}) - J_n(\theta_\star, \pi)] &= \sum_{t=1}^n \mathbb{E}_{D_{t-1}}\left[\mathbb{E}_{t-1}\left[Q^{\pi^\star_{\mathrm{M}}}(D_{t-1}, X'_t, Y_{X'_t}) - Q^{\pi^\star_{\mathrm{M}}}(D_{t-1}, X_t, Y_{X_t})\right]\right] \\ &= \mathbb{E}\sum_{t=1}^n \mathbb{E}_{t-1}\left[q_t(\theta_\star, X'_t, Y_{X'_t}) - q_t(\theta_\star, X_t, Y_{X_t})\right], \end{aligned} \tag{8.11}$$

where we have defined

$$q_t(\theta_\star, x, y) = Q^{\pi^\star_{\mathrm{M}}}(D_{t-1}, x, y). \tag{8.12}$$

Note that the randomness in q_t stems from its dependence on $\theta_\star$ and future observations.

8.3.2 Comparing $\pi^{\mathrm{PS}}_{\mathrm{M}}$ against $\pi^\star_{\mathrm{M}}$ and $\pi^\star_{\mathrm{G}}$

We will let $\tilde{\mathbb{P}}_{t-1}$ denote the distribution of X_t given D_{t-1}; i.e. $\tilde{\mathbb{P}}_{t-1}(\cdot) = \mathbb{P}_{t-1}(X_t = \cdot)$. The density (Radon-Nikodym derivative) $\tilde{p}_{t-1}$ of $\tilde{\mathbb{P}}_{t-1}$ can be expressed as $\tilde{p}_{t-1}(x) = \int_\Theta p_\star(x|\theta_\star = \theta)p(\theta_\star = \theta|D_{t-1})\mathrm{d}\theta$ where $p_\star(x|\theta_\star = \theta)$ is the density of the maximiser of λ given $\theta_\star = \theta$ and $p(\theta_\star = \cdot|D_{t-1})$ is the posterior density of $\theta_\star$ conditoned on D_{t-1}. Note that $p_\star(x|\theta_\star = \theta)$ puts all its mass at the maximiser of $\lambda^+(\theta, D_{t-1}, x)$. Hence, X_t has the same distribution as X'_t; i.e. $\mathbb{P}_{t-1}(X'_t = \cdot) = \tilde{\mathbb{P}}_{t-1}(\cdot)$. This will form a key intuition in our analysis. To this end, we begin with a technical result, whose proof is adapted from Russo and Van Roy [213]. We will denote by $\mathrm{I}_{t-1}(A; B)$ the mutual information between two variables A, B under the posterior measure after having seen D_{t-1}; i.e. $\mathrm{I}_{t-1}(A; B) = \mathrm{KL}(\mathbb{P}_{t-1}(A, B)\|\mathbb{P}_{t-1}(A) \cdot \mathbb{P}_{t-1}(B))$.

Lemma 75. *Assume that we have collected a data sequence D_{t-1}. Let the action taken by $\pi^{\mathrm{PS}}_{\mathrm{M}}$ at time instant t with D_{t-1} be X_t and the action taken by $\pi^\star_{\mathrm{M}}$ be X'_t. Then,*

$$\begin{aligned}&\mathbb{E}_{t-1}[q_t(\theta_\star, X'_t, Y_{X'_t}) - q_t(\theta_\star, X_t, Y_{X_t})] = \\ &\qquad\qquad \sum_{x\in\mathcal{X}} \left(\mathbb{E}_{t-1}[q_t(\theta_\star, x, Y_x)|X'_t = x] - \mathbb{E}_{t-1}[q_t(\theta_\star, x, Y_x)]\right)\tilde{\mathbb{P}}_{t-1}(x) \\ &\mathrm{I}_{t-1}(X'_t; (X_t, Y_{X_t})) = \sum_{x_1, x_2\in\mathcal{X}} \mathrm{KL}(\mathbb{P}_{t-1}(Y_{x_1}|X'_t = x_2)\|\mathbb{P}_{t-1}(Y_{x_1}))\,\tilde{\mathbb{P}}_{t-1}(x_1)\tilde{\mathbb{P}}_{t-1}(x_2)\end{aligned}$$

Proof. The proof for both results uses the fact that $\mathbb{P}_{t-1}(X_t = x) = \mathbb{P}_{t-1}(X'_t = x) = \tilde{\mathbb{P}}_{t-1}(x)$.

For the first result,

$$\begin{aligned}
&\mathbb{E}_{t-1}[q_t(\theta_\star, X'_t, Y_{X'_t}) - q_t(\theta_\star, X_t, Y_{X_t})] \\
&= \sum_{x\in\mathcal{X}} \mathbb{P}_{t-1}(X'_t = x)\mathbb{E}_{t-1}[q_t(\theta_\star, X'_t, Y_{X'_t})|X'_t = x] - \\
&\qquad\qquad\qquad\qquad \sum_{x\in\mathcal{X}} \mathbb{P}_{t-1}(X_t = x)\mathbb{E}_{t-1}[q_t(\theta_\star, X_t, Y_{X_t})|X_t = x] \\
&= \sum_{x\in\mathcal{X}} \mathbb{P}_{t-1}(X'_t = x)\mathbb{E}_{t-1}[q_t(\theta_\star, x, Y_x)|X'_t = x] - \sum_{x\in\mathcal{X}} \mathbb{P}_{t-1}(X_t = x)\mathbb{E}_{t-1}[q_t(\theta_\star, x, Y_x)] \\
&= \sum_{x\in\mathcal{X}} \big(\mathbb{E}_{t-1}[q_t(\theta_\star, x, Y_x)|X'_t = x] - \mathbb{E}_{t-1}[q_t(\theta_\star, x, Y_x)]\big)\tilde{\mathbb{P}}_{t-1}(x)\,.
\end{aligned}$$

The second step uses that the observation Y_x does not depend on the fact that x may have been chosen by $\pi^{\mathrm{PS}}_{\mathrm{M}}$; this is because $\pi^{\mathrm{PS}}_{\mathrm{M}}$ makes its decisions based on past data D_{t-1} and is independent of $\theta_\star$ given D_{t-1}. Y_x however can depend on the fact that x may have been the action chosen by $\pi^{\star}_{\mathrm{M}}$ which knows $\theta_\star$. For the second result,

$$\begin{aligned}
&\mathrm{I}_{t-1}(X'_t; (X_t, Y_{X_t})) = \mathrm{I}_{t-1}(X'_t; X_t) + \mathrm{I}_{t-1}(X'_t; Y_{X_t}|X_t) = \mathrm{I}_{t-1}(X'_t; Y_{X_t}|X_t) \\
&\quad= \sum_{x_1\in\mathcal{X}} \mathbb{P}_{t-1}(X_t = x_1)\,\mathrm{I}_{t-1}(X'_t; Y_{X_t}|X_t = x) = \sum_{x_1\in\mathcal{X}} \tilde{\mathbb{P}}_{t-1}(x_1)\,\mathrm{I}_{t-1}(X'_t; Y_{x_1}) \\
&\quad= \sum_{x_1\in\mathcal{X}} \tilde{\mathbb{P}}_{t-1}(x_1) \sum_{x_2\in\mathcal{X}} \mathbb{P}_{t-1}(X'_t = x_2)\,\mathrm{KL}(\mathbb{P}_{t-1}(Y_{x_1}|X'_t = x_2)\|\mathbb{P}_{t-1}(Y_{x_1})) \\
&\quad= \sum_{x_1,x_2\in\mathcal{X}} \mathrm{KL}(\mathbb{P}_{t-1}(Y_{x_1}|X'_t = x_2)\|\mathbb{P}_{t-1}(Y_{x_1}))\,\tilde{\mathbb{P}}_{t-1}(x_1)\tilde{\mathbb{P}}_{t-1}(x_2)
\end{aligned}$$

The first step uses the chain rule for mutual information. The second step uses that X_t is chosen based on an external source of randomness and D_{t-1}; therefore, it is independent of $\theta_\star$ and hence X'_t given D_{t-1}. The fourth step uses that Y_{x_1} is independent of X_t. The fifth step uses Lemma 9. □

We are now ready to prove theorem 71.

Proof of Theorem 71: Using the first result of Lemma 75, we have,

$$\begin{aligned}
&\mathbb{E}_{t-1}[q_t(\theta_\star, X'_t, Y_{X'_t}) - q_t(\theta_\star, X_t, Y_{X_t})]^2 \\
&\quad= \left(\sum_{x\in\mathcal{X}} \tilde{\mathbb{P}}_{t-1}(x)\big(\mathbb{E}_{t-1}[q_t(\theta_\star, x, Y_x)|X'_t = x] - \mathbb{E}_{t-1}[q_t(\theta_\star, x, Y_x)]\big)\right)^2 \\
&\quad\overset{(a)}{\leq} |\mathcal{X}| \sum_{x\in\mathcal{X}} \tilde{\mathbb{P}}_{t-1}(x)^2\big(\mathbb{E}_{t-1}[q_t(\theta_\star, x, Y_x)|X'_t = x] - \mathbb{E}_{t-1}[q_t(\theta_\star, x, Y_x)]\big)^2 \\
&\quad\overset{(b)}{\leq} |\mathcal{X}| \sum_{x_1,x_2\in\mathcal{X}} \tilde{\mathbb{P}}_{t-1}(x_1)\tilde{\mathbb{P}}_{t-1}(x_2)\big(\mathbb{E}_{t-1}[q_t(\theta_\star, x_1, Y_{x_1})] - \mathbb{E}_{t-1}[q_t(\theta_\star, x_1, Y_{x_1})|X'_t = x_2]\big)^2
\end{aligned}$$

$$\overset{(c)}{\leq} |\mathcal{X}| \sum_{x_1,x_2\in\mathcal{X}} \tilde{\mathbb{P}}_{t-1}(x_1)\tilde{\mathbb{P}}_{t-1}(x_2)\mathbb{E}_{Y_{x_1}}\Big[\big(\mathbb{E}_{t-1}[q_t(\theta_\star,x_1,y)|Y_{x_1}=y] - \tag{8.13}$$

$$\mathbb{E}_{t-1}[q_t(\theta_\star,x_1,y)|X'_t=x_2,Y_{x_1}=y]\big)^2\Big]$$

$$\overset{(d)}{\leq} \frac{|\mathcal{X}|}{2} \sum_{x_1,x_2\in\mathcal{X}} \tau_{n-t}\tilde{\mathbb{P}}_{t-1}(x_1)\tilde{\mathbb{P}}_{t-1}(x_2) \times$$

$$\mathbb{E}_{Y_{x_1}}\big[\mathrm{KL}(\mathbb{P}_{t-1}(Y_{x_1}|X'_t=x_2,Y_{x_1}=y)\|\mathbb{P}_{t-1}(Y_{x_1}|Y_{x_1}=y))\big]$$

$$\overset{(e)}{\leq} \frac{|\mathcal{X}|}{2} \sum_{x_1,x_2\in\mathcal{X}} \tau_{n-t}\tilde{\mathbb{P}}_{t-1}(x_1)\tilde{\mathbb{P}}_{t-1}(x_2)\mathrm{KL}(\mathbb{P}_{t-1}(Y_{x_1}|X'_t=x_2)\|\mathbb{P}_{t-1}(Y_{x_1}))$$

$$\overset{(f)}{=} \frac{1}{2}|\mathcal{X}|\tau_n \mathrm{I}_{t-1}(X'_t;(X_t,Y_{X_t})) \overset{(g)}{\leq} \frac{1}{2}|\mathcal{X}|\tau_n \mathrm{I}_{t-1}(\theta_\star;(X_t,Y_{X_t}))$$

Here, step (a) uses the Cauchy-Schwarz inequality and step (b) uses the fact that the previous line can be viewed as the diagonal terms in a sum over x_1, x_2. Step (c) conditions on $Y_{x_1} = y$ and applies Jensen's inequality. Step (e) uses the definition of conditional KL divergence. Step (f) uses the second result of Lemma 75, and step (g) uses Lemma 10 and the fact that X'_t is a deterministic function of $\theta_\star$ given D_{t-1}. For step (d), we use the version of Pinsker's inequality given in Lemma 8 in conjunction with Condition 70. Precisely, we let H in Condition 70 to be $D_{t-1} \uplus \{(x,y)\}$. Now using (8.12) and (8.9), and the fact that $\pi^\star_{\mathrm{M}}$ is deterministic, we can write,

$$q_t(\theta_1,x,y) - q_t(\theta_2,x,y) = \lambda(\theta_1, D_{t-1}\uplus\{(x,y)\}) - \lambda(\theta_2, D_{t-1}\uplus\{(x,y)\}) +$$

$$\sum_{j=1}^{n}\Big(\mathbb{E}_{Y,t+1:n|\theta_1}\big[\lambda(\theta_1, D_{t-1}\uplus\{(x,y)\}\uplus F_{j,1})\big] -$$

$$\mathbb{E}_{Y,t+1:n|\theta_2}\big[\lambda(\theta_2, D_{t-1}\uplus\{(x,y)\}\uplus F_{j,2})\big]\Big)$$

$$\leq 1 + \sum_{j=1}^{n-t}\epsilon_j \leq \sqrt{\tau_{n-t}}.$$

Here, $F_{n,i}$ is the data collected by $\pi^\star_{\mathrm{M}}$ when $\theta_\star = \theta_i$, having observed H, and $F_{j,i}$ is its prefix of length j. The last step uses Condtion 70. Hence, by Lemma 8, the term with the squared paranthesis in (8.13) can be bounded by $\tau_{n-t}\mathrm{KL}(\mathbb{P}_{t-1}(Y_{x_1}|X'_t=x_2)\|\mathbb{P}_{t-1}(Y_{x_1}))$.

Now, using (8.11) and the Cauchy-Schwarz inequality we have,

$$\mathbb{E}[J_n(\theta_\star,\pi^\star_{\mathrm{M}}) - J_n(\theta_\star,\pi^{\mathrm{PS}}_{\mathrm{M}})]^2 \leq n\sum_{t=1}^{n}\frac{1}{2}|\mathcal{X}|\tau_n\mathrm{I}_{t-1}(\theta_\star;(X_t,Y_{X_t})) = \frac{1}{2}|\mathcal{X}|\tau_n\mathrm{I}(\theta_\star;D_n)$$

Here the last step uses the chain rule of mutual information in the following form,

$$\sum_t \mathrm{I}_{t-1}(\theta_\star;(X_t,Y_{X_t})) = \sum_t \mathrm{I}(\theta_\star;(X_t,Y_{X_t})|\{(X_j,Y_{X_j})\}_{j=1}^{t-1}) = \mathrm{I}(\theta_\star;\{(X_j,Y_{X_j})\}_{j=1}^{n}).$$

The claim follows from the observation, $\mathrm{I}(\theta_\star;D_n) \leq \Psi_n$. □

Proof of Theorem 73

In this section, we will let $D_m^{\star\star}$ be the data collected $\pi_{\mathrm{G}}^\star$ in m steps and $D_n^\star$ be the data collected by $\pi_{\mathrm{M}}^\star$ in n steps. We will use the following result on adaptive submodular maximisation from [74].

Lemma 76. *(Theorem 38 in Golovin and Krause [74], modified) Under condition 72, we have for all $\theta_\star \in \Theta$,*

$$\mathbb{E}_Y[\lambda(\theta_\star, D_n^\star)] \geq (1 - e^{-n/m})\mathbb{E}_Y[\lambda(\theta_\star, D_m^{\star\star})]$$

Lemma 10 controls the approximation error when we approximate the globally optimal policy which knows $\theta_\star$ with the myopic policy which knows $\theta_\star$. Our proof of theorem 73, combines the above result with Theorem 71, to show that MPS can approximate $\pi_{\mathrm{G}}^\star$ under suitable conditions.

Proof of Theorem 73. Let D_n be the data collected by $\pi_{\mathrm{M}}^{\mathrm{PS}}$. By monotonicity of λ, and the fact that the maximum is larger than the average we have $\mathbb{E}[\lambda(\theta_\star, D_n)] \geq \frac{1}{n}\sum_{t=1}^n \mathbb{E}[\lambda(\theta_\star, D_t)] = \frac{1}{n}\mathbb{E}[\Lambda(\theta_\star, D_n)]$. Using theorem 71 the following holds for all m,

$$\begin{aligned}
\mathbb{E}[\lambda(\theta_\star, D_n)] &\geq \frac{1}{n}\left(\mathbb{E}\left[\Lambda(\theta_\star, D_n^\star)\right] - \sqrt{\frac{|\mathcal{X}|\tau_n n \Psi_n}{2}}\right) \\
&= \frac{1}{n}\sum_{t=1}^n \mathbb{E}_{\theta_\star}[\mathbb{E}_Y[\lambda(\theta_\star, D_t^\star)]] - \sqrt{\frac{|\mathcal{X}|\tau_n \Psi_n}{2n}} \\
&\geq \mathbb{E}[\lambda(\theta_\star, D_m^{\star\star})]\frac{1}{n}\sum_{t=1}^n (1 - e^{-t/m}) - \sqrt{\frac{|\mathcal{X}|\tau_n \Psi_n}{2n}} \\
&\geq \mathbb{E}[\lambda(\theta_\star, D_m^{\star\star})](1 - \frac{m}{n}e^{-1/m} - \frac{1}{n}e^{-1/m}) - \sqrt{\frac{|\mathcal{X}|\tau_n \Psi_n}{2n}}.
\end{aligned}$$

Here, the first step uses Theorem 71, the second step rearranges the expectations noting that λ takes the expectation over the observations. The third step uses Lemma 76 for each t. The last step bounds the sum by an integral as follows,

$$\sum_{t=1}^n e^{-t/m} \leq e^{-1/m} + \int_1^\infty e^{-t/m}\mathrm{d}t \leq e^{-1/m} + me^{-1/m}.$$

The result follows by using $m = \gamma n$. □

8.3.3 On Conditions 70 and 72

The following proposition shows that when the myopic policy has value 1, and achieves this at a fast enough rate, for all values of θ, we satisfy Condition 70. For this, let $\theta, \theta', \pi_{\mathrm{M}}^\theta, \pi_{\mathrm{M}}^{\theta'}, D_n, D'_n, \mathbb{E}_{Y,t+1:}$ be as defined in Condition 70.

Proposition 77. *($\pi^\star_{\mathrm{M}}$ has value 1). Let π^θ_{M} denote the myopically optimal policy when $\theta_\star = \theta$. Assume there exists a sequence $\{\epsilon'_n\}_{n\geq 1}$ such that,*

$$\sup_{\theta\in\Theta}\sup_{H\in\mathcal{D}}\left(1-\mathbb{E}_{Y,|H|+1}[\lambda(\theta, H\uplus D_n)]\right)\leq \epsilon'_n.$$

Then, Condition 70 is satisfied with $\epsilon_n = \epsilon'_n$.

Proof. Let $H\in\mathcal{D}$ and $\theta,\theta'\in\Theta$. Then,

$$\begin{aligned}&\mathbb{E}_{Y,|H|+1|\theta}\lambda(\theta, H\uplus D_n)-\mathbb{E}_{Y,|H|+1|\theta'}\lambda(\theta', H\uplus D'_n)\\ &\quad=\left(\mathbb{E}_{Y,|H|+1|\theta}\lambda(\theta, H\uplus D_n)-1\right)+\left(1-\mathbb{E}_{Y,|H|+1|\theta'}\lambda(\theta', H\uplus D'_n)\right)\ \leq\ \epsilon'_n,\end{aligned}$$

since the first term is always negative. □

We next show two examples of DOE problems where the condition in Proposition 77 is satisfied.

Bandits & Bayesian Optimisation

In both settings, the parameter $\theta_\star$ specifies a function $f_{\theta_\star}:\mathcal{X}\to\mathbb{R}$. When we choose a point $X\in\mathcal{X}$ to evaluate the function, we observe $Y_X = f_{\theta_\star}(X)+\epsilon$ where $\mathbb{E}[\epsilon]=0$. In the bandit framework, we can define the reward to be $\lambda(\theta_\star, D_n) = 1+f_{\theta_\star}(X_n)-\max_{x\in\mathcal{X}} f_{\theta_\star}(x)$ which is equivalent to maximising the instantaneous reward. In Bayesian optimisation, one is interested in simply finding a single value close to the optimum and hence $\lambda(\theta_\star, D_n) = 1+\max_{t\leq n} f_{\theta_\star}(X_t)-\max_{x\in\mathcal{X}} f_{\theta_\star}(x)$.

In both cases, since $\pi^\star_{\mathrm{M}}$ knows it will always choose $\operatorname{argmax}_{x\in\mathcal{X}} f_{\theta_\star}(x)$ achieving reward 1. Thus Proposition 77 is satisfied with $\epsilon_n = 0$ and $\tau_n = 1$.

An Active Learning Example

We describe an active learning task on a Bayesian linear regression problem, and outline how it can be formulated to satisfy Condition 70. In this example, our parameter space is $\Theta = \{\theta = (\beta,\eta^2)|\beta\in\mathbb{R}^k, \eta^2\in[a,b]\}$ for some positive numbers $b>a>0$. We will assume the following prior on $\theta_\star = (\beta_\star,\eta^2_\star)$,

$$\beta_\star\sim\mathcal{N}(\mathbf{0}_k, \mathrm{P}_0^{-1}),\quad \eta^2_\star\sim\mathrm{Unif}(a,b),$$

where $\mathrm{P}_0\in\mathbb{R}^{k\times k}$ is the non-singular precision matrix of the Gaussian prior for $\beta_\star$. Our domain $\mathcal{X} = \{x\in\mathbb{R}^k; \|x\|_2\leq 1\}$ is the unit ball in $\mathbb{R}^k$ and $\mathcal{Y}=\mathbb{R}$. When we query the model at $x\in\mathcal{X}$, we observe $Y_x = \beta^\top x+\epsilon$ where $\epsilon\sim\mathcal{N}(0,\eta^2)$. Our goal in DOE is to choose a sequence of experiments $\{X_t\}_t\subset\mathcal{X}$ so as to estimate β well.

Given a dataset $D_n = \{(x_j,y_j)\}_{j=1}^n$, a natural quantity to characterise how well we have estimated $\beta_\star$ in the Bayesian setting is via the entropy of the posterior for β. This ensures that the data is sampled also considering the uncertainty in the prior. For example, if the prior

covariance is small along certain directions, an active learning agent is incentivised to collect data so as to minimise the variance along other directions. Specifically, in this example, we wish to minimise $\mathrm{Ent}(\beta_\star | D_n = D_n, \eta_\star^2 = \eta_\star^2)$, the entropy of $\beta_\star$ assuming we have collected data D_n and the true $\eta_\star^2$ value were to be revealed at the end. It is straightforward to see that, $\mathbb{P}(\beta_\star | \eta_\star^2, D_n) = \mathcal{N}(\mu_n, \mathrm{P}_n^{-1})$, where,

$$\mathrm{P}_n = \mathrm{P}_0 + \frac{1}{\eta_\star^2} \sum_{j=1}^{n} x_j x_j^\top, \qquad \mu_n = \mathrm{P}_n \sum_{j=1}^{n} y_j x_j.$$

The entropy of this posterior is

$$\mathrm{Ent}(\beta_\star | D_n = D_n, \eta_\star^2 = \eta_\star^2) \; = \frac{1}{2} \log \det(2\pi e \mathrm{P}_n^{-1}) \; = \frac{k}{2} \log(2\pi e) - \frac{1}{2} \log \det \mathrm{P}_n.$$

Minimising the posterior entropy can be equivalently formulated as maximising the following reward function,

$$\lambda(\theta_\star, D_n) = 1 - \frac{1}{\det \mathrm{P}_n} = 1 - \frac{1}{\det \left(\mathrm{P}_0 + \frac{1}{\eta_\star^2} \sum_{j=1}^{n} x_j x_j^\top \right)}. \tag{8.14}$$

The reward depends on $\theta_\star$ due to the $\eta_\star^2$ term, and an adaptive policy can be expected to do better than a non-adaptive one since the observations $\{y_j\}_{j=1}^n$ can inform us about the true value of $\eta_\star^2$.

Note that since $\lambda(\theta_\star, D_n)$ is a multi-set function, D_n can be viewed as a (non-ordered) mulit-set and the $\uplus$ operator is simply the union operator. We will now demonstrate that λ satisfies the two conditions set out in Chapter 8.1.3.

Condition 70: We will show that it satisfies the condition in Proposition 77. Let c be the smallest eigenvalue of P_0. For a given data set $H = \{(x_j, y_j)\}_{j=1}^m$ of size m, denote $\mathrm{P}_0^H = \mathrm{P}_0 + \frac{1}{\eta_\star^2} \sum_{j=1}^{m} x_j x_j^\top$. Moreover, assume that the points chosen by $\pi_{\mathrm{M}}^\star$ in $\mathcal{X}$ are $z_1, z_2, \ldots$. Note that this is a deterministic sequence since $\pi_{\mathrm{M}}^\star$ knows $\eta_\star^2$ and the reward does not depend on the observations.

Let $\mathrm{P}_n^H = \mathrm{P}_0^H + \frac{1}{\eta_\star^2} \sum_{i=1}^{n} z_j z_j^\top$ and denote its eigenvalues by $\sigma_1 > \sigma_2 > \cdots > \sigma_k$. Note that since the myopic policy chooses actions to maximise the reward at the next step, it will choose $z_{n+1} = \mathrm{argmax}_{\|z\|=1} \det(\mathrm{P}_n^H + \frac{1}{\eta_\star^2} z z^\top)$. We therefore have,

$$\det \mathrm{P}_{n+1}^H = \max_{\|z\|=1} \det \left(\mathrm{P}_n^H + \frac{1}{\eta_\star^2} z z^\top \right) \geq \left(\sigma_1 + \frac{1}{\eta_\star^2} \right) \prod_{j=2}^{k} \sigma_j$$

Noting that $\mathrm{P}_0^H - cI_k$ is positive definite, we have, via an inductive argument $\det \mathrm{P}_n^H \geq c^{k-1}(c + n\eta_\star^{-2})$. Letting $D_n^\star$ be the data collected by $\pi_{\mathrm{M}}^\star$, we have

$$1 - \lambda(\theta_\star, D_n^\star) \leq \frac{1}{c^{k-1}(c + nb)} \triangleq \epsilon_n',$$

as $\eta_\star^2 \leq b$. This leads to $\epsilon_n', \epsilon_n \in \mathcal{O}(1/n)$ and hence $\tau_n \in \mathcal{O}(\log n)$ in Proposition 77 and Condition 70. We next look at the adaptive submodularity condition.

Condition 72 (Adaptive Submodularity): Let $D_n = \{(x_j, y_j)\}_{j=1}^n$ $D_m = \{(x_j, y_j)\}_{j=1}^m$ be two data sets such that $D_m \subset D_n$ and $m < n$. Let $Q_m = \mathrm{P}_0 + \frac{1}{\eta_\star^2}\sum_{j=1}^n x_j x_j^\top$ and $Q_n = \mathrm{P}_0 + \frac{1}{\eta_\star^2}\sum_{j=1}^m x_j x_j^\top = Q_m + \frac{1}{\eta_\star^2}\sum_{j=m+1}^n x_j x_j^\top$. Let (x, Y_x) be a new observation. We then have,

$$\begin{aligned}\mathbb{E}[\lambda(\theta_\star, D_n \uplus \{(x, Y_x)\})] - \lambda(\theta_\star, D_n) &= \frac{1}{\det(Q_n)} - \frac{1}{\det(Q_n + xx^\top)} \\ &= \frac{\det(Q_n + xx^\top) - \det(Q_n)}{\det(Q_n)\det(Q_n + xx^\top)} = \frac{1 + x^\top Q_n^{-1} x}{\det(Q_n + xx^\top)},\end{aligned}$$

and similarly for Q_m. Here the last step uses the identity $\det(A + uv^\top) = \det(A)(1 + v^\top A^{-1} u)$. Submodularity follows by observing that Q_m, Q_n are positive definite and $Q_n - Q_m$ is positive semidefinite. Hence,

$$\frac{1 + x^\top Q_m^{-1} x}{\det(Q_m + xx^\top)} \geq \frac{1 + x^\top Q_n^{-1} x}{\det(Q_n + xx^\top)}.$$

Rewards with State-like structure

Here, we will show that $\pi_{\mathrm{M}}^{\mathrm{PS}}$ can achieve sublinear regret with respect to $\pi_{\mathrm{M}}^\star$, when there is additional structure in the rewards. In particular, we will assume that there exists a set of "states" $\mathcal{S}$ and a mapping $\sigma : \Theta \times \mathcal{D} \to \mathcal{S}$ from parameter, data sequence pairs to states. Moreover, λ takes the form $\lambda(\theta_\star, D) = \lambda_S(\theta_\star, \sigma(\theta_\star, D))$ for some known function $\lambda_S : \Theta \times \mathcal{S} \to [0, 1]$. We will also assume that the state transitions are Markovian, in that for any $S \in \mathcal{S}$, let $D_S = \{D \in \mathcal{D} : \sigma(\theta_\star, D) = S\}$. Then, for all $x \in \mathcal{X}$, $y \in \mathcal{Y}$ and $D, D' \in D_S$, $\sigma(\theta_\star, D \cup \{(x, y)\}) = \sigma(\theta_\star, D' \cup \{(x, y)\})$.

Now, for any policy π, define,

$$\begin{aligned}V_n(\pi, D; \theta) &= \frac{1}{n}\mathbb{E}\left[\sum_{j=1}^n \lambda(\theta, D \uplus D_j) \;\middle|\; \theta_\star = \theta, D, D_n \sim \pi\right] \\ V(\pi, D; \theta) &= \lim_{n\to\infty} V_n(\pi, D; \theta)\end{aligned}$$

V_n is the expected sum of future rewards in n steps for a policy π when $\theta_\star = \theta$, and it starts from a prefix D. The expectation is over the observations and any randomness in π. V is the limit of V_n. A common condition used in reinforcement learning is that the associated Markov chain mixes when starting from any state $S \in \mathcal{S}$. Under this condition, V does not depend on the prefix D and we will simply denote it by $V(\pi; \theta)$. We have the following result.

Proposition 78. *Assume that there exists a sequence $\{\nu_n\}_{n\geq 1}$, such that $\nu_n \in o(1/\sqrt{n})$, and the following two statements are true.*

1. *$V(\pi_{\mathrm{M}}^\theta; \theta) = V(\pi_{\mathrm{M}}^{\theta'}; \theta')$ for all $\theta, \theta' \in \Theta$.*
2. *For all θ, and all data sequences H, H', $|V_n(\pi_{\mathrm{M}}^\theta, H; \theta) - V(\pi_{\mathrm{M}}^\theta; \theta)| \leq \nu_n$.*

Then Theorem 71 holds with $\sqrt{\tau_n} = 1 + 2n\nu_n$.

The second condition is similar to the requirements in Definition 5 in [137]. While they only use a thresholding behaviour, we assume a uniform rate of convergence, where our bounds depend on this rate. However, while results for non-episodic RL settings are given in terms of the mixing characteristics of the globally optimal policy, our results are in terms of the myopic policy.

Proof of Proposition 78. We will turn to our proof of Theorem 71, where we need to bound $q_t(\theta_1, x, y) - q_t(\theta_2, x, y)$. We will use Proposition 78 with $H = D_{t-1} \uplus \{(x, y)\}$ and have,

$$\begin{aligned}
&q_t(\theta_1, x, y) - q_t(\theta_2, x, y) \\
&\quad = \lambda(\theta_1, D_{t-1} \uplus \{(x, y)\}) - \lambda(\theta_2, D_{t-1} \uplus \{(x, y)\}) + \\
&\qquad\qquad \sum_{j=1}^{n} \Big(\mathbb{E}_{Y,t+1:n|\theta_1} \left[\lambda(\theta_1, D_{t-1} \uplus \{(x, y)\} \uplus F_{j,1}) \right] - \\
&\qquad\qquad\qquad\qquad \mathbb{E}_{Y,t+1:n|\theta_2} \left[\lambda(\theta_2, D_{t-1} \uplus \{(x, y)\} \uplus F_{j,2}) \right] \Big) \\
&\quad \leq 1 + (n-t) \left(V_n(\pi^{\theta}_{\mathrm{M}}, D_{t-1} \uplus \{(x, y)\}; \theta) - V_{n-t}(\pi^{\theta'}_{\mathrm{M}}, D_{t-1} \uplus \{(x, y)\}; \theta') \right) \\
&\quad \leq 1 + (n-t) \Big(|V_{n-t}(\pi^{\theta}_{\mathrm{M}}, D_{t-1} \uplus \{(x, y)\}; \theta) - V(\pi^{\theta}_{\mathrm{M}}; \theta')| + \\
&\qquad\qquad |V_{n-t}(\pi^{\theta'}_{\mathrm{M}}, D_{t-1} \uplus \{(x, y)\}; \theta') - V(\pi^{\theta'}_{\mathrm{M}}; \theta')| \Big) \\
&\quad \leq 1 + 2(n-t)\nu_{n-t} = \sqrt{\tau_{n-1}}
\end{aligned}$$

Here, the second step uses that λ is bounded in $[0, 1]$, the third step simply uses the first condition in Proposition 78 along with the triangle inequality, and the fourth step uses the second condition. The remainder of the proof carries through by applying Pinksker's inequality with this bound in (8.13). □

Conditions of the above form are necessary in non-episodic undiscounted settings for RL [137], and we show that under similar conditions, $\pi^{\mathrm{PS}}_{\mathrm{M}}$ achieves sublinear regret with $\pi^{\star}_{\mathrm{M}}$.

Chapter 9

Conclusion

9.1 Summary

We studied bandits and other settings for sequential decision making under uncertainty in stateless environments. Many tasks in science, engineering, and finance can be framed as such problems. With a few exceptions, we operated primarily in the Bayesian paradigm, which uses introspective Bayesian models to ingest outcomes of past experiments and plan future experiments. The goal of this thesis was to study techniques to address the challenges and exploit new opportunities when scaling up such decision making problems to large scale settings.

On the bandit side, our work on additive models for Bayesian optimisation scales gracefully with the number of dimensions when compared to naive techniques. Multi-fidelity bandits allow us to leverage cheap approximations of f to optimise f efficiently; it uses the cheap approximations to discard poor regions of the domain and deploys the expensive evaluations only in a promising region. Our methods for parallelising bandits allow us to determine prudent locations to manage exploration and exploitation when conducting multiple experiments at the same time; computationally, they scale gracefully with the amount of parallelism while statistically, they perform almost as well as if the evaluations were done in sequence. NASBOT allows us to optimise functions defined on combinatorially structured spaces. A pertinent application for this framework is neural architecture search, which optimises the architecture of a neural network for cross validation error on a given problem. Our open source software platform, Dragonfly, integrates the above techniques into a scalable and robust framework for Bandit optimisation.

We then took a step back from the bandit setting and studied a more general setting for stateless decision making. The MPS framework allows a practitioner to specify the goal of experimentation via a reward function. This finds applications in many problems where the goals of experimentation can be very application-specific. Moreover, it provides an intuitive mechanism to incorporate domain expertise which is useful in applications where experiments are very expensive and we need to rely on prior knowledge to reduce the sample complexity. Theoretically, we see that it is competitive with globally optimal policies that know system characteristics under natural regularity conditions.

In almost all cases above, our methods come with theoretical underpinnings. Moreover, empirically they outperform vanilla bandit techniques and other baselines for the setting.

Next, we discuss some interesting questions for future research arising out of this thesis. We first discuss short term questions that arise from the limitations in our work in individual chapters. Then, we discuss applications where our work on scalable decision making has potential for high impact. Finally, we conclude with thoughts and open questions on the topic of scalability in decision making under uncertainty.

9.2 Future Work on Individual Chapters

High Dimensional Bandits

In its current form, Add-GP-UCB comes without any theoretical guarantees. While it performs demonstrably better empirically due to reasons discussed in Chapter 3, theoretically identifying conditions under which it has sublinear regret is an open problem. One could also study other additive models for high dimensional bandits (e.g. [54, 117]) and more generally other structural assumptions on f to statistically and computationally simplify high dimensional bandits.

Multi-fidelity Bandits

On the K-armed multi-fidelity bandit, an immediate question arising out of our work is to close the gap between the upper and lower bounds on the regret. Another important question is to study different notions of cumulative regret for the multi-fidelity setting. For example, consider a mining robot where each high fidelity play is a real world experiment of the robot and incurs cost $\lambda^{(2)}$. However, a vastly cheaper computer simulation which incurs cost $\lambda^{(1)}$ may approximate a robot's real world behaviour. In such applications, $\lambda^{(1)} \ll \lambda^{(2)}$. However, lower fidelity plays do not have any reward, since they are just simulations. As a different example, consider clinical trials, where the regret due to a bad treatment at the highest fidelity would be catastrophic. However, a bad treatment at a lower fidelity may not warrant a large penalty.

In the Gaussian process settings, we wish to extend our theoretical results to more general settings. For instance, in BOCA, we believe a stronger bound on the regret might be possible if $\phi_{\mathcal{Z}}$ is a finite dimensional kernel. However, since finite dimensional kernels are typically not radial [236], our analysis techniques will not carry over straightforwardly.

Another line of work, applicable to all settings above is to develop multi-fidelity methods with theoretical guarantees for other acquisitions such as Thompson sampling or expected improvement. Currently, Dragonfly uses the BOCA/MF-GP-UCB strategy of first using any given acquisition to determine the next evaluation point, and then determining the fidelity by studying the variance across fidelity space at the chosen domain point. However, this does not come with theoretical guarantees.

Parallel Bandits

One of the open theoretical questions in our work is to bound the regret for asyTS without the initialisation procedure. Second, we are interested in other models for evaluation times, such as to capture correlations between the evaluation time and the query point $x_j \in \mathcal{X}$ that arise in practice. For example, in neural architecture search, the time to evaluate a network depends on the size of the network. We believe that in such problems, even sequential algorithms might be different from the conventional methods. One could also consider models where some workers are slower than the rest. Third, in the asynchronous setting, there might be instances where the algorithm might choose to kill an evaluation in progress based on the result of a completed job. The algorithm might also choose to wait for another evaluation to finish without immediately deploying a free worker, so as to incorporate additional information.

Neural Architecture Search & Bandits on Graphs

In its current form, many of the parameters in the OTMANN distance are assumed fixed. Using GP marginal likelihood based methods to learn these parameters as part of the optimisation procedure will improve the performance of NASBOT when we have the computational budget to evaluate many networks. Secondly, the current trend in architecture search, which has yielded state of the art results, is to constrain the search space of architectures to cells which are repeated throughout the network. While NASBOT's search space, in its current form is very general, it would be useful to see how well it performs in such constrained spaces.

At a higher level, an interesting question for future work is to extend the ideas in OTMANN and NASBOT to optimisation of other graph-structured objects, such as drugs, chemical molecules, crystal structures, and social networks. Developing theoretical guarantees for bandit methods on such complex domains is also an intriguing line for future work.

Dragonfly

Many of the techniques employed to make Bayesian optimisation robust in Dragonfly, especially those based on ensemble techniques, come without theoretical underpinnings. More generally, in the author's opinion, developing robust methods for selecting the GP hyperparameters is an open problem in Bayesian optimisation [263].

One of the main challenges for the Bayesian optimisation community is that Gaussian processes are not scalable for a very large number of experiments. In many practical problems, this is not a serious bottleneck since evaluations can be quite expensive – hence, we usually cannot afford a large number of experiments. However, we are fast reaching the point where we might conceivably run several tens of thousands of experiments, at least in some applications. For instance, with multi-fidelity methods we can carry out a large number of cheap approximations. Moreover, with increasingly better computational capabilities, experiments that were historically very expensive, can today be run significantly faster (e.g. [115, 227]). While other Bayesian

models for the reward function have been explored [103, 233], GPs still remain the most effective and popular method for BO. Combining BO with methods for scaling up GPs [231, 268, 269] will enable the application of such methods to very large scale problems.

Going forward, we wish to continue development of Dragonfly, and integrate many other features for scalable Bayesian optimisation, including those discussed above.

Bayesian Design of Experiments

A natural question arising out of our work is to study real life DOE tasks where conditions 70 and 72 hold and other conditions under which a myopic strategy such as MPS will work. Moreover, similar to how MPS is inspired by Thompson sampling for bandits, it is worth asking if other policies are possible, say inspired by upper confidence bound strategies or information theoretic criteria [44, 118, 119, 145, 146]. It will also be interesting to study computationally efficient methods for fully optimal policies, and those with k-step lookahead, which interpolate between myopic policies and fully optimal ones. Analysing the trade-off of computation and optimality for such k-step policies will also be an interesting question for theoretical study.

Finally, we should also mention that Chapter 8 only studies one class of problems for stateless decision making, and aims to provide a general algorithm for this class. Studying if more efficient algorithms can be developed for a subset of this class and developing new methods for other classes of problems are both intriguing directions for future work.

9.3 Impactful Application Domains for Bandits & Stateless Decision Making

We now discuss some potential applications for bandits and other stateless decision making techniques. The thoughts here stem primarily from my understanding of the technological feasibility and opportunity in the respective domains and the potential for commercial and social benefit. The first three applications below have been classical motivations for studying bandit and design of experiments formalisms. However, these methods have not seen widespread adoption due to the large disconnect between theoretical innovations and practical challenges. In the last two applications, bandit methods have been previously unexplored but where we believe they could have significant impact in the modern economy.

For each application, I outline some unique challenges arising, elaborate how some of them can be addressed by the work in this thesis, and discuss open questions that need to be solved to make automated decision making technology feasible and realistic in said domain.

Materials Design

Materials design is a blanket term used to describe the process of developing new materials for new applications. Traditionally, these tasks were primarily performed by experts who manually tested out different designs and chose one that satisfied the required criteria the most. This exercise is expensive for materials companies since it requires a materials scientist to use their intuition and expertise in selecting the next design. Moreover, it is laborious for said materials scientists as it requires digesting and reasoning with large high dimensional datasets. However, recently, there has been a surge of interest in computational and data-driven methods in materials science. This is in part driven by the rising popularity of machine learning as a field, and in part, by the above challenges in the conventional modus operandi. In particular, many recent work have begun viewing the design and discovery of new materials for new applications as bandit and design of experiment tasks [85, 105, 193]. This includes the collaboration with the Scott Institute for Energy at CMU, highlighted in Chapter 7.4.4, on using bandit methods for electrolyte design.

That said, there are many unique challenges to applying such methods to materials design. For example, domains tend to be quite high dimensional. Moreover, the types of experiments vary from simple laboratory tests which take a few hours to real-world tests which can take several days. Our work in Chapters 3 and 4 on high dimensional and multi-fidelity decision making are relevant in such settings. Further, materials scientists are interested in a variety of tasks such as multi-objective optimisation and active learning and wish to incorporate domain expertise in their models – both of which are addressable by our work in Chapter 8 on design of experiments.

Drug Discovery

The average R&D cost for bringing a new drug to the market is estimated at more than $2 billion. Many pharmaceutical experts believe that the current approach to drug discovery, which predominantly relies on chemists to design new drugs, should change drastically to reduce this cost. One of the challenges in this space is that small molecules, which have been the traditional focus of the industry, are not easily modelled computationally. However, recently, there has been optimism in using biologics which are more amenable for computational methods.

From a design of experiments perspective, drug discovery shares many challenges to materials design, including high dimensional domains and a need to incorporate domain expertise. One notable difference though is the availability of high throughput screening infrastructure that can be used to conduct hundreds, if not thousands of experiments in parallel [102], where, work on parallel Bayesian optimisation in Chapter 5 is pertinent. One unsolved problem in this space is developing good representations for drug molecules, which can be viewed as 3D graph-like objects. There is some recent progress in this front (e.g. [76]), and we believe our ideas on OTMANN in Chapter 6 can be used here as well.

Online Tuning of Industrial Systems & Manufacturing Apparatuses

The performance of systems such as those found in industrial manufacturing facilities depend on crucial configuration parameters. A crucial challenge in optimising such systems is that they need to be tuned online, i.e. when the systems are running and being used for a real world downstream application. Hence, bandit algorithms should be careful in choosing parameters to control. For example, when optimising an industrial manufacturing apparatus [271], we should be careful in which configurations we try out as bad configurations can reduce production and result in losses. Similarly, consider tokamaks, which are devices using a magnetic field to confine hot plasma in order to generate thermonuclear fusion energy. The stability and the total energy generated by such devices are controlled by the power, energy fraction, and half energy fraction of each beam in the tokamak [64]. Choosing bad configurations can result in unstable systems and potentially lead to explosions. Hence, it is necessary to develop methods which can explore the configurations safely [14, 239].

Another potential challenge in such systems is that the decision making may need to be done very fast or in real-time. For example, when tuning tokamaks, new recommendations should be provided in under 250ms. Many bandit methods are not computationally efficient, usually requiring solving an NP-hard problem on each iteration. Hence, they can be quite slow especially in large and complex domains. Developing efficient bandit methods for such problems is an important and open unsolved problem.

Optimising Computing Infrastructure

Computing infrastructure has become increasingly complex over the last few years, with several hundred, if not thousands of configuration parameters. For example; data storage and processing tools can have hundreds of configuration knobs, modern JVMs have more than 700 tunable parameters, and other containers, OS kernels, and VMs have dozens of settings. Practitioners are interested in optimizing for several criteria such as latency, throughput, and cost when managing complex infrastructure systems, depending on the application at hand. Today, in institutions where performance of infrastructure systems are essential to the business, these configurations are manually tuned by performance engineers, such as DevOps or site reliability engineers, who sift through log files to study how different knob configurations affected the desired performance criteria. With increasingly complex systems and an explosion in the number of such systems deployed in an organization, this approach is not scalable any more.

Many of the methods developed in this thesis are relevant in this domain. For example, the domains tend to be high dimensional, many experiments can be executed in parallel, and these experiments can be approximated to various extents by running shorter lower fidelity versions of relevant jobs. However, it also ushers in new challenges. For example, the performance of these systems tend to depend on the hardware and the workload, whose characteristics may be entirely or partly unknown. This will require new methodological developments in contextual bandits [143]. Secondly, infrastructure systems are tuned in a staging environment and then

deployed in production settings. Combining bandit methods with transfer learning [189] is necessary to account for differences between staging and production.

Allocation of Computation in Modern Networks

Modern computing networks have become increasingly sophisticated, be it a small IoT network which collects data from mobile devices and/or home appliances (edge) and communicates them to a computer (core), or an enterprise network used for online retail which collects consumer data from various geographic locations (edge) and communicates them to a central server (core). An important problem that arises in such networks is the allocation of various computing tasks to different parts of the network so as to optimise desired performance criteria.

Performing computation at the core allows us to synchronise data coming from various sources and potentially leverage powerful computing resources. However, communicating data frequently from the edge will induce undue stress on the network's capacity. Moreover, due to network latency, the data at the core is necessarily stale, i.e. not up to date with the data available at the edge. In contrast, performing computation on edge devices will allow us to respond to changes in the environment in real time or very fast. However, heavy computation might not be possible and there might be limited availability of data from other nodes in the network.

Like many other applications we have seen above, not all characteristics of these systems can be modeled analytically, and hence we need to resort to repeated experimentaton to optimise these neworks. Moreover, similar to computing infrastructure systems, the performance of a given design depends on the performance requirements (service level objectives), the properties of the network, and the workload; some of this contextual information is observed, while some are not. Similar to the computing infrastructure use case, we will require new methodological innovations in contextual bandits and transfer learning. In addition, this will also require being able to optimise over complex and structured domains.

9.4 Open Problems & Future Research Directions

In this section, I discuss some open problems and other research directions that need to be solved so as to make data driven decision making systems more scalable and practical, both in stateless and stateful environments. This discussion is motivated by the following question,

What other gaps need to be filled so that automated decision making methods can be used pervasively in industrial, engineering, and scientific applications?

The thoughts here are both inspired and fashioned by my interactions with practitioners and domain experts from potential application domains for this technology, both in academia and industry. Some of the ideas here have already been alluded to in the previous section. To better follow the discussion, I would like to draw the attention of the reader to Table 1.1 in Chapter 1, which outlines various delineations in decision making.

Stateless Decision Making

The scalability topics in this thesis has primarily, albeit not exclusively, focused on the bandit setting. Many of these ideas will translate straightforwardly to general stateless decision making. For example, I expect that using additive models in high dimensional settings, and posterior sampling techniques for executing several experiments in parallel will apply with no or very minimal modifications to design of experiment problems. However, extending our multi-fidelity work will require more work; the intuition of proceeding to the higher fidelity once the lower fidelity uncertainty has shrunk sufficiently might work for problems such as active learning, but not in the specific way developed in Chapter 4. We hope that future research will flesh out these ideas in the coming years. Next, we take look at other topics not covered in this thesis.

Interpretability: One of the main practical challenges in widespread adoption of automated decision making systems is that many practitioners view these methods as "too black-box". Developing systems that can provide interpretable recommendations might go a long way in alleviating such concerns. For example, in addition to recommending a design for the next experiment, we can also provide reasons as to why said design was recommended (e.g. we expect it to be an optimum of the function or we expect to learn a certain aspect of the system).

Interaction with Practitioners: Developing autonomous systems that can interact with practitioners will allow AI systems to work synergistically with human experts to achieve the desired outcome. From a socio-economic perspective, it could also reduce the barrier to entry for adoption by domain experts who might view such technologies as an entity that might replace them as opposed to a tool that could be helpful to them. One setting for an expert-in-the-loop model would allow an expert to score the suggestion by an algorithm or compare multiple suggestions [275, 276]; the algorithm then re-evaluates whether or not it should still run the experiment. Another setting would allow an expert to either entirely pass the suggestion or modify it before running the experiment. A third setting would allow the expert to first suggest a design and have the algorithm critique it. Exploring these and other models for expert interaction would be interesting from a theoretical standpoint and have significant practical impact.

User-friendly and Robust Bayesian Inference Engines: Our work in Chapter 8 allows one to incorporate domain expertise in the form of custom Bayesian models. To leverage such methods, it is necessary to develop flexible tools to specify such models and perform posterior inference on them. The field of probabilistic programming (PP) aims to do precisely this [16, 31, 248]. While there have been many strides in scalable black-box Bayesian inference in the recent past [179, 180] and attempts to incorporate PP into Bayesian optimisation [182], this is still far from a solved problem. Using available PP tools requires a fair amount of machine learning expertise. Moreover, inference tends to become slow and brittle when the complexity of the model increases. This can be especially problematic in fields such as materials science, drug discovery, and computing infrastructure where models can be large and complex. This is further exacerbated by the fact that practitioners in their respective fields who use these tools may not be familiar with machine learning, Bayesian inference, and related topics. Developing robust and user-friendly inference engines that can be used by non-experts will facilitate widespread adoption of these methods.

Resource Management: Currently, we do not have good techniques for resource management when conducting experiments, especially when they are available in parallel. For instance, assume that we have a few parallel workers, but unlike the setting in Chapter 5, we are able to combine them to execute the same experiment sooner. As an example, assume one is given four GPUs to tune the parameters of a deep network. A straightforward solution is to treat this as a parallel bandit optimisation algorithm with four different workers, as we have done in Chapter 5. But suppose we had the option to combine all four GPUs to execute the same experiment and obtain the result twice as fast. We receive feedback sooner and are consequently able to make a better decision at the next instant. However, we lose throughput as we can now only complete two experiments in the time that it took to complete four. Such trade-offs are ubiquitous in several set ups where parallel computation is available, such as cosmological simulations in astrophysics and molecular dynamics or density functional theory calculations in materials science. Developing strategies that can account for this trade-off between throughput and information accumulation when making decisions is an important open problem.

Contextual Decision Making: Many real world problems fall naturally into the contextual bandit formalism, where we need to find the optimal design for different contexts. For example, in personalised advertising, the optimal ad for one user (context) will be different from another's since user preferences are different. While contextual bandits is fairly well studied in settings where this context is known [143, 147, 148], in many applications they are either unknown, partly known, or known only after the action has been made, such as the computing infrastructure optimisation example above. Studying such new formalisms for contextual bandits and extending our scalability techniques to the contextual case will be interesting avenues for future work.

Stateful Decision Making

The next step is of course extending many of these scalability ideas to the stateful setting. Reinforcement learning (RL) is presently a popular area of research [90, 126, 174, 226, 227, 240], but has primarily been shown to be most successful in settings where an abundance of experiments are possible. This limits in applicability in applications such as materials science and drug discovery where each experiment can be expensive. On the other hand, many problems in decision making have a notion of state where bandit methods are inadequate. Hence, it is necessary to develop RL methods that can work well with few samples, even on complex domains.

Many of the ideas in this thesis can be applied to various extents in the RL paradigm. For example, additive models and other similar simplifying structural assumptions on the feedback model can help reduce the sample complexity and multi-fidelity methods in RL can help leverage cheap approximations to an experiment (such as simulations) to speed up the optimisation process. We also believe that RL methods based on Thompson sampling (e.g. [185]) can be easily parallelised in a manner similar to our approach in Chapter 5 for the bandit case. Furthermore, many of the challenges outlined above for stateless decision making, including interpretability, resource management, and scalable models are challenges that need to be addressed.

Finally, we mention that stateful decision making paradigms exist outside of reinforcement learn-

ing. A pertinent example that might find applications in many industrial settings is that of system identification, where the goal is to learn how a black box system, such as a tokamak, responds to inputs at different states. To our best knowledge, besides some work studying very simple systems (e.g. where the output is a linear function of the input and the state [20]), there has been no work studying such problems in their fullest generality. We believe that our work in Chapter 8 provides a good starting point towards solving such stateful decision making systems.

9.5 Final Thoughts

I would like to conclude this thesis with my thoughts on the current state of decision making under uncertainty, and its readiness for real world applications. I make two high level take-aways based on the discussions above.

1. **More work needs to be done for stateless decision making, but it is ready-to-go for many applications.**
   Bandit methods have already been tested and validated on real world applications such as online advertising, content recommendations, and hyperparameter tuning. While they have not been widely adopted, recent developments for high dimensional optimisation, parallelisation, safe exploration, multi-fidelity methods, and contextual formalisms have enabled their application for problems in different domains today. In addition to bandits, the machine learning community has made significant progress in other stateless decision making systems, such as active learning, design of experiments, and posterior estimation. By engaging with different application domains, and applying these methods on real world problems, we can also learn about new problem formalisms for theoretical study.
2. **Stateful decision making is far from solved for the real world.**
   While there have been a few success stories for RL in the recent years, they have been in simulated environments where several hundred thousand or more than a million experiments have been possible. This severely limits their applicability in real world applications where experiments can be significantly more expensive. It is therefore necessary to develop methods for sample-efficient RL and/or methods for leveraging information from cheap approximations such as simulations. In addition, our progress in stateless decision making in implicit reward settings (see Table 1.1) is still immature.

Appendix A

Notation

This appendix summarises the notation used throughout this thesis. We first provide notation common across all chapters and then the notation specific to each chapter or sub-chapter(s). Note that some symbols may be overloaded, and differ from chapter to chapter.

General Notation

f	:	The payoff function to be maximised.
$\mathcal{X}$	:	The domain over which we are optimising f.
$x_\star$	:	The optimum point of f, $x_\star \in \operatorname{argmax}_{x\in\mathcal{X}} f(x)$.
$f(x_\star)$	:	The optimum value of f, $f(x_\star) = \max_{x\in\mathcal{X}} f(x)$.
$\mathcal{D}_t$	:	Unless otherwise specified, $\mathcal{D}_t$ is the set of all previous queries at time t consisting of query-feedback pairs, i.e. $\mathcal{D}_t = \{(x_i, y_i)\}_{i=1}^{t-1}$.
μ_{t-1}	:	The mean of the GP at time t conditioned on the previous queries.
σ_{t-1}	:	The standard deviation of the GP at time t conditioned on the previous queries.
β_t	:	Unless otherwise specified, the coefficient trading off exploration and exploitation in the GP-UCB. See Theorem 1 and 2. May be overloaded in other chapters.
x_t	:	The point in the domain $\mathcal{X}$ queried at time t.
φ_t	:	Typically, the acquisition at time t used to decide the point for evaluation x_t in Bayesian optimisation, i.e. $x_t = \operatorname{argmax}_{x\in\mathcal{X}} \varphi_t(x)$.
Ψ_n	:	The maximum information gain in a Gaussian process, See Definition 1.

δ	:	Typically, the failure probability for a theoretical statement. For example, in Theorem 1, the bound holds with probability $> 1 - \delta$.
$\overline{A}$	:	The complement of a set $A \subset \mathcal{X}$. $\overline{A} = \mathcal{X} \backslash A$.
$\lvert A \rvert$	:	The cardinality of a set $A \subset \mathcal{X}$ if it is countable.
$\vee, \wedge$	:	Logical *Or* and *And* respectively.
$\lesssim, \gtrsim, \asymp$	:	Inequalities and equality ignoring constant and $\mathrm{polylog}$ terms.
$\mathbb{P}, \mathbb{E}$	:	Probabilities and expectations. In Bayesian settings, unless otherwise specified, they are under the prior measure.
$\mathrm{Ent}, \mathrm{I}, \mathrm{KL}$	:	The Shannon entropy, Shannon mutual information, and Kullback Leibler Divergence [47]. In Bayesian settings, unless otherwise specified, they are under the prior measure.

Chapter 3

$\{\mathcal{X}^{(j)}\}_{j=1}^{M}$	:	The decomposition of the domain $\mathcal{X} = [0,1]^d$ in the additive model where $\mathcal{X}^{(j)}$ is composed of p_j dimensions.
M	:	The number of groups in the additive model.
$x^{(j)}$	:	For given $x \in \mathcal{X}$, $x^{(j)}$ denotes the coordinates of the dimensions in $\mathcal{X}^{(j)}$.
$f^{(j)}$	:	$f^{(j)} : \mathcal{X}^{(j)} \to \mathbb{R}$ is the function on the domain $\mathcal{X}^{(j)}$ in the additive model. $f : \mathcal{X} \to \mathbb{R}$ is given by $f = \sum_{j=1}^{M} f^{(j)}$
$\kappa^{(j)}$	:	The kernel for the j^{th} GP corresponding to function $f^{(j)}$.
$\mu_{t-1}^{(j)}$	:	The mean of the j^{th} GP at time t conditioned on the previous queries. See (3.2).
$\sigma_{t-1}^{(j)}$	:	The standard deviation of the j^{th} GP at time t conditioned on the previous queries. See (3.2).
p_j	:	The dimensionality of the j^{th} group, $\mathcal{X}^{(j)}$.
p	:	The maximum group size, $p = \max_j p_j$.

Chapters 4.1 & 4.4

$\mathcal{X}$	:	All arms in the bandit problem, i.e. the domain. $\mathcal{X} = \{1, \ldots, K\}$.
M	:	The number of fidelities (approximations).

$\theta_k, \theta_k^{(M)}$	:	The distribution from which the rewards are drawn for arm $k \in \mathcal{X}$ at the highest fidelity.
$\theta_k^{(m)}$	:	The distribution of the m^{th} fidelity approximation of $\theta_k^{(M)}$.
$\mu_k^{(m)}$	:	The mean of $\theta_k^{(m)}$.
$\zeta^{(m)}$	:	Bound on the maximum difference between $\mu_k^{(m)}$ and $\mu_k^{(M)}$, $\lvert\mu_k^{(M)} - \mu_k^{(m)}\rvert \leq \zeta^{(m)}$.
$\mathcal{X}_\star$	:	The set of optimal arms, $\operatorname{argmax}_{k\in\mathcal{X}} \mu_k^{(M)}$.
I_t	:	$I_t \in \mathcal{X}$ is the arm pulled at time t.
m_t	:	$m_t \in \{1, \ldots, M\}$ is the fidelity queried at time t.
Λ	:	Denotes the capital of some resource which is expended during the execution of the algorithm.
$\lambda^{(m)}$	:	The cost, i.e. amount of capital expended upon one arm pull at fidelity m.
N	:	The random number of queries at any fidelity within capital Λ. $N = \max\{n \geq 1 : \sum_{t=1}^{n} \lambda^{(m_t)} \leq \Lambda\}$
$R(\Lambda)$	:	The cumulative regret after spending capital Λ in the multi-fidelity setting. See (1.1).
ν, ψ	:	Quantities which characterise the concentration of the stochastic rewards around their mean, see (4.2).
$T_{k,t}^{(m)}$	:	The number of pulls of arm k at fidelity m in the first t steps.
$Q_t^{(m)}$	:	The number of pulls of all arms at fidelity m in the first t steps, $Q_t^{(m)} = \sum_l T_{k,t}^{(m)}$.
$\overline{X}_{k,s}^{(m)}$	:	The mean of s samples drawn from $\theta_k^{(m)}$.
$\Delta_k^{(m)}$	:	Denotes the quantity $\Delta_k^{(m)} = \mu_\star - \mu_k^{(m)} - \zeta^{(m)}$.
$\mathcal{B}_{k,t}^{(m)}$	:	An upper confidence bound on the mean $\mu_k^{(m)}$ of arm k at the m^{th} fidelity at time step t. See (4.3).
$\mathcal{B}_{k,t}$	:	An upper confidence bound on $\mu_k^{(M)} = \mu_k$ at time step t which combines $\{\mathcal{B}_{k,t}^{(m)}\}_{m=1}^{M}$. See (4.3).
ρ	:	A parameter in the MF-UCB algorithm used in constructing the upper confidence bounds. See (4.3).
$\gamma^{(m)}$	:	The parameter in MF-UCB for switching from the m^{th} fidelity to the $(m+1)^{\text{th}}$. See (4.4).

$\mathcal{J}_\eta^{(m)}$	:	The set of arms whose fidelity m mean is within η of $\mu_\star$, i.e $\mathcal{J}_\eta^{(m)} = \{k \in \mathcal{X};\, \mu_\star - \mu_k^{(m)} \leq \eta\}$.
$\{\mathcal{X}^{(m)}\}_{m=1}^M$	:	A partitioning of the arms $\mathcal{X} = \bigcup_{m=1}^M \mathcal{X}^{(m)}$ where $\mathcal{X}^{(m)}$ are the arms that will be played at the m^{th} fidelity by MF-UCB, but are mostly excluded from higher fidelities using information at fidelity m. See (4.6) and Figure 4.2.
$[\![k]\!]$	:	For an arm $k \in \mathcal{X}$, $[\![k]\!]$ denotes the set in the partition $\{\mathcal{X}^{(m)}\}_{m=1}^M$ that k belongs to.
Λ_0	:	The minimum cost that needs to be expended before the bounds in Theorem 16.
n_Λ	:	$n_\Lambda = \lfloor \Lambda/\lambda^{(M)} \rfloor$. Number of plays by a strategy playing only at fidelity M within capital Λ; also a lower bound on N, the number of plays by a multi-fidelity strategy.
$\mathcal{X}_{\checkmark}^{(m)}, \mathcal{X}_{\text{✗}}^{(m)}$	:	A further partitioning of $\mathcal{X}^{(m)} = \mathcal{X}_{\checkmark}^{(m)} \cup \mathcal{X}_{\text{✗}}^{(m)}$ where the lower bound in Theorem 17 is tight in $\mathcal{X}_{\checkmark}^{(m)}$ and not in $\mathcal{X}_{\text{✗}}^{(m)}$. See (4.7).
$\mathcal{L}_m(k)$	:	$\mathcal{L}_m(k) = \{\ell < m : \Delta_k^{(\ell)} > 0\} \cup \{m\}$ is the union of the m^{th} fidelity and all fidelities smaller than m for which $\Delta_k^{(\ell)} > 0$. See Theorem 17.
$\tilde{R}_k(\Lambda), \tilde{R}_{kn}$	:	$\tilde{R}_k(\Lambda) = \sum_{m=1}^M \lambda^{(m)} \Delta_k^{(M)} T_{k,N}^{(m)}$ is the regret incurred by arm k within capital Λ and $\tilde{R}_{kn} = \mathbb{E}[\tilde{R}_k(\Lambda) \vert N = n]$.
$\phi_t^{(m)}$	:	Denotes the quantity, $\phi_t^{(m)} = \lfloor \frac{\rho \log(t)}{\psi(\gamma^{(m)})} \rfloor$.
$\tilde{\mathbb{P}}, \tilde{\mathbb{E}}$	:	Probabilities and expectations in the modified construction for the lower bound.

Chapters 4.2 & 4.5

$\mathbb{P}_{\mathcal{GP}}, \mathbb{E}_{\mathcal{GP}}$	:	Probabilities and expectations when $f^{(1)}, \ldots, f^{(M)}$ are sampled from $\mathcal{GP}(0, \kappa)$.
$\mathbb{P}, \mathbb{E}$	:	Probabilities and expectations under the prior, which includes condition **A2** after $f^{(1)}, \ldots, f^{(M)}$ are sampled from $\mathcal{GP}(0, \kappa)$.
$\xi_{\mathbf{A2}}$	:	A lower bound on the probability that condition **A2** holds when $f^{(1)}, \ldots, f^{(M)}$ are sampled, see (4.11).
Q	:	The function which controls the probability on the supremum of a GP, see Lemma 7.
M	:	The number of fidelities.
$f^{(m)}$	:	The m^{th} fidelity approximation of $f^{(M)} = f$.

Λ	:	Denotes the capital of some resource which is expended during the execution of the algorithm.
$\lambda^{(m)}$	:	The cost, i.e. amount of capital expended, upon querying at fidelity m.
m_t	:	$m_t \in \{1, \ldots, M\}$ is the fidelity queried at time t.
N	:	The random number of queries at any fidelity within capital Λ. $N = \max\{n \geq 1 : \sum_{t=1}^{n} \lambda^{(m_t)} \leq \Lambda\}$
q_t, r_t	:	The instantaneous reward and regret respectively. $q_t = f^{(M)}(x_t)$ if $m_t = M$ and $-\infty$ if $m_t \neq M$. $r_t = f(x_\star) - q_t$.
$S(\Lambda)$	:	The simple regret after spending capital Λ in the multi-fidelity setting. $S(\Lambda) = f(x_\star) - \min_{t \in \{1,\ldots,N\}\ m_t = M} f(x_t)$. See (4.9).
$\zeta^{(m)}$	:	The bound on the maximum difference between $f^{(m)}$ and $f^{(M)}$, $\|f^{(M)} - f^{(m)}\|_\infty \leq \zeta^{(m)}$.
$\kappa_t^{(m)}$	:	The covariance of the m^{th} fidelity GP $f^{(m)}$ conditioned on $\mathcal{D}_t^{(m)}$ at time t.
$\sigma_t^{(m)}$	:	The standard deviatiation of the m^{th} fidelity GP $f^{(m)}$ conditioned on $\mathcal{D}_t^{(m)}$ at time t.
x_t, y_t	:	The queried point and observation at time t.
m_t	:	The queried fidelity at time t.
$\mathcal{D}_n^{(m)}$	:	The set of queries at the m^{th} fidelity until time n $\{(x_t, y_t)\}_{t:m_t=m}$.
β_t	:	The coefficient trading off exploration and exploitation in the UCB. See Theorems 21 and 22.
$\varphi_t^{(m)}(x)$	:	The upper confidence bound (UCB) provided by the m^{th} fidelity on $f^{(M)}(x)$. $\varphi_t^{(m)}(x) = \mu_{t-1}^{(m)}(x) + \beta_t^{1/2}\sigma_{t-1}^{(m)}(x) + \zeta^{(m)}$.
$\varphi_t(x)$	:	The combined UCB provided by all fidelities on $f^{(M)}(x)$. $\varphi_t(x) = \min_m \varphi_t^{(m)}(x)$.
$\gamma^{(m)}$	:	The parameter in MF-GP-UCB for switching from the m^{th} fidelity to the $(m+1)^{\text{th}}$.
$\tilde{R}_n$	:	The M^{th} fidelity cumulative regret after n rounds. See (4.37)
$T_n^{(m)}(A)$	:	The number of queries at fidelity m in subset $A \subset \mathcal{X}$ until time n.
$T_n^{(>m)}(A)$	:	Number of queries at fidelities greater than m in any subset $A \subset \mathcal{X}$ until time n.
n_Λ	:	$n_\Lambda = \lfloor \Lambda / \lambda^{(M)} \rfloor$. Number of plays by a strategy querying only at fidelity M within capital Λ; also a lower bound on N, the number of plays by a multi-fidelity strategy.

$\overline{n}_\Lambda$	:	An upper bound on N, the number of plays by a multi-fidelity strategy within capital Λ. $\overline{n}_\Lambda = \lfloor \Lambda/\lambda^{(1)} \rfloor$.
$\Psi_n(A)$	:	The maximum information gain of a set $A \subset \mathcal{X}$ after n queries in A. See Definition 1.
$\Delta^{(m)}(x)$	:	$\Delta^{(m)}(x) = f(x_\star) - f^{(m)} - \zeta^{(m)}$.
$\mathcal{J}_\eta^{(m)}$	:	The points in $\mathcal{X}$ whose $f^{(m)}$ value is within $\zeta^{(m)} + \eta$ of the optimum $f(x_\star)$. $\mathcal{J}_\eta^{(m)} = \{x \in \mathcal{X}; \Delta^{(m)}(x) \leq \eta\}$.
$\mathcal{H}^{(m)}$	:	$(\mathcal{H}^{(m)})_{m=1}^M$ is a partitioning of $\mathcal{X}$. See Equation (4.12). The analysis of MF-GP-UCB hinges on these partitioning.
$\widehat{\mathcal{H}}^{(m)}, \breve{\mathcal{H}}^{(m)}$	:	The arms "above"/"below" $\mathcal{H}^{(m)}$. $\widehat{\mathcal{H}}^{(m)} = \bigcup_{\ell=m+1}^M \mathcal{H}^{(\ell)}$, $\breve{\mathcal{H}}^{(m)} = \bigcup_{\ell=1}^{m-1} \mathcal{H}^{(\ell)}$.
$\mathcal{H}_{\tau,n}^{(m)}$	:	An n-dependent dilation of $\mathcal{H}_\tau^{(m)}$ in the continuous setting. See Section 4.2.3.
$\widehat{\mathcal{H}}_\tau^{(m)}, \breve{\mathcal{H}}_\tau{}^{(m)}$	:	The arms "above"/"below" $\mathcal{H}_\tau^{(m)}$. $\widehat{\mathcal{H}}_\tau^{(m)} = \bigcup_{\ell=m+1}^M \mathcal{H}_\tau^{(\ell)}$, $\breve{\mathcal{H}}_\tau{}^{(m)} = \bigcup_{\ell=1}^{m-1} \mathcal{H}_\tau^{(\ell)}$.
$\mathcal{X}_g$	:	The good set for $M = 2$ fidelity problems. $\mathcal{X}_g = \{x \in \mathcal{X}; f(x_\star) - f^{(1)}(x) \leq \zeta^{(1)}\}$.
$\widetilde{\mathcal{X}}_{g,\rho}$	:	The inflated good set for MF-GP-UCB. $\widetilde{\mathcal{X}}_{g,\rho} = \{x; f(x_\star) - f^{(1)}(x) \leq \zeta^{(1)} + 3\gamma\}$.
$\Omega^{(\varepsilon)}(A)$	:	The ε–covering number of a subset $A \subset \mathcal{X}$ in the $\|\cdot\|_2$ metric.
Λ_1, Λ_2	:	The minimum capitals that need to be expended before the bound on $S(\Lambda)$ hold in Theorems 21 and 22.

Chapters 4.3 & 4.6

$\mathcal{Z}$	:	The fidelity space.
g	:	$g : \mathcal{Z} \times \mathcal{X} \to \mathbb{R}$ defined on the product of the fidelity space and domain characterises the pay-off function and its connection to the approximations.
$z_\bullet$	:	$z_\bullet \in \mathcal{Z}$ is the fidelity of interest, i.e. the fidelity at which we wish to maximise g.
f	:	The pay-off function $f : \mathcal{X} \to \mathbb{R}$, given by $f(x) = g(z_\bullet, x)$.
λ	:	A known cost function $\lambda : \mathcal{Z} \to \mathbb{R}_+$ where, $\lambda(z)$ is the cost of evaluating g at fidelity $z \in \mathcal{Z}$.
z_t	:	The point in the fidelity space queried at time t.

n_Λ : $n_\Lambda = \lfloor \Lambda/\lambda(z_\bullet) \rfloor$ is the number of evaluations by a single fidelity method within capital Λ.

$S(\Lambda)$: The simple regret after spending capital Λ in the multi-fidelity setting. $S(\Lambda) = f(x_\star) - \min_{t \in \{1,\ldots,N\}\ z_t = z_\bullet} f(x_t)$. See (4.19).

$\phi_{\mathcal{Z}}$: $\phi_{\mathcal{Z}} : \mathbb{R}_+ \to \mathbb{R}_+$ is a decreasing function with $\|\phi_{\mathcal{Z}}\|_\infty = \phi_{\mathcal{Z}}(0) = 1$ used to characterise a radial kernel defined on the fidelity space.

ξ : The information gap for radial kernels defined on the fidelity space. $\xi : \mathcal{Z} \to [0,1]$ is given by $\xi(z) = \sqrt{1 - \phi_{\mathcal{Z}}(\|z - z_\bullet\|)^2}$. See (4.21).

ν_n, τ_n : The posterior mean and standard deviation of the joint GP conditioned on n past evaluations. See Chapter 4.3.1 and equation (4.22).

φ_t : The upper confidence bound in the multi-fidelity setting. See (4.23).

$\mathcal{Z}_t$: $\mathcal{Z}_t(x_t)$ is a subset of the fidelities chosen at time t based on x_t which will be considered for evaluation at time t. See (4.24).

$\mathcal{D}_t$: All previous queries at time t consisting of fidelity, domain point, and feedback triples, $\mathcal{D}_t = \{(z_i, x_i, y_i)\}_{i=1}^{t-1}$.

$\gamma(z)$: $\gamma(z)$ is a threshold function used in defining $\mathcal{Z}_t(x_t)$ at time t. See (4.24).

q : The exponent for the cost ratios used in defining $\gamma(z)$. See (4.24) and Remark 5.

$\Psi_n(A)$: The maximum information gain of a set $A \subset \mathcal{X}$ after n queries in A. See Definition 1.

$\mathcal{X}_{\rho,n}$: After n queries at any fidelity, BOCA will use most of its $z_\bullet$ evaluations in $\mathcal{X}_{\rho,n}$. See (4.60) for the definition.

$\mathcal{X}_\rho$: The limit of $\mathcal{X}_{\rho,n}$ as $n \to \infty$. See (4.25).

T_n : For any $U \subset \mathcal{Z} \times \mathcal{X}$, $T_n(U)$ denotes the number of queries by BOCA in U within the first n time steps. When $A \subset \mathcal{Z}$ and $B \subset \mathcal{X}$, we will denote $T_n(A, B) = T_n(A \times B)$.

$[> z]$: For $z \in \mathcal{Z}$, $[> z]$ will denote the fidelities which are more expensive than z, i.e. $[> z] = \{z' \in \mathcal{Z} : \lambda(z') > \lambda(z)\}$.

$\tilde{\mathcal{H}}_n, \mathcal{H}'_n$: Subsets of $\mathcal{Z} \times \mathcal{X}$ which are used in our analysis. See Chapter 4.6.1.

$\mathcal{H}_n$: A subset of $\mathcal{Z} \times \mathcal{X}$ at time n which is used to control the number of queries at various parts of the space. See (4.61).

F_n : A partitioning of the compact continuous domain $\mathcal{X}$ at time step n, based off a $\frac{\sqrt{d}}{2n^{\frac{\alpha}{2d}}}$-covering of $\mathcal{X}$.

Q_t : $Q_t : 2^{\mathcal{X}} \to 2^{\mathcal{Z}}$ maps subsets of $\mathcal{X}$ to subsets of $\mathcal{Z}$. For $A \subset \mathcal{X}$, $Q_t(A)$ are the fidelities where the information gap is smaller than $(f(x_\star) - f(x))/(2\rho\beta_t^{1/2})$ for all $x \in A$. See (4.62).

θ_t : $\theta_t(A)$ is the cheapest fidelity in $Q_t(A)$ for any $A \subset \mathcal{X}$. See (4.63).

$\mathcal{F}_n$: A subset of $\mathcal{Z} \times \mathcal{X}$ where we show BOCA does not spend too many high fidelity queries. See (4.64).

λ_r : $\lambda_r = \lambda_{min}/\lambda(z_\bullet)$ where $\lambda_{min} = \min_{z \in \mathcal{Z}} \lambda(z)$.

Chapter 5

M : The number of parallel workers, available in the synchronous or asynchronous parallel setting.

S'_T : The simple regret after time T, when we analyse the regret with time as the resource (instead of the number of evaluations) in the parallel setting. See (5.2).

N : The (possibly random) number of evaluations completed by all workers within time T by a method for parallel BO.

$\mathcal{D}_j$: The data from previous evaluations when deploying the j^{th} evaluation. In sequential settings, $|\mathcal{D}_j| = j - 1$, in asynchronous parallel settings, $|\mathcal{D}_j| = j - M$, and in synchronous parallel settings, $j - M \leq |\mathcal{D}_j| \leq j - 1$.

ξ_M : A bound on the maximum mutual information between the function f and M evaluations. See (5.3).

C_κ : A kernel dependent constant which bounds ξ_M after a suitable initialisation scheme [52]. See Chapter 5.3.

γ_M : The number of evaluations required in the initialisation scheme so that $\xi_M \leq C_\kappa$. $\gamma_M \asymp \text{poly}(M)$ for the SE kernel and $\gamma_M \asymp \text{polylog}(M)$ for the Matérn kernel. See Chapter 5.3.

n_{seq} : The expected number of evaluations for a sequential BO algorithm with one worker in time T. See Table 5.1.

$n_{\text{syn}}, n_{\text{asy}}$: The expected number of evaluations for synchronous and asynchronous parallel BO algorithms with M workers in time T. See Table 5.1.

Unif : Unif(a, b) denotes the uniform distribution on the continuous interval (a, b). See Table 5.1.

$\mathcal{HN}$: $\mathcal{HN}(\zeta^2)$ denotes the half-normal distribution with width parameter ζ^2. See Table 5.1.

Exp	:	$\text{Exp}(\lambda)$ denotes the exponential distribution with parameter λ. See Table 5.1.
β_n	:	A parameter used to construct an upper confidence bound for f at time step n, which is used in the proof for TS. See (5.4).
ν_j, τ_j	:	In our analysis, we construct a discretisatin of the domain $\mathcal{X}$ of size $\lvert\nu_j\rvert = \tau_j^d$. See Chapter 5.5.2.
$[x]_j$	:	For any $x \in \mathcal{X}$, $[x]_j$ denotes the closest point to x in the discretisation ν_j.
U_j	:	$U_j = \mu_{\mathcal{D}_j} + \beta_j^{1/2}\sigma_{\mathcal{D}_j} : \mathcal{X} \to \mathbb{R}$ is an upper confidence bound for f when conditioned on the data available at time j.
V_j	:	$U_j = \mu_{\mathcal{D}_j} + \beta_j^{1/2}\sigma_{j-1} : \mathcal{X} \to \mathbb{R}$ is a quantity used in our proof of parallel TS.
Ξ	:	The expected supremum of a GP $f \sim \mathcal{GP}(0, \kappa)$. See Lemma 6.
θ	:	In Chapter 5.5.3, θ denotes the expected time to complete a single evaluation in any of the random evaluation models.
$T_{\alpha,\delta}$	:	In Chapter 5.5.3, $T_{\alpha,\delta}$ is defined such that for all $T > T_{\alpha,\delta}$, the random number of evaluations N concentrates in an interval around its expected value with probability at least $1-\delta$. α determines the width of this interval.

Chapter 6

$\mathcal{G}, (\mathcal{L}, \mathcal{E})$	:	Typically, $\mathcal{G} = (\mathcal{L}, \mathcal{E})$ represents the architecture of a neural network where $\mathcal{L}$ are the layers of the network and $\mathcal{E}$ are its directed edges.
$\ell\ell$	:	For a layer u, $\ell\ell(u)$ denotes its layer label, i.e. the type of operations performed at the layer.
ℓu	:	For a layer u, $\ell u(u)$ denotes the number of computational units, e.g. number of computational filters, at the layer.
$u_{\text{ip}}, u_{\text{op}}$	:	The input and output layers respectively for each network, which exist uniquely for all neural network architectures. $\ell\ell(u_{\text{ip}}) = \texttt{ip}$ and $\ell\ell(u_{\text{op}}) = \texttt{op}$.
$d, \bar{d}$	:	The OTMANN distance and its normalised version respectively. See Chapter 6.2 and (6.1).
ℓm	:	For a layer u, $\ell m(u)$ denotes the mass of the layer used in the computation of the OTMANN distance.
ζ	:	$\zeta \in (0, 1)$ is a coefficient used to define the layer mass for input, output, and decision layers relative to the total mass of the network.

$\delta^{\text{sp}}_{\text{ip}(u)}$, $\delta^{\text{sp}}_{\text{op}}(u)$	:	The shortest path length from the input layer u_{ip} to a layer u and the shortest path length from u to the output layer u_{op} respectively.
$\delta^{\text{lp}}_{\text{ip}(u)}$, $\delta^{\text{lp}}_{\text{op}}(u)$	:	The longest path length from the input layer u_{ip} to a layer u and the longest length path from u to the output layer u_{op} respectively.
$\delta^{\text{rw}}_{\text{ip}(u)}$, $\delta^{\text{rw}}_{\text{op}}(u)$	:	The random walk path length from the input layer u_{ip} to a layer u and the random walk path length from u to the output layer u_{op} respectively.
$d, \bar{d}$	:	The OTMANN distance and its normalised version used to quantify the dissimilarity between two neural networks. See Chapter 6.2.
$\langle \cdot, \cdot \rangle$	:	In Chapter 6, this typically refers to the Frobenius matrix inner product.
ϕ_{lmm}	:	The label mismatch penalty in OTMANN. See (6.1) and Chapter 6.2.
ϕ_{str}	:	The structural penalty in OTMANN. See (6.1) and Chapter 6.2.
ϕ_{nas}	:	The non-assignment penalty in OTMANN. See (6.1) and Chapter 6.2.
M	:	The $L \times L$ label penalty matrix used to define ϕ_{lmm} in OTMANN, where L is the number of label types. See Chapter 6.2.
C_{lmm}	:	The label penalty matrix used to define ϕ_{lmm} in OTMANN. ϕ_{lmm} is given by $\phi_{\text{lmm}}(Z) = \langle Z, C_{\text{lmm}} \rangle$, where Z is the matrix of matchings between the two networks. See Chapter 6.2.
C_{str}	:	The structral penalty matrix used to define ϕ_{str} in OTMANN. ϕ_{str} is given by $\phi_{\text{str}}(Z) = \langle Z, C_{\text{str}} \rangle$, where Z is the matrix of matchings between the two networks. See Chapter 6.2.

Chapters 8.1 & 8.3

Θ	:	The parameter space. Any $\theta \in \Theta$ specifies the system characteristics.
$\mathcal{X}$	:	The action space. At each time step, the decision-maker chooses an experiment/action $x \in \mathcal{X}$.
$\mathcal{Y}$	:	The outcome space. Upon conducting an experiment, the decision-maker observes an outcome $y \in \mathcal{Y}$.
$\theta_\star$	:	The true parameter unknown to the decision-maker. $\theta_\star \in \Theta$.
ρ_0	:	The prior distribution from which $\theta_\star$ is drawn.
$\mathbb{P}(\cdot \vert x, \theta_\star)$	:	The likelihood model for the outcomes. Upon making an action $x \in \mathcal{X}$, the decision-maker observes $Y_x \sim \mathbb{P}(\cdot \vert x, \theta_\star)$.
D_n	:	A data sequence n which is an ordered multiset of action-observation pairs $D_n = \{(X_j, Y_j)\}_{j=1}^n$ collected by the decision-maker after n rounds. For $t < n$, $D_t = \{(X_j, Y_j)\}_{j=1}^t$ is a prefix of length t of the data sequence D_n.

$\mathcal{D}_n$	:	The set of all data sequences of length n.
$\mathcal{D}$	:	The set of all possible data sequences, $\mathcal{D} = \bigcup_{t=1}^{n} \mathcal{D}_t$.
λ	:	$\lambda : \Theta \times \mathcal{D} \to \mathbb{R}$ is the reward function. After n rounds, we wish to collect a data sequence D_n to maximise $\lambda(\theta_\star, D_n)$.
Λ	:	$\Lambda(\theta_\star, D_n) = \sum_{t=1}^{n} \lambda(\theta_\star, D_t)$ is the sum of rewards in the first n rounds. See (8.1).
$\uplus$	:	For two data sequences D, D', $D \uplus D'$ denotes the concatenation of the two sequences.
$\prec, \succ$	:	For two data sequences D, D', $D \prec D'$ and $D' \succ$ both equivalently denote that D is a prefix of D'.
π	:	Typically denotes a policy for adaptive goal oriented design of experiments.
$\pi^\star_{\mathrm{M}}, \pi^\star_{\mathrm{G}}$	:	The myopically optimal and globally optimal policies respectively, which operate with knowledge of the true parameter $\theta_\star$. See Chapter 8.1.1.
$\pi^{\mathrm{PS}}_{\mathrm{M}}$	:	The myopic posterior sampling (MPS) policy. See Chapter 8.1.2 and Algorithm 11.
X_t	:	The action taken by a decision-maker at time t.
Y_x	:	The outcome when we take an action $x \in \mathcal{X}$.
λ^+	:	$\lambda^+ : \Theta \times \mathcal{D} \times \mathcal{X} \to \mathbb{R}$, is the expected look-ahead reward, where, $\lambda^+(\theta, D, x)$ is the expected reward at the next time step if $\theta \in \Theta$ were the true parameter, D was the current data sequence collected, and we were to take action $x \in \mathcal{X}$. See (8.2).
$\mathbb{E}[\cdot \vert D_n \sim \pi]$	:	The expectation of some quantity (typically a function of the data sequence), when the data sequence is collected by following policy π.
$\mathbb{E}_{Y,t+1:\vert\theta}$	:	The expectation over all observations generated from time $t+1$ onwards, when the true parameter is θ, i.e. $\theta_\star = \theta$.
π^θ_{M}	:	The myopically optimal policy $\pi^\star_{\mathrm{M}}$ when the true parameter is θ, i.e. $\theta_\star = \theta$.
ϵ_n, τ_n	:	Sequences which determine the rate of convergence of $\pi^{\mathrm{PS}}_{\mathrm{M}}$. See Condition 70.
Ψ_n	:	The maximum information gain, defined with respect to $\theta_\star$, i.e. $\Psi_n = \max_{D_n \subset \mathcal{D}_n} \mathrm{I}(\theta_\star; D_n)$. See (8.3).
$\mathbb{P}_t, \mathbb{E}_t$	:	Probabilities and expectations when conditioning on actions and observations up to and including time t, e.g. for any event E, $\mathbb{P}_t(E) = \mathbb{P}(E \vert D_t)$.
J_n	:	$J_n(\theta_\star, \pi) = \mathbb{E}[\Lambda(\theta_\star, D_n) \vert \theta_\star, D_n \sim \pi]$ denotes the expected sum of cumulative rewards for a fixed policy π after n rounds. See Chapter 8.3.1.

Q^π	:	$Q^\pi(D_t, x, y)$ will denote the expected sum of future rewards when, having collected the data sequence D_n, we take action $x \in \mathcal{X}$, observe $y \in \mathcal{Y}$ and then execute policy π for the remaining $n-t-1$ steps. See (8.9).
q_t	:	$q_t(\theta_\star, x, y) = Q^{\pi^\star_{\mathrm{M}}}(D_{t-1}, x, y)$ is expected future rewards when we follow $\pi^\star_{\mathrm{M}}$.
$\tilde{\mathbb{P}}_{t-1}, \tilde{p}_{t-1}$	:	$\tilde{\mathbb{P}}_{t-1}$ is the distribution of X_t given D_{t-1}; i.e. $\tilde{\mathbb{P}}_{t-1}(\cdot) = \mathbb{P}_{t-1}(X_t = \cdot)$. $\tilde{p}_{t-1}$ is the density (Radon-Nikodym derivative) of $\tilde{\mathbb{P}}_{t-1}$.
$D^\star_n, D^{\star\star}_n$	:	The data collected by $\pi^\star_{\mathrm{M}}$ and $\pi^\star_{\mathrm{G}}$ respectively in n steps.

Chapter 8.2

Θ	:	The space of parameters where we can query the likelihood $\mathcal{L}(\theta)$ for any $\theta \in \Theta$.
P_θ	:	The prior distribution on Θ for the parameters.
$\mathcal{L}$	:	The likelihood $\mathcal{L}(\theta) = \mathbb{P}(\mathbf{X_{obs}}\|\theta)$.
$P_{\theta\|\mathbf{X_{obs}}}$	:	$P_{\theta\|\mathbf{X_{obs}}}(\theta\|\mathbf{X_{obs}})$ is the posterior distribution for the parameters given the observations $\mathbf{X_{obs}}$. See (8.4).
$\widehat{\mathcal{P}}^A_n(\mathbf{X_{obs}}, \theta)$	:	Estimate of the log joint density obtained using a dataset $A_n = \{\theta_i, \mathcal{L}_i\}_{i=1}^n$ of query and likelihood-value pairs.
$\widehat{P}^A_n(\theta\|\mathbf{X_{obs}})$	:	Estimate of the posterior distribution obtained using a dataset $A_n = \{\theta_i, \mathcal{L}_i\}_{i=1}^n$ of query and likelihood-value pairs. See (8.5).
$u_t^{\mathrm{NED}}, u_t^{\mathrm{EV}}$	:	The negative exponentiated divergence and exponentiated variance utility functions respectively for active posterior estimation. See (8.7) and (8.8)
$D(\cdot\|\cdot)$	:	A divergence between two distributions, used in u_t^{NED}.

Appendix B

Abbreviations

ABC	:	Approximate Bayesian Computing [168]
ActiveSel	:	Active Select [36]
AGPR	:	Active Gaussian Process Regression [219]
Add-GP-UCB	:	Additive Gaussian Process Upper Confidence Bound. See Algorithm 2.
asyEI	:	Asynchronous Expected Improvement
asyHTS	:	Asynchronous Thompson Sampling with Hallucinations
asyHUCB	:	Asynchronous Upper Confidence Bound with Hallucinations
asyRAND	:	Asynchronous Random Sampling
asyTS	:	Asynchronous Thompson Sampling. See Algorithm 7.
asyUCB	:	Asynchronous Upper Confidence Bound
BAPE	:	Bayesian Active Posterior Estimation. See Algorithm 12
BO	:	Bayesian Optimisation
BOCA	:	Bayesian Optimisation with Continuous Approximations. See Algorithm 5.
DiRect	:	The DIviding RECTangles algorithm [112]
DOE	:	Design of Experiments
EA	:	Evolutionary Algorithm
ESP	:	Elementary Symmetric Polynomial [165]
EV	:	Exponentiated Variance. See Chapter 8.2.1.
GP-EI	:	(Gaussian Process) Expected Improvement [113]
GP-UCB	:	Gaussian Process Upper Confidence Bound [235]. See Algorithm 1.
GRID	:	Grid Search. Typically a policy which queries a Euclidean domain on a uniform grid.
KDE	:	Kernel Density Estimation [265]
MCMC	:	Markov Chain Monte Carlo [32]
MCMC-DE	:	MCMC with Density Estimation. A heuristic for posterior estimation which performs kernel density estimation on the points collected with MCMC.

MCMC-R	:	MCMC with Regression. A heuristic for posterior estimation which regresses on the likelihood values on the points collected with MCMC.
MF-GP-UCB	:	Multi-fidelity Gaussian Process Upper Confidence Bound. See Algorithm 4.
MF-NAIVE	:	A naive multi-fidelity method described in Chapter 4.2.5.
MF-SKO	:	Multi-fidelity Sequential Kriging Optimisation [100]
MF-UCB	:	Multi-fidelity Upper Confidence Bound. See Algorithm 3.
MOO	:	Multi-objective Optimisation
MOORS	:	Multi-objective Optimisation with Random Scalarisations [190]
MPS	:	Myopic Posterior Sampling. See Algorithm 11.
NASBOT	:	Neural Architecture Search with Bayesian Optimisation and Optimal Transport. See Chaper 6.3.
NED	:	Negative Exponentiated Divergence. See Chapter 8.2.1.
OTMANN	:	Optimal Transport Metrics for Architectures of Neural Networks. See Chaper 6.2.
PDOO	:	Parallel Deterministic Optimistic Optimisation [86]
RAND	:	Random Search. Typically a policy which queries the domain randomly.
REMBO	:	Random Embeddings for Bayesian Optimisation [264].
SE	:	Squared Exponential (in reference to the kernel)
seqTS	:	Sequential Thompson Sampling. See Algorithm 6.
synBUCB	:	Synchronous Batch Upper Confidence Bound [52]
synUCBPE	:	Synchronous Upper Confidence Bound with Pure Exploration [43]
synRAND	:	Synchronous Random Sampling.
synTS	:	Synchronous Thompson Sampling. See Algorithm 8.
TreeBO	:	Tree structured Bayesian Optimisation [108]
TS	:	Thompson Sampling, a sampling strategy for bandits first discussed in Thompson [246].
UCB	:	Upper Confidence Bound, a bandit strategy first appearing in Auer [10].

Appendix C

Open Source Software Released with this Thesis

We state and provide links to open source software tools released as part of this thesis. Unless otherwise specified, all software is released under the MIT License. The repositories also contain links for downloading relevant datasets and/or code for generating data.

- add-gp-bandits: github.com/kirthevasank/add-gp-bandits
  A matlab implementation of additive upper confidence bound methods in Gaussian process bandits, as described in Chapter 3. Released under the GNU GPL v3 license.
- Dragonfly: dragonfly.github.io
  A python library for scalable Bayesian optimisation, that integrates many of the methods in this thesis. It provides an array of tools to scale up Bayesian optimisation. These include the methods in Chapters 3, 4.3, 5, 6, and 7, among others.
- gp-parallel-ts: github.com/kirthevasank/gp-parallel-ts
  A python implementation of parallelised Bayesian optimisation using Thompson sampling as described in Chapter 5. We provide implementations of both the synchronous and asynchronous versions along with experimental set ups in synthetic settings where the evaluation time is modeled as a random variable.
- mf-gp-ucb: github.com/kirthevasank/mf-gp-ucb
  A matlab implementation of the MF-GP-UCB method described in Chapter 4.2 for multi-fidelity Bayesian optimisation with a finite number of approximations. Released under the GNU GPL v3 license.
- mps: github.com/kirthevasank/mps
  A python implementation of the MPS algorithm described in Chapter 8.1 for adaptive Bayesian design of experiments.
- nasbot: github.com/kirthevasank/nasbot
  A python implementation of NASBOT as described in Chapter 6. The library also provides

code for tuning and training the architectures of convolutional neural networks and multi-layer perceptrions.

Please see cs.cmu.edu/∼kkandasa/software.html for up-to-date information on the above and other related software packages and relevant datasets.

Bibliography

[1] Yahoo Streaming Benchmarks. https://github.com/yahoo/streaming-benchmarks. Accessed: 2018-09-30. 7.4.4

[2] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mane, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viegas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv:1603.04467*, 2016. 5.4

[3] Ryan Prescott Adams, Iain Murray, and David J. C. MacKay. The Gaussian Process Density Sampler. In *NIPS*, pages 9–16, 2008. 8.2.1

[4] Robert J Adler. *An Introduction to Continuity, Extrema, and Related Topics for General Gaussian Processes*. IMS, 1990. 6, 2, 56, 5.5.3

[5] Alekh Agarwal, John C Duchi, Peter L Bartlett, and Clement Levrard. Oracle inequalities for computationally budgeted model selection. In *COLT*, 2011. 4

[6] Shipra Agrawal and Navin Goyal. Analysis of thompson sampling for the multi-armed bandit problem. In *Conference on Learning Theory (COLT)*, 2012. 2.2

[7] Zeyuan Allen-Zhu, Yuanzhi Li, Aarti Singh, and Yining Wang. Near-optimal design of experiments via regret minimization. In *International Conference on Machine Learning*, pages 126–135, 2017. 8.1

[8] Alexandr Andoni, Piotr Indyk, Thijs Laarhoven, Ilya Razenshteyn, and Ludwig Schmidt. Practical and optimal lsh for angular distance. In *Advances in Neural Information Processing Systems*, pages 1225–1233, 2015. 7.4.4

[9] Jean-Yves Audibert, Rémi Munos, and Csaba Szepesvári. Exploration-exploitation Tradeoff Using Variance Estimates in Multi-armed Bandits. *Theor. Comput. Sci.*, 2009. 4.1.2, 4.4.2, 4.4.2

[10] Peter Auer. Using Confidence Bounds for Exploitation-exploration Trade-offs. *J. Mach. Learn. Res.*, 2003. 3, 4.1, 4.1.2, B.1

[11] TG Authors. Gpyopt: A bayesian optimization framework in python, 2016. 7.4

[12] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning. *arXiv preprint arXiv:1611.02167*, 2016. 6

[13] James Bergstra, Dan Yamins, and David D Cox. Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms. In *Proceedings of the 12th Python in Science Conference*, pages 13–20. Citeseer, 2013. 7.4

[14] Felix Berkenkamp, Matteo Turchetta, Angela Schoellig, and Andreas Krause. Safe model-based reinforcement learning with stability guarantees. In *Advances in Neural Information Processing Systems*, pages 908–918, 2017. 9.3

[15] Hildo Bijl, Thomas B Schön, Jan-Willem van Wingerden, and Michel Verhaegen. A Sequential Monte Carlo approach to Thompson sampling for Bayesian optimization. *arXiv preprint arXiv:1604.00169*, 2016. 2.2

[16] Eli Bingham, Jonathan P Chen, Martin Jankowiak, Fritz Obermeyer, Neeraj Pradhan, Theofanis Karaletsos, Rohit Singh, Paul Szerlip, Paul Horsfall, and Noah D Goodman. Pyro: Deep universal probabilistic programming. *arXiv preprint arXiv:1810.09538*, 2018. 8.1.2, 9.4

[17] Ilija Bogunovic, Jonathan Scarlett, Andreas Krause, and Volkan Cevher. Truncated variance reduction: A unified approach to bayesian optimization and level-set estimation. In *Advances in Neural Information Processing Systems*, pages 1507–1515, 2016. 4

[18] Stéphane Boucheron and Maud Thomas. Concentration inequalities for order statistics. *Electronic Communications in Probability*, 2012. 5.5.3, 59

[19] Stéphane Boucheron, Gábor Lugosi, and Pascal Massart. *Concentration inequalities: A nonasymptotic theory of independence*. Oxford University Press, 2013. 2.4, 12, 2.4, 13, 5.5.3, 61

[20] Stephen Boyd and Craig Barratt. Linear controller design: limits of performance. Technical report, Stanford University Stanford United States, 1991. 9.4

[21] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004. 7.3.5

[22] Gary Bradski and Adrian Kaehler. *Learning OpenCV*. O'Reilly Media Inc., 2008. 3.2

[23] Eric Brochu, Vlad M. Cora, and Nando de Freitas. A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning. *CoRR*, 2010. 3.2, 4.2.4, 4.3.4, 7.3.4

[24] Brent Bryan, Jeff Schneider, Robert Nichol, Christopher Miller, Christopher Genovese, and Larry Wasserman. Active learning for identifying function threshold boundaries. In *Advances in Neural Information Processing Systems 18*, pages 163–170. MIT Press, Cambridge, MA, 2006. 8.2

[25] Sébastien Bubeck and Nicolò Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends in Machine Learning*, 2012. 3, 4.4.2, 4.4.3

[26] Sébastien Bubeck, Rémi Munos, Gilles Stoltz, and Csaba Szepesvári. X-armed bandits. *Journal of Machine Learning Research*, 12(May):1655–1695, 2011. 4.2.3

[27] Adam D. Bull. Convergence Rates of Efficient Global Optimization Algorithms. *Journal of Machine Learning Research*, 2011. 2.2, 3, 3.1.2, 8.1.2

[28] Jonathan Kenneth Bunn, Jianjun Hu, and Jason R Hattrick-Simpers. Semi-supervised approach to phase identification from combinatorial sample diffraction patterns. *JOM*, 68 (8):2116–2125, 2016. 1.2, 8.1

[29] Krisztian Buza. Feedback prediction for blogs. In *Data analysis, machine learning and knowledge discovery*, pages 145–152. Springer, 2014. 6.4, 7.4.3

[30] Luis M Candanedo, Véronique Feldheim, and Dominique Deramaix. Data driven prediction models of energy use of appliances in a low-energy house. *Energy and Buildings*, 140:81–97, 2017. 7.4.3

[31] Bob Carpenter, Andrew Gelman, Matthew D Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell. Stan: A probabilistic programming language. *Journal of statistical software*, 76(1), 2017. 8.1.2, 9.4

[32] George Casella and Christian P. Robert. A Short History of Markov Chain Monte Carlo: Subjective Recollections from Incomplete Data. Economics Papers from University Paris Dauphine 123456789/3549, Paris Dauphine University, 2011. B.1

[33] Rui Castro, Rebecca Willett, and Robert Nowak. Faster rates in regression via active learning. In *NIPS*, 2005. 8.2.1

[34] Deepayan Chakrabarti, Ravi Kumar, Filip Radlinski, and Eli Upfal. Mortal Multi-Armed Bandits. In *NIPS*, 2008. 1.1

[35] Olivier Chapelle and Lihong Li. An empirical evaluation of thompson sampling. In *Advances in neural information processing systems*, pages 2249–2257, 2011. 8.1.4

[36] Kamalika Chaudhuri, Sham M Kakade, Praneeth Netrapalli, and Sujay Sanghavi. Convergence rates of active learning for maximum likelihood estimation. In *Advances in Neural Information Processing Systems*, pages 1090–1098, 2015. 8.1.4, 8.1.4, B.1

[37] Bo Chen, Rui Castro, and Andreas Krause. Joint Optimization and Variable Selection of High-dimensional Gaussian Processes. In *Int'l Conference on Machine Learning*, 2012. 3

[38] Yuxin Chen and Andreas Krause. Near-optimal batch mode active learning and adaptive submodular optimization. *ICML (1)*, 28:160–168, 2013. 8.1

[39] Yuxin Chen, Hiroaki Shioi, Cesar Fuentes Montesinos, Lian Pin Koh, Serge Wich, and Andreas Krause. Active detection via adaptive submodularity. In *ICML*, pages 55–63, 2014. 8.1

[40] Yuxin Chen, S Hamed Hassani, Andreas Krause, et al. Near-optimal bayesian active learning with correlated and noisy tests. *Electronic Journal of Statistics*, 11(2):4969–5017, 2017. 8.1, 1

[41] Herman Chernoff. *Sequential analysis and optimal design*. Siam, 1972. 8.1

[42] Sayak Ray Chowdhury and Aditya Gopalan. On kernelized multi-armed bandits. *arXiv:1704.00445*, 2017. 2.2

[43] Emile Contal, David Buffoni, Alexandre Robicquet, and Nicolas Vayatis. Parallel Gaussian process optimization with upper confidence bound and pure exploration. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 225–240. Springer, 2013. 5, 2, 5.1, 5.4, B.1

[44] Emile Contal, Vianney Perchet, and Nicolas Vayatis. Gaussian process optimization with mutual information. In *International Conference on Machine Learning*, pages 253–261, 2014. 9.2

[45] Andrea Coraddu, Luca Oneto, Aessandro Ghio, Stefano Savio, Davide Anguita, and Massimo Figari. Machine learning approaches for improving condition-based maintenance of naval propulsion plants. *Proceedings of the Institution of Mechanical Engineers, Part M: Journal of Engineering for the Maritime Environment*, 230(1):136–153, 2016. 6.2.4, 6.4, 7.4.3

[46] Corinna Cortes, Xavi Gonzalvo, Vitaly Kuznetsov, Mehryar Mohri, and Scott Yang. Adanet: Adaptive structural learning of artificial neural networks. *arXiv preprint arXiv:1607.01097*, 2016. 6

[47] Thomas M Cover and Joy A Thomas. *Elements of Information Theory*. John Wiley & Sons, 2012. A.1

[48] Mark Cutler, Thomas J. Walsh, and Jonathan P. How. Reinforcement Learning with Multi-Fidelity Simulators. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2014. 4, 4

[49] Christoph Dann and Emma Brunskill. Sample complexity of episodic fixed-horizon reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 2818–2826, 2015. 8.1.3

[50] T. M. Davis et al. Scrutinizing Exotic Cosmological Models Using ESSENCE Supernova Data Combined with Other Cosmological Probes. *Astrophysical Journal*, 2007. (document), 1.1, 4.2.5, 4.3.4, 7.4.2, 7.15, 8.1.4, 8.2.2

[51] Kalyanmoy Deb. Multi-objective optimization. In *Search methodologies*, pages 403–449. Springer, 2014. 7.3.5

[52] Thomas Desautels, Andreas Krause, and Joel W Burdick. Parallelizing exploration-exploitation tradeoffs in Gaussian process bandit optimization. *Journal of Machine Learning Research*, 15(1):3873–3923, 2014. 5, 2, 5.1, 5.3, 47, 5.4, 5.5.2, 52, 5.5.2, 5.5.2, 7.2, A.6, B.1

[53] Josip Djolonga, Andreas Krause, and Volkan Cevher. High-Dimensional Gaussian Process Bandits. In *Advances in Neural Information Processing Systems*, 2013. 3

[54] David K. Duvenaud, Hannes Nickisch, and Carl Edward Rasmussen. Additive gaussian processes. In *Advances in Neural Information Processing Systems*, 2011. 3, 7.2, 7.3.2, 9.2

[55] Michael Emmerich and Jan-willem Klinkenberg. The computation of the expected improvement in dominated hypervolume of pareto front approximations. *Rapport technique, Leiden University*, 34, 2008. 7.3.5

[56] Valerii Vadimovich Fedorov. *Theory of optimal experiments*. Elsevier, 1972. 8.1

[57] Kelwin Fernandes, Pedro Vinagre, and Paulo Cortez. A proactive intelligent decision support system for predicting the popularity of online news. In *Portuguese Conference on Artificial Intelligence*, pages 535–546. Springer, 2015. 6.4, 7.4.3

[58] Dario Floreano, Peter Dürr, and Claudio Mattiussi. Neuroevolution: from architectures to learning. *Evolutionary Intelligence*, 1(1):47–62, 2008. 6

[59] Daniel Foreman-Mackey, David W. Hogg, Dustin Lang, and Jonathan Goodman. emcee: The MCMC Hammer, January 2013. 8.2, 8.2.2

[60] Forrester, Alexander I. J. and Sóbester, András and Keane, Andy J. Multi-fidelity optimization via surrogate modelling. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science*, 2007. 4, 4.2.5

[61] Peter I Frazier. A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018. 8.1

[62] Peter I Frazier, Warren B Powell, and Savas Dayanik. A knowledge-gradient policy for sequential information collection. *SIAM Journal on Control and Optimization*, 47(5): 2410–2439, 2008. 8.1

[63] Roy Frostig, Rong Ge, Sham M Kakade, and Aaron Sidford. Competing with the empirical risk minimizer in a single pass. In *Conference on learning theory*, pages 728–763, 2015. 8.1.4

[64] Yichen Fu, Egemen Kolemen, Mark Boyer, and Qiming Hu. Machine learning-based real-time plasma control of diii-d. *Bulletin of the American Physical Society*, 2018. 9.3

[65] Kenji Fukumizu, Le Song, and Arthur Gretton. Kernel Bayes' Rule: Bayesian Inference with Positive Definite Kernels. *Journal of Machine Learning Research*, 14:3753–3783, 2014. 8.2

[66] Victor Gabillon, Branislav Kveton, Zheng Wen, Brian Eriksson, and S Muthukrishnan. Adaptive submodular maximization in bandit setting. In *Advances in Neural Information Processing Systems*, pages 2697–2705, 2013. 8.1

[67] Victor Gabillon, Branislav Kveton, Zheng Wen, Brian Eriksson, and S Muthukrishnan. Large-scale optimistic adaptive submodularity. In *AAAI*, pages 1816–1823, 2014. 8.1

[68] Xinbo Gao, Bing Xiao, Dacheng Tao, and Xuelong Li. A survey of graph edit distance. *Pattern Analysis and applications*, 13(1):113–129, 2010. 6, 6.2.2

[69] Jacob Gardner, Chuan Guo, Kilian Weinberger, Roman Garnett, and Roger Grosse. Discovering and exploiting additive structure for bayesian optimization. In *Artificial Intelligence and Statistics*, pages 1311–1319, 2017. 3

[70] Jacob R Gardner, Matt J Kusner, Zhixiang Eddie Xu, Kilian Q Weinberger, and John P Cunningham. Bayesian optimization with inequality constraints. In *ICML*, pages 937–945, 2014. 8.1.4

[71] Kevin L Gering. Prediction of electrolyte viscosity for aqueous and non-aqueous systems: Results from a molecular model based on ion solvation and a chemical physics framework. *Electrochimica Acta*, 51(15):3125–3138, 2006. 1.2, 8.1, 8.1.4

[72] Subhashis Ghosal and Anindya Roy. Posterior consistency of Gaussian process prior for nonparametric binary regression". *Annals of Statistics*, 2006. 2.2, 7, 2

[73] David Ginsbourger, Janis Janusevskis, and Rodolphe Le Riche. Dealing with asynchronicity in parallel gaussian process based global optimization. In *Conference of the ERCIM WG on Computing and Statistics*, 2011. 5, 1, 5.4, 6.3.3, 7.2

[74] Daniel Golovin and Andreas Krause. Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *Journal of Artificial Intelligence Research*, 42:427–486, 2011. 8.1, 8.1.2, 72, 8.3.2, 76

[75] Daniel Golovin, Andreas Krause, and Debajyoti Ray. Near-optimal bayesian active learning with noisy observations. In *Advances in Neural Information Processing Systems*, pages 766–774, 2010. 8.1

[76] Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4 (2):268–276, 2018. 9.3

[77] Javier Gonzalez, Joseph Longworth, David James, and Neil Lawrence. Bayesian Optimization for Synthetic Gene Design. In *NIPS Workshop on Bayesian Optimization in Academia and Industry*, 2014. 1.1, 3

[78] Javier González, Zhenwen Dai, Philipp Hennig, and Neil D Lawrence. Batch bayesian optimization via local penalization. *arXiv preprint arXiv:1505.08052*, 2015. 5, 7.2

[79] Aditya Gopalan and Shie Mannor. Thompson sampling for learning parameterized markov decision processes. In *Conference on Learning Theory*, pages 861–898, 2015. 8.1

[80] Aditya Gopalan, Shie Mannor, and Yishay Mansour. Thompson sampling for complex online problems. In *International Conference on Machine Learning*, pages 100–108, 2014. 8.1

[81] Alkis Gotovos, Nathalie Casati, Gregory Hitz, and Andreas Krause. Active Learning for Level Set Estimation. In *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*, 2013. 8.1, 8.1.4, 8.2

[82] Alkis Gotovos, Nathalie Casati, Gregory Hitz, and Andreas Krause. Active learning for level set estimation. In *IJCAI*, pages 1344–1350, 2013. 8.1.3

[83] Yashodhar Govil, Egemen Kolemen, Yichen Fu, Florian Laggner, et al. Machine learning algorithms for profile control and prediction in tokomaks. *Bulletin of the American Physical Society*, 2018. 1.1

[84] Franz Graf, Hans-Peter Kriegel, Matthias Schubert, Sebastian Pölsterl, and Alexander Cavallaro. 2d image registration in ct images using radial image descriptors. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 607–614. Springer, 2011. 6.2.4, 6.4, 7.4.3

[85] Jarosław M Granda, Liva Donina, Vincenza Dragone, De-Liang Long, and Leroy Cronin.

Controlling an organic synthesis robot with machine learning to search for new reactivity. *Nature*, 559(7714):377, 2018. 9.3

[86] Jean-Bastien Grill, Michal Valko, and Rémi Munos. Black-box optimization of noisy functions with unknown smoothness. In *Advances in Neural Information Processing Systems*, pages 667–675, 2015. 7.3.3, 7.4, B.1

[87] László Györfi, Micael Kohler, Adam Krzyzak, and Harro Walk. *A Distribution Free Theory of Nonparametric Regression*. Springer Series in Statistics, 2002. 3, 8.2.2

[88] T. J. Hastie and R. J. Tibshirani. *Generalized Additive Models*. London: Chapman & Hall, 1990. 3

[89] W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 1970. 7.1.2

[90] Ji He, Jianshu Chen, Xiaodong He, Jianfeng Gao, Lihong Li, Li Deng, and Mari Ostendorf. Deep reinforcement learning with a natural language action space. *arXiv preprint arXiv:1511.04636*, 2015. 9.4

[91] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. (document), 5.4, 6, 6.3.3, 6.3.3, 6.6

[92] James Hensman, Magnus Rattray, and Neil D Lawrence. Fast variational inference in the conjugate exponential family. In *Advances in neural information processing systems*, pages 2888–2896, 2012. 8.1.2

[93] Daniel Hernández-Lobato, Jose Hernandez-Lobato, Amar Shah, and Ryan Adams. Predictive entropy search for multi-objective bayesian optimization. In *International Conference on Machine Learning*, pages 1492–1501, 2016. 7.3.5, 7.4.4, 8.1.4

[94] José Miguel Hernández-Lobato, Matthew W Hoffman, and Zoubin Ghahramani. Predictive Entropy Search for Efficient Global Optimization of Black-box Functions. In *NIPS*, 2014. 2.2, 7.1.1

[95] José Miguel Hernández-Lobato, Michael A Gelbart, Matthew W Hoffman, Ryan P Adams, and Zoubin Ghahramani. Predictive entropy search for bayesian optimization with unknown constraints. 2015. 7.1.1

[96] José Miguel Hernández-Lobato, James Requeima, Edward O Pyzer-Knapp, and Alán Aspuru-Guzik. Parallel and distributed thompson sampling for large-scale accelerated exploration of chemical space. *arXiv preprint arXiv:1706.01825*, 2017. 5, 8.1.4

[97] Matthew D Hoffman and Andrew Gelman. The no-u-turn sampler: adaptively setting path lengths in hamiltonian monte carlo. *Journal of Machine Learning Research*, 15(1): 1593–1623, 2014. 7.1.2

[98] Matthew W Hoffman and Bobak Shahriari. Modular mechanisms for bayesian optimization. In *NIPS Workshop on Bayesian Optimization*, pages 1–5. Citeseer, 2014. 7.1.2

[99] G. S. Hornby, A. Globus, D.S. Linden, and J.D Lohn. Automated Antenna Design with Evolutionary Algorithms. *American Institute of Aeronautics and Astronautics*, 2006. 1.1, 3

[100] D. Huang, T.T. Allen, W.I. Notz, and R.A. Miller. Sequential kriging optimization using multiple-fidelity evaluations. *Structural and Multidisciplinary Optimization*, 2006. 4, 4.2.5, 4.3, 4.3.4, B.1

[101] Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017. 6

[102] James P Hughes, Stephen Rees, S Barrett Kalindjian, and Karen L Philpott. Principles of early drug discovery. *British journal of pharmacology*, 162(6):1239–1249, 2011. 5.3, 9.3

[103] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Sequential Model-based Optimization for General Algorithm Configuration. In *LION*, 2011. 1.1, 6, 7.4, 9.2

[104] Brett W Israelsen, Nisar R Ahmed, Kenneth Center, Roderick Green, and Winston Bennett. Towards adaptive training of agent-based sparring partners for fighter pilots. In *AIAA Information Systems-AIAA Infotech@ Aerospace*, page 0343, 2017. 5

[105] Yuma Iwasaki, Ichiro Takeuchi, Valentin Stanev, Aaron Gilad Kusne, Masahiko Ishida, Akihiro Kirihara, Kazuki Ihara, Ryohto Sawada, Koichi Terashima, Hiroko Someya, et al. Machine-learning guided discovery of a high-performance spin-driven thermoelectric material. *arXiv preprint arXiv:1805.02303*, 2018. 9.3

[106] Thomas Jaksch, Ronald Ortner, and Peter Auer. Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research*, 11(Apr):1563–1600, 2010. 8.1, 8.1.3

[107] Janis Janusevskis, Rodolphe Le Riche, David Ginsbourger, and Ramunas Girdziusas. Expected Improvements for the Asynchronous Parallel Global Optimization of Expensive Functions: Potentials and Challenges. In *Learning and Intelligent Optimization*, 2012. 5, 1, 3

[108] Rodolphe Jenatton, Cedric Archambeau, Javier González, and Matthias Seeger. Bayesian optimization with tree-structured dependencies. In *International Conference on Machine Learning*, pages 1655–1664, 2017. 6, 6.4, 6.4, B.1

[109] Nan Jiang, Akshay Krishnamurthy, Alekh Agarwal, John Langford, and Robert E Schapire. Contextual decision processes with low bellman rank are pac-learnable. *arXiv preprint arXiv:1610.09512*, 2016. 6

[110] Shali Jiang, Gustavo Malkomes, Matthew Abbott, Benjamin Moseley, and Roman Garnett. Efficient nonmyopic batch active search. In *Advances in Neural Information Processing Systems*, pages 1107–1117, 2018. 8.1

[111] Thorsten Joachims. A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. Technical report, DTIC Document, 1996. 4.3.4

[112] D. R. Jones, C. D. Perttunen, and B. E. Stuckman. Lipschitzian Optimization Without the Lipschitz Constant. *J. Optim. Theory Appl.*, 1993. 4.2.4, 4.2.5, 4.3.4, 7.3.3, B.1

[113] Donald R. Jones, Matthias Schonlau, and William J. Welch. Efficient global optimization of expensive black-box functions. *J. of Global Optimization*, 1998. 2.2, 4.2.5, 4.3.4, 5.4, 6.3.3, B.1

[114] Pooria Joulani, Andras Gyorgy, and Csaba Szepesvári. Online learning under delayed feedback. In *International Conference on Machine Learning (ICML)*, 2013. 5

[115] Norman P Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, et al. In-datacenter performance analysis of a tensor processing unit. In *Computer Architecture (ISCA), 2017 ACM/IEEE 44th Annual International Symposium on*, pages 1–12. IEEE, 2017. 9.2

[116] Kwang-Sung Jun, Kevin Jamieson, Robert Nowak, and Xiaojin Zhu. Top arm identification in multi-armed bandits with batch arm pulls. In *Artificial Intelligence and Statistics (AISTATS)*, 2016. 5

[117] Kirthevasan Kandasamy and Yaoliang Yu. Additive approximations in high dimensional nonparametric regression via the salsa. In *International Conference on Machine Learning*, pages 69–78, 2016. 1.5, 3, 4.2.5, 7.2, 7.3.2, 7.4.3, 9.2

[118] Kirthevasan Kandasamy, Akshay Krishnamurthy, Barnabas Poczos, Larry Wasserman, and James M Robins. Influence functions for machine learning: Nonparametric estimators for entropies, divergences and mutual informations. *arXiv preprint arXiv:1411.4342*, 2014. 1.5, 9.2

[119] Kirthevasan Kandasamy, Akshay Krishnamurthy, Barnabas Poczos, Larry Wasserman, et al. Nonparametric von mises estimators for entropies, divergences and mutual informations. In *Advances in Neural Information Processing Systems*, pages 397–405, 2015. 1.5, 9.2

[120] Kirthevasan Kandasamy, Jeff Schenider, and Barnabás Póczos. High Dimensional Bayesian Optimisation and Bandits via Additive Models. In *International Conference on Machine Learning*, 2015. (document), 1.3, 3, 3, 14, 3.1.1, 1, 2, 4.2.4, 7.2, 7.3.2, 8.2

[121] Kirthevasan Kandasamy, Jeff Schneider, and Barnabás Póczos. Bayesian Active Learning for Posterior Estimation. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015. (document), 1.1, 1.2, 1.3, 7.4.2, 8.1, 8.1.4, 8.1.4, 8.2, 12

[122] Kirthevasan Kandasamy, Maruan Al-Shedivat, and Eric P Xing. Learning hmms with nonparametric emissions via spectral decompositions of continuous matrices. In *Advances in Neural Information Processing Systems*, pages 2865–2873, 2016. 1.5

[123] Kirthevasan Kandasamy, Gautam Dasarathy, Junier Oliva, Jeff Schenider, and Barnabás Póczos. Gaussian Process Bandit Optimisation with Multi-fidelity Evaluations. In *Advances in Neural Information Processing Systems*, 2016. (document), 1.3, 4, 4, 4.3, 4.3.1, 4.3.2, 4.3.2, 4.3.4, 4.6.3, 5.4

[124] Kirthevasan Kandasamy, Gautam Dasarathy, Junier B Oliva, Jeff Schneider, and Barnabas Poczos. Multi-fidelity Gaussian Process Bandit Optimisation. *arXiv preprint arXiv:1603.06288*, 2016. (document), 1.3, 4, 4

[125] Kirthevasan Kandasamy, Gautam Dasarathy, Jeff Schneider, and Barnabas Poczos. The Multi-fidelity Multi-armed Bandit. In *Advances in Neural Information Processing Systems*, 2016. (document), 1.3, 4, 3, 4.2.1, 4.2.3, 3

[126] Kirthevasan Kandasamy, Yoram Bachrach, Ryota Tomioka, Daniel Tarlow, and David

Carter. Batch policy gradient methods for improving neural conversation models. *arXiv preprint arXiv:1702.03334*, 2017. 1.5, 9.4

[127] Kirthevasan Kandasamy, Gautam Dasarathy, Jeff Schneider, and Barnabás Póczos. Multi-fidelity Bayesian Optimisation with Continuous Approximations. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1799–1808. JMLR. org, 2017. (document), 1.3, 4, 4.2.1, 5, 7.2, 7.4.1

[128] Kirthevasan Kandasamy, Jeff Schneider, and Barnabás Póczos. Query Efficient Posterior Estimation in Scientific Experiments via Bayesian Active Learning. *Artificial Intelligence*, 243:45–56, 2017. (document), 1.3, 12

[129] Kirthevasan Kandasamy, Akshay Krishnamurthy, Jeff Schneider, and Barnabás Póczos. Parallelised Bayesian Optimisation via Thompson Sampling. In *International Conference on Artificial Intelligence and Statistics*, pages 133–142, 2018. (document), 1.3, 2, 7, 8, 7.2, 8.1, 8.1.4

[130] Kirthevasan Kandasamy, Willie Neiswanger, Jeff Schneider, Barnabas Poczos, and Eric Xing. Neural Architecture Search with Bayesian Optimisation and Optimal Transport. In *Advances in Neural Information Processing Systems (NIPS)*, 2018. (document), 1.3, 9, 7.3.1, 7.3.2, 7.4.3

[131] Kirthevasan Kandasamy, Willie Neiswanger, Reed Zhang, Akshay Krishnamurthy, Jeff Schneider, and Barnabas Poczos. Myopic Bayesian Design of Experiments via Posterior Sampling and Probabilistic Programming. *arXiv preprint arXiv:1805.09964*, 2018. (document), 1.3, 11

[132] Kirthevasan Kandasamy, Willie Neiswanger, Reed Zhang, Akshay Krishnamurthy, Jeff Schneider, and Barnabas Poczos. Myopic Posterior Sampling for Adaptive Goal Oriented Design of Experiments. 2019. (document), 1.3, 11

[133] Kirthevasan Kandasamy, Karun Raju Vysyaraju, Willie Neiswanger, Biswajit Paria, Christopher R Collins, Jeff Schneider, Barnabas Poczos, and Eric P Xing. Tuning Hyperparameters without Grad Students: Scalable and Robust Bayesian Optimisation with Dragonfly. *arXiv preprint arXiv:1903.06694*, 2019. (document), 1.3, 7, 10

[134] Tarun Kathuria, Amit Deshpande, and Pushmeet Kohli. Batched gaussian process bandit optimization via determinantal point processes. In *Advances in Neural Information Processing Systems*, pages 4206–4214, 2016. 5, 2, 5.1, 7.2

[135] Emilie Kaufmann, Nathaniel Korda, and Rémi Munos. Thompson sampling: An asymptotically optimal finite-time analysis. In *ALT*, volume 12, pages 199–213. Springer, 2012. 2.2, 8.1.3

[136] Jaya Kawale, Hung H Bui, Branislav Kveton, Long Tran-Thanh, and Sanjay Chawla. Efficient thompson sampling for online matrix factorization recommendation. In *Advances in neural information processing systems*, pages 1297–1305, 2015. 8.1

[137] Michael Kearns and Satinder Singh. Near-optimal reinforcement learning in polynomial time. *Machine learning*, 49(2-3):209–232, 2002. 8.1, 8.1.3, 8.3.3

[138] Hiroaki Kitano. Designing neural networks using genetic algorithms with graph genera-

tion system. *Complex systems*, 4(4):461–476, 1990. 6

[139] A. Klein, S. Bartels, S. Falkner, P. Hennig, and F. Hutter. Towards efficient Bayesian Optimization for Big Data. In *BayesOpt*, 2015. 4, 2

[140] Joshua Knowles. Parego: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation*, 10(1):50–66, 2006. 7.3.5, 7.4.4

[141] Risi Imre Kondor and John Lafferty. Diffusion kernels on graphs and other discrete input spaces. In *ICML*, volume 2, pages 315–322, 2002. 6.2.2

[142] Andreas Krause and Carlos Guestrin. Nonmyopic Active Learning of Gaussian Processes: An Exploration-exploitation Approach. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 449–456, New York, NY, USA, 2007. ACM. 8.2

[143] Andreas Krause and Cheng S Ong. Contextual gaussian process bandit optimization. In *Advances in Neural Information Processing Systems*, pages 2447–2455, 2011. 9.3, 9.4

[144] Andreas Krause, Ajit Singh, and Carlos Guestrin. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research*, 9(Feb):235–284, 2008. 5.3, 47, 5.5.2

[145] Akshay Krishnamurthy, Kirthevasan Kandasamy, Barnabas Poczos, and Larry Wasserman. Nonparametric estimation of renyi divergence and friends. In *International Conference on Machine Learning*, pages 919–927, 2014. 9.2

[146] Akshay Krishnamurthy, Kirthevasan Kandasamy, Barnabas Poczos, and Larry A Wasserman. On estimating l22 divergence. In *AISTATS*, 2015. 9.2

[147] Akshay Krishnamurthy, Alekh Agarwal, and Miro Dudik. Contextual semibandits via supervised learning oracles. In *Advances In Neural Information Processing Systems*, pages 2388–2396, 2016. 9.4

[148] Akshay Krishnamurthy, Zhiwei Steven Wu, and Vasilis Syrgkanis. Semiparametric contextual bandits. *arXiv preprint arXiv:1803.04204*, 2018. 9.4

[149] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images, 2009. 5.4, 6.4

[150] Harold J Kushner. A New Method of Locating the Maximum Point of An Arbitrary Multipeak Curve in the Presence of Noise. *Journal of Basic Engineering*, 1964. 7.1.1

[151] T. L. Lai and H. Robbins. Asymptotically Efficient Adaptive Allocation Rules. *Adv. in App. Math.*, 1985. 4.1, 4.4.3

[152] David Landau and Kurt Binder. *A Guide to Monte Carlo Simulations in Statistical Physics*. Cambridge University Press, New York, NY, USA, 2005. ISBN 0521842387. 8.2

[153] Chun-Liang Li, Kirthevasan Kandasamy, Barnabás Póczos, and Jeff Schneider. High dimensional bayesian optimization via restricted projection pursuit models. In *Artificial Intelligence and Statistics*, pages 884–892, 2016. 3

[154] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar.

Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization. *arXiv preprint arXiv:1603.06560*, 2016. 4, 7.4.3

[155] Chenxi Liu, Barret Zoph, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. *arXiv preprint arXiv:1712.00559*, 2017. 6, 6.4

[156] Hanxiao Liu, Karen Simonyan, Oriol Vinyals, Chrisantha Fernando, and Koray Kavukcuoglu. Hierarchical representations for efficient architecture search. *arXiv preprint arXiv:1711.00436*, 2017. 6, 6.4

[157] Jun S. Liu. *Monte Carlo strategies in Scientific computing*. Springer, 2001. ISBN 0387952306. 8.2

[158] Yao Liu and Emma Brunskill. When simple exploration is sample efficient: Identifying sufficient conditions for random exploration to yield pac rl algorithms. *arXiv:1805.09045*, 2018. 8.1

[159] Daniel Lizotte, Tao Wang, Michael Bowling, and Dale Schuurmans. Automatic gait optimization with gaussian process regression. In *in Proc. of IJCAI*, pages 944–949, 2007. 1.1

[160] M. Osborne and D. Duvenaud and R. Garnett and C. Rasmussen and S. Roberts and Z. Ghahramani. Active Learning of Model Evidence Using Bayesian Quadrature. In *Neural Information Processing Systems (NIPS)*, 2012. 8.2, 2, 8.2.1

[161] Yifei Ma, Roman Garnett, and Jeff Schneider. Active Area Search via Bayesian Quadrature. In *International Conference on Artificial Intelligence and Statistics*, 2014. 8.2

[162] Yifei Ma, Tzu-Kuo Huang, and Jeff G Schneider. Active search and bandits on graphs using sigma-optimality. In *UAI*, pages 542–551, 2015. 8.1.3

[163] Yifei Ma, Dougal J. Sutherland, Roman Garnett, and Jeff G. Schneider. Active Pointillistic Pattern Search. In *International Conference on Artificial Intelligence and Statistics, AISTATS*, 2015. 8.1

[164] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008. 6.2.4

[165] Ian Grant Macdonald. *Symmetric functions and Hall polynomials*. Oxford university press, 1998. 7.2, B.1

[166] David J. C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003. 8.2

[167] Gustavo Malkomes, Charles Schaff, and Roman Garnett. Bayesian optimization for automated model selection. In *Advances in Neural Information Processing Systems*, pages 2900–2908, 2016. 7.1.2

[168] Jean-Michel Marin, Pierre Pudlo, Christian P. Robert, and Robin J. Ryder. Approximate Bayesian computational methods. *Statistics and Computing*, 22(6):1167–1180, 2012. 8.2, 8.2.2, B.1

[169] Paul Marjoram, John Molitor, Vincent Plagnol, and Simon Tavaré. Markov Chain Monte

Carlo without Likelihoods. *Proceedings of the National Academy of Sciences of the United States of America*, 100(26):15324–15328, 2003. 8.2, 8.2.2

[170] R. Martinez-Cantin, N. de Freitas, A. Doucet, and J. Castellanos. Active Policy Learning for Robot Planning and Exploration under Uncertainty. In *Proceedings of Robotics: Science and Systems*, 2007. 1.1

[171] Ruben Martinez-Cantin. Locally-biased bayesian optimization using nonstationary gaussian processes. In *Neural Information Processing Systems (NIPS) workshop on Bayesian Optimization*, 2015. 7.3.4

[172] Bruno T Messmer and Horst Bunke. A new algorithm for error-tolerant subgraph isomorphism detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5): 493–504, 1998. 6, 6.2.2

[173] Risto Miikkulainen, Jason Liang, Elliot Meyerson, Aditya Rawal, Dan Fink, Olivier Francon, Bala Raju, Arshak Navruzyan, Nigel Duffy, and Babak Hodjat. Evolving deep neural networks. *arXiv preprint arXiv:1703.00548*, 2017. 6

[174] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013. 9.4

[175] J.B. Mockus and L.J. Mockus. Bayesian approach to global optimization and application to multiobjective and constrained problems. *Journal of Optimization Theory and Applications*, 1991. 8.2

[176] Jonas Mockus. Application of Bayesian approach to numerical methods of global and stochastic optimization. *Journal of Global Optimization*, 1994. 2.2

[177] Radford M Neal. Slice sampling. *Annals of statistics*, pages 705–741, 2003. 7.1.2

[178] Renato Negrinho and Geoff Gordon. DeepArchitect: Automatically Designing and Training Deep Architectures. submitted, 2017. 6

[179] Willie Neiswanger and Eric Xing. Post-inference prior swapping. *arXiv preprint arXiv:1606.00787*, 2016. 9.4

[180] Willie Neiswanger, Chong Wang, and Eric Xing. Asymptotically exact, embarrassingly parallel mcmc. *arXiv preprint arXiv:1311.4780*, 2013. 9.4

[181] Willie Neiswanger, Chong Wang, and Eric Xing. Embarrassingly parallel variational inference in nonconjugate models. *arXiv preprint arXiv:1510.04163*, 2015. 8.1.2

[182] Willie Neiswanger, Kirthevasan Kandasamy, Barnabas Poczos, Jeff Schneider, and Eric Xing. Probo: a framework for using probabilistic programming in bayesian optimization. *arXiv preprint arXiv:1901.11515*, 2019. 9.4

[183] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functionsi. *Mathematical Programming*, 14(1): 265–294, 1978. 8.1.2

[184] Ian Osband and Benjamin Van Roy. Near-optimal reinforcement learning in factored mdps. In *Advances in Neural Information Processing Systems*, pages 604–612, 2014. 8.1

[185] Ian Osband, Daniel Russo, and Benjamin Van Roy. (more) efficient reinforcement learning via posterior sampling. In *Advances in Neural Information Processing Systems*, pages 3003–3011, 2013. 8.1, 9.4

[186] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. In *Advances in Neural Information Processing Systems (NIPS)*, 2016. 5

[187] Michael Osborne, Roman Garnett, Zoubin Ghahramani, David K Duvenaud, Stephen J Roberts, and Carl E Rasmussen. Active learning of model evidence using bayesian quadrature. In *Advances in neural information processing systems*, pages 46–54, 2012. 8.1

[188] Long Ouyang, Michael Henry Tessler, Daniel Ly, and Noah Goodman. Practical optimal experiment design with probabilistic programs. *arXiv preprint arXiv:1608.05046*, 2016. 8.1, 8.1.2

[189] Sinno Jialin Pan, Qiang Yang, et al. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010. 9.3

[190] Biswajit Paria, Kirthevasan Kandasamy, and Barnabás Póczos. A flexible multi-objective bayesian optimization approach using random scalarizations. *arXiv preprint arXiv:1805.12168*, 2018. (document), 1.3, 1.5, 7.3.5, 7.21, 8.1.4, B.1

[191] David Parkinson, Pia Mukherjee, and Andrew R Liddle. A Bayesian model selection analysis of WMAP3. *Physical Review*, 2006. 1.1, 3, 4, 8.1.4, 8.2

[192] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014. 7.4.4

[193] Kedar M Perkins, Chetali Gupta, Emily N Charleson, and Newell R Washburn. Surfactant properties of pegylated lignins: Anomalous interfacial activities at low grafting density. *Colloids and Surfaces A: Physicochemical and Engineering Aspects*, 530:200–208, 2017. 9.3

[194] Gareth W. Peters, Y. Fan, and Scott A. Sisson. On sequential Monte Carlo, partial rejection control and approximate Bayesian computation. *Statistics and Computing*, 22(6):1209–1222, 2012. 8.2.2

[195] Gabriel Peyré and Marco Cuturi. *Computational Optimal Transport*. Available online, 2017. 6.2.2, 6.5

[196] Victor Picheny. Multiobjective optimization using gaussian process emulators via stepwise uncertainty reduction. *Statistics and Computing*, 25(6):1265–1280, 2015. 7.3.5

[197] Matthias Poloczek, Jialei Wang, and Peter I Frazier. Multi-Information Source Optimization. *arXiv preprint arXiv:1603.00389*, 2016. 4

[198] Wolfgang Ponweiser, Tobias Wagner, Dirk Biermann, and Markus Vincze. Multiobjective optimization on a limited budget of evaluations using model-assisted metric selection. In *International Conference on Parallel Problem Solving from Nature*, pages 784–794. Springer, 2008. 7.3.5, 7.4.4

[199] Chao Qin, Diego Klabjan, and Daniel Russo. Improving the expected improvement algo-

rithm. In *Advances in Neural Information Processing Systems*, pages 5381–5391, 2017. 7.1.1

[200] Kent Quanrud and Daniel Khashabi. Online learning with adversarial delays. In *Advances in Neural Information Processing Systems (NIPS)*, 2015. 5

[201] Tom Rainforth. *Automating Inference, Learning, and Design using Probabilistic Programming*. PhD thesis, PhD thesis, 2017. 8.1

[202] PS Rana. Physicochemical properties of protein tertiary structure data set, 2013. 6.4

[203] C.E. Rasmussen and C.K.I. Williams. *Gaussian Processes for Machine Learning*. Adaptative computation and machine learning series. University Press Group Limited, 2006. 2.1, 2.1, 2.2, 1, 3.1.2, 4.2.4, 4.3.4, 5.1

[204] Pradeep Ravikumar, John Lafferty, Han Liu, and Larry Wasserman. Sparse Additive Models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 2009. 3

[205] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Quoc Le, and Alex Kurakin. Large-scale evolution of image classifiers. *arXiv preprint arXiv:1703.01041*, 2017. 6

[206] Herbert Robbins. Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, 1952. 8.1

[207] Herbert Robbins. Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, 1952. 2.3, 4.1

[208] Christian P. Robert and George Casella. *Monte Carlo Statistical Methods (Springer Texts in Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005. 8.2

[209] Paul Rolland, Jonathan Scarlett, Ilija Bogunovic, and Volkan Cevher. High-dimensional bayesian optimization via additive models with overlapping groups. *arXiv preprint arXiv:1802.07028*, 2018. 3

[210] Stephane Ross and J Andrew Bagnell. Reinforcement and imitation learning via interactive no-regret learning. *arXiv preprint arXiv:1406.5979*, 2014. 8.3.1

[211] Daniel Russo and Benjamin Van Roy. Learning to optimize via posterior sampling. *Mathematics of Operations Research*, 39(4):1221–1243, 2014. 2.2, 2, 5.5.2, 5.5.2

[212] Daniel Russo and Benjamin Van Roy. An information-theoretic analysis of thompson sampling. *The Journal of Machine Learning Research*, 17(1):2442–2471, 2016. 8.1

[213] Daniel Russo and Benjamin Van Roy. An Information-theoretic analysis of Thompson sampling. *Journal of Machine Learning Research*, 2016. 2.2, 2.4, 9, 8.1.3, 8.3.2

[214] A Sabharwal, H Samulowitz, and G Tesauro. Selecting near-optimal learners via incremental data allocation. In *AAAI*, 2015. 4

[215] John Salvatier, Thomas V Wiecki, and Christopher Fonnesbeck. Probabilistic programming in python using pymc3. *PeerJ Computer Science*, 2:e55, 2016. 7.1.2

[216] MW. Seeger, SM. Kakade, and DP. Foster. Information Consistency of Nonparametric Gaussian Process Methods. *IEEE Transactions on Information Theory*, 2008. 2.2, 3.3, 15, 3.3, 3.3, 1

[217] Rajat Sen, Kirthevasan Kandasamy, and Sanjay Shakkottai. Multi-fidelity black-box optimization with hierarchical partitions. In *International Conference on Machine Learning*, pages 4545–4554, 2018. 4

[218] Rajat Sen, Kirthevasan Kandasamy, and Sanjay Shakkottai. Noisy blackbox optimization with multi-fidelity queries: A tree search approach. *arXiv preprint arXiv:1810.10482*, 2018. 4

[219] Sambu Seo, Marko Wallat, Thore Graepel, and Klaus Obermayer. Gaussian Process Regression: Active Data Selection and Test Point Rejection. In *International Joint Conference on Neural Networks*, 2000. 8.2, 8.2.1, B.1

[220] Burr Settles. Active Learning Literature Survey. Technical report, University of Wisconsin-Madison, 2010. 8.2.1

[221] Amar Shah and Zoubin Ghahramani. Parallel predictive entropy search for batch global optimization of expensive objective functions. In *Advances in Neural Information Processing Systems*, pages 3330–3338, 2015. 5, 3, 7.2

[222] Bobak Shahriari, Ziyu Wang, Matthew W Hoffman, Alexandre Bouchard-Côté, and Nando de Freitas. An Entropy Search Portfolio for bayesian Optimization. *arXiv preprint arXiv:1406.4625*, 2014. 2.2, 7.1.1

[223] Xuedong Shang, Emilie Kaufmann, and Michal Valko. Adaptive black-box optimization got easier: Hct only needs local smoothness. In *European Workshop on Reinforcement Learning*, 2017. 4.2.3

[224] John Shawe-Taylor, Nello Cristianini, et al. *Kernel methods for pattern analysis*. Cambridge university press, 2004. 7.2, 7.3.2

[225] VK Shchigolev. Calculating luminosity distance versus redshift in flrw cosmology via homotopy perturbation method. *Gravitation and Cosmology*, 23(2):142–148, 2017. 7.4.2

[226] David Silver and Demis Hassabis. Alphago: Mastering the ancient game of go with machine learning. *Research Blog*, 2016. 9.4

[227] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017. 9.2, 9.4

[228] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. (document), 6, 6.3.3, 6.6, 6.4

[229] John Skilling. Nested sampling for general Bayesian computation. *Bayesian Anal.*, 2006. 8.2

[230] Alexander J Smola and Risi Kondor. Kernels and regularization on graphs. In *Learning theory and kernel machines*, pages 144–158. Springer, 2003. 6.2.2

[231] Edward Snelson and Zoubin Ghahramani. Sparse gaussian processes using pseudo-inputs. In *Advances in neural information processing systems*, pages 1257–1264, 2006. 9.2

[232] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical Bayesian Optimization of Machine Learning Algorithms. In *Advances in Neural Information Processing Systems*, 2012. 1.1, 2.1, 4.2.4, 6, 6, 6.3.3, 7.1.2, 7.3.2, 7.4, 8.1.4

[233] Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mr Prabhat, and Ryan Adams. Scalable bayesian optimization using deep neural networks. In *International Conference on Machine Learning*, pages 2171–2180, 2015. 9.2

[234] Jialin Song, Yuxin Chen, and Yisong Yue. A general framework for multi-fidelity bayesian optimization with gaussian processes. *arXiv preprint arXiv:1811.00755*, 2018. 4

[235] Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias Seeger. Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design. In *International Conference on Machine Learning*, 2010. (document), 2.1, 2.2, 1, 1, 2.2, 1, 2.4, 5, 3, 3.1.2, 1, 4.2.3, 4.3.2, 4.3.4, 4.5.1, 4.5.3, 4.6.2, 4.6.2, 5.4, 5.5.2, 8.1.2, 8.1.3, 2, 8.2, B.1

[236] Bharath Sriperumbudur et al. On the optimal estimation of probability measures in weak and strong topologies. *Bernoulli*, 22(3):1839–1893, 2016. 9.2

[237] Kenneth O Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127, 2002. 6

[238] Alexander L Strehl, Lihong Li, and Michael L Littman. Reinforcement learning in finite mdps: Pac analysis. *Journal of Machine Learning Research*, 10(Nov):2413–2444, 2009. 8.1

[239] Yanan Sui, Alkis Gotovos, Joel Burdick, and Andreas Krause. Safe exploration for optimization with gaussian processes. In *International Conference on Machine Learning*, pages 997–1005, 2015. 9.3

[240] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998. 6, 9.4

[241] Kevin Swersky, Jasper Snoek, and Ryan P Adams. Multi-task bayesian optimization. In *NIPS*, 2013. 4

[242] Kevin Swersky, Jasper Snoek, and Ryan Prescott Adams. Freeze-thaw bayesian optimization. *arXiv preprint arXiv:1406.3896*, 2014. (document), 7.4, 7.2

[243] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015. 6

[244] M. Tegmark et al. Cosmological Constraints from the SDSS Luminous Red Galaxies. *Physical Review*, December 2006. (document), 3.2, 7.14, 7.4.2, 8.1.4, 8.2.2

[245] Lothar Thiele, Kaisa Miettinen, Pekka J Korhonen, and Julian Molina. A preference-based evolutionary algorithm for multi-objective optimization. *Evolutionary computation*, 17 (3):411–436, 2009. 7.3.5

[246] W. R. Thompson. On the Likelihood that one Unknown Probability Exceeds Another in View of the Evidence of Two Samples. *Biometrika*, 1933. (document), 2.2, 5.2, 6, 8.1,

8.1, B.1

[247] Joaquín Torres-Sospedra, Raúl Montoliu, Adolfo Martínez-Usó, Joan P Avariento, Tomás J Arnau, Mauri Benedito-Bordonau, and Joaquín Huerta. Ujiindoorloc: A new multi-building and multi-floor database for wlan fingerprint-based indoor localization problems. In *Indoor Positioning and Indoor Navigation (IPIN), 2014 International Conference on*, pages 261–270. IEEE, 2014. 6.2.4, 6.4, 7.4.3

[248] Dustin Tran, Matthew D Hoffman, Rif A Saurous, Eugene Brevdo, Kevin Murphy, and David M Blei. Deep probabilistic programming. *arXiv preprint arXiv:1701.03757*, 2017. 8.1.2, 8.1.4, 9.4

[249] Long Tran-Thanh, Lampros C. Stavrogiannis, Victor Naroditskiy, Valentin Robu, Nicholas R. Jennings, and Peter Key. Efficient Regret Bounds for Online Bid Optimisation in Budget-Limited Sponsored Search Auctions. In *UAI*, 2014. 4

[250] Chris Urmson, Joshua Anhalt, Drew Bagnell, Christopher Baker, Robert Bittner, MN Clark, John Dolan, Dave Duggins, Tugrul Galatali, Chris Geyer, et al. Autonomous Driving in Urban Environments: Boss and the Urban Challenge. *Journal of Field Robotics*, 2008. 4

[251] Dana Van Aken, Andrew Pavlo, Geoffrey J Gordon, and Bohan Zhang. Automatic database management system tuning through large-scale machine learning. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 1009–1024. ACM, 2017. 1.1

[252] Vladimir Naumovich Vapnik and Vlamimir Vapnik. *Statistical learning theory*. Wiley New York, 1998. 4

[253] Cédric Villani. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008. 6, 6.2.3

[254] Paul A. Viola and Michael J. Jones. Rapid Object Detection using a Boosted Cascade of Simple Features. In *Computer Vision and Pattern Recognition*, 2001. 3.2, 4.2.5, 7.4.4

[255] S Vichy N Vishwanathan, Nicol N Schraudolph, Risi Kondor, and Karsten M Borgwardt. Graph kernels. *Journal of Machine Learning Research*, 11(Apr):1201–1242, 2010. 6, 6.2.2

[256] Walter D Wallis, Peter Shoubridge, M Kraetz, and D Ray. Graph distances using graph union. *Pattern Recognition Letters*, 22(6-7):701–704, 2001. 6, 6.2.2

[257] Jialei Wang, Scott C Clark, Eric Liu, and Peter I Frazier. Parallel bayesian global optimization of expensive functions. *arXiv preprint arXiv:1602.05149*, 2016. 3, 5, 1, 3

[258] Yingfei Wang and Warren B Powell. Finite-time analysis for the knowledge-gradient policy. *SIAM Journal on Control and Optimization*, 56(2):1105–1129, 2018. 8.1

[259] Zi Wang and Stefanie Jegelka. Max-value entropy search for efficient bayesian optimization. *arXiv preprint arXiv:1703.01968*, 2017. 7.1.1

[260] Zi Wang, Bolei Zhou, and Stefanie Jegelka. Optimization as estimation with gaussian processes in bandit settings. In *Artificial Intelligence and Statistics*, pages 1022–1031, 2016. 7.1.1

[261] Zi Wang, Chengtao Li, Stefanie Jegelka, and Pushmeet Kohli. Batched high-dimensional bayesian optimization via structural kernel learning. *arXiv:1703.01973*, 2017. 5

[262] Zi Wang, Clement Gehring, Pushmeet Kohli, and Stefanie Jegelka. Batched large-scale bayesian optimization in high-dimensional spaces. In *AISTATS*, 2018. 5, 7.2

[263] Zi Wang, Beomjoon Kim, and Stefanie Jegelka. Regret bounds for meta bayesian optimization with an unknown gaussian process prior. In *Advances in Neural Information Processing Systems*, 2018. 9.2

[264] Ziyu Wang, Masrour Zoghi, Frank Hutter, David Matheson, and Nando de Freitas. Bayesian Optimization in High Dimensions via Random Embeddings. In *International Joint Conference on Artificial Intelligence*, 2013. 3, 3, 3.2, 8.2.2, B.1

[265] Larry Wasserman. All of nonparametric statistics. *New York*, 2006. B.1

[266] Kai Wei, Rishabh Iyer, and Jeff Bilmes. Submodularity in data subset selection and active learning. In *International Conference on Machine Learning*, pages 1954–1963, 2015. 8.1.2

[267] Rebecca Willett, Robert Nowak, and Rui M Castro. Faster rates in regression via active learning. In *Advances in Neural Information Processing Systems*, pages 179–186, 2006. 8.1.4

[268] Andrew Wilson and Hannes Nickisch. Kernel interpolation for scalable structured gaussian processes (kiss-gp). In *International Conference on Machine Learning*, pages 1775–1784, 2015. 9.2

[269] Andrew Gordon Wilson, Christoph Dann, and Hannes Nickisch. Thoughts on massively scalable gaussian processes. *arXiv preprint arXiv:1511.01870*, 2015. 9.2

[270] Jian Wu and Peter Frazier. The parallel knowledge gradient method for batch bayesian optimization. In *Advances In Neural Information Processing Systems*, pages 3126–3134, 2016. 5, 3

[271] Thorsten Wuest, Daniel Weimer, Christopher Irgens, and Klaus-Dieter Thoben. Machine learning in manufacturing: advantages, challenges, and applications. *Production & Manufacturing Research*, 4(1):23–45, 2016. 9.3

[272] Yingce Xia, Haifang Li, Tao Qin, Nenghai Yu, and Tie-Yan Liu. Thompson Sampling for Budgeted Multi-Armed Bandits. In *IJCAI*, 2015. 4

[273] Lingxi Xie and Alan Yuille. Genetic cnn. *arXiv preprint arXiv:1703.01513*, 2017. 6

[274] Shifeng Xiong, Peter Z. G. Qian, and C. F. Jeff Wu. Sequential design and analysis of high-accuracy and low-accuracy computer codes. *Technometrics*, 2013. 4.2.5, 4.2.5

[275] Yichong Xu, Hongyang Zhang, Kyle Miller, Aarti Singh, and Artur Dubrawski. Noise-tolerant interactive learning using pairwise comparisons. In *Advances in Neural Information Processing Systems*, pages 2431–2440, 2017. 9.4

[276] Yichong Xu, Sivaraman Balakrishnan, Aarti Singh, and Artur Dubrawski. Interactive linear regression with pairwise comparisons. 2018. 9.4

[277] Daniel Yamins, David Tax, and James S. Bergstra. Making a Science of Model Search:

Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures. In *International Conference on Machine Learning*, 2013. 3

[278] Kaifeng Yang, Longmei Li, André Deutz, Thomas Back, and Michael Emmerich. Preference-based multiobjective optimization using truncated expected hypervolume improvement. In *Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), 2016 12th International Conference on*, pages 276–281. IEEE, 2016. 7.3.5

[279] C. Zhang and K. Chaudhuri. Active Learning from Weak and Strong Labelers. In *NIPS*, 2015. 4

[280] Qingfu Zhang, Wudong Liu, Edward Tsang, and Botond Virginas. Expensive multiobjective optimization by moea/d with gaussian process model. *IEEE Transactions on Evolutionary Computation*, 14(3):456–474, 2010. 7.3.5

[281] Zhao Zhong, Junjie Yan, and Cheng-Lin Liu. Practical network blocks design with q-learning. *arXiv preprint arXiv:1708.05552*, 2017. 6

[282] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016. 6, 6.4

[283] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. *arXiv preprint arXiv:1707.07012*, 2017. 6