POST-INFERENCE METHODS FOR SCALABLE PROBABILISTIC MODELING AND SEQUENTIAL DECISION MAKING

WILLIE NEISWANGER

August 2019
CMU-ML-19-113

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

Thesis Committee

Eric P. Xing, Chair
Jeff Schneider
Ruslan Salakhutdinov
Ryan P. Adams (Princeton University)
Yee Whye Teh (University of Oxford and DeepMind)

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

This research was supported by the National Science Foundation award DBI0546594, the Air Force Research Laboratory award FA87501220324, and the Office of Naval Research awards N000141410684 and N000141712463

Keywords: machine learning, probabilistic modeling, Bayesian inference, prior knowledge, sequential decision making, Bayesian optimization, distributed inference, parallel algorithms

To my family and friends :)

ABSTRACT

Probabilistic modeling refers to a set of techniques for modeling data that allows one to specify assumptions about the processes that generate data, incorporate prior beliefs about models, and infer properties of these models given observed data. Benefits include uncertainty quantification, multiple plausible solutions, reduction of overfitting, better performance given small data or large models, and explicit incorporation of *a priori* knowledge and problem structure. In recent decades, an array of inference algorithms have been developed to estimate these models.

This thesis focuses on *post-inference methods*, which are procedures that can be applied after the completion of standard inference algorithms to allow for increased efficiency, accuracy, or parallelism when learning probabilistic models of big data sets. These methods also allow for scalable computation in distributed or online settings, incorporation of complex prior information, and better use of inference results in downstream tasks. A few examples include:

- **Embarrassingly parallel inference.** Large data sets are often distributed over a collection of machines. We first compute an inference result (e.g. with Markov chain Monte Carlo or variational inference) on each machine, in parallel, without communication between machines. Afterwards, we combine the results to yield an inference result for the full data set.
- **Prior swapping.** Certain model priors limit the number of applicable inference algorithms, or increase their computational cost. We first choose any "convenient prior" (e.g. a conjugate prior, or a prior that allows for computationally cheap inference), and compute an inference result. Afterwards, we use this result to efficiently perform inference with other, more sophisticated priors or regularizers.
- **Sequential decision making and optimization.** Model-based sequential decision making and optimization methods use models to define acquisition functions. We compute acquisition functions using the inference result from any probabilistic program or model framework, and perform efficient inference in sequential settings.

We also describe the benefits of combining the above methods, present methodology for applying the embarrassingly parallel procedures when the number of machines is dynamic or unknown at inference time, illustrate how these methods can be applied for spatiotemporal analysis and in covariate dependent models, show ways to optimize these methods by incorporating test-functions of interest, and demonstrate how these methods can be implemented in probabilistic programming frameworks for automatic deployment.

ACKNOWLEDGEMENTS

There are quite a few people who I'd like to thank for their impact on this thesis and on my life in Pittsburgh over the past few years.

To begin with, I want to thank my advisor Eric Xing for his support and guidance. I've had a great time in the SAILING lab over these years, and by fostering such a rich academic environment with many friendly and talented group members, I have been able to learn so much.

I'd like to thank my batchmates in the Machine Learning Department: Benjo Cowley, David (Wei) Dai, Kirstin Early, Jessica Chemali, Junier Olivia, Micol Marchetti-Bowick, and Nicole Rafidi. Each of you has had a large impact on my life. I'd also like to thank my close MLD friends in nearby years: Avinava Dubley, Calvin Murdock, Kirthevasan Kandasamy, Anthony Platanios, Dan Schwartz, Christoph Dann, Mariya Toneva, Otilia Stretcu, Mrinmaya Sachan, Pete Lund, Manzil Zaheer, George Montanez, and Yu-Xiang Wang. Thanks also to Dan Howarth, Tim Hyde, Gabi Huffman, and Lacey Konopasek. I am very grateful as well to my older friends in MLD for introducing me to the department and for their friendship: Ankur Parikh, Aaditya Ramdas, Leila Wehbe, Ina Fiterau, Akshay Krishnamurthy, Jing Xiang, and Yifei Ma. Finally, I must give a huge thanks to Diane Stidle, who makes life a lot easier for all of us in MLD.

I've had a number of long term collaborators and mentors who have been great sources of help, instruction, and inspiration for my research. I want to thank Chong Wang, Jeff Schneider, Barnabas Poczos, Frank Wood, Chris Wiggins, and Viveka Mayya. A particular thanks to my collaborator and startup co-conspirator Kirthevasan Kandasamy. I want to thank my younger collaborators as well: Youngseog Chung, Ian Char, Hai Pham, Ksenia Korovina, and Yusha Liu. I've learned a lot (and continue to learn) from each of you. Additionally, I'm grateful to other members of my thesis committee for their time and feedback: Yee Whye Teh, Ryan P. Adams, and Ruslan Salakhutdinov.

To the SAILING Lab: thanks for making our weekly (sometimes more) meetings fun over the years, and something I could always look forward to. I want to give a particular shout-out to Ben Lengerich, Bryon Aragam, Haohan Wang, Maruan Al-Shedivat, Xun Zheng, Zhiting Hu, Jinliang Wei, Pengtao Xie, Aurick Qiao, Yaoliang Yu, Qirong Ho, Lisa Lee, Helen Zhou, Xiaodan Liang, and Amy Protos. And I'd like to say thanks to a few others in the broader CMU community: Newell Washburn, Kevin Tran, Dougal Sutherland, Jimmy Williams, Bill McDowell, and Colin White.

I truly appreciate my Group-X friends for their camaraderie, consistency, and hard work over the years, and in particular want to say thanks to Karen Edwards, Caitlin Tenison, Chris MacLellan, Cassie Eng, Chris Collins, Nick Golio, Bowen Yu, Josh Gyory, Min Kyung Lee, Suyoun Kim, Juliet Shafto, Yigit Yakupoglu, and Zachary

Lee. I've also had many great gym friends in grad school including Ankur Parikh, Veeranjaneyulu Sadhanala, Ryan Tibshirani, Yaoliang Yu, and Xun Zheng. Thanks for keeping me honest and spotting me. And a big thanks to my friends from the climbing wall: Steph Wilson, Ada Zhang, Natalie Sauerwald, Bentley Wingert, DJ Grant, Kartik Goyal, and the inexhaustible Ramsey Hanna.

I need to give a big thanks to my Bloomfield & Lawrenceville friends (and sometimes roommates, research collaborators, travel buddies, and movie subjects): Matt Barnes, Zhe Zhang, Maria De-Arteaga, and Benedikt Boecking. Thanks as well to some of my great friends who I have met over the years, sprinkled throughout the city of Pittsburgh: Jan-Tosh Gerling, Alexandria Maruca, Avigayil Diamond, Elijah Lawrence, Mia Davis, Nathan Colosimo, and Janna Arnold. Finally, thank you to my very talented movie-making friends, including Louise Zhang, Tori Vilseck, Lauren Kunde, John Cantine, and Ben Speiser—I had a wonderful time working with you all at Filmmakers. Thanks also to Spi Cat, Eggs, and Princess.

To my past and present partners, I want to express my sincere gratitude for being sources of support, patience, and happiness as I slogged through graduate school and pursued this thesis. Thank you Amanda, Kristie, and especially Vanessa (sorry that I've been up so late working on this, I am nearly done!).

Finally, I would like to give a heartfelt thanks to my mom, my dad, and my twin sister Lila. Thanks for helping me get started on this path, and for supporting me along the way. I think we are all glad that I am finally finishing up.

PUBLICATIONS

Some ideas and figures have appeared previously in the following publications:

1. Willie Neiswanger, Frank Wood, and Eric P Xing. "The dependent dirichlet process mixture of objects for detection-free tracking and object modeling." In: *The Seventeenth International Conference on Artificial Intelligence and Statistics (AISTATS 2014)*. 2014

2. Willie Neiswanger, Chong Wang, Qirong Ho, and Eric P Xing. "Modeling citation networks using latent random offsets." In: *Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence (UAI 2014)*. 2014

3. Willie Neiswanger, Chong Wang, and Eric Xing. "Asymptotically Exact, Embarrassingly Parallel MCMC." in: *Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence (UAI 2014)*. 2014

4. Willie Neiswanger, Chong Wang, and Eric Xing. "Embarrassingly Parallel Variational Inference in Nonconjugate Models." In: *arXiv preprint arXiv:1510.04163*. 2015

5. François Caron, Willie Neiswanger, Frank Wood, Arnaud Doucet, and Manuel Davy. "Generalized Pólya urn for time-varying Pitman-Yor processes." In: *Journal of Machine Learning Research (JMLR)* 18.27 (2017)

6. Willie Neiswanger, Xue Liu, and Eric Xing. "Low Communication Distributed Black Box VI." in: *The International Conference on Probabilistic Programming*. 2018

7. Willie Neiswanger, Avinava Dubey, Chong Wang, and Eric Xing. *Embarrassingly Parallel MCMC in Quasi-ergodic Settings*. Tech. rep. 2016

8. Willie Neiswanger. *Embarrassingly Parallel MCMC for Fast and Flexible Analysis of Spatiotemporal Data*. Tech. rep. 2016

9. Willie Neiswanger and Eric Xing. "Post-inference prior swapping." In: *Proceedings of the 34th International Conference on Machine Learning—Volume 70 (ICML 2017)*. 2017, pp. 2594–2602

10. Willie Neiswanger, Kirthevasan Kandasamy, Barnabas Poczos, Jeff Schneider, and Eric P. Xing. "ProBO: Versatile Bayesian Optimization Using Any Probabilistic Programming Language." In: *arXiv preprint arXiv:1901.11515* (2019)

CONTENTS

LIST OF FIGURES

LIST OF ALGORITHMS

ACRONYMS

MCMC Markov chain Monte Carlo

SMC Sequential Monte Carlo

PMCMC Particle Markov chain Monte Carlo

HMC Hamiltonian Monte Carlo

IS Importance sampling

VI Variational inference

BBVI Black box variational inference

PPL Probabilistic programming language

DDPMO Dependent Dirichlet process mixture of objects

LRO Latent random offset

PS Prior swapping

BO Bayesian optimization

DOE Design of experiments

ProBO Probabilistic programming Bayesian optimization

1 INTRODUCTION

Probabilistic modeling allows us to incorporate assumptions about the processes that generate data, explore prior beliefs about our models, infer properties of these models when given observations, and quantify uncertainty about our inferences. These types of models are often called probabilistic graphical models, or generative latent variable models. They are typically defined via a generative process that produces data $x \in \mathcal{X}$, and consist of latent variables $\theta \in \Theta$, a prior distribution $p(\theta)$ over these variables, and a generating distribution $p(x|\theta)$ that describes how the data is generated given θ. Bayesian inference algorithms are used to compute or approximate distributions associated with a given model. One primary task of these algorithms is to infer the conditional distribution over the latent variables given the data, known as the posterior distribution $p(\theta|x)$. Posterior inference can become more difficult in the following settings:

- **Big data:** when datasets contain large numbers of observations, there can be a high computational cost of many inference algorithms.
- **Distributed data:** when data are partitioned over multiple machines or locations, it may be difficult for inference algorithms to operate in these distributed settings without high communication costs, especially if the data are private and cannot be moved or pooled.
- **Streaming data:** when data are collected in a streaming manner, it may be difficult to process the data and perform correct inference, especially when we want to use multiple machines to process the data without storing it.
- **Rich priors:** when we have complicated or sophisticated prior modeling assumptions, there may be a much higher computational cost of many inference algorithms.

This thesis concerns new techniques that allow posterior inference to be more-easily performed in the above settings. It also focuses on the development of new models and procedures for downstream tasks in computer vision, network and text analysis, and in model-based sequential decision making and optimization.

1.1 POST-INFERENCE METHODS

Algorithmically, our main strategy involves *post-inference methods*. These are procedures that can be applied after the completion of standard inference algorithms to provide increased efficiency, accuracy, or parallelism when learning probabilistic

models. These methods allow for scalable computation in distributed or online settings, incorporation of complex prior information, and better use of inference results in downstream tasks. Three examples of post-inference methods are:

1. **Embarrassingly parallel inference.** Large data sets are often distributed over a collection of machines. We first compute an inference result (e.g. with Markov chain Monte Carlo or variational inference) on each machine, in parallel, without communication between machines. Afterwards, we combine the results to yield an inference result for the full data set. These methods can allow for efficient inference on large, distributed, and streaming datasets.

2. **Prior swapping.** Certain model priors limit the number of applicable inference algorithms, or increase their computational cost. We first choose any "convenient prior" (e.g. a conjugate prior, or a prior that allows for computationally cheap inference), and compute an inference result. Afterwards, we use this result to efficiently perform inference with other, more sophisticated priors or regularizers. This also lets us efficiently incorporate new or updated prior information, post inference. These methods can allow for efficient inference on models with complex priors, and can also aid in the application of embarrassingly parallel inference methods.

3. **Sequential Decision Making and Optimization** Model-based sequential decision making and optimization methods use probabilistic models to define acquisition functions, which are used to determine subsequent queries or decisions. Today, Gaussian process models are predominantly used, which allow for easy computation of acquisition functions. However, we may wish to use a broader set of modeling tools and techniques. We develop methods to compute acquisition functions using the inference result from any probabilistic program or model framework, and to perform efficient inference in sequential settings.

In this thesis, we aim to develop scalable inference methods that can be applied to large, streaming, and distributed datasets, and to probabilistic models with complex priors. We also aim to demonstrate how these methods can help probabilistic modeling in downstream applications. Towards this end, we develop new Bayesian models for text, network, and video data, and new procedures for flexibly defining and using models for improved sequential decision making and Bayesian optimization.

Concretely, in Part 1 of this thesis, we present new probabilistic models for tasks in computer vision and analysis of citation networks; in Part 2, we focus on embarrassingly parallel inference methods, which allow for more efficient inferences given large and distributed datasets; and in Part 3, we focus on methods to flexibly incorporate useful structure into both models and model-based sequential decision making procedures. This includes methods for prior swapping, which allow for more efficient inferences given models with complex priors, and a system called

ProBO, which more easily allows for models defined using different probabilistic modeling frameworks to be used for model-based sequential decision making and optimiation.

In this thesis, we also describe strategies for combining embarrassingly parallel and prior swapping methods, present methodology for applying the parallel procedures when the number of machines is dynamic or unknown at inference time, develop randomized algorithms for efficient application of post-inference methods in distributed environments, show ways to optimize the post-inference methods by incorporating test-functions of interest, outline how these methods can be applied to aid in the analysis of spatiotemporal and streaming data, and demonstrate how these methods can be implemented in probabilistic programming frameworks for automatic deployment.

1.2 USE CASES FOR PROBABILISTIC MODELING

In this section, we aim to describe the benefits of probabilistic modeling over other non-probabilistic or non-Bayesian strategies for machine learning, give application areas where it is useful, and detail various downstream applications.

WHY IS PROBABILISTIC MODELING USEFUL? It is important to consider and summarize the benefits of probabilistic modeling over other strategies for modeling and machine learning, and discuss the settings where it is most useful. To be concrete, we use the term *probabilistic modeling* to mean using latent variable probabilistic graphical models to define a generative process, and then using the mechanisms of Bayesian inference to update our prior beliefs about the latent variables to posterior beliefs about these variables given a set of observed data. A few of the main benefits of probabilistic modeling are that it can

- Quantify uncertainty over estimated quantities: performing inference in probabilistic models returns a posterior distribution over possible models (or some function of this distribution).
- Provide multiple plausible solutions: inference algorithms can be used to compute multiple models, of which each could plausibly explain an observed set of data.
- Allow for explicit incorporation of prior knowledge and problem structure: this additional information can be incorporated both by specifying how the data is generated and by specifying prior distributions over latent variables.
- Help reduce overfitting: prior distributions in these models often provide regularization, which helps reduce overfitting to observed (training) data.
- Yield better performance given small data or large models: due to the incorporation of constraints, regularization, and additional model structure, proba-

bilistic models can perform well when learning large models, i.e. models with many parameters relative to the number of observations.

A few example areas of machine learning where the above properties are useful include unsupervised machine learning, such as generative modeling, or analysis of large unstructured data; supervised machine learning that requires uncertainty quantification, such as Bayesian classification or Gaussian process regression; and sequential decision making, such as Bayesian optimization or Bayesian design of experiments.

USE CASES IN INDUSTRY We now describe a few popular real-world and industry applications in which these models are used.

Topic modeling, such as the method of latent Dirichlet allocation (LDA) [28] is used for feature exactraction for text and web corpora [88, 211], user modeling, and recommendation systems, by many large technology companies. It is also used for data compression, representation, and embedding. Related models are probabilistic clustering models, record linkage models [179], and probabilistic matrix factorization models [160]. Inference in these models can be carried out with sampling methods (e.g. Gibbs sampling), variational inference (e.g. stochastic variational inference), or optimization to a maximum a posteriori (MAP) point estimate.

Bayesian versions of standard parametric models are often used when it is beneficial to quantify the uncertainty of estimation results. For example, in simple classification or regression tasks, Bayesian logistic regression [66, 67] or Bayesian linear regression [113] are used in industry settings. Inference in these models can sometimes be carried out exactly (depending on the model and choice of prior) or via sampling or variational inference methods.

Bayesian optimization is a popular method for query-efficient hyperparameter tuning and model selection [174] in industry [47, 167]. A popular recent use case is for tuning the hyperparameters of large neural network models [75]. Bayesian models are particularly useful here, because Bayesian optimization algorithms leverage the models' uncertainty estimates to effectively manage the tradeoff between exploration and exploitation, which allows them to achieve efficient optimization performance [32]. For this application, Gaussian process models are typically used, where inference can often be carried out exactly.

Hierarchical Bayesian models (particularly parametric regression and classification models) are used in marketing and decision sciences. Here, the ability to incorporate structure between groups or hierarchies of data, quantification of uncertainty over estimated quantities, and incorporation of specific prior assumptions, is particularly useful [157]. In these settings, even when data grows large, there may still be high levels of uncertainty over fine-grained or local components of models, such as those corresponding to individual consumers [156].

Deep generative models are used for generating complex, structured, or high dimensional data (such as images) [76, 100, 154]. A majority of these models have

been developed in recent times and the broad potential for industry impact is still being explored.

1.3 A FEW KEY CONCEPTS AND NOTATION

Here we include common notation and definitions used throughout this thesis, describe the distributed and streaming data settings, and formalize the typical goals of Bayesian inference.

Data. Suppose we have n data points in p dimensions, denoted

$$x^n = (x_1, \ldots, x_n) \in \mathcal{X} \subseteq \mathbb{R}^{p \times n}. \tag{1}$$

Likelihood. Assume that the data are drawn from a class of models parameterized by $\theta \in \Theta \subseteq \mathbb{R}^d$, with a likelihood function

$$\mathcal{L}(\theta) = p(x^n|\theta) \tag{2}$$

where $p(x^n|\theta)$ denotes the probability density function (PDF) of a conditional distribution over $\mathcal{X}$.

Prior. Suppose we've chosen a prior distribution over Θ, with PDF $p(\theta)$. Note that, at various points in this thesis, when noted, we denote the prior PDF as $\pi(\theta)$.

Joint distribution. The likelihood and prior can be used to define a joint distribution over $\Theta \times \mathcal{X}$, with PDF

$$p(\theta, x^n) = p(\theta)p(x^n|\theta). \tag{3}$$

Posterior distribution. In Bayesian inference, we are interested in the posterior distribution, a conditional of this joint distribution, with PDF defined to be

$$p(\theta|x^n) = \frac{p(\theta)p(x^n|\theta)}{\int p(\theta)p(x^n|\theta)\, d\theta} = \frac{p(\theta)p(x^n|\theta)}{p(x^n)} \tag{4}$$

Data-distributed setting. Suppose that we have paritioned our data into M groups, i.e.

$$x^n = \{x_1^{n_1}, \ldots, x_M^{n_L}\}, \tag{5}$$

$$\text{where } x_m^{n_m} = (x_{m,1}, \ldots, x_{m,n_m}) \in \mathcal{X} \subseteq \mathbb{R}^{p \times n_m} \tag{6}$$

and the m^{th} set has n_m data points. We will refer to these as local data sets. We can also write the full or aggregate set of data by $x^n = \bigcup_{m=1}^{M} x_m^{n_m}$.

Factorization assumption. We assume that each local data set is independent given a parameter θ, i.e. that the likelihood factorizes in the following way:

$$\mathcal{L}(\theta) = p\left(\bigcup_{m=1}^{M} x_m^{n_m} | \theta\right) = \prod_{m=1}^{M} p(x_m^{n_m} | \theta). \tag{7}$$

Posterior expectation. When performing Bayesian inference, we are often interested in the following task: for a chosen prior $p(\theta)$ and set of observations x^n, sample from or compute the PDF of the associated target posterior $p(\theta|x^n)$—or, more generally, for some test function $h(\theta)$, compute the expectation

$$\mu_h = \mathbb{E}_p\left[h(\theta)\right] \tag{8}$$

with respect to the target posterior.

1.4 AN INTUITIVE MODEL OVERVIEW

In this section we give a non-technical overview of different model types that will be encountered in this thesis. Namely, we describe models with global latent variables, models with local latent variables, and models that are dependent on some covariate. In the following chapters, we will develop post-inference methods—e.g. algorithms that allow for distributed inference or efficient inferences with additional prior information—for these different model types.

1.4.1 *Global Variable Models*

An initial class of models to consider is the set of probabilistic models for a dataset $x_{i\,i=1}^{n}$ with a single global latent variable θ. A couple examples of this are:

- Bayesian logistic regression: in this model [66, 67], a global parameter dictates the probability of binary observations.
- Bayesian neural networks: in this model [119, 135], a global parameter corresponding to neural network weights yields predictions.

Note that the global parameter θ is associated with (i.e. is assumed to control generation of) the full set of data $x_{i\,i=1}^{n}$. We visualize a graphical model with a global latent variable, using plate notation, in Figure 1 (left).

1.4.2 *Local Variable Models*

Some models have local variables associated with data points or subsets of data, possibly in addition to global variables associated with an entire data set. A couple examples of this are:

- Mixture models: in some formulations [22, 49], there are assignment latent variables for each data point, which specify the mixture component that a given data belongs to.
- Topic models: in some formulations [28, 78] there are latent variables corresponding to each document in a corpus, and other latent variables corresponding to each word in a document.

We visualize a graphical model with local latent variables, using plate notation, in Figure 1 (right). Here, we assume that each observation x_i has an associated local latent variable z_i.

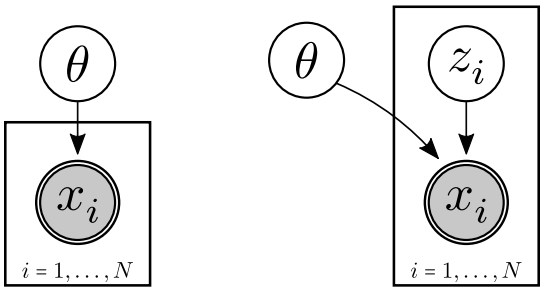


Figure 1: Model with global latent variables θ (left), and model with both global latent variables θ and local latent variables z_i (right).

1.4.3 *Covariate Dependent Models*

Some models have repetitive structure that depends on a certain covariate, such as time, spatial position, or network structure. In this thesis, we refer to these models as *dependent models*. Dependent models can have both local and global latent variables. A couple examples of this are:

- Hidden Markov models: in HMMs [72, 159], there is typically a latent variable at each time step, which is generated dependent on the latent variable at the previous time step. Observations are drawn given the latent variable at each time step.
- Network dependent models: in dependent models with network covariates [57, 102], a directed network is defined over a collection of models (where each model is associated with a node of the graph), and each model is dependent on its parents in the graph.

We visualize a covariate dependent graphical model with both global and local latent variables, using plate notation, in Figure 2. Here, the dependent model is the graphical model with local variables shown in Figure 1 (right), and the covariate is

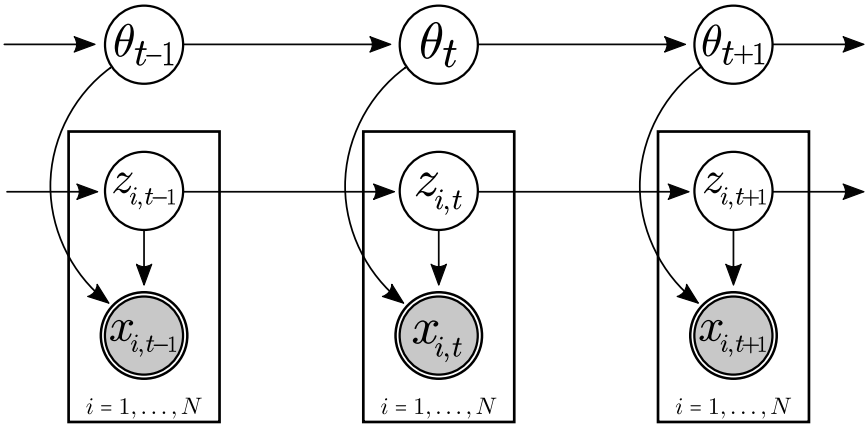


Figure 2: Covariate dependent model with both global and local time-varying latent variables (θ_t and $z_{i,t}$, respectively). Here, the covariate is the time step t.

the time step t for a sequence $1, \ldots, T$. We assume that both global and local latent variables are dependent on the respective latent variables of the previous time step.

In the following sections we give examples of Bayesian models for spatio-temporal and network data. These models are typically dependent on some covariate, such as time or network structure. We will define these probabilistic graphical models and describe typical inference algorithms for these models. In later chapters, we will develop new inference algorithms for models like these, and show how these models and inference procedures can be used for sequential decision making.

1.5 OVERVIEW OF PROBABILISTIC PROGRAMMING

Probabilistic programming languages (PPLs) are modern tools for specifying Bayesian models and performing inference. At a high level, most PPLs:

1. Allow users to specify a Bayesian model, including the generative process that yields data, the unknown (latent) model parameters, and prior beliefs over these latent parameters.
2. Allows users to provide a set of observed data.
3. Compute and return the posterior beliefs over the latent model parameters in an automatic fashion.

PPLs allow for easy incorporation of prior knowledge and model structure, composition of models, quick deployment, and automatic inference, often in the form of samples from or variational approximations to a posterior distribution. PPLs may be used to specify and run inference in a variety of models, such as graphical models, GPs, deep Bayesian models, hierarchical models, and implicit (simulator-based) models, to name a few [7, 21, 38, 50, 117, 121, 127, 161, 186, 202]. In this

thesis, we will discuss the interplay between post-inference methods, probabilistic programming, and sequential decision making.

1.6 CHAPTER OVERVIEW

In Part 1, we develop probabilistic models for text, network, and video data, and derive approximate inference algorithms for these models. In Chapter 2, we introduce the latent random offset (LRO) model for citation networks. In Chapter 3, we introduce the dependent Dirichlet process mixture of objects for detection-free tracking and object modeling.

In Part 2, we present algorithms for scalable approximate inference on big data and in distributed settings. In Chapter 4, we introduce methods for embarrassingly parallel Markov chain Monte Carlo (MCMC). In Chapter 5, we introduce methods for embarrassingly parallel variational inference (VI), and for low-communication black box variational inference (BBVI). In Chapter 6, we describe methods for embarrassingly parallel inference in quasi-ergodic settings and in dependent models or models with local latent variables.

In Part 3, we focus on methods that allow for the incorporation of structure: either prior structure in models, or model structure in model-based sequential decision making and optimization procedures. In Chapter 7, we introduce methods for prior swapping for efficient incorporation of prior information. In Chapter 8, we introduce methods for allowing arbitrary probabilistic models, defined via probabilistic programs, to be used in Bayesian optimization and other sequential decision making procedures.

We visualize the components and contributions of this thesis (with a focus on applications, models, and inference methods) in Figure 3.

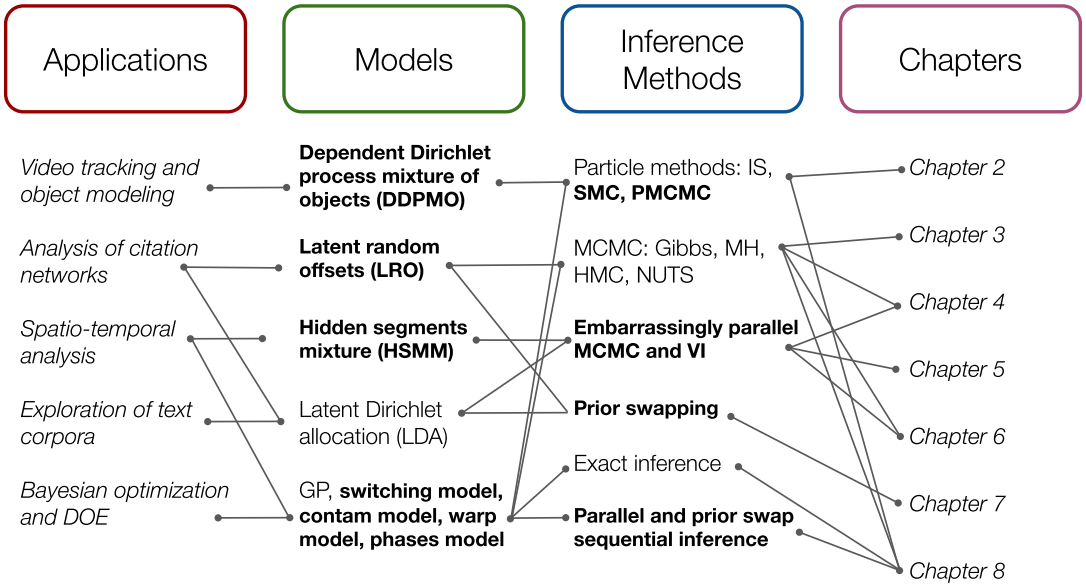


Figure 3: A visualization of the topics and chapters in this thesis. The primary contributions of this thesis are in **bold**.

Part I

PROBABILISTIC MODELS FOR TEXT, NETWORK, AND VIDEO DATA

2

MODELING CITATION NETWORKS USING LATENT RANDOM OFFSETS

2.1 CHAPTER SUMMARY

Out of the many potential factors that determine which links form in a document citation network, two in particular are of high importance: first, a document may be cited based on its subject matter—this can be modeled by analyzing document content; second, a document may be cited based on which other documents have previously cited it—this can be modeled by analyzing citation structure. Both factors are important for users to make informed decisions and choose appropriate citations as the network grows. In this chapter, we present a novel model that integrates the merits of content and citation analyses into a single probabilistic framework. We demonstrate our model on three real-world citation networks. Compared with existing baselines, our model can be used to effectively explore a citation network and provide meaningful explanations for links while still maintaining competitive citation prediction performance.

2.2 INTRODUCTION

Many large citation networks—Wikipedia, arXiv, and PubMed[1], to name a few—continue to quickly grow in size, and the structure of these networks continues to increase in complexity. To effectively explore large-scale and complex data like these and extract useful information, users rely more and more on various types of guidance for help. An important type of guidance comes from the citations (or links) in the network. Citations serve as paths that users can easily follow, and do not require users to specify certain keywords in advance. In scientific research, for example, researchers often find potentially interesting articles by following citations made in other articles. In Wikipedia, users often find explanations of certain terms by following the links made by other Wikipedia users. Thus, generating relevant citations is important for many users who may frequently rely on these networks to explore data and find useful information.

We believe that, among many, two important factors largely determine how a document citation network is formed: the documents' contents and the existing citation structure. Take as an example a citation network of computer science articles. A research paper about "support vector machines (SVMs)", for instance, might be cited

1 http://www.wikipedia.org/, http://arxiv.org/, and http://www.ncbi.nlm.nih.gov/pubmed

by several other articles that develop related methods, based on the subject matter alone. This type of information can be well captured by analyzing the content of the documents. However, the existing citation structure is also important. If this SVM paper included great results on a computer vision dataset, for example, it might be cited by many vision papers that are *not* particularly similar in content. Though different in content, this SVM paper could be very important to users in a different topic area, and should be considered by these users when choosing citations. This type of information cannot be easily captured by analyzing document content, but can be discovered by analyzing the existing citation structure among documents while studying the contents of the papers that generated these citations.

Given these observations, we present a probabilistic model to accurately model citation networks by integrating content and citation/link information into a single framework. We name our approach a *latent random offset* (LRO) model. The basic idea is as follows: we first represent the content of each document using a latent vector representation (i.e. "topics") that summarizes the document content. Then, each latent representation is augmented in an additive manner with a random offset vector; this vector models information from the citation structure that is not well captured by document content. The final augmented representation is then used to model how this document is cited by other documents. To motivate this representation, we present sample outputs from running LRO on the Simple English Wikipedia.

Examples from Simple English Wikipedia. The first graph in the top row of Figure 4 shows, for the *Sistine Chapel* article in the Simple English Wikipedia, the latent vector representation, which is concentrated around three topics: `countries` (*italy, italian, china, russian*), `Christianity` (*church*, *christ*, *jesus*, *god*), and `architecture` (*built, side, large, design*). Here we've listed the top four words in each topic (in parens). The incoming links to the *Sistine Chapel* article are also shown; these citing documents determine the random offsets for *Sistine Chapel*. The random offsets can be thought of as "corrections" to the latent vector representation, based on the content of citing documents—for example, the two largest positive offsets are `Christianity` (*church, christ, jesus, god*) and `Anglicanism` (*english, knight, translated, restoration*), meaning that the citing documents strongly exhibit these two topics (compared to the *Sistine Chapel* article). On the other hand, there is a large negative offset on `architecture` (*built, side, large, design*), indicating that the citing documents do not exhibit this topic as much as *Sistine Chapel*.

Notably, the topic `Anglicanism` (containing words related to Christianity in England) is found in the random offsets for *Sistine Chapel*, but is absent from its latent vector representation. This is because the Sistine Chapel is in the Vatican City, and thus its article does not emphasize content relating to England or Anglicanism (even though they are all related to Christianity). However, documents that link to *Sistine Chapel*, such as *Chapel*, talk about the Anglican Church in England. This is an ex-

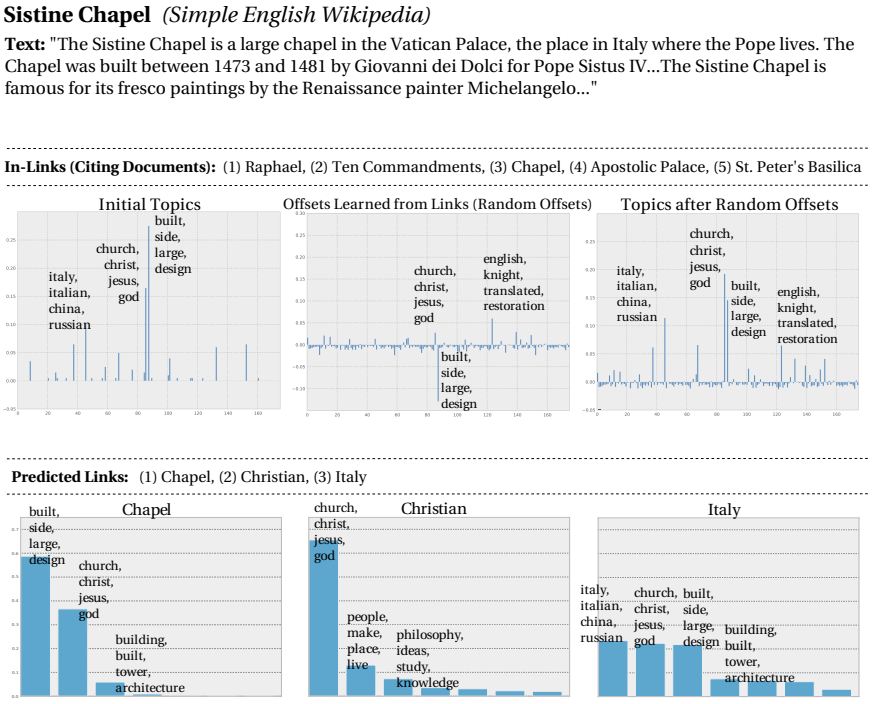


Figure 4: Analysis of content, latent offsets, and predicted links for the *Sistine Chapel* document in the Simple English Wikipedia dataset. The first row shows an example passage from the document. The next row shows the names of the documents that cite *Sistine Chapel*. The next row shows the initial latent topics (first column), the latent offsets learned from links (second column), and the latent topics after applying the offsets (third column). The final row shows interpretable link predictions; for each predicted link, we show the relative weight that each latent topic contributed to the prediction.

ample where pertinent information is found in the citation structure, but not in the document content. By capturing this citation information, the LRO model provides insights into the context surrounding a document.

Following this idea, we can add the latent vector and random offsets together to obtain the "augmented representation" of a document (i.e. the "topics after random offsets" graph in Figure 4), which takes into account not just its content, but the content of its citing documents as well. Link predictions in the LRO model are based upon the intuition that a document i cites document j only if both documents have similar representations. This intuition is captured in the bottom row of graphs in Figure 4, which explains three out-links predicted by the LRO model for the *Sistine Chapel* document. For each predicted link, we show the topics that contributed most to the prediction, and not surprisingly, the most important topics for each link also feature strongly in the augmented representation for the *Sistine Chapel*. Knowing which topics contributed to the prediction of links not only helps users interpret existing links within a document corpus, but also gives users an explanation for

every new link predicted by the LRO model—for instance, a user might invoke LRO to recommend citations for an academic paper, and such "link explanations" give the user a quick overview of why each recommendation is relevant.

We note that of the three predicted out-links for *Sistine Chapel*, two of them (*Chapel*, *Italy*) are actual out-links in *Sistine Chapel*, while the third, *Christian*, is obviously relevant but not found in the document. This motivates another application of LRO: predicting relevant but missing links in document corpora; in this case, we are completing the references for a Wikipedia article. Another application context is academic paper writing: LRO can be used to recommend important (but otherwise overlooked) citations for a newly-written academic paper.

The rest of this chapter is organized as follows: we begin by formalizing latent random offset modeling, and then show how we can use it to model citation networks. We then develop a fast learning algorithm with linear complexity in the size of the number of citations, and empirically evaluate our approach using three real-world citation networks. Compared with several baselines, our model not only improves citation prediction performance, but also provides meaningful explanations for citations within the networks. By studying latent random offset representations, we show these explanations can be used to effectively interpret why our model predicts links for given documents and to explore citation networks.

2.3 LATENT RANDOM OFFSET MODELS

We introduce the general framework of latent random offsets for citation network modeling. Suppose our citation network consists of D documents (i.e. nodes), $\mathcal{D} = \{x_1, x_2, ..., x_D\}$. We use $y_{ij} = 1$ or 0 to indicate whether document i cites document j or not. Note that y_{ij} is *directed*, meaning y_{ij} is not necessarily the same as y_{ji}.

Each document x_j is usually a high-dimensional vector in $\mathbb{R}^V$, where V is the vocabulary size, so it is desirable to represent x_j using a low-dimensional vector θ_j. In other words, the mapping

$$\theta_j = \theta_j(x_j) \tag{9}$$

serves as a summarization of the original document content x_j, and these summarizations can be used to measure the content similarities of different documents.

However, in real citation networks, a document can be cited by others for reasons outside of its content information. For example, a target document might provide an influential idea that can be used in many different fields and thus be cited by a diverse set of documents. This information is encoded not in the document content but in the citation network structure. We choose to model this phenomenon by allowing a random offset vector ϵ_j to augment the low-dimensional vector θ_j, which gives the augmented representation

$$\nu_j = \theta_j + \epsilon_j. \tag{10}$$

The offset vector ϵ_j is used to capture the network structure information that is *not* contained in the document's content. One important property of this augmented representation is that the random offset ϵ_j is aligned in the same space as θ_j. If the dimension of θ_j has some semantic explanations, then ϵ_j can be understood as modifications of those explanations.

Finally we consider using a function f to model the citation from document i to document j, such that

$$f(\theta_i, \theta_j + \epsilon_j) \approx y_{ij} \quad \text{(for all } i, j\text{)}$$

where y_{ij} is the citation indicator from document i to document j. Notice the asymmetric structure here for document i and j—we do not consider the offset vector ϵ_i for document i in our function f. In real citation networks, when a new document joins the citation network by citing some other documents, this new document is effectively "not in" the network. It will be most likely to cite other documents based only on their content and their citations, as no network information exists for this new document. One advantage of this formulation is that we can make citation predictions for a brand new document by only using its content information.

In the next two sections, we first describe how we create the low-dimensional document content representation θ_j and how we use the latent random offset model for citation network modeling.

2.3.1 *Probabilistic Topic Models for Representing the Contents of Documents*

There are many potential ways to create the low-dimensional document content representation described in Eq. 9. Here we choose to use probabilistic topic models. Topic models [24] are used to discover a set of "topics" (or themes) from a large collection of documents. These topics are distributions over terms, which are biased to be associated under a single theme. One notable property of these models is that they often provide an interpretable low-dimensional representation of the documents [41]. They have been used for tasks like corpus exploration [39], information retrieval [197] and recommendation [190].

Here we describe the simplest topic model, latent Dirichlet allocation (LDA) [26] and use it to create the low-dimensional document content representations. Assume there are K topics, β_k, $k = 1, ..., K$ and each β_k is a distribution over a fixed vocabulary. For each document j, the generative process is as follows,

1. Draw topic proportions $\theta_j \sim \text{Dirichlet}(\alpha)$
2. For each word x_{jn} in document j,
   a) Draw topic assignment $z_{jn} \sim \text{Mult}(\theta_j)$
   b) Draw word $x_{jn} \sim \text{Mult}(\beta_{z_{jn}})$

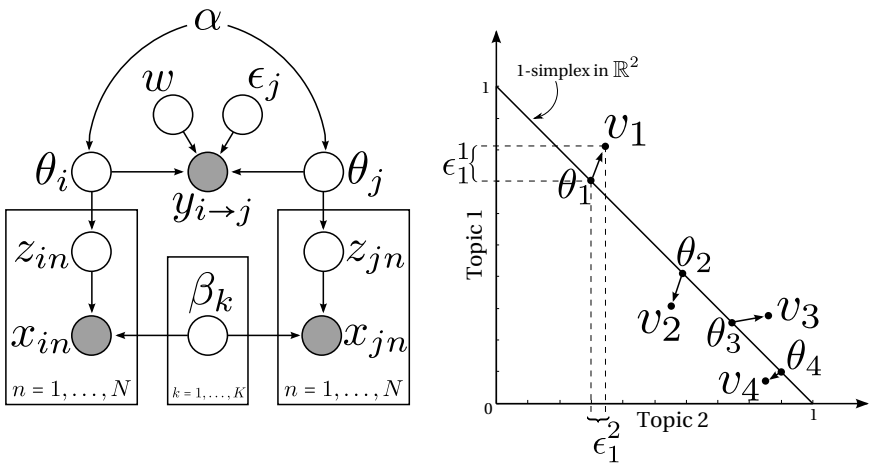


Figure 5: Left: The LRO graphical model. Only two documents (i and j) and one citation (from i to j) are shown. The augumented latent representation representation for document j is $\nu_j = \theta_j + \epsilon_j$. Right: An illustration of the random offsets. We show each document's content vector θ_j (which lies on the simplex), its offsets ϵ_j due to link structure (the superscript indicates the dimension for ϵ_j), and the resulting augmented latent representation ν_j.

This process describes how the words of a document are generated from a mixture of topics that are shared by the corpus. The topic proportions θ_j are document-specific and we use these topic proportions as our low-dimensional document content representation.

Given a document collection, the only observations are the words in the documents. The topics, topic proportions for each document, and topic assignments for each word, are all latent variables that have to be determined from the data. LDA has been extensively studied in the literature and many efficient algorithms have been proposed to fit the LDA model variables [26, 86, 172]. For example, standard learning algorithms like variational EM or Gibbs sampling can be used to estimate these quantities [26]. These methods give us the estimated document content representations θ_j in terms of an approximate posterior distribution or point estimates.

2.3.2 *Modeling Citations via Random Offsets*

Having described how we represent the documents in a low dimensional space, we now consider how to create the augmented representations introduced in Eq. 10. We model our latent random offset vector ϵ_j with a multivariate Gaussian distribution

$$\epsilon_j \sim \mathcal{N}(0, \lambda^{-1} I_K).$$

where λ is a scalar precision parameter for the latent random offsets.

Using the general idea of latent random offset modeling shown in Eq. 10 and probabilistic topic models described in Section 2.3.1, our *latent random offset model*

(LRO) for citation network modeling has the following generative process (Figure 5 shows the graphical model). Assuming K topics, $\beta_{1:K}$,

1. For each document j,
   a) Draw topic proportions $\theta_j \sim \text{Dirichlet}(\alpha)$
   b) Draw latent random offset $\epsilon_j \sim \mathcal{N}(0, \lambda^{-1} I_K)$ and set the document augmented representation as $\nu_j = \theta_j + \epsilon_j$
   c) For each word x_{jn},
      i. Draw topic assignment $z_{jn} \sim \text{Mult}(\theta)$
      ii. Draw word $x_{jn} \sim \text{Mult}(\beta_{z_{jn}})$
2. For each *directed* pair of documents (i, j), draw the citation indicator

$$y_{ij} \sim \mathcal{N}(y | w\theta_i^\top \nu_j, \tau_{ij}^{-1}).$$

where $w \in \mathbb{R}_+$ is a global scaling parameter to account for potential inefficiencies of the topic proportions θ_i, which are constrained to the simplex.[2] We chose a Gaussian response to model the citations, in similar fashion to [190]. Notation τ_{ij}^{-1} is the precision parameter for the Gaussian distribution. Here, we choose to stray from a formal generative process and also treat the y_{ij} as parameters, such that τ_{ij} satisfies

$$\tau_{ij} = \begin{cases} \tau_1 & \text{if } y_{ij} = 1 \\ \tau_0 & \text{if } y_{ij} = 0 \,. \end{cases}$$

In this formulation, τ_1 specifies the precision if a link exists from document i to j, while τ_0 is for the case where the link does not exist. We set τ_0 to be much smaller (i.e. higher noise) than τ_1 — this is similar to the assumption made in [190], which models the fact that $y_{ij} = 0$ could either mean it is not appropriate for document i to cite document j, or simply that document i should cite document j but has inadvertently neglected to cite it. This also enables a fast learning algorithm with complexity linear in the number of citations (See Section 2.4 for details).

The expectation of the citation can be computed as

$$\mathbb{E}[y_{ij}] = w\theta_i^\top \nu_j = w(\theta_i^\top \theta_j) + w(\theta_i^\top \epsilon_j).$$

This reveals how likely it is for a citation from document i to document j to occur under our model. If the documents have similar content or document j has certain large positive offsets, it is more likely to be cited by document i.

For a document j, our latent representation θ_j is over a simplex. In Figure 5 (right), we show how the random offsets ϵ_j produce the augmented representation ν_j.

2 Our experiments show that optimizing the global scaling parameter w is important for obtaining good results.

2.3.3 *Citation Prediction*

In a system for citation prediction, it is more realistic to suggest citations than to make hard decisions for the users. This is common in many recommender systems [89, 190]. For a particular document i, we rank the potential citations according to the score

$$S_{ij} = w\theta_i^\top \nu_j,$$

for all other documents j, and suggest citations based on this score (excluding document i and all pre-existing citations).

2.4 LEARNING ALGORITHM

We use maximum a posteriori (MAP) estimation to learn the latent parameters of the LRO, where we perform a coordinate ascent procedure to carry out the optimization. Maximization of the posterior is equivalent to maximizing the complete log likelihood of $\nu_{1:D}$, $\theta_{1:D}$ and $\beta_{1:K}$, which we can write as

$$\begin{aligned}\mathcal{L} = &-\frac{\lambda}{2}\sum_j (\nu_j - \theta_j)^\top(\nu_j - \theta_j) - \sum_{i\neq j}\frac{\tau_{ij}}{2}(y_{ij} - w\theta_i^\top \nu_j)^2 \\ &+ \sum_j \sum_n \log\left(\sum_k \theta_{jk}\beta_{k,x_{jn}}\right).\end{aligned}$$

where we have omitted a constant and set $\alpha = 1$.

First, given topics $\beta_{1:K}$ and augmented representations $\nu_{1:D}$, for all documents, we describe how to learn the topic proportions θ_j. We first define $\phi_{jnk} = q(z_{jn} = k)$. Then we separate the items that contain θ_j and apply Jensen's inequality,

$$\begin{aligned}\mathcal{L}(\theta_j) &\geqslant -\frac{\lambda}{2}\sum_j (\nu_j - \theta_j)^\top(\nu_j - \theta_j) \\ &\quad + \sum_n \sum_k \phi_{jnk}\left(\log \theta_{jk}\beta_{k,x_{jn}} - \log \phi_{jnk}\right) \\ &= \mathcal{L}(\theta_j, \mathbf{ffi_j}).\end{aligned}$$

where $\mathbf{ffi_j} = (\phi_{jnk})_{n=1,k=1}^{D\times K}$. The optimal ϕ_{jnk} then satisfies

$$\phi_{jnk} \propto \theta_{jk}\beta_{k,x_{jn}}.$$

The $\mathcal{L}(\theta_j, \mathbf{ffi_j})$ gives the *tight* lower bound of $\mathcal{L}(\theta_j)$. We cannot optimize θ_j analytically, but we can use the projection gradient [18] method for optimization.[3]

3 On our data, we found that simply fixing θ_j as the estimate from the LDA model gives comparable performance and saves computation.

Second, given this **ffi**, we can optimize the topics $\beta_{1:K}$ with

$$\beta_{kx} \propto \sum_j \sum_n \phi_{jnk} 1[x_{jn} = x].$$

This is the same M-step update for topics as in LDA [26].

Next, we would like to optimize the augmented representations $\nu_{1:D}$. We can write the component of the log likelihood with terms containing ν_j as

$$\begin{aligned}\mathcal{L}(\nu_j) = &-\frac{\lambda}{2}(\nu_j - \theta_j)^\top(\nu_j - \theta_j) \\ &- \sum_{i,i\neq j} \frac{\tau_{ij}}{2}(y_{ij} - w\theta_i^\top \nu_j)^2.\end{aligned}$$

To maximize this quantity, we take the gradient of $\mathcal{L}(\nu_j)$ with respect to ν_j and set it to 0, which gives an update for ν_j

$$\begin{aligned}\nu_j^* \leftarrow &\left(\lambda I_K + w^2\left((\tau_1 - \tau_0)\sum_{i\in\{i:i\to j\}} \theta_i\theta_j^\top + \tau_0 \sum_{i,i\neq j} \theta_i\theta_j^\top\right)\right)^{-1} \\ &\times \left(\theta_j + w\tau_1 \sum_{i\in\{i:i\to j\}} \theta_i\right)\end{aligned} \tag{11}$$

where $\{i : i \to j\}$ denotes the set of documents that cite document j. For the second line of Eq. 11, we can see that the augmented representation ν_j is affected by two main parts: the first is the content from document j (topic proportions θ_j) and the second is the content from other documents who cite document j (topic proportions θ_i, where $i \in \{i : i \to j\}$).

Next, we want to optimize the global scaling variable w. Isolating the terms in the complete log likelihood that contain w gives

$$\mathcal{L}(w) = -\sum_{i\neq j} \frac{\tau_{ij}}{2}(y_{ij} - w\theta_i^\top \nu_j)^2.$$

In a similar manner as the previous step, to maximize this quantity we take the gradient of $\mathcal{L}(w)$ with respect to w and set it to 0, which gives its update[4]

$$\begin{aligned}w^* \leftarrow &\left(\sum_j \left((\tau_1 - \tau_0)\sum_{i\in\{i:i\to j\}} (\theta_i^\top \nu_j)^2 + \tau_0 \sum_{i,i\neq j} (\theta_i^\top \nu_j)^2\right)\right)^{-1} \\ &\times \left(\tau_1 \sum_j \sum_{i\in\{i:i\to j\}} \theta_i^\top \nu_j\right).\end{aligned} \tag{12}$$

Empirically, we found that an optimal trade-off between computation time and performance involves performing LDA [26] initially to learn the latent representations θ_j, and then performing coordinate ascent to learn the augmented representations ν_j and global parameter w. We detail this procedure in Algorithm 1.

4 In theory, this update could lead to a negative value. However, in our experiments, we did not see this happen.

Algorithmus 1 : MAP Parameter Learning from Neiswanger et al. [141]

Input : A citation network of documents $\{x_j\}_{j=1}^D$ with directed links y_{ij} for $i, j \in \{1, \ldots, D\}$, and stopping criteria δ

Output : Latent content representations θ_j, link-offset representations ν_j, and global scale parameter w

```
Run LDA [26] on {x_j}_{j=1}^D to learn θ_{1:D}
Initialize ν_{1:D} = θ_{1:D} and eps = ∞
while eps > δ do
    Update w ← w*                                   ▷ Equation 12
    for j = 1 to D do
        Update ν_j ← ν_j*                           ▷ Equation 11
    Set eps ← ‖ν_{1:D} − ν̃_{1:D}‖
```

Computational efficiency. We now show that our learning algorithm (Algorithm 1) has runtime complexity linear in the number of documents and citations.

First, estimating the topic proportions θ_j, $j = 1, ..., D$ has the same complexity as the standard learning algorithm for LDA, which is linear in the number of documents.

Second, the augmented representations ν_j, $j = 1, ..., D$ and global scaling parameter w can be estimated in linear time, via a caching strategy — this is similar to the method adopted by [89, 190]. We now describe this strategy.

For the augmented representation ν_j (Eq. 11), we cache $\theta_0 = \sum_i \theta_i$. This allows us to update ν_j (Eq. 11) using the identity

$$\sum_{i, i \neq j} \theta_i = \theta_0 - \theta_j.$$

Every time we update a θ_j, we also update the cache θ_0, and this takes constant time w.r.t. the number of documents and citations.

For the global scaling parameter w (Eq. 12), we can compute

$$\begin{aligned} \sum_{i, i \neq j} (\theta_i^\top \nu_j)^2 &= \sum_{i, i \neq j} \nu_j^\top \theta_i \theta_i^\top \nu_j \\ &= \nu_j^\top (\sum_{i, i \neq j} \theta_i \theta_i^\top) \nu_j \\ &= \nu_j^\top (\sum_i \theta_i \theta_i^\top) \nu_j - \nu_j^\top \theta_j \theta_j^\top \nu_j \end{aligned}$$

in $O(K^2)$ time (constant in the number of docs and citations) by simply caching $\Theta_0 = \sum_i \theta_i \theta_i^\top$. This cache variable also requires $O(K^2)$ time to update whenever we modify some θ_j.

The remaining sums in Eqs 11,12 touch every citation exactly once, therefore a single update sweep over all ν_j and w only requires constant work per edge (treating K as constant). We have therefore shown that Algorithm 1 is linear in the number of documents and citations. Moreover, we have attained linear scalability without

resorting to treating missing citations as hidden data. This gives our LRO a data advantage over methods that hide missing citations, such as the RTM [40].

2.5 RELATED WORK

Our proposed work focuses on two aspects of citation network modeling: 1) network understanding/exploration and 2) citation prediction. We therefore divide the related work section into these two categories.

Network understanding/exploration. Network exploration is a broad empirical task concerned with, amongst other things, understanding the overall structure of the network [148], understanding the context of individual nodes [4], and discovering anomalous nodes or edges [169]. In addition to methods that operate on purely graph data, there are techniques that leverage both the graph as well as textual content, such as relational topic models (RTM) [40], Link-PLSA-LDA [131], and TopicFlow [133]. The idea behind such hybrid methods is that text and graph data are often orthogonal, providing complementary insights [85].

Our LRO model incorporates network information by modeling per-document random offsets that capture topical information from connected neighbors. These random offsets represent relevant topics that would otherwise not be found in the documents through content analysis. The Simple English Wikipedia analysis from the introduction provides a good example: the *Sistine Chapel* article's random offsets (the top row of Figure 4) contain the topic `Anglicanism` (which is also related to Christianity), even though the article text's latent topic representation makes no mention of it. In this manner, the LRO model helps us understand the context of network nodes (a.k.a. documents), and helps us to detect anomalous nodes (such as documents whose random offsets diverge greatly from their latent topic vectors).

Citation prediction. The citation prediction task can be approached by considering text features, network features, or a combination of both. In the text-only setting, approaches based on common text features (e.g., TF-IDF scores [20]) and latent space models (e.g., topic models [24]) can be used to the measure similarities between two documents, allowing for ranking and prediction. However, text-only approaches cannot account for citation behavior due to the network structure.

In the network-only setting without document content, there are a number of commonly-used measures of node similarity, such as the Jaccard Coefficient, the Katz measure [99] and the Adamic/Adar measure [1]. Latent space models such as matrix factorization (MF) methods [104] can be used here. However, when test documents are out-of-sample with respect to the network (when we consider newly-written papers with no preexisting citations), these measures are inapplicable.

Finally, there are methods that combine both document content and network structure to predict citations. One such method is the relational topic models (RTM) [40], in which link outcomes depend on a reweighted inner product between latent po-

sitions (under the LDA model). The weights are learned for each latent dimension (topic), but are not specific to any document, and thus only capture network behavior due to topic-level interactions. In contrast, our random offsets are learned on a per-document basis, capturing interaction patterns specific to each document, which in turn yields better predictive performance as shown in our empirical study. In [114], in addition to the document content, author information is also considered to model the citation structure. In [131], citations were treated as a parallel document (of citations) as to the document content of words. Neither of these methods use per-document offsets to model citation structure.

2.6 EMPIRICAL STUDY

We will empirically demonstrate the use of our model for modeling citation networks. We will first show quantitative results for citation prediction then present qualitative results using our model to explore citation networks.

Datasets. We use three citation network datasets,

1. The ACL Anthology paper citation network (*ACL*) contains 16,589 documents and 94,973 citations over multiple decades.
2. The arXiv high energy physics citation network (*arXiv*) contains 34,546 arXiv/hep-th articles and 421,578 citations from January 1993 through April 2003.
3. The Simple English Wikipedia citation network (*Wikipedia*) contains 27,443 articles, and 238,957 citations corresponding to user-curated hyperlinks between articles.

2.6.1 *Citation Prediction*

For citation prediction, we compare against the RTM [40], matrix factorization (MF) [104], LDA-based predictions [26], and three common baseline algorithms. A detailed description is given below.

The first task is predicting held-out citations. Here we used a five-fold cross validation: for each document that has cited more than 5 documents, we held out 20% of the documents into test set and the rest into the training set.

The second task is predicting citations for new documents. To simulate this scenario, we train our model using all the citations before a certain year and predict the citations of the new documents published in that year. This task is important for a real citation prediction system, where user may input some text without existing citations. For this experiment, we excluded MF from the comparisons, because it cannot perform this task.

Evaluation metric. Our goal is to make citation predictions, where it is more realistic to provide a rank list of citation predictions than to make hard decisions for

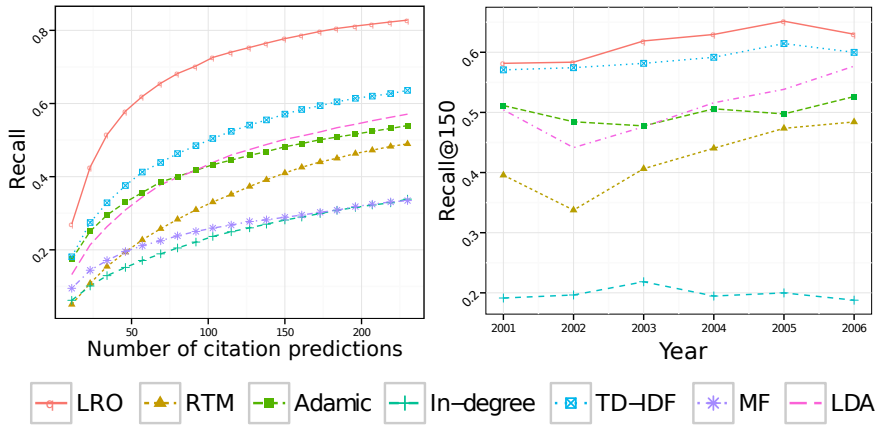


Figure 6: Left: Citation prediction performance on the ACL dataset for task one (predicting held-out citations).Right: Citation prediction performance on task two (predicting citations for new documents) on subsets of the ACL dataset for 7 years. In both cases, the LRO yields the highest recall over all ranges.

the users. For a given set of M predicted citations, we use a performance metric, Recall@M,

$$\text{Recall@M} = \frac{\text{number of citations in the predicted set}}{\text{total number of citations}}$$

which can be viewed as the proportion of "true" citations successfully predicted by a given method, when the method is allowed to provide M guesses.

Comparison methods. We compare our model with a number of competing strategies, starting with the RTM [40]. In order to make predictions using the RTM, we learn a latent representation for each document and predict citations using a similarity function between these representations (detailed in [40]). The second comparison is an LDA-based prediction strategy, in which document predictions are determined by the similarity between the latent document representation vectors θ_j. The similarity is computed using inverse of the Hellinger distance [25]

$$S_{ij} = H(\theta_i, \theta_j)^{-1} = \sqrt{2}\left\|\sqrt{\theta_i} - \sqrt{\theta_j}\right\|^{-1}.$$

Third, we compare with matrix factorization (MF), but only on the first task. (MF cannot make the citation predictions for a brand new document.) Finally, we compare with three simple baseline methods on both tasks. The first is that of Adamic/Adar [1], described in Section 2.5. The second is based on term frequence-inverse document frequency (TF-IDF) scores, where citations are predicted based on similarities in the documents' scores [20]. The third baseline is called "in-degree", where each document is given a score proportional to the number of times it is cited; in this case, the same set of predictions are given for every test document. Hyperparameters are set via cross validation.

Task one: predicting held-out citations. Given the document contents and the remaining links, the task is to predict the held out citations for each document. We show results for our model and six comparison methods on the ACL dataset in Figure 6. Our model (LRO) achieves a significantly higher recall over all ranges of the number of predictions, and we observed similar results for the other two datasets.

We also wanted to determine how our method performs across different datasets. To make the results comparable, we normalized the number of predictions M by setting it to a fraction of the total number of documents in each respective dataset. The results are shown in Figure 7: LRO performs well on all three datasets, though we note that ACL has a much better score than the other two. We attribute this to the fact that ACL contains only refereed academic papers, and is therefore more structured than either arXiv (which is unrefereed) or Simple English Wikipedia (whose articles are not always subject to editorial attention).

Task two: predicting citations for new documents. The second task is to predict citations for documents with no prior citation information, corresponding to scenarios in which one needs to suggest citations for newly written documents. This task is often referred to as the "cold start problem" in recommender systems.

We simulate the process of introducing newly written papers into a citation network by dividing them according to publication year. Specifically, from the ACL citation network dataset, we select the citations and documents that existed before the year Y as training data, for Y ranging from 2001 to 2006. After training on this subset, the task is then to predict the citations occurring in year Y for the new documents written in year Y.

For this task, we compared our model against the same comparison methods used in the previous task, except for matrix factorization, which cannot make citation predictions for new documents. Figure 6 (right) shows the results. We fix the number of citation predictions $M = 150$ (other M values have similar trends). Again, our model achieves the best performance over a majority of the M values in all six years, and increases its lead over the comparison methods in later years, after a larger portion of the citation network has formed and can be used as training data.

Hyperparameter sensitivity. We also study how different hyperparameters affect performance, including the number of topics K, precision parameters τ_0 and τ_1, and latent random offset precision parameter λ (Figure 7, right). Again, we fix $M = 150$. First, we varied the number of topics from 75 to 250, and found an optimal value of approximately 175 topics. Next, in order to find the optimal balance between parameters τ_0 and τ_1, we fixed $\tau_1 = 1$ and varied τ_0 from 1/10000 to 1, finding an optimal value of approximately $\tau_0 = 1/100$. Finally, we varied the parameter λ from 5 to 40, and found an optimal value at approximately $\lambda = 9$.

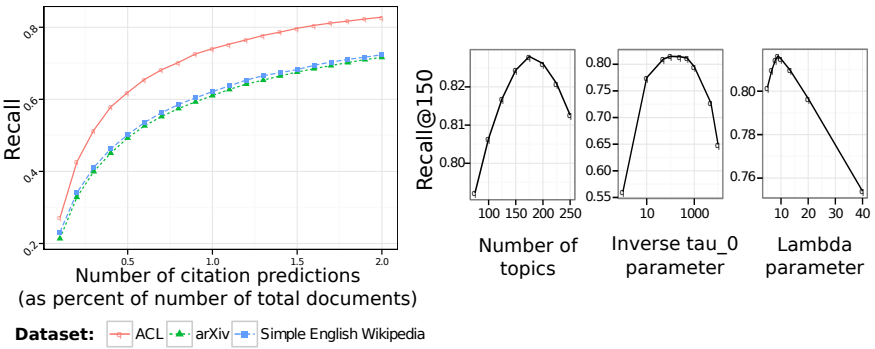


Figure 7: Left: citation prediction performance of our LRO model on three real-world datasets. The ACL dataset has a better score than the other two datasets. See main text for details. Right: citation prediction performance for a range of hyperparameter settings, including the number of topics K, the non-link variance parameter τ_0, and the latent random offset variance parameter λ.

2.6.2 *Exploring Citation Networks*

The latent random offsets can yield useful information that allows for analysis and exploration of documents in the citation network. Our model provides, for each document, a content representation vector θ_j, which captures the topics associated with the content of the document, and a latent offset vector ϵ_j, which captures topics not necesarily contained within the document but expressed by others who cited the document. Highly positive latent offsets may capture the topics where a given document has been influential within the context of the citation network; alternatively, negative offsets can represent topics that are expressed highly in a document, but that have not proven to be influential within the context of the network.

Given a document, we can therefore explore its contents by examining the learned set of topics, and we can explore its role in the citation network (and see the topics of documents that it has influenced) by examining the latent offsets. In Figures 4 and 8 we show the latent topic representations of document contents, the learned random offsets, and the final augmented representations (the sum of topic representations and random offsets), for a document in each of the Simple English Wikipedia and ACL datasets. The augmented representations provide information on both the content and context of a document: they incorporate information contained in the document as well as in other documents that cite it.

For highly cited documents, we have a great deal of information from the citing documents (i.e. the in-links), and this information can be used to more strongly offset the latent topic representations. Intuitively, the content is like a prior belief about a document's latent representation, and as more sources start citing the document, this outside information further offsets the latent topic representations. Additionally, the offsets do not only "add" more information to the latent representation from the citing documents. In Figure 8 (top row), the offsets acted primarily to reduce the

weights of many of the largest topics in the content representation, and only added weight to two topics. Here, the offsets served to dampen many of the content topics that did not appear to be relevant to the citing documents, and for this reason, the augmented representation is more sparse than the initial content representation.

Figure 8: Interpreting citation predictions for the document *Automatic Recognition Of Chinese Unknown Words Based On Roles Tagging* in the ACL dataset. For each predicted link, we show the relative weight that each latent topic (denoted by the top four words) contributed to the prediction. These provide reasons why each predicted link was chosen, in terms of the topics.

Interpreting predictions. In addition to maintaining competitive prediction performance, our model allows for interpretable link prediction: for each predicted link we can use our latent representations to give users an understanding of why the link was returned. In particular, we can find the contribution that each topic provides to the final prediction score in order to determine the "reasons" (in terms of the latent topics) why a given document was predicted. We illustrate this in Figures 4 and 8 (bottom row of graphs). In Figure 4, for the *Sistine Chapel* document, *Chapel* is cited largely due to three topics (`architecture`, `Christianity`, and `buildings`), *Christian* is cited primarily due to a single topic (`Christianity`), and *Italy* is mainly cited due to six lower-weighted topics (`countries`, `Christianity`, `architecture`, `buildings`, `music`, and `populace`). Since *Italy* is a highly cited document and its augmented latent representation emphasizes a large number of topics (many of those expressed

by its in-links), it was predicted due to a slight similarity in a number of topics as opposed to a strong similarity in just a few.

In Figure 8 we show three predictions for the document *Automatic Recognition of Chinese Unknown Words Based on Roles Tagging*. We can see that each of the predicted documents was due to a different aspect of this paper: the document *Automatic Rule Induction For Unknown-Word Guessing* was chosen primaily due to the `unknown-word` topic (related to the paper's goal of recognizing unknown words), the document *Word Identification for Mandarin Chinese Sentences* was chosen primarily due to the `China` topic (related to the paper's language domain area), and the document *A Knowledge-Free Method For Capitalized Word Disambiguation* was chosen primarily due to the `pronoun` topic (related to the paper's use of names, locations, and roles).

2.7 CONCLUSION

In this chapter, we proposed a probabilistic approach for citation network modeling that integrates the merits of both content and link analyses. Our empirical results showed improved performance compared with several popular approaches for citation prediction. Furthermore, our approach can suggest citations for brand new documents without prior citations—an essential ability for building a real citation recommendation system.

Qualitatively, our approach provides meaningful explanations for how predictions are made, through the latent random offsets. These explanations provide additional information that can be useful for making informed decisions. For example, in a citation recommendation system, we can inform users whether a citation is suggested more due to content similarities or due to the existing network structure, and we can show the relative amounts that individual topics contributed to the prediction. In future work, we would like to conduct user studies to quantify how this additional information helps users find more relevant citations in a more efficient way.

3

THE DEPENDENT DIRICHLET PROCESS MIXTURE OF OBJECTS FOR DETECTION-FREE TRACKING

3.1 CHAPTER SUMMARY

This chapter explores a probabilistic model for finding, tracking, and representing arbitrary objects in a video without a predefined method for object detection. We present a model that localizes objects via unsupervised tracking while learning a representation of each object, avoiding the need for pre-built detectors. Our model uses a dependent Dirichlet process mixture to capture the uncertainty in the number and appearance of objects and requires only spatial and color video data that can be efficiently extracted via frame differencing. We give two inference algorithms for use in both online and offline settings, and use them to perform accurate detection-free tracking on multiple real videos. We demonstrate our method in difficult detection scenarios involving occlusions and appearance shifts, on videos containing a large number of objects, and on a recent human-tracking benchmark where we show performance comparable to state of the art detector-based methods.

3.2 INTRODUCTION

Algorithms for automated object detection and tracking in video have found application in a wide range of fields, including robotic vision, cell tracking, sports analysis, video indexing, and video surveillance [188, 209]. The goal of these algorithms is to find the sequences of positions held by each object of interest in a video. A majority of modern methods require a pre-trained object detector or make use of prior knowledge about the objects' physical characteristics (such as their color or shape) to perform detection [31]. Often, these methods will apply the detector in each frame of a video, and then use the detection results in tracking or data association algorithms. Other algorithms use heuristics to find, or require manual initialization of, object positions and then search for similar image patches in consecutive frames to perform tracking [124]. Both techniques require some predefined detection strategy for each type of object they intend to find and track.

When the objects to be tracked have highly variable appearance, if one wishes to track many different types of objects, or if one simply does not know the types of objects in advance, it is often hard to find a suitable detection strategy [13]. Furthermore, common video conditions such as variable lighting, low quality images, non-uniform backgrounds, and object occlusions can all reduce detection accuracy [203].

Cases such as these, where it is difficult to construct an object detector in advance, prompt the need for a method to automatically localize and track arbitrary objects. Some methods towards this end have involved background subtraction and blob tracking, which segment foreground patches to localize objects, and optical flow-based tracking, which separate objects based on their relative motion. Both have trouble consistently and accurately segmenting objects and tracking through occlusion [15, 189]. A recent work introduced the term "detection-free tracking" for this task, and proposed a method based on spectral clustering of trajectories [59].

Bayesian models have also been employed to capture the components of a video, and a number of recent works have incorporated nonparametric Bayesian priors for finding the patterns of motion in scenes [56, 195]. However, there has been little work towards building Bayesian models of arbitrary objects in order to perform detection-free tracking.

In this chapter, we develop a nonparametric Bayesian model for jointly learning a representation of each object and performing unsupervised tracking, thereby allowing for accurate localization of arbitrary objects. We combine a dependent Dirichlet process mixture with object and motion models to form the dependent Dirichlet process mixture of objects (DDPMO). The advantages of our model are that it can (a) accurately localize and track arbitrary video objects in a fully unsupervised fashion, (b) jointly learn a time-varying model for each object and use these models to increase the localization/tracking performance, (c) infer a distribution over the number of distinct objects present in a video, (d) incorporate a model for the motion of each object, and (e) begin tracking as objects enter the video frame, stop when they exit, and track through periods of partial or full occlusion.

3.3 DEPENDENT DIRICHLET PROCESS MIXTURE OF OBJECTS

To find and track arbitrary video objects, the DDPMO models spatial and color features that are extracted as objects travel within a video scene (described in Section 3.3.1). The model isolates independently moving video objects and learns object models for each that capture their shape and appearance. The learned object models allow for tracking through occlusions and in crowded videos. The unifying framework is a dependent Dirichlet process mixture, where each component is a (time-varying) object model. This setup allows us to estimate the number of objects in a video and track moving objects that may undergo changes in orientation, perspective, and appearance.

3.3.1 *Preliminaries*

Dependent Dirichlet process prior. Dirichlet process (DP) priors for component weights in mixture models have long been used as nonparametric Bayesian tools to estimate the number of clusters in data [9]. Dependent Dirichlet process (DDP)

mixtures extend this by allowing cluster parameters to vary with some covariate [118]. In our case, a DDP object mixture lets us estimate, and capture the uncertainty in, the number of objects while modeling their time-varying parameters.

A DDP known as a generalized Polya urn (GPU) [36] has the desired properties that, when used in a mixture model, clusters can be created and die off and cannot merge or split. In this model, the n^{th} data point at time t, $\mathbf{x}_{t,n}$, has an assignment $c_{t,n}$ to a cluster $k \in \{1, \ldots, K_{t,n}\}$ (where $K_{t,n}$ denotes the total number of assigned clusters after reaching $\mathbf{x}_{t,n}$). Each assignment increases the cluster's size $m^k_{t,n}$ by one. After each time step, cluster sizes may decrease when observations are uniformly "unassigned" in a deletion step. The generative process for the GPU, at each time step t, is

1. For $k = 1, \ldots, K_{t-1,N_{t-1}}$
   a) Draw $\Delta m^k_{t-1} \sim \text{Binom}(m^k_{t-1,N_{t-1}}, \rho)$
   b) Set $m^k_{t,0} = m^k_{t-1,N_{t-1}} - \Delta m^k_{t-1}$
2. For $n = 1, \ldots, N_t$
   a) Draw $c_{t,n} \sim \text{Cat}\left(\frac{m^1_{t,n-1}}{\alpha+\sum_k m^k_{t,n-1}}, \frac{m^{K_{t,n-1}}_{t,n-1}}{\alpha+\sum_k m^k_{t,n-1}}, \frac{\alpha}{\alpha+\sum_k m^k_{t,n-1}}\right)$
   b) If $c_{t,n} \leqslant K_{t,n-1}$:
      Set $m^{c_{t,n}}_{t,n} = m^{c_{t,n}}_{t,n-1} + 1$, $m^{\setminus c_{t,n}}_{t,n} = m^{\setminus c_{t,n}}_{t,n-1}$, and $K_{t,n} = K_{t,n-1}$
   c) If $c_{t,n} > K_{t,n-1}$:
      Set $m^{c_{t,n}}_{t,n} = 1$, $m^{\setminus c_{t,n}}_{t,n} = m^{\setminus c_{t,n}}_{t,n-1}$, and $K_{t,n} = K_{t,n-1} + 1$.

where Cat is the categorical distribution, $m^{\setminus c_{t,n}}_{t,n}$ is the set $\{m^1_{t,n}, \ldots, m^{K_{t,n}}_{t,n}\} \setminus \{m^{c_{t,n}}_{t,n}\}$, Binom is the binomial distribution, α is the DP concentration parameter, and ρ is a deletion parameter that controls temporal dependence of the DDP. We will refer to this process as GPU(α, ρ).

Data. At each frame t, we assume we are given a set of N_t foreground pixels, extracted via some background subtraction method (such as those detailed in [209]). These methods primarily segment foreground objects based on their motion relative to the video background. For example, an efficient method applicable for stationary videos is frame differencing: in each frame t, one finds the pixel values that have changed beyond some threshold, and records their positions $\mathbf{x}^s_{t,n} = (x^{s_1}_{t,n}, x^{s_2}_{t,n})$. In addition to the position of each foreground pixel, we extract color information. The spectrum of RGB color values is discretized into V bins, and the local color distribution around each pixel is described by counts of surrounding pixels (in an $m \times m$ grid) that fall into each color bin, denoted $\mathbf{x}^c_{t,n} = (x^{c_1}_{t,n}, \ldots, x^{c_V}_{t,n})$. Observations are therefore of the form

$$\mathbf{x}_{t,n} = (\mathbf{x}^s_{t,n}, \mathbf{x}^c_{t,n}) = (x^{s_1}_{t,n}, x^{s_2}_{t,n}, x^{c_1}_{t,n}, \ldots, x^{c_V}_{t,n}) \tag{13}$$

Examples of spatial pixel data extracted via frame differencing are shown in Figure 9 (a)-(g).

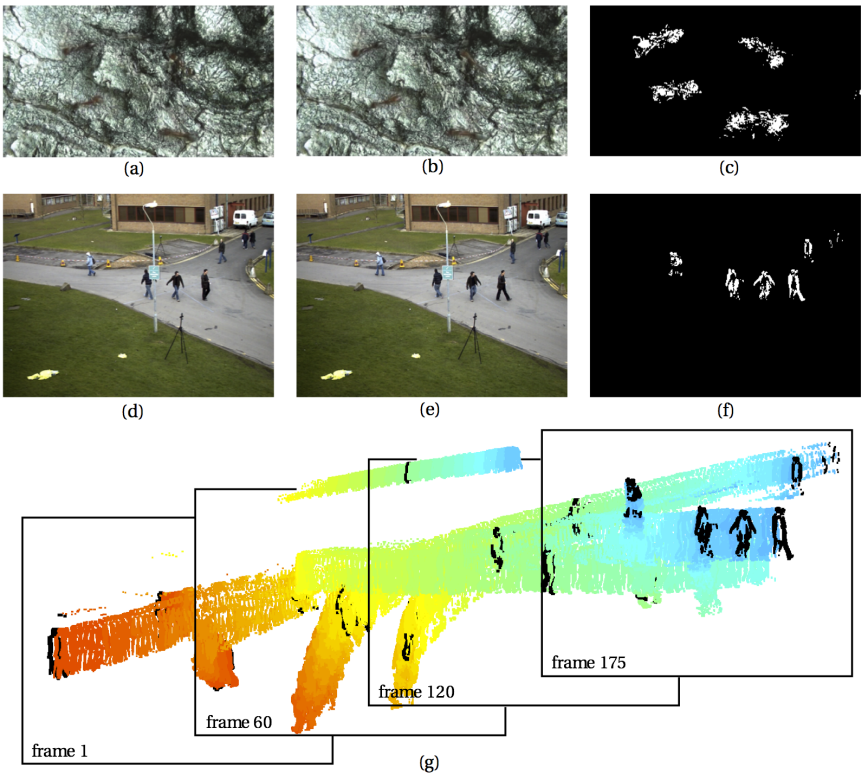


Figure 9: (a - f) Two pairs of consecutive frames and the spatial observations $\mathbf{x}^s_{t,n}$ extracted by taking the pixel-wise frame difference between each pair. (g) The results of frame differencing over a sequence of images (from the PETS2010 dataset).

3.3.2 *DDPMO*

Our object model $F(\theta^k_t)$ is a distribution over pixel data, where θ^k_t represents the parameters of the k^{th} object at time t. We wish to keep our object model general enough to be applied to arbitrary video objects, but specific enough to learn a representation that can aid in tracking. Here, we model each object with

$$\mathbf{x}_{t,n} \sim F(\theta^k_t) = \text{Normal}(\mathbf{x}^s_{t,n}|\mu_t, \Sigma_t)\text{Mult}(\mathbf{x}^c_{t,n}|\delta_t) \tag{14}$$

where object parameters $\theta_t = \{\mu_t, \Sigma_t, \delta_t\}$, and $\sum_{j=1}^{V} \delta^j_t = 1$. The object model captures the objects' locus and extent with the multivariate Gaussian and color distribution with the multinomial. We demonstrate in Section 3.5 that this representation can capture the physical characteristics of a wide range of objects while allowing objects with different shapes, orientations, and appearances to remain isolated during tracking.

We would also like to model the motion of objects. Assuming as little as possible, we take each object's parameters θ_t^k to be a noisy version of the previous parameters θ_{t-1}^k (if the object existed at the previous time step) and define

$$\theta_t^k|\theta_{t-1}^k \sim \begin{cases} T(\theta_{t-1}^k) \text{ if } k \leqslant K_{t-1,N_{t-1}} \\ G_0 \text{ if } k > K_{t-1,N_{t-1}} \end{cases} \tag{15}$$

where T denotes a transition kernel, the $k > K_{t-1,N_{t-1}}$ case is when a new cluster has been created at time t, and G_0 is the base distribution of the dependent Dirichlet process, which represents the prior distribution over object parameters. We define G_0 to be

$$G_0(\theta_t^k) = \text{NiW}(\mu_t^k, \Sigma_t^k|\mu_0, \kappa_0, \nu_0, \Lambda_0)\text{Dir}(\delta_t^k|q_0) \tag{16}$$

where NiW denotes the normal-inverse-Wishart distribution and Dir denotes the Dirichlet distribution; these act as a conjugate prior to the object model. We can therefore write the generative process of the DDPMO as, for each time step $t = 1, \ldots, T$:

1. Draw $\{c_{t,1:N_t}, K_{t,N_t}, m_{t,0}^{1:K_{t-1,N_{t-1}}}\} \sim \text{GPU}(\alpha, \rho)$
2. For $k = 1, \ldots, K_{t,N_t}$:

$$\text{draw } \theta_t^k \sim \begin{cases} T(\theta_{t-1}^k) \text{ if } k \leqslant K_{t-1,N_{t-1}} \\ G_0(\mu_0, \kappa_0, \nu_0, \Lambda_0, q_0) \text{ if } k > K_{t-1,N_{t-1}} \end{cases}$$

3. For $n = 1, \ldots, N_t$: draw $\mathbf{x}_{t,n} \sim F(\theta_t^{c_{t,n}})$

where the notation $c_{1,1:N_1} = \{c_{1,1}, \ldots, c_{1,N_1}\}$. A graphical model for the DDPMO is shown in Figure 10.

To meet technical requirements of the GPU, the transition kernel T must satisfy

$$\int G_0(\theta_{t-1}^k)T(\theta_t^k|\theta_{t-1}^k)d\theta_{t-1}^k = G_0(\theta_t^k) \tag{17}$$

or, equivalently, its invariant distribution must equal the base distribution [61]. One way to satisfy this while providing a reasonable transition kernel is to introduce a set of M auxiliary variables $\mathbf{z}_t^k = (z_{t,1}^k, \ldots, z_{t,M}^k)$ for cluster k at time t such that

$$P(\theta_t^k|\theta_{t-1}^k) = \int P(\theta_t^k|\mathbf{z}_t^k)P(\mathbf{z}_t^k|\theta_{t-1}^k)d\mathbf{z}_t^k \tag{18}$$

With this addition, object parameters do not directly depend on their values at a previous time, but are instead dependent through an intermediate sequence of variables. This allows the cluster parameters at each time step to be marginally distributed according to the base distribution G_0 while maintaining simple time varying behavior. We can therefore sample from the transition kernel using $\theta_t^k \sim T(\theta_{t-1}^k) = T_2 \circ T_1(\theta_{t-1}^k)$, where

$$z_{t,1:M}^k \sim T_1(\theta_{t-1}^k) = \text{Normal}(\mu_{t-1}^k, \Sigma_{t-1}^k)\text{Mult}(\delta_{t-1}^k) \tag{19}$$

$$\mu_t^k, \Sigma_t^k, \delta_t^k \sim T_2(z_{t,1:M}^k) = \text{NiW}(\mu_M, \kappa_M, \nu_M, \Lambda_M)\text{Dir}(q_M) \tag{20}$$

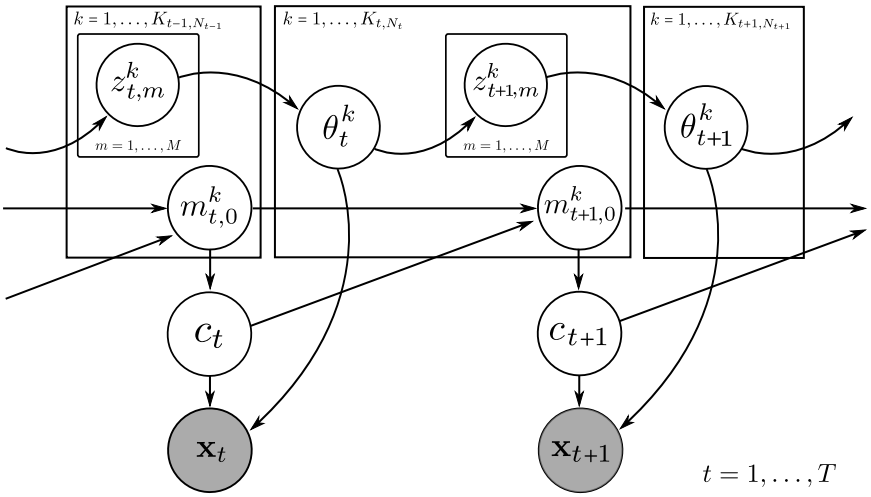


Figure 10: Graphical model of the dependent Dirichlet process mixture of objects (DDPMO). All observations at time t are denoted as $\mathbf{x}_t$ and their assignments as c_t.

where $\mu_M, \kappa_M, \nu_M, \Lambda_M$ and q_M are posterior NiW and Dir parameters, given the auxiliary variables $z_{t,1:M}$ (formulas given in Section 3.4.1.1).

3.4 INFERENCE

We describe two inference algorithms for the DDPMO: sequential Monte Carlo (SMC) with local Gibbs iterations, and Particle Markov Chain Monte Carlo (PMCMC).

3.4.1 *Sequential Monte Carlo*

We first derive an SMC (particle filter) inference algorithm where we draw samples from a proposal distribution by iterating through local Gibbs updates (detailed in Section 3.4.1.1). SMC allows us to make a single pass through the data and draw posterior samples in an online fashion.

3.4.1.1 *Local Gibbs Updates*

We perform Gibbs sampling on the assignments and object parameters (at a given t) to draw SMC proposals; this allows for the proposal of well-mixed samples given newly introduced data in a particular frame. For an assignment $c_{t,n}$, we can compute a value proportional to the posterior for each possible assignment value $1, \ldots, K_{t,n}$, and then sample from the resulting categorical distribution (after normalizing). The first proposal distribution Q_1 is the probability of an assignment

Algorithmus 2 : SMC for the DDPMO from Neiswanger et al. [144]

Input : Extracted pixel data $\{\mathbf{x}_{1,1:N_1}, \ldots, \mathbf{x}_{T,1:N_T}\}$, number of particles L, number of local Gibbs iterations S, and prior parameters α, ρ, μ_0, κ_0, ν_0, Λ_0 and q_0.

Output : Posterior samples $\left\{\theta_1^{1:K_{1,N_1}}, \ldots, \theta_T^{1:K_{T,N_T}}\right\}^{(1:L)}$ of the object model parameters.

for $t = 1$ ***to*** T **do**
  for $l = 1$ ***to*** L **do**
    for *iter* $= 1$ ***to*** S **do**
      Sample $(c_{t,1:N_t})^{(l)} \sim Q_1$ and $(\theta_t^{1:K_{t,N_t}})^{(l)} \sim Q_2$
    for $k = 1$ ***to*** K_{t,N_t} **do**
      Sample $(\Delta m_t^k)^{(l)} \sim \text{Binom}((m_{t,N_t}^k)^{(l)}, \rho)$
      Set $(m_{t+1,0}^k)^{(l)} = (m_{t,N_t}^k)^{(l)} - (\Delta m_t^k)^{(l)}$
      Sample $(z_{t+1,1:M}^k)^{(l)} \sim T_1((\theta_t^k)^{(l)})$
    Compute particle weight $\tilde{w}_t^{(l)}$
  Normalize particle weights and resample particles

$c_{t,n}$ given current cluster sizes, cluster parameters, and concentration parameter α, written

$$Q_1\left(c_{t,n} | m_{t,n-1}^{1:K_{t,n-1}}, \theta_t^{1:K_{t,n-1}}, \alpha\right) \propto \text{Cat}(m_{t,n-1}^1, \ldots, m_{t,n-1}^{K_{t,n-1}}, \alpha) \times \begin{cases} F(\mathbf{x}_{t,n} | \theta_t^{c_{t,n}}) & \text{if } c_{t,n} \leqslant K_{t,n-1} \\ \int P(\mathbf{x}_{t,n} | \theta) G_0(\theta) d\theta & c_{t,n} > K_{t,n-1} \end{cases} \tag{21}$$

where we set the number of clusters $K_{t,n}$ and their sizes $m_{t,n}^{1:K_{t,n}}$ appropriately as each $c_{t,n}$ is assigned, and assume $K_{1,0} = 0$ for consistency at $t = 1$. The integral in the case of a new cluster ($k > K_{t,n-1}$) has an analytic solution

$$\int P(\mathbf{x}_{t,n} | \theta) G_0(\theta) d\theta = t_{\nu_0 - 1}\left(\mathbf{x}_{t,n}^s \mid \mu_0, \frac{\Lambda_0(\kappa_0 + 1)}{\kappa_0(\nu_0 - 1)}\right) \times \prod_{j=1}^{V} \frac{\Gamma(\mathbf{x}_{t,n}^c)}{\Gamma(q_0)} \times \frac{\Gamma(\sum_{j=1}^{V} q_0)}{\Gamma(\sum_{j=1}^{V} \mathbf{x}_{t,n}^c)} \tag{22}$$

where $t_{\nu_0 - 1}$ denotes the multivariate t-distribution with $\nu_0 - 1$ degrees of freedom, where we follow the three-value parameterization [64], and Γ denotes the gamma function.

The conjugacy of appearance model and transition kernel allow us to sample from the second proposal distribution Q_2, which is the posterior distribution over

the object parameters given current observations, auxiliary variables, and previous time object parameters, written

$$\begin{aligned} Q_2(\theta_t^k|\theta_{t-1}^k, \mathbf{x}_{t,1:N_t}^k, z_{t,1:M}^k) &= F(\mathbf{x}_{t,1:N_t}^k|\theta_t^k) T_2(\theta_t^k|z_{t,1:M}^k) \\ &= \text{NiW}(\mu_t^k, \Sigma_t^k|\mu_N, \kappa_N, \nu_N, \Lambda_N) \\ &\times \text{Dir}(\delta_t^k|q_N) \end{aligned} \tag{23}$$

where $\mathbf{x}_{t,1:N_t}^k = \{x_{t,n} \in x_{t,1:N_t}|c_{t,n} = k\}$ and the parameters for the NiW and Dir distributions are given when $\mathbf{x}_{t,1:N_t}^k$ and $z_{t,1:M}^k$ are taken to be the "observations" in the following posterior updates

$$\kappa_N = \kappa_0 + N \tag{24}$$

$$\nu_N = \nu_0 + N \tag{25}$$

$$\mu_N = \frac{\kappa_0}{\kappa_0 + N}\mu_0 + \frac{N}{\kappa_0 + N}\bar{\mathbf{x}}^s \tag{26}$$

$$\Lambda_N = \Lambda_0 + S_{\mathbf{x}^s} \tag{27}$$

$$q_N = q_0 + \sum_{i=1}^{N} \mathbf{x}_i^c \tag{28}$$

where N is the number of observations, $\{\mu_0, \kappa_0, \nu_0, \Lambda_0\}$ are the NiW prior parameters, q_0 is the Dir prior parameter, $\mathbf{x}^s$ and $\mathbf{x}^c$ respectively denote the spatial and color features of the observations, and $\bar{\mathbf{x}}$ and $S_{\mathbf{x}}$ respectively denote the sample mean and sample covariance of the observations.

3.4.1.2 *Particle Weights*

At each time, the particle weights are set to be

$$\tilde{w}_t^{(l)} = \frac{P\left((c_{t,1:N_t})^{(l)}, (\theta_t^{1:K_{t,N_t}})^{(l)}, \mathbf{x}_{t,1:N_t}|\Lambda\right)}{P\left((c_{t,1:N_t})^{(l)}, (\theta_t^{1:K_{t,N_t}})^{(l)}|\Lambda\right)} \tag{29}$$

where we've defined

$$\Lambda = \{(\theta_{t-1}^{1:K_{t-1,N_{t-1}}})^{(l)}, (m_{t,0}^{1:K_{t-1,N_{t-1}}})^{(l)}\} \tag{30}$$

Note that the numerator decomposes into

$$\begin{aligned} &P\left(\mathbf{x}_{t,1:N_t}|(c_{t,1:N_t})^{(l)}, (\theta_t^{1:K_{t,N_t}})^{(l)}\right) \\ &\times P\left((c_{t,1:N_t})^{(l)}, (\theta_t^{1:K_{t,N_t}})^{(l)}|\Lambda\right) \end{aligned} \tag{31}$$

which can be computed using the DDPMO local probability equations defined in Section 3.3.2, and the denominator can be computed using equations 21 and 23. After the particle weights are computed, they are normalized; particles are then redrawn based on their normalized weights in a multinomial resampling procedure [52].

3.4.1.3 *Computational Cost*

Assume N extracted pixels per frame, T frames, L particles, M auxiliary variables, S local Gibbs iterations, and fewer than K sampled objects ($K^{(1:L)}_{T,N_T} \leqslant K$). In the SMC inference algorithm, each local Gibbs iterations is $O(KN + M)$ and evaluating each particle weight is $O(K + N)$; the SMC algorithm therefore scales as $O(TL(K(SN + M) + SM + N))$. If we neglect the number of auxiliary variables M, as we can usually fix this at a small value, the algorithm scales as $O(TLKSN)$. We have empirically found that an SMC implementation in MATLAB, while not tuned for speed, usually requires 4-20 seconds for every 1 second of video, depending on the number of objects (after frame-rate has been subsampled to approximately 3 images/second in all cases). It is not unreasonable to believe that this could be scaled to real time tracking, given parallel computation and efficient image processing.

3.4.2 *Particle Markov Chain Monte Carlo*

SMC provides an efficient, online method for posterior inference, but can suffer from degeneracy; notably, a large majority of the returned particles correspond to a single, non-optimal tracking hypothesis. Ideally, we would like to infer a full posterior over object paths. MCMC methods are guaranteed to yield true posterior samples as the number of samples tends to infinity; however, we have found batch Gibbs sampling to be impractical for inference in the DDPMO, as samples tend to remain stuck in local posterior optima (often when a track begins on one object before switching to another) and cannot converge to a high accuracy tracking hypothesis in a reasonable amount of time.

PMCMC [8] is a Markov chain Monte Carlo method that attempts to remedy these problems by using SMC as an intermediate sampling step to move efficiently through high dimensional state spaces. We implement a specific case known as the Particle Gibbs (PG) algorithm, where we sample from the conditional distributions used in Gibbs sampling via a modified version of Algorithm 1 referred to as conditional sequential Monte Carlo.

3.4.2.1 *Conditional SMC*

Conditional SMC [8] allows for SMC to be used as a proposal distribution in a Gibbs sampling algorithm. We must first introduce the notion of a particle's lineage. Let $A_t^{1:L}$ denote the indices of the L particles chosen during the resampling step in time t (in Algorithm 1). The lineage $B_{1:T}^{(l)}$ of a particle is recursively defined as $B_T^{(l)} = l$ and for $t = (T-1), \ldots, 1$, $B_t^{(l)} = A_t^{B_{t+1}^{(l)}}$. More intuitively, for the l^{th} particle, which contains the variables $\Theta_{1:T}^{(l)}$ at the final time T, $B_t^{(l)}$ denotes the index of the particle that contained the variables $\Theta_{1:t}^{(l)} \subset \Theta_{1:T}^{(l)}$ at time t.

Conditional SMC uses lineages to ensure that a given particle will survive all resampling steps, whereas the remaining particles are generated as before. We define conditional SMC for the DDPMO in Algorithm 2. Note that computation of particle weights and resampling (for relevant particles) is performed in the same manner as in SMC inference (Algorithm 1).

Algorithmus 3 : Conditional SMC for the DDPMO from Neiswanger et al. [144]

Input : Extracted pixel data $\{\mathbf{x}_{1,1:N_1}, \ldots, \mathbf{x}_{T,1:N_T}\}$, number of particles L, number of local Gibbs iterations S, condition particle $\Phi_{1:T}^{(\eta)}$ with lineage $B_{1:T}^{(\eta)}$ $(\eta \in \{1, \ldots, L\})$.

Output : Particle-conditional posterior samples $\{\Theta_{1:T}\}^{(1:L)}$ of all latent model variables.

for $t = 1$ ***to*** T **do**
  for $l = 1$ ***to*** L **do**
    if $l \neq B_t^{(\eta)}$ **then**
      for *iter* $= 1$ ***to*** S **do**
        Sample $(c_{t,1:N_t})^{(l)} \sim Q_1$ and $(\theta_t^{1:K_{t,N_t}})^{(l)} \sim Q_2$
      for $k = 1$ ***to*** K_{t,N_t} **do**
        Sample $(\Delta m_t^k)^{(l)} \sim \text{Binom}((m_{t,N_t}^k)^{(l)}, \rho)$
        Set $(m_{t+1,0}^k)^{(l)} = (m_{t,N_t}^k)^{(l)} - (\Delta m_t^k)^{(l)}$
        Sample $(z_{t+1,1:M}^k)^{(l)} \sim T_1((\theta_t^k)^{(l)})$
      Set $\Theta_t^{(l)} = \left\{(c_{t,1:N_t})^{(l)}, (\theta_t^{1:K_{t,N_t}})^{(l)}, (m_{t+1,0}^{1:K_{t,N_t}})^{(l)}, (z_{t+1,1:M}^{1:K_{t,N_t}})^{(l)}\right\}$
    else
      Set $\Theta_t^{(l)} = \Phi_t^{(\eta)}$
    Compute particle weight $\tilde{w}_t^{(l)}$
  for $l = 1$ ***to*** L **do**
    if $l \neq B_t^{(\eta)}$ **then**
      Normalize particle weights and resample particles

3.4.2.2 *Particle Gibbs*

In the particle Gibbs (PG) algorithm [8], the model variables are first initialized, and then conditional SMC (Algorithm 2) is run for a number of iterations. More specifically, at the end of each iteration, a sample is drawn from the set of weighted particles returned by conditional SMC, and this sample is conditioned upon in the next iteration. The PG algorithm for the DDPMO is formalized in Algorithm 3.

As the PG inference requires all variables to be initialized, SMC inference (Algorithm 1) can be used as a quick way to provide near-MAP initialization of variables.

Algorithmus 4 : PMCMC (Particle Gibbs) for the DDPMO from Neiswanger et al. [144]

Input : Extracted pixel data $\{\mathbf{x}_{1,1:N_1}, \ldots, \mathbf{x}_{T,1:N_T}\}$, number of global Gibbs iterations G, number of particles L, number of local Gibbs iterations S

Output : Posterior samples $\left\{\theta_1^{1:K_{1,N_1}}, \ldots, \theta_T^{1:K_{T,N_T}}\right\}^{1:G}$ of the object model parameters

Initialize all model variables to $\Phi_0^{(L)}$

for $g = 1$ ***to*** G **do**

  Run conditional SMC (Algorithm 3) with input $\{\mathbf{x}_{1,1:N_1}, \ldots, \mathbf{x}_{T,1:N_T}\}$, L, S, and conditional on particle $\Phi_{g-1}^{(L)}$ to get particle set $\{\Theta_{1:T}^{(1)}, \ldots, \Theta_{1:T}^{(L)}\}$

  Draw $\Phi_g^{(L)} \sim \text{Unif}(\{\Theta_{1:T}^{(1)}, \ldots, \Theta_{1:T}^{(L)}\})$

Return $\left\{\theta_1^{1:K_{1,N_1}}, \ldots, \theta_T^{1:K_{T,N_T}}\right\}^{1:G} \in \Phi_{1:G}^{(L)}$

3.5 EXPERIMENTS

We demonstrate the DDPMO on three real video datasets: a video of foraging ants, where we show improved performance over other detection-free methods; a human tracking benchmark video, where we show comparable performance against object-specific methods designed to detect humans; and a T cell tracking task where we demonstrate our method on a video with a large number of objects and show how our unsupervised method can be used to automatically train a supervised object detector.

Detection-free comparison methods. Detection-free tracking strategies aim to find and track objects without any prior information about the objects' characteristics nor any manual initialization. One type of existing strategy uses optical flow or feature tracking algorithms to produce short tracklets, which are then clustered into full object tracks. We use implementations of Large Displacement Optical Flow (LDOF) [34] and the Kanade-Lucas-Tomasi (KLT) feature tracker [184] to produce tracklets [1]. Full trajectories are then formed using the popular normalized-cut (NCUT) method [168] to cluster the tracklets or with a variant that uses non-negative matrix factorization (NNMF) to cluster motion using tracklet velocity information [42] [2]. We also compare against a detection-free blob-tracking method, where extracted foreground

1 The LDOF implementation can be found at http://www.seas.upenn.edu/~katef/LDOF.html and the KLT implementation at http://www.ces.clemson.edu/~stb/klt/.

2 The NCUT implementation can be found at http://www.cis.upenn.edu/~jshi/software/ and the NNMF implementation at http://www.ornl.gov/~czx/research.html.

pixels are segmented into components in each frame [178] and then associated with the nearest neighbor criterion [209].

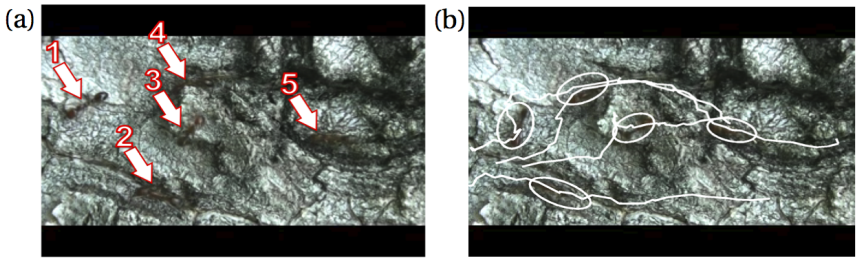
Performance metrics. For quantitative comparison, we report two commonly used performance metrics for object detection and tracking, known as the sequence frame detection accuracy (SFDA) and average tracking accuracy (ATA) [98]. These metrics compare detection and tracking results against human-authored ground-truth, where SFDA$\in [0,1]$ corresponds to detection performance and ATA$\in [0,1]$ corresponds to tracking performance. We authored the ground-truth for all videos with the Video Performance Evaluation Resource (ViPER) tool [51].

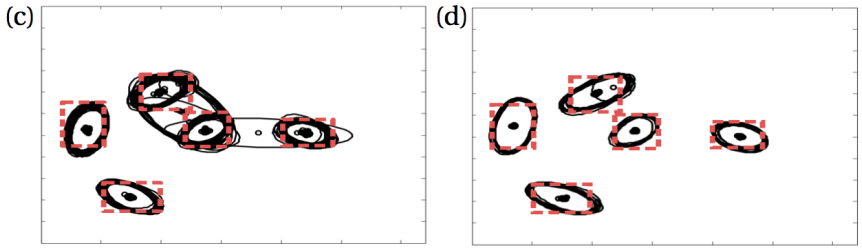
3.5.1 *Insect Tracking*

In this experiment, we aim to demonstrate the ability of the DDPMO to find and track objects in a difficult detection scenario. The video contains six ants with a similar texture and color distribution as the background. The ants are hard to discern, and it is unclear how a predefined detection criteria might be constructed. Futher, the ants move erratically and the spatial data extracted via frame differencing does not yield a clear segmentation of the objects in individual frames. A still image from the video, with ant locations shown, is given in Figure 3(a). We compare the SMC and PMCMC inference algorithms, and find that PMCMC yields more accurate posterior samples (3(d)) than SMC (3(c)). Ground-truth bounding boxes (dashed) are overlaid on the posterior samples. The MAP PMCMC sample is shown in 3(b) and posterior samples of the object tracks are shown in 3(f), along with overlaid ground-truth tracks (dashed). SFDA and ATA performance metrics for all comparison methods are shown in 3(e). The DDPMO yields higher metric values than all other detection-free comparison methods, with PMCMC inference scoring higher than SMC. The comparison methods seemed to suffer from two primary problems: very few tracklets could follow object positions for an extended sequence of frames, and clustering tracklets into full tracks sharply decreased in accuracy when the objects came into close contact with one another.

3.5.2 *Comparisons with Detector-based Methods*

In this experiment we aim to show that our general-purpose algorithm can compete against state of the art object-specific algorithms, even when it has no prior information about the objects. We use a benchmark human-tracking video from the International Workshop on Performance Evaluation of Tracking and Surveillance (PETS) 2009-2013 conferences [55], due to its prominence in a number of studies (listed in Figure 12(f)). It consists of a monocular, stationary camera, 794 frame video sequence containing a number of walking humans. Due to the large number of frames and objects in this video, we perform inference with the SMC algorithm only.





(e)

Method Name	SFDA	ATA
LDOF + NCUT	0.26	0.15
KLT + NCUT	0.22	0.16
LDOF + NNMF	0.25	0.19
KLT + NNMF	0.11	0.19
Blob-Tracking	0.29	0.06
DDPMO (SMC)	0.44	0.41
DDPMO (PMCMC)	0.50	0.47

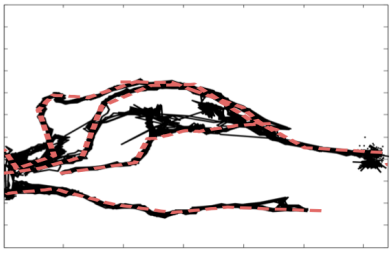


Figure 11: The ants in (a) are difficult to discern (positions labeled). We plot 100 samples from the inferred posterior over object parameters (using SMC (c) and PMCMC (d)) with ground-truth bounding boxes overlaid (dashed). PMCMC proves to give more accurate object parameter samples. We also plot samples over object tracks (sequences of mean parameters) using PMCMC in (f) , and its MAP sample in (b). We show the SFDA and ATA scores for all comparison methods in (e).

The DDPMO is compared against ten object-specific detector-based methods from the PETS conferences. These methods all either leverage assumptions about the orientation, position, or parts of humans, or explicitly use pre-trained human detectors. For example, out of the three top scoring comparison methods, [31] uses a state of the art pedestrian detector, [208] performs head and feet detection, and [43] uses assumptions about human geometry and orientation to segment humans and remove shadows.

In Figure 4(a-d), the MAP sample from the posterior distribution over the object parameters is overlayed on the extracted data over a sequence of frames. The first 50 frames from the video are shown in 4(e), where the assignment of each data point is represented by color and marker type. We show the SFDA and ATA values for all methods in 4(f), and can see that the DDPMO yields comparable results, receiving the fourth highest SFDA score and tying for the second highest ATA score.

3.5.3 *Tracking Populations of T Cells*

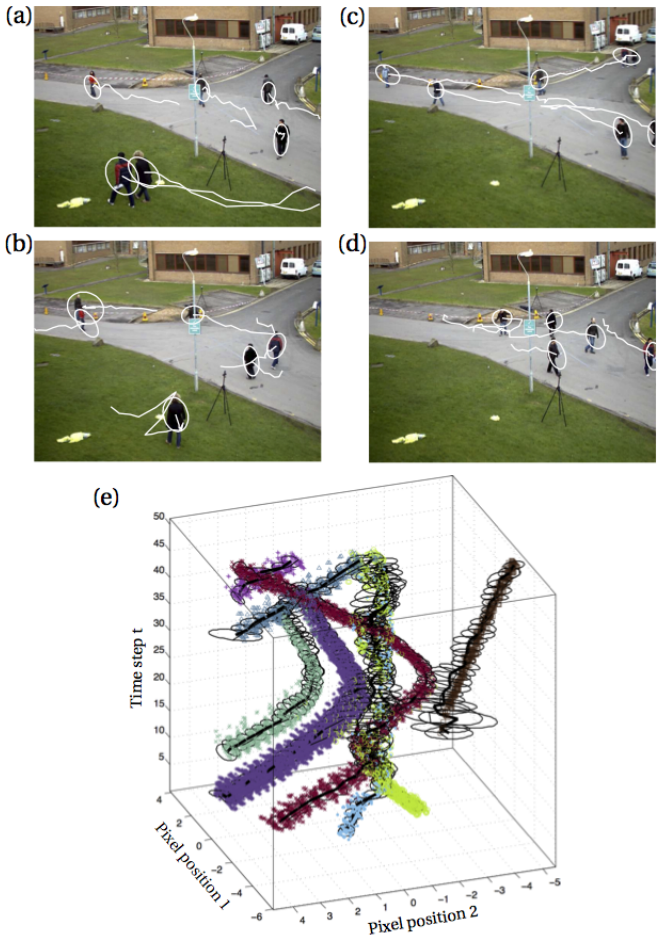
Automated tracking tools for cells are useful for cell biologists and immunologists studying cell behavior. We present results on a video containing T cells that are hard to detect using conventional methods due to their low constrast appearance against a background (Figure 5(a)). Furthermore, there are a large number of cells (roughly 60 per frame, 92 total). In this experiment, we aim to demonstrate the ability of the DDPMO to perform a tough detection task while scaling up to a large number of objects. Ground-truth bounding boxes for the cells at a single frame [123] are shown in 5(b) and PMCMC inference results (where the MAP sample is plotted) are shown in in 5(c). A histogram illustrating the inferred posterior over the total number of cells is shown in 5(e). It peaks around 87, near the true value of 92 cells.

Manually hand-labeling cell positions to train a detector is feasible but time consuming; we show how unsupervised detection results from the DDPMO can be used to automatically train a supervised cell detector (a linear SVM), which can then be applied (via a sliding window across each frame) as a secondary, speedy method of detection (Figure 5(d)). This type of strategy in conjunction with the DDPMO could allow for an ad-hoc way of constructing detectors for arbitrary objects on the fly, which could be taken and used in other vision applications, without needing an explicit predefined algorithm for object detection.

3.6 CONCLUSION

The DDPMO provides the ability to find, track, and learn representations of arbitrary objects in a video, in a single model framework, in order to accomplish detection-free tracking. We detail inference algorithms that can be used in both online and offline settings and provide results on a number of real video datasets. We consistently achieve better performance than other detection-free tracking strategies and even achieve competitive performance with object-specific detector-based methods on a human tracking benchmark video. Furthermore, we've demonstrated the ability of our model to perform accurate localization and tracking in videos with large numbers of objects, and in those that contain instances of full or partial occlusion, objects with shifting appearance or orientation, and objects for which it is difficult to construct an explicit detection strategy.

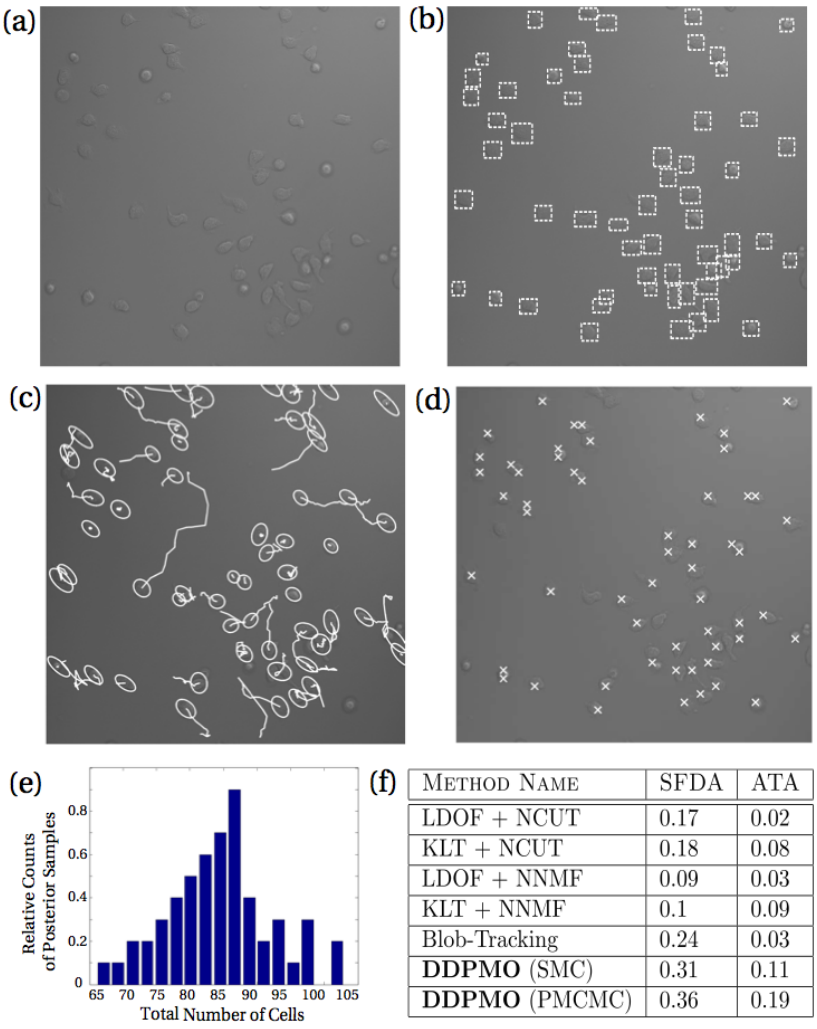
We've also shown how the DDPMO can provide an unsupervised, detection-free way to train a discriminative object detector for arbitary objects. This combination could provide a way to build object detectors for unknown objects on the fly and increase the accuracy or speed of localization and tracking within our model framework. We envision the DDPMO to be particularly useful in settings where the number and type of objects are unknown, or the objects' appearances are highly variable, and a high-quality general-purpose object localization and tracking method is desirable.



(f)

Method Name	SFDA	ATA
Breitenstein [9]	0.57	0.30
Yang [32]	0.55	0.45
Conte [13]	0.53	0.06
DDPMO (SMC)	0.51	0.30
Berclaz [7]	0.48	0.15
Alahi 1 [1]	0.43	0.04
Alahi 2 [1]	0.42	0.05
Bolme 1 [8]	0.41	NA
Ge [20]	0.38	0.04
Bolme 2 [8]	0.34	NA
Arsic [4]	0.18	0.02

Figure 12: DDPMO results on the PETS human tracking benchmark dataset and comparison with object-detector-based methods. The MAP object parameter samples are overlaid on four still video frames (a-d). The MAP object parameter samples are also shown for a sequence of frames (a 50 time-step sequence) along with spatial pixel observations (e) (where the assignment variables $c_{t,n}$ for each pixel are represented by marker type and color). The SFDA and ATA performance metric results for the DDPMO and ten human-specific, detection-based tracking algorithms are shown in (f), demonstrating that the DDPMO achieves comparable performance to these human-specific methods. Comparison results were provided by the authors of [55].



(f)

Method Name	SFDA	ATA
LDOF + NCUT	0.17	0.02
KLT + NCUT	0.18	0.08
LDOF + NNMF	0.09	0.03
KLT + NNMF	0.1	0.09
Blob-Tracking	0.24	0.03
DDPMO (SMC)	0.31	0.11
DDPMO (PMCMC)	0.36	0.19

Figure 13: T cells are numerous, and hard to detect due to low contrast images (a). For a single frame, ground-truth bounding boxes are overlaid in (b), and inferred detection and tracking results are overlaid in (c). A histogram showing the posterior distribution over the total number of cells is shown in (e). The SFDA and ATA for the detection-free comparison methods are shown in (f). Inferred cell positions (unsupervised) were used to automatically train an SVM for supervised cell detection; SVM detected cell positions for a single frame are shown in (d).

Part II

SCALABLE AND DISTRIBUTED INFERENCE ALGORITHMS

4

EMBARRASSINGLY PARALLEL MCMC

4.1 CHAPTER SUMMARY

Communication costs, resulting from synchronization requirements during learning, can greatly slow down many parallel machine learning algorithms. In this chapter, we present a parallel Markov chain Monte Carlo (MCMC) algorithm in which subsets of data are processed independently, with very little communication. First, we arbitrarily partition data onto multiple machines. Then, on each machine, any classical MCMC method (e.g., Gibbs sampling) may be used to draw samples from a posterior distribution given the data subset. Finally, the samples from each machine are combined to form samples from the full posterior. This embarrassingly parallel algorithm allows each machine to act independently on a subset of the data (without communication) until the final combination stage. We prove that our algorithm generates asymptotically exact samples and empirically demonstrate its ability to parallelize burn-in and sampling in several models.

4.2 INTRODUCTION

Markov chain Monte Carlo (MCMC) methods are popular tools for performing approximate Bayesian inference via posterior sampling. One major benefit of these techniques is that they guarantee asymptotically exact recovery of the posterior distribution as the number of posterior samples grows. However, MCMC methods may take a prohibitively long time, since for N data points, most methods must perform $O(N)$ operations to draw a sample. Furthermore, MCMC methods might require a large number of "burn-in" steps before beginning to generate representative samples. Further complicating matters is the issue that, for many big data applications, it is necessary to store and process data on multiple machines, and so MCMC must be adapted to run in these data-distributed settings.

Researchers currently tackle these problems independently, in two primary ways. To speed up sampling, multiple independent chains of MCMC can be run in parallel [110, 130, 199]; however, each chain is still run on the entire dataset, and there is no speed-up of the burn-in process (as each chain must still complete the full burn-in before generating samples). To run MCMC when data is partitioned among multiple machines, each machine can perform computation that involves a subset of the data and exchange information at each iteration to draw a sample [109, 147, 173]; however, this requires a significant amount of communication between machines, which can

greatly increase computation time when machines wait for external information [2, 84].

We aim to develop a procedure to tackle both problems simultaneously, to allow for quicker burn-in and sampling in settings where data are partitioned among machines. To accomplish this, we propose the following: on each machine, run MCMC on only a subset of the data (independently, without communication between machines), and then combine the samples from each machine to algorithmically construct samples from the full-data posterior distribution. We'd like our procedure to satisfy the following four criteria:

1. Each machine only has access to a portion of the data.
2. Each machine performs MCMC independently, without communicating (i.e. the procedure is "embarrassingly parallel").
3. Each machine can use any type of MCMC to generate samples.
4. The combination procedure yields provably asymptotically exact samples from the full-data posterior.

The third criterion allows existing MCMC algorithms or software packages to be run directly on subsets of the data—the combination procedure then acts as a post-processing step to transform the samples to the correct distribution. Note that this procedure is particularly suitable for use in a MapReduce [46] framework. Also note that, unlike current strategies, this procedure does not involve multiple "duplicate" chains (as each chain uses a different portion of the data and samples from a different posterior distribution), nor does it involve parallelizing a single chain (as there are multiple chains operating independently). We will show how this allows our method to, in fact, parallelize and greatly reduce the time required for burn-in.

In this chapter we will (1) introduce and define the *subposterior* density—a modified posterior given a subset of the data—which will be used heavily, (2) present methods for the embarrassingly parallel MCMC and combination procedure, (3) prove theoretical guarantees about the samples generated from our algorithm, (4) describe the current scope of the presented method (i.e. where and when it can be applied), and (5) show empirical results demonstrating that we can achieve speedups for burn-in and sampling while meeting the above four criteria.

4.3 EMBARRASSINGLY PARALLEL MCMC

The basic idea behind our method is to partition a set of N i.i.d. data points $x^N = \{x_1, \cdots, x_N\}$ into M subsets, sample from the *subposterior*—the posterior given a data subset with an underweighted prior—in parallel, and then combine the resulting samples to form samples from the full-data posterior $p(\theta|x^N)$, where $\theta \in \mathbb{R}^d$ and $p(\theta|x^N) \propto p(\theta)p(x^N|\theta) = p(\theta)\prod_{i=1}^N p(x_i|\theta)$.

More formally, given data x^N partitioned into M subsets $\{x^{n_1}, \ldots, x^{n_M}\}$, the procedure is:

1. For $m = 1, \ldots, M$ (in parallel):
   Sample from the subposterior p_m, where
$$p_m(\theta) \propto p(\theta)^{\frac{1}{M}} p(x^{n_m}|\theta). \tag{32}$$
2. Combine the subposterior samples to produce samples from an estimate of the subposterior density product $p_1 \cdots p_M$, which is proportional to the full-data posterior, i.e. $p_1 \cdots p_M(\theta) \propto p(\theta|x^N)$.

We want to emphasize that we do not need to iterate over these steps and the combination stage (step 3) is the only step that requires communication between machines. Also note that sampling from each subposterior (step 2) can typically be done in the same way as one would sample from the full-data posterior. For example, when using the Metropolis-Hastings algorithm, one would compute the likelihood ratio as $\frac{p(\theta^*)^{\frac{1}{M}} p(x^{n_m}|\theta^*)}{p(\theta)^{\frac{1}{M}} p(x^{n_m}|\theta)}$ instead of $\frac{p(\theta^*)p(x^N|\theta^*)}{p(\theta)p(x^N|\theta)}$, where θ^* is the proposed move. In the next section, we show how the combination stage (step 3) is carried out to generate samples from the full-data posterior using the subposterior samples.

4.4 COMBINING SUBPOSTERIOR SAMPLES

Our general idea is to combine the subposterior samples in such a way that we are implicitly sampling from an estimate of the subposterior density product function $\widehat{p_1 \cdots p_M}(\theta)$. If our density product estimator is consistent, then we can show that we are drawing asymptotically exact samples from the full posterior. Further, by studying the estimator error rate, we can explicitly analyze how quickly the distribution from which we are drawing samples is converging to the true posterior (and thus compare different combination algorithms).

In the following three sections we present procedures that yield samples from different estimates of the density product. Our first example is based on a simple parametric estimator motivated by the Bernstein-von Mises theorem [111]; this procedure generates approximate (asymptotically biased) samples from the full posterior. Our second example is based on a nonparametric estimator, and produces asymptotically exact samples from the full posterior. Our third example is based on a semiparametric estimator, which combines beneficial aspects from the previous two estimators while also generating asymptotically exact samples.

4.4.1 *Approximate Posterior Sampling with Parametric Density Product Estimation*

The first method for forming samples from the full posterior given subposterior samples involves using an approximation based on the Bernstein-von Mises (Bayesian

central limit) theorem, an important result in Bayesian asymptotic theory. Assuming that a unique, true data-generating model exists and is denoted θ_0, this theorem states that the posterior tends to a normal distribution concentrated around θ_0 as the number of observations grows. In particular, under suitable regularity conditions, the posterior $P(\theta|x^N)$ is well approximated by $\mathcal{N}_d(\theta_0, F_N^{-1})$ (where F_N is the fisher information of the data) when N is large [111]. Since we aim to perform posterior sampling when the number of observations is large, a normal parametric form often serves as a good posterior approximation. A similar approximation was used in [3] in order to facilitate fast, approximately correct sampling. We therefore estimate each subposterior density with $\widehat{p}_m(\theta) = \mathcal{N}_d(\theta|\widehat{\mu}_m, \widehat{\Sigma}_m)$ where $\widehat{\mu}_m$ and $\widehat{\Sigma}_m$ are the sample mean and covariance, respectively, of the subposterior samples. The product of the M subposterior densities will be proportional to a Gaussian pdf, and our estimate of the density product function $p_1 \cdots p_M(\theta) \propto p(\theta|x^N)$ is

$$\widehat{p_1 \cdots p_M}(\theta) = \widehat{p}_1 \cdots \widehat{p}_M(\theta) \propto \mathcal{N}_d\left(\theta|\widehat{\mu}_M, \widehat{\Sigma}_M\right),$$

where the parameters of this distribution are

$$\widehat{\Sigma}_M = \left(\sum_{m=1}^{M} \widehat{\Sigma}_m^{-1}\right)^{-1} \tag{33}$$

$$\widehat{\mu}_M = \widehat{\Sigma}_M \left(\sum_{m=1}^{M} \widehat{\Sigma}_m^{-1} \widehat{\mu}_m\right). \tag{34}$$

These parameters can be computed quickly and, if desired, online (as new subposterior samples arrive).

4.4.2 *Asymptotically Exact Posterior Sampling with Nonparametric Density Product Estimation*

In the previous method we made a parametric assumption based on the Bernstein-von Mises theorem, which allows us to generate approximate samples from the full posterior. Although this parametric estimate has quick convergence, it generates asymptotically biased samples, especially in cases where the posterior is particularly non-Gaussian. In this section, we develop a procedure that implicitly samples from the product of nonparametric density estimates, which allows us to produce asymptotically exact samples from the full posterior. By constructing a consistent density product estimator from which we can generate samples, we ensure that the distribution from which we are sampling converges to the full posterior.

Given T samples[1] $\{\theta_{t_m}^m\}_{t_m=1}^T$ from a subposterior p_m, we can write the kernel density estimator $\widehat{p}_m(\theta)$ as,

$$\begin{aligned}\widehat{p}_m(\theta) &= \frac{1}{T}\sum_{t_m=1}^{T}\frac{1}{h^d}K\left(\frac{\|\theta-\theta_{t_m}^m\|}{h}\right)\\ &= \frac{1}{T}\sum_{t_m=1}^{T}\mathcal{N}_d(\theta|\theta_{t_m}^m, h^2 I_d),\end{aligned}$$

where we have used a Gaussian kernel with bandwidth parameter h. After we have obtained the kernel density estimator $\widehat{p}_m(\theta)$ for M subposteriors, we define our nonparametric density product estimator for the full posterior as

$$\begin{aligned}\widehat{p_1\cdots p_M}(\theta) &= \widehat{p}_1\cdots\widehat{p}_M(\theta)\\ &= \frac{1}{T^M}\prod_{m=1}^{M}\sum_{t_m=1}^{T}\mathcal{N}_d(\theta|\theta_{t_m}^m, h^2 I_d)\\ &\propto \sum_{t_1=1}^{T}\cdots\sum_{t_M=1}^{T} w_{t\cdot}\,\mathcal{N}_d\left(\theta\middle|\bar{\theta}_{t\cdot}, \frac{h^2}{M}I_d\right).\end{aligned} \tag{35}$$

This estimate is the probability density function (pdf) of a mixture of T^M Gaussians with *unnormalized* mixture weights $w_{t\cdot}$. Here, we use $t\cdot = \{t_1,\ldots,t_M\}$ to denote the set of indices for the M samples $\{\theta_{t_1}^1,\ldots,\theta_{t_M}^M\}$ (each from a separate machine) associated with a given mixture component, and we define

$$\bar{\theta}_{t\cdot} = \frac{1}{M}\sum_{m=1}^{M}\theta_{t_m}^m \tag{36}$$

$$w_{t\cdot} = \prod_{m=1}^{M}\mathcal{N}_d\left(\theta_{t_m}^m|\bar{\theta}_{t\cdot}, h^2 I_d\right). \tag{37}$$

Although there are T^M possible mixture components, we can efficiently generate samples from this mixture by first sampling a mixture component (based on its unnormalized component weight $w_{t\cdot}$) and then sampling from this (Gaussian) component. In order to sample mixture components, we use an independent Metropolis within Gibbs (IMG) sampler. This is a form of MCMC, where at each step in the Markov chain, a single dimension of the current state is proposed (i.e. sampled) independently of its current value (while keeping the other dimensions fixed) and then is accepted or rejected. In our case, at each step, a new mixture component is proposed by redrawing one of the M current sample indices $t_m \in t\cdot$ associated with the component uniformly and then accepting or rejecting the resulting proposed

1 For ease of description, we assume each machine generates the same number of samples, T. In practice, they do not have to be the same.

component based on its mixture weight. We give the IMG algorithm for combining subposterior samples in Algorithm 5.[2]

In certain situations, Algorithm 5 may have a low acceptance rate and therefore may mix slowly. One way to remedy this is to perform the IMG combination algorithm multiple times, by first applying it to groups of $\tilde{M} < M$ subposteriors and then applying the algorithm again to the output samples from each initial application. For example, one could begin by applying the algorithm to all $\frac{M}{2}$ pairs (leaving one subposterior alone if M is odd), then repeating this process—forming pairs and applying the combination algorithm to pairs only—until there is only one set of samples remaining, which are samples from the density product estimate.

Algorithmus 5 : Asymptotically Exact Sampling via Nonparametric Density Product Estimation from Neiswanger et al. [142]

Input : Subposterior samples: $\{\theta^1_{t_1}\}_{t_1=1}^T \sim p_1(\theta), \ldots, \{\theta^M_{t_M}\}_{t_M=1}^T \sim p_M(\theta)$.
Output : Posterior samples (as $T \to \infty$): $\{\theta_i\}_{i=1}^T \sim p_1 \cdots p_M(\theta) \propto p(\theta|x^n)$

Draw $t_\cdot = \{t_1, \ldots, t_M\} \overset{\text{iid}}{\sim} \text{Unif}(\{1, \ldots, T\})$
for $i = 1, \ldots, T$ **do**
  Set $h \leftarrow i^{-1/(4+d)}$
  for $m = 1, \ldots, M$ **do**
    Set $c_\cdot \leftarrow t_\cdot$
    Draw $c_m \sim \text{Unif}(\{1, \ldots, T\})$
    Draw $u \sim \text{Unif}([0, 1])$
    if $u < w_{c_\cdot}/w_{t_\cdot}$ **then**
      Set $t_\cdot \leftarrow c_\cdot$
  Draw $\theta_i \sim \mathcal{N}_d(\bar{\theta}_{t_\cdot}, \frac{h^2}{M} I_d)$

4.4.3 *Asymptotically Exact Posterior Sampling with Semiparametric Density Product Estimation*

Our first example made use of a parametric estimator, which has quick convergence, but may be asymptotically biased, while our second example made use of a nonparametric estimator, which is asymptotically exact, but may converge slowly when the number of dimensions is large. In this example, we implicitly sample from a semiparametric density product estimate, which allows us to leverage the fact that the full posterior has a near-Gaussian form when the number of observations is large, while still providing an asymptotically unbiased estimate of the posterior density, as the number of subposterior samples $T \to \infty$.

2 Again for simplicity, we assume that we generate T samples to represent the full posterior, where T is the number of subposterior samples from each machine.

We make use of a semiparametric density estimator for p_m that consists of the product of a parametric estimator $\widehat{f}_m(\theta)$ (in our case $\mathcal{N}_d(\theta|\widehat{\mu}_m, \widehat{\Sigma}_m)$ as above) and a nonparametric estimator $\widehat{r}(\theta)$ of the correction function $r(\theta) = p_m(\theta)/\widehat{f}_m(\theta)$ [83]. This estimator gives a near-Gaussian estimate when the number of samples is small, and converges to the true density as the number of samples grows. Given T samples $\{\theta^m_{t_m}\}_{t_m=1}^T$ from a subposterior p_m, we can write the estimator as

$$\begin{aligned}
\widehat{p}_m(\theta) &= \widehat{f}_m(\theta)\widehat{r}(\theta) \\
&= \frac{1}{T}\sum_{t_m=1}^{T} \frac{1}{h^d} K\left(\frac{\|\theta - \theta^m_{t_m}\|}{h}\right) \frac{\widehat{f}_m(\theta)}{\widehat{f}_m(\theta^m_{t_m})} \\
&= \frac{1}{T}\sum_{t_m=1}^{T} \frac{\mathcal{N}_d(\theta|\theta^m_{t_m}, h^2 I_d)\mathcal{N}_d(\theta|\widehat{\mu}_m, \widehat{\Sigma}_m)}{\mathcal{N}_d(\theta^m_{t_m}|\widehat{\mu}_m, \widehat{\Sigma}_m)},
\end{aligned}$$

where we have used a Gaussian kernel with bandwidth parameter h for the nonparametric component of this estimator. Therefore, we define our semiparametric density product estimator to be

$$\begin{aligned}
\widehat{p_1 \cdots p_M}(\theta) &= \widehat{p}_1 \cdots \widehat{p}_M(\theta) \\
&= \frac{1}{T^M}\prod_{m=1}^{M}\sum_{t_m=1}^{T} \frac{\mathcal{N}_d(\theta|\theta^m_{t_m}, hI_d)\mathcal{N}_d(\theta|\widehat{\mu}_m, \widehat{\Sigma}_m)}{h^d \mathcal{N}_d(\theta^m_{t_m}|\widehat{\mu}_m, \widehat{\Sigma}_m)} \\
&\propto \sum_{t_1=1}^{T} \cdots \sum_{t_M=1}^{T} W_{t_\cdot}\, \mathcal{N}_d\left(\theta|\mu_{t_\cdot}, \Sigma_{t_\cdot}\right).
\end{aligned}$$

This estimate is proportional to the pdf of a mixture of T^M Gaussians with unnormalized mixture weights,

$$W_{t_\cdot} = \frac{w_{t_\cdot}\, \mathcal{N}_d\left(\bar{\theta}_{t_\cdot}|\widehat{\mu}_M, \widehat{\Sigma}_M + \frac{h}{M} I_d\right)}{\prod_{m=1}^{M} \mathcal{N}_d(\theta^m_{t_m}|\widehat{\mu}_m, \widehat{\Sigma}_m)},$$

where $\bar{\theta}_{t_\cdot}$ and $w_{t_\cdot}$ are given in Eqs. 36 and 37. We can write the parameters of a given mixture component $\mathcal{N}_d(\theta|\mu_{t_\cdot}, \Sigma_{t_\cdot})$ as

$$\begin{aligned}
\Sigma_{t_\cdot} &= \left(\frac{M}{h} I_d + \widehat{\Sigma}_M^{-1}\right)^{-1}, \\
\mu_{t_\cdot} &= \Sigma_{t_\cdot}\left(\frac{M}{h} I_d \bar{\theta}_{t_\cdot} + \widehat{\Sigma}_M^{-1}\widehat{\mu}_M\right),
\end{aligned}$$

where $\widehat{\mu}_M$ and $\widehat{\Sigma}_M$ are given by Eq. 33 and 34. We can sample from this semiparametric estimate using the IMG procedure outlined in Algorithm 5, replacing the component weights $w_{t_\cdot}$ with $W_{t_\cdot}$ and the component parameters $\bar{\theta}_{t_\cdot}$ and $\frac{h}{M} I_d$ with $\mu_{t_\cdot}$ and $\Sigma_{t_\cdot}$.

We also have a second semiparametric procedure that may give higher acceptance rates in the IMG algorithm. We follow the above semiparametric procedure, where each component is a normal distribution with parameters $\mu_{t\cdot}$ and $\Sigma_{t\cdot}$, but we use the nonparametric component weights $w_{t\cdot}$ instead of $W_{t\cdot}$. This procedure is also asymptotically exact, since the semiparametric component parameters $\mu_{t\cdot}$ and $\Sigma_{t\cdot}$ approach the nonparametric component parameters $\bar{\theta}_{t\cdot}$ and $\frac{h}{M} I_d$ as $h \to 0$, and thus this procedure tends to the nonparametric procedure given in Algorithm 5.

4.5 METHOD COMPLEXITY

Given M data subsets, to produce T samples in d dimensions with the nonparametric or semiparametric asymptotically exact procedures (Algorithm 1) requires $O(dTM^2)$ operations. The variation on this algorithm that performs this procedure $M-1$ times on pairs of subposteriors (to increase the acceptance rate; detailed in Section 4.4.2) instead requires only $O(dTM)$ operations.

We have presented our method as a two step procedure, where first parallel MCMC is run to completion, and then the combination algorithm is applied to the M sets of samples. We can instead perform an online version of our algorithm: as each machine generates a sample, it immediately sends it to a master machine, which combines the incoming samples[3] and performs the accept or reject step (Algorithm 1, lines 3-12). This allows the parallel MCMC phase and the combination phase to be performed in parallel, and does not require transfering large volumes of data, as only a single sample is ever transferred at a time.

The total communication required by our method is transferring $O(dTM)$ scalars (T samples from each of M machines), and as stated above, this can be done online as MCMC is being carried out. Further, the communication is unidirectional, and each machine does not pause and wait for any information from other machines during the parallel sampling procedure.

4.6 THEORETICAL RESULTS

Our second and third procedures aim to draw asymptotically exact samples by sampling from (fully or partially) nonparametric estimates of the density product. We prove the asymptotic correctness of our estimators, and bound their rate of convergence. This will ensure that we are generating asymptotically correct samples from the full posterior as the number of samples T from each subposterior grows.

3 For the semiparametric method, this will involve an online update of mean and variance Gaussian parameters.

4.6.1 *Density Product Estimate Convergence and Risk Analysis*

To prove (mean-square) consistency of our estimator, we give a bound on the mean-squared error (MSE), and show that it tends to zero as we increase the number of samples drawn from each subposterior. To prove this, we first bound the bias and variance of the estimator. The following proofs make use of similar bounds on the bias and variance of the nonparametric and semiparametric density estimators, and therefore the theory applies to both the nonparametric and semiparametric density product estimators.

Throughout this analysis, we assume that we have T samples $\{\theta_{t_m}^m\}_{t_m=1}^T \subset \mathcal{X} \subset \mathbb{R}^d$ from each subposterior ($m = 1, \ldots, M$), and that $h \in \mathbb{R}_+$ denotes the bandwidth of the nonparametric density product estimator (which is annealed to zero as $T \to \infty$ in Algorithm 5). Let Hölder class $\Sigma(\beta, L)$ on $\mathcal{X}$ be defined as the set of all $\ell = \lfloor \beta \rfloor$ times differentiable functions $f : \mathcal{X} \to \mathbb{R}$ whose derivative $f^{(\ell)}$ satisfies

$$|f^{(\ell)}(\theta) - f^{(\ell)}(\theta')| \leqslant L \left|\theta - \theta'\right|^{\beta-\ell} \quad \text{for all } \theta, \theta' \in \mathcal{X}.$$

We also define the class of densities $\mathcal{P}(\beta, L)$ to be

$$\mathcal{P}(\beta, L) = \left\{ p \in \Sigma(\beta, L) \;\middle|\; p \geqslant 0, \int p(\theta) d\theta = 1 \right\}.$$

We also assume that all subposterior densities p_m are bounded, i.e. that there exists some $b > 0$ such that $p_m(\theta) \leqslant b$ for all $\theta \in \mathbb{R}^d$ and $m \in \{1, \ldots, M\}$.

First, we bound the bias of our estimator. This shows that the bias tends to zero as the bandwidth shrinks.

Lemma 4.6.1. *The bias of the estimator* $\widehat{p_1 \cdots p_M}(\theta)$ *satisfies*

$$\sup_{p_1, \ldots, p_M \in \mathcal{P}(\beta, L)} \left| \mathbb{E}\left[\widehat{p_1 \cdots p_M}(\theta)\right] - p_1 \cdots p_M(\theta) \right| \leqslant \sum_{m=1}^{M} c_m h^{m\beta}$$

for some $c_1, \ldots, c_M > 0$.

Proof. For all $p_1, \ldots, p_M \in \mathcal{P}(\beta, L)$,

$$\begin{aligned}
\left|\mathbb{E}\left[\widehat{p_1 \cdots p_M}\right] - p_1 \cdots p_M\right| &= \left|\mathbb{E}\left[\widehat{p}_1 \cdots \widehat{p}_M\right] - p_1 \cdots p_M\right| \\
&= \left|\mathbb{E}\left[\widehat{p_1}\right] \cdots \mathbb{E}\left[\widehat{p}_M\right] - p_1 \cdots p_M\right| \\
&\leqslant \left|(p_1 + \tilde{c}_1 h^\beta) \cdots (p_M + \tilde{c}_M h^\beta) - p_1 \cdots p_M\right| \\
&\leqslant \left|c_1 h^\beta + \ldots + c_M h^{M\beta}\right| \\
&\leqslant \left|c_1 h^\beta\right| + \ldots + \left|c_M h^{M\beta}\right| \\
&= \sum_{m=1}^{M} c_m h^{m\beta}
\end{aligned}$$

where we have used the fact that $\left|\mathbb{E}\left[\widehat{p_m}\right] - p\right| \leqslant \tilde{c}_m h^\beta$ for some $\tilde{c}_m > 0$. □

Next, we bound the variance of our estimator. This shows that the variance tends to zero as the number of samples grows large and the bandwidth shrinks.

Lemma 4.6.2. *The variance of the estimator* $\widehat{p_1 \cdots p_M}(\theta)$ *satisfies*

$$\sup_{p_1,\ldots,p_M \in \mathcal{P}(\beta,L)} \mathbb{V}\left[\widehat{p_1 \cdots p_M}(\theta)\right] \leqslant \sum_{m=1}^{M} \binom{M}{m} \frac{c_m}{T^m h^{dm}}$$

for some $c_1, \ldots, c_M > 0$ *and* $0 < h \leqslant 1$.

Proof. For all $p_1, \ldots, p_M \in \mathcal{P}(\beta, L)$,

$$\begin{aligned}
\mathbb{V}[\widehat{p_1 \cdots p_M}] &= \mathbb{E}\left[\widehat{p}_1^2\right] \cdots \mathbb{E}\left[\widehat{p}_M^2\right] - \mathbb{E}\left[\widehat{p}_1\right]^2 \cdots \mathbb{E}\left[\widehat{p}_M\right]^2 \\
&= \left(\prod_{m=1}^{M} \mathbb{V}\left[\widehat{p}_m\right] + \mathbb{E}\left[\widehat{p}_m\right]^2\right) - \left(\prod_{m=1}^{M} \mathbb{E}\left[\widehat{p}_m\right]^2\right) \\
&\leqslant \sum_{m=0}^{M-1} \binom{M}{m} \frac{\tilde{c}^m c^{M-m}}{T^{M-m} h^{d(M-m)}} \\
&\leqslant \sum_{m=1}^{M} \binom{M}{m} \frac{c_m}{T^m h^{dm}}
\end{aligned}$$

where we have used the facts that $\mathbb{V}\left[\widehat{p}_m\right] \leqslant \frac{c}{Th^d}$ for some $c > 0$ and $\mathbb{E}\left[\widehat{p}_m\right]^2 \leqslant \tilde{c}$ for some $\tilde{c} > 0$. □

Finally, we use the bias and variance bounds to bound the MSE, which shows that our estimator is consistent.

Theorem 4.6.3. *If* $h \asymp T^{-1/(2\beta+d)}$, *the mean-squared error of the estimator* $\widehat{p_1 \cdots p_M}(\theta)$ *satisfies*

$$\sup_{p_1,\ldots,p_M \in \mathcal{P}(\beta,L)} \mathbb{E}\left[\int \left(\widehat{p_1 \cdots p_M}(\theta) - p_1 \cdots p_M(\theta)\right)^2 d\theta\right] \leqslant \frac{c}{T^{2\beta/(2\beta+d)}}$$

for some $c > 0$ *and* $0 < h \leqslant 1$.

Proof. For all $p_1, \ldots, p_M \in \mathcal{P}(\beta, L)$, using the fact that the mean-squared error is equal to the variance plus the bias squared, we have that

$$\begin{aligned}
\mathbb{E}\left[\int \left(\widehat{p_1 \cdots p_M}(\theta) - p_1 \cdots p_M(\theta)\right)^2 d\theta\right] &\leqslant \left(\sum_{m=1}^{M} c_m h^{m\beta}\right)^2 + \sum_{m=1}^{M} \binom{M}{m} \frac{\tilde{c}_m}{T^m h^{dm}} \\
&\leqslant k T^{-2\beta/(2\beta+d)} + \frac{\tilde{k}}{T^{1-d(2\beta+d)}} \quad \text{(for some } k, \tilde{k} > 0\text{)} \\
&\leqslant \frac{c}{T^{2\beta/(2\beta+d)}}
\end{aligned}$$

for some $c_1, \ldots, c_M > 0$ and $\tilde{c}_1, \ldots, \tilde{c}_M > 0$. □

4.7 METHOD SCOPE

The theoretical results and algorithms in this chapter hold for posterior distributions over finite-dimensional real spaces. These include generalized linear models (e.g. linear, logistic, or Poisson regression), mixture models with known weights, hierarchical models, and (more generally) finite-dimensional graphical models with unconstrained variables. This also includes both unimodal and multimodal posterior densities (such as in Section 4.9.2). However, the methods and theory presented here do not yet extend to cases such as infinite dimensional models (e.g. nonparametric Bayesian models [68]) nor to distributions over the simplex (e.g. topics in latent Dirichlet allocation [28]). In the future, we hope to extend this work to these domains.

4.8 RELATED WORK

In [3, 151, 198], the authors develop a way to sample approximately from a posterior distribution when only a small randomized mini-batch of data is used at each step. In [103], the authors used a hypothesis test to decide whether to accept or reject proposals using a small set of data (adaptively) as opposed to the exact Metropolis-Hastings rule. This reduces the amount of time required to compute the acceptance ratio. Since all of these algorithms are still sequential, they can be directly used in our algorithm to generate subposterior samples to further speed up the entire sampling process.

Several parallel MCMC algorithms have been designed for specific models, such as for topic models [147, 173] and nonparametric mixture models [200]. These approaches still require synchronization to be correct (or approximately correct), while ours aims for more general model settings and does not need synchronization until the final combination stage.

Consensus Monte Carlo [164] is perhaps the most relevant work to ours. In this algorithm, data is also portioned into different machines and MCMC is performed independently on each machine. Thus, it roughly has the same time complexity as our algorithm. However, the prior is not explicitly reweighted during sampling as we do in Eq 32, and final samples for the full posterior are generated by averaging subposterior samples. Furthermore, this algorithm has few theoretical guarantees. We find that this algorithm can be viewed as a relaxation of our nonparametric, asymptotically exact sampling procedure, where samples are generated from an evenly weighted mixture (instead of each component having weight $w_{t\cdot}$) and where each sample is set to $\bar{\theta}_{t\cdot}$ instead of being drawn from $\mathcal{N}\left(\bar{\theta}_{t\cdot}, \frac{h}{M} I_d\right)$. This algorithm is one of our experimental baselines.

4.9 EMPIRICAL STUDY

In the following sections, we demonstrate empirically that our method allows for quicker, MCMC-based estimation of a posterior distribution, and that our consistent-estimator-based procedures yield asymptotically exact results. We show our method on a few Bayesian models using both synthetic and real data. In each experiment, we compare the following strategies for parallel, communication-free sampling:[4]

- **Single chain full-data posterior samples** (`regularChain`)—Typical, single-chain MCMC for sampling from the full-data posterior.
- **Parametric subposterior density product estimate** (`parametric`)—For M sets of subposterior samples, the combination yielding samples from the parametric density product estimate.
- **Nonparametric subposterior density product estimate** (`nonparametric`)—For M sets of subposterior samples, the combination yielding samples from the nonparametric density product estimate.
- **Semiparametric subposterior density product estimate** (`semiparametric`)—For M sets of subposterior samples, the combination yielding samples from the semiparametric density product estimate.
- **Subposterior sample average** (`subpostAvg`)—For M sets of subposterior samples, the average of M samples consisting of one sample taken from each subposterior.
- **Subposterior sample pooling** (`subpostPool`)—For M sets of subposterior samples, the union of all sets of samples.
- **Duplicate chains full-data posterior sample pooling** (`duplicateChainsPool`)—For M sets of samples from the full-data posterior, the union of all sets of samples.

To assess the performance of our sampling and combination strategies, we ran a single chain of MCMC on the full data for 500,000 iterations, removed the first half as burn-in, and considered the remaining samples the "groundtruth" samples for the true posterior density. We then needed a general method to compare the distance between two densities given samples from each, which holds for general densities (including multimodal densities, where it is ineffective to compare moments such as the mean and variance[5]). Following work in density-based regression

4 We did not directly compare with the algorithms that require synchronization since the setup of these experiments can be rather different. We plan to explore these comparisons in future work.

5 In these cases, dissimilar densities might have similar low-order moments. See Section 4.9.2 for an example.

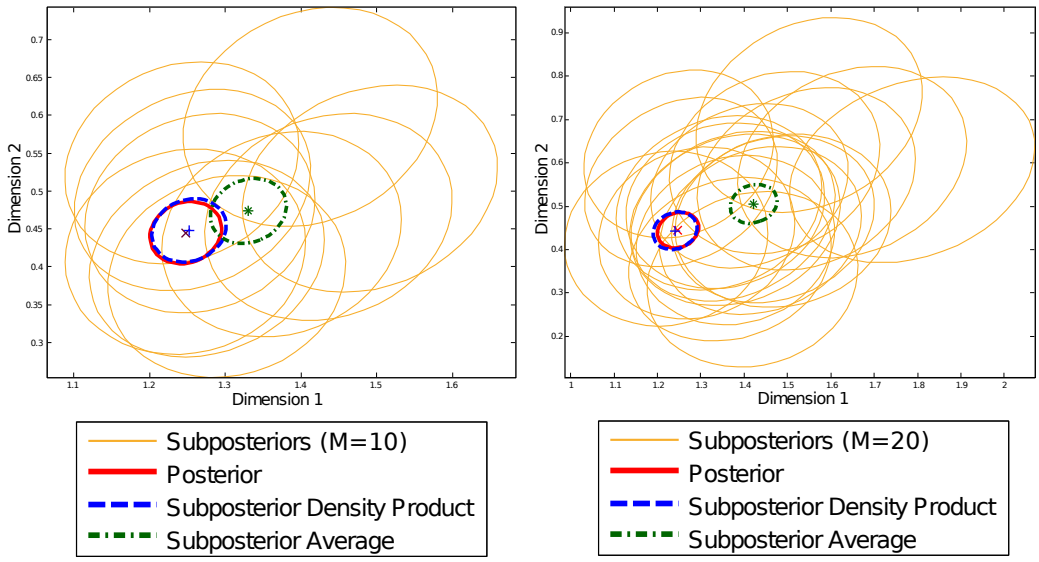


Figure 14: Bayesian logistic regression posterior ovals. We show the posterior 90% probability mass ovals for the first 2-dimensional marginal of the posterior, the M subposteriors, the subposterior density product (via the `parametric` procedure), and the subposterior average (via the `subpostAvg` procedure). We show M=10 subsets (left) and M=20 subsets (right). The subposterior density product generates samples that are consistent with the true posterior, while the `subpostAvg` produces biased results, which grow in error as M increases.

[150], we use an estimate of the L_2 distance, $d_2(p, \hat{p})$, between the groundtruth posterior density p and a proposed posterior density $\hat{p}$, where $d_2(p, \hat{p}) = \|p - \hat{p}\|_2 = \left(\int (p(\theta) - \hat{p}(\theta))^2 d\theta\right)^{1/2}$.

In the following experiments involving timing, to compute the posterior L_2 error at each time point, we collected all samples generated before a given number of seconds, and added the time taken to transfer the samples and combine them using one of the proposed methods. In all experiments and methods, we followed a fixed rule of removing the first $\frac{1}{6}$ samples for burn-in (which, in the case of combination procedures, was applied to each set of subposterior samples before the combination was performed).

Experiments were conducted with a standard cluster system. We obtained subposterior samples by submitting batch jobs to each worker since these jobs are all independent. We then saved the results to the disk of each worker and transferred them to the same machine which performed the final combination.

4.9.1 *Generalized Linear Models*

Generalized linear models are widely used for many regression and classification problems. Here we conduct experiments, using logistic regression as a test case,

on both synthetic and real data to demonstrate the speed of our parallel MCMC algorithm compared with typical MCMC strategies.

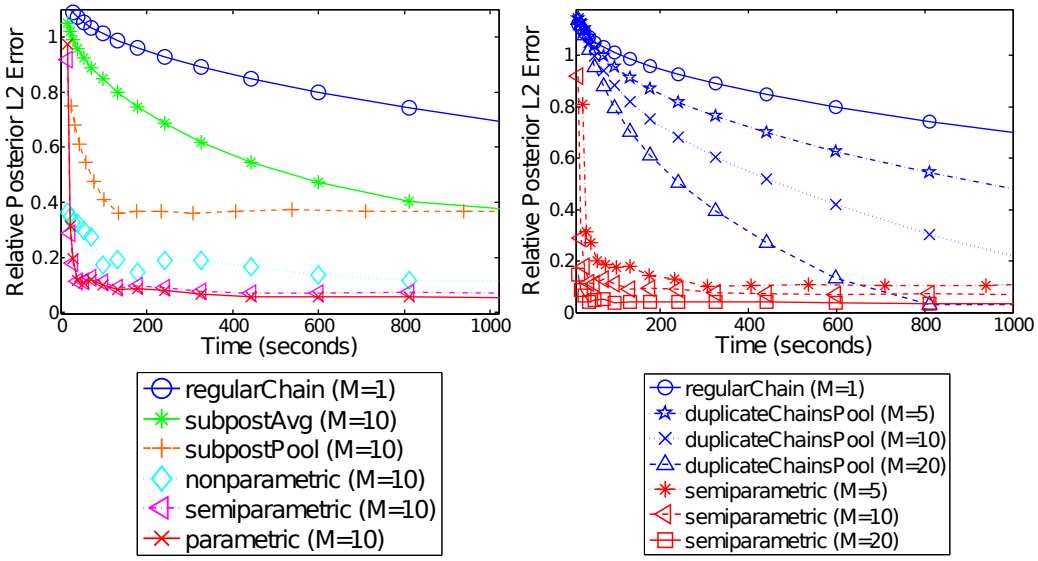


Figure 15: Posterior L_2 error vs time for logistic regression. Left: the three combination strategies proposed in this chapter (`parametric`, `nonparametric`, and `semiparametric`) reduce the posterior error much more quickly than a single full-data Markov chain; the `subpostAvg` and `subpostPool` procedures yield biased results. Right: we compare with multiple full-data Markov chains (`duplicateChainsPool`); our method yields faster convergence to the posterior even though only a fraction of the data is being used by each chain.

4.9.1.1 *Synthetic data*

Our synthetic dataset contains 50,000 observations in 50 dimensions. To generate the data, we drew each element of the model parameter β and data matrix X from a standard normal distribution, and then drew each outcome as $y_i \sim \text{Bernoulli}(\text{logit}^{-1}(X_i\beta))$ (where X_i denotes the i^{th} row of X)[6]. We use Stan, an automated Hamiltonian Monte Carlo (HMC) software package,[7] to perform sampling for both the true posterior (for groundtruth and comparison methods) and for the subposteriors on each machine. One advantage of Stan is that it is implemented with C++ and uses the No-U-Turn sampler for HMC, which does not require any user-provided parameters [87].

In Figure 14, we illustrate results for logistic regression, showing the subposterior densities, the subposterior density product, the subposterior sample average, and the true posterior density, for the number of subsets M set to 10 (left) and 20 (right). Samples generated by our approach (where we draw samples from the subposterior density product via the `parametric` procedure) overlap with the true posterior

6 Note that we did not explicitly include the intercept term in our logistic regression model.
7 http://mc-stan.org

much better than those generated via the `subpostAvg` (subposterior sample average) procedure— averaging of samples appears to create systematic biases. Futher, the error in averaging appears to increase as M grows. In Figure 15 (left) we show the posterior error vs time. A regular full-data chain takes much longer to converge to low error compared with our combination methods, and simple averaging and pooling of subposterior samples gives biased solutions.

We next compare our combination methods with multiple independent "duplicate" chains each run on the full dataset. Even though our methods only require a fraction of the data storage on each machine, we are still able to achieve a significant speed-up over the full-data chains. This is primarily because the duplicate chains cannot parallelize burn-in (i.e. each chain must still take some n steps before generating reasonable samples, and the time taken to reach these n steps does not decrease as more machines are added). However, in our method, each subposterior sampler can take each step more quickly, effectively allowing us to decrease the time needed for burn-in as we increase M. We show this empirically in Figure 15 (right), where we plot the posterior error vs time, and compare with full duplicate chains as M is increased.

Using a Matlab implementation of our combination algorithms, all (batch) combination procedures take under twenty seconds to complete on a 2.5GHz Intel Core i5 with 16GB memory.

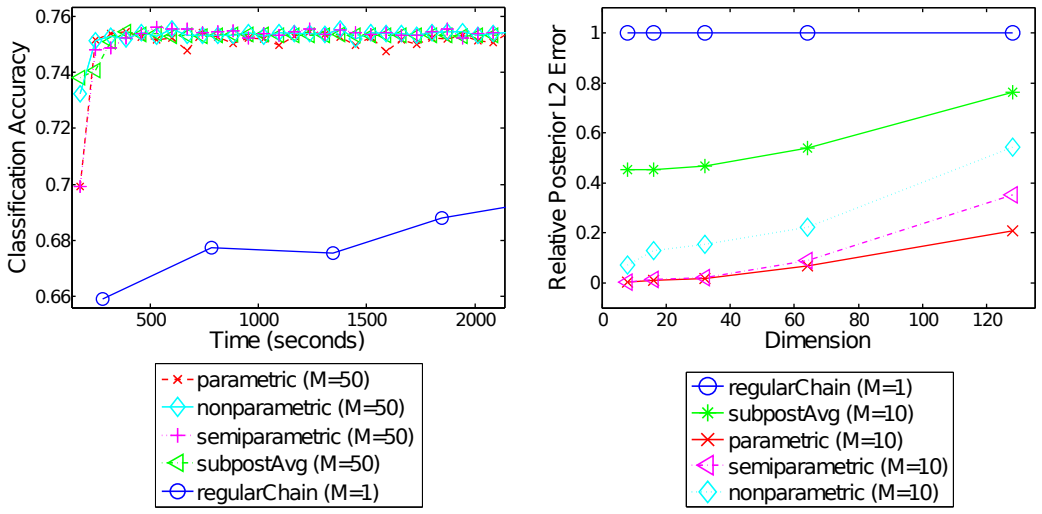


Figure 16: Left: Bayesian logistic regression classification accuracy vs time for the task of predicting forest cover type. Right: Posterior error vs dimension on synthetic data at 1000 seconds, normalized so that `regularChain` error is fixed at 1.

4.9.1.2 *Real-world data*

Here, we use the *covtype* (predicting forest cover types)[8] dataset, containing 581,012 observations in 54 dimensions. A single chain of HMC running on this entire dataset takes an average of 15.76 minutes per sample; hence, it is infeasible to generate groundtruth samples for this dataset. Instead we show classification accuracy vs time. For a given set of samples, we perform classification using a sample estimate of the posterior predictive distribution for a new label y with associated datapoint x, i.e.

$$\begin{aligned} P(y|x, y^N, x^N) &= \int P(y|x, \beta, y^N, x^N) P(\beta|x^N, y^N) \\ &\approx \frac{1}{S} \sum_{s=1}^{S} P(y|x, \beta_s) \end{aligned}$$

where x^N and y^N denote the N observations, and $P(y|x, \beta_s) = \text{Bernoulli}(\text{logit}^{-1}(x^\top \beta_s))$. Figure 16 (left) shows the results for this task, where we use M=50 splits. The parallel methods achieve a higher accuracy much faster than the single-chain MCMC algorithm.

4.9.1.3 *Scalability with dimension*

We investigate how the errors of our methods scale with dimensionality, to compare the different estimators implicit in the combination procedures. In Figure 16 (right) we show the relative posterior error (taken at 1000 seconds) vs dimension, for the synthetic data with M=10 splits. The errors at each dimension are normalized so that the `regularChain` error is equal to 1. Here, the `parametric` (asymptotically biased) procedure scales best with dimension, and the `semiparametric` (asymptotically exact) procedure is a close second. These results also demonstrate that, although the `nonparametric` method can be viewed as implicitly sampling from a nonparametric density estimate (which is usually restricted to low-dimensional densities), the performance of our method does not suffer greatly when we perform parallel MCMC on posterior distributions in much higher-dimensional spaces.

4.9.2 *Gaussian Mixture Models*

In this experiment, we aim to show correct posterior sampling in cases where the full-data posterior, as well as the subposteriors, are multimodal. We will see that the combination procedures that are asymptotically biased suffer greatly in these scenarios. To demonstrate this, we perform sampling in a Gaussian mixture model. We generate 50,000 samples from a ten component mixture of 2-d Gaussians. The resulting posterior is multimodal; this can be seen by the fact that the component labels can be arbitrarily permuted and achieve the same posterior value. For example,

8 http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets

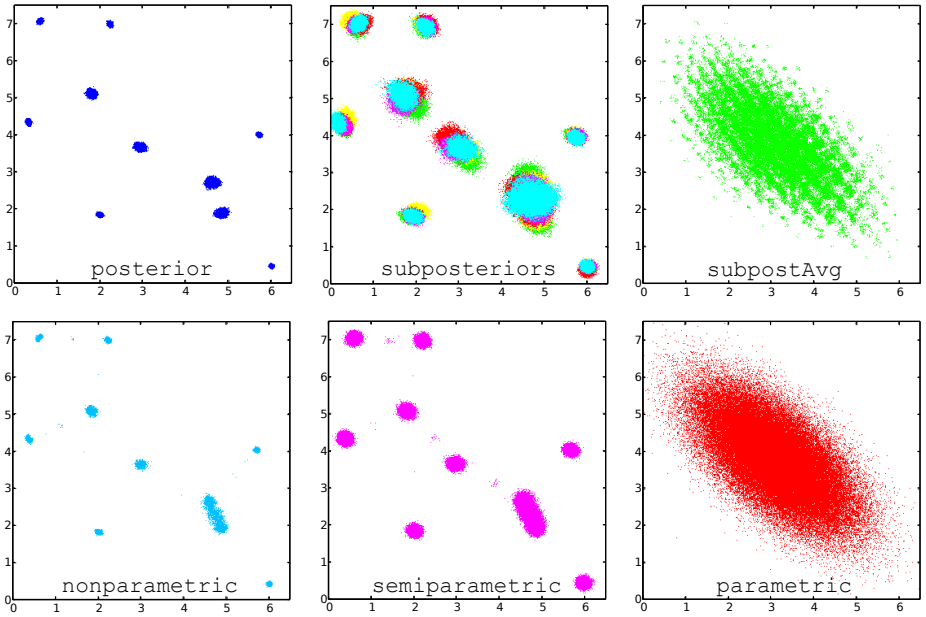


Figure 17: Gaussian mixture model posterior samples. We show 100,000 samples from a single 2-d marginal (corresponding to the posterior over a single mean parameter) of the full-data posterior (top left), all subposteriors (top middle—each one is given a unique color), the subposterior average via the `subpostAvg` procedure (top right), and the subposterior density product via the `nonparametric` procedure (bottom left), `semiparametric` procedure (bottom middle), and `parametric` procedure (bottom right).

we find after sampling that the posterior distribution over each component mean has ten modes. To sample from this multimodal posterior, we used the Metropolis-Hastings algorithm, where the component labels were permuted before each step (note that this permutation results in a move between two points in the posterior distribution with equal probability).

In Figure 17 we show results for M=10 splits, showing samples from the true posterior, overlaid samples from all five subposteriors, results from averaging the subposterior samples, and the results after applying our three subposterior combination procedures. This figure shows the 2-d marginal of the posterior corresponding to the posterior over a single mean component. The `subpostAvg` and `parametric` procedures both give biased results, and cannot capture the multimodality of the posterior. We show the posterior error vs time in Figure 18 (left), and see that our asymptotically exact methods yield quick convergence to low posterior error.

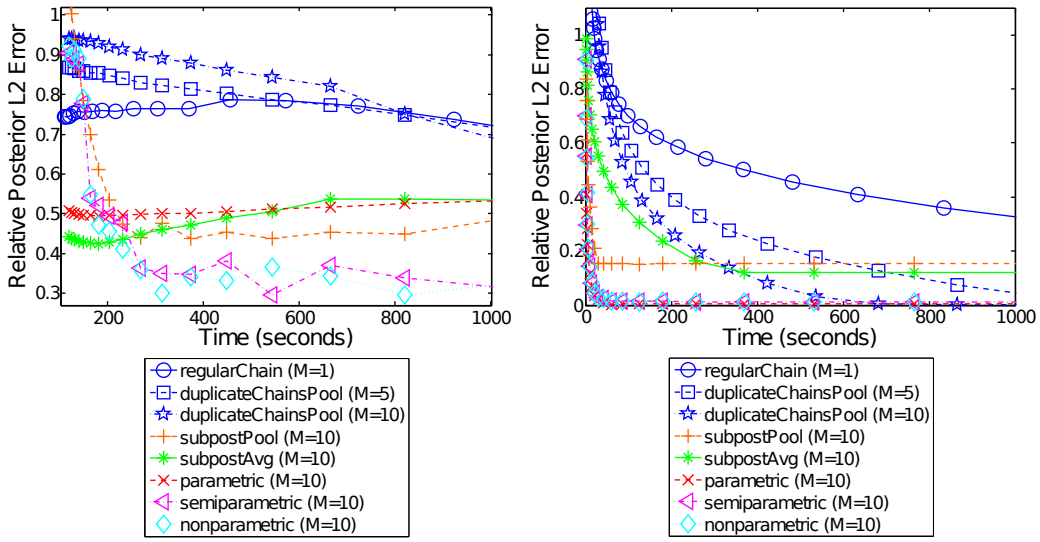


Figure 18: Left: Gaussian mixture model posterior error vs time results. Right: Poisson-gamma hierarchical model posterior error vs time results.

4.9.3 *Hierarchical Models*

We show results on a hierarchical Poisson-gamma model of the following form

$$\begin{aligned} a &\sim \text{Exponential}(\lambda) \\ b &\sim \text{Gamma}(\alpha, \beta) \\ q_i &\sim \text{Gamma}(a, b) \text{ for } i = 1, \ldots, N \\ x_i &\sim \text{Poisson}(q_i t_i) \text{ for } i = 1, \ldots, N \end{aligned}$$

for N=50,000 observations. We draw $\{x_i\}_{i=1}^N$ from the above generative process (after fixing values for a, b, λ, and $\{t_i\}_{i=1}^N$), and use M=10 splits. We again perform MCMC using the Stan software package.

In Figure 18 (right) we show the posterior error vs time, and see that our combination methods complete burn-in and converge to a low posterior error very quickly relative to the `subpostAvg` and `subpostPool` procedures and full-data chains.

4.10 FUTURE WORK ON EMBARRASSINGLY PARALLEL INFERENCE

4.10.1 *The Unknown M Case*

Embarrassingly parallel inference methods do not, in general, infer a correct posterior on each machine (given the subset of data on each machine); they instead compute a quantity, the subposterior, that depends on the total number of machines M. Further, the full collection of M machines must be combined, while any arbitrary subset of machines typically cannot be combined to construct any meaningful result.

More concretely, in the future, we would like to mitigate the following issues of embarrassingly parallel inference:

1. We must specify M in advance of parallel inference.
2. We cannot include additional groups of data after inference using other data is underway (or complete) without starting over.
3. In some cases, using an M that is too large (i.e. splitting the data too finely) can cause difficulties for inference algorithms in practice, as the prior becomes "too underweighted".
4. We do not begin with a meaningful uncertainty estimate for the subset of data on a given machine, which might be of independent interest.
5. We cannot combine an arbitrary subset of inference results, and must combine exactly all M inference results.
6. There are few guarantees about the existence or uniqueness of underweighted prior distributions and their resulting subposterior distributions.

The main issue behind the above issues lies in the fact that each subposterior is defined with respect to M. Namely, we write the subposterior as

$$p_m(\theta) \propto p(\theta)^{\frac{1}{M}} p(x^{n_m}|\theta).$$

Running inference algorithms during the parallel MCMC or VI step therefore requires M to be specified in advance. Ideally, we would instead perform parallel inference on, and then run combination procedures using, the *actual* posterior given a subset of data (without the underweighted prior), i.e.

$$p(\theta|x^{n_m}) \propto p(\theta)p(x^{n_m}|\theta).$$

In the future, we aim to develop methods that carry out the following two things: (1) every local machine performs a meaningful local inference, which gives the correct posterior given the local set of data. (2) any subset of the machines can combine to yield a correct combined posterior given the pooled subset of data. We discuss methods that make progress toward this goal in Chapter 7.

4.11 DISCUSSION

In this chapter, we present an embarrassingly parallel MCMC algorithm and provide theoretical guarantees about the samples it yields. Experimental results demonstrate our method's potential to speed up burn-in and perform faster asymptotically correct sampling. Further, it can be used in settings where data are partitioned onto multiple machines that have little intercommunication—this is ideal for use in a MapReduce setting. Currently, our algorithm works primarily when the posterior samples are real, unconstrained values and we plan to extend our algorithm to more general settings in future work.

5

LOW-COMMUNICATION DISTRIBUTED VARIATIONAL INFERENCE

5.1 CHAPTER SUMMARY

In this chapter, we develop parallel variational inference (VI) procedures for use in data-distributed settings, where each machine only has access to a subset of data and runs VI independently, with limited communication to other machines. However, in many cases it is not possible to directly extend this procedure to VI methods without requiring certain restrictive exponential family conditions on the form of the model. Furthermore, most existing (nonparallel) VI methods are restricted to use on conditionally conjugate models, which limits their applicability. To combat these issues, we propose two methods. The first makes use of the recently proposed nonparametric VI to facilitate an embarrassingly parallel VI procedure that can be applied to a wider scope of models, including to nonconjugate models. For the second procedure, we describe how similar parallelization methods can be applied to black box variational inference, which uses sampling to perform gradient updates. We derive our low-communication VI algorithms, analyze our methods theoretically, and demonstrate our methods empirically on a few nonconjugate models.

5.2 INTRODUCTION

Many large, modern datasets are collected and stored in a distributed fashion by multiple sensors or data-collecting agents. Examples of this include medical data recorded in hospitals throughout a country, weather data gathered by a collection of sensors, web data scraped by a network of machines, and cell phone data collected on users' phones. Inference algorithms that can operate in these distributed settings—by processing subsets of data separately and in parallel—are particularly advantageous. This is because they mitigate the need for transfering data to a central location for analysis, reduce both the memory usage and computation time of inference [115, 142], allow for continuous data collection from independently operating agents [35], and allow for sensitive data to be processed independently in secure locations (which can yield privacy guarantees [149]).

Variational inference (VI) methods are general procedures for approximate inference in Bayesian models, and they have been applied successfully in a wide variety of domains [16, 95]. This chapter is concerned with developing better VI methods for use in distributed settings. One major issue with most existing parallel methods is that they often require synchronization between machines at regular intervals

[132, 207, 210]. Communication between machines due to this synchronization can greatly reduce the efficiency of these procedures, as each machine must wait for information from other machines before proceeding with computation. Furthermore, communication requirements may increase the difficulty of system implementation and maintenance [115], and necessitate the transfer of (potentially sensitive) data between machines [200].

We aim to develop a new "embarrassingly parallel" algorithm for VI in data-distributed settings, which is a type of parallel algorithm where there is no regular communication between machines. Given a dataset partitioned over a collection of machines, embarrassingly parallel VI methods carry out the following two steps:

1. Perform variational inference on the subset of data on each machine in parallel (independently, without communication between machines).
2. Combine the results from all machines to yield a variational inference result for the full-data posterior distribution.

These two steps are only performed once, and there is only communication between machines at one point in the second step, when collecting results from each of the local instances of VI.

Recently, progress has been made toward this goal for mean field variational inference methods limited to models with certain exponential family restrictions on the likelihood and prior distribution [33, 35]. These methods use a decomposition of the posterior that takes advantage of closedness properties of exponential family densities under products and quotients. However, these modeling assumptions are fairly restrictive, and this decomposition cannot be applied to many popular models (including logistic regression, correlated topic models, and nonlinear matrix factorization models). Additionally, this approximation family is typically inadequate to capture multimodal densities [69].

A separate line of work has aimed to develop "nonconjugate" variational inference methods for models without tractable exponential family conditional distributions [69, 192]. Similar to these methods, we would like a general inference algorithm that can be applied to a wide class of Bayesian models, yet operates in this embarrassingly parallel setting. However, the variational families employed by these nonconjugate methods are not in a form that allows us to apply the above-mentioned decomposition strategy for parallelization.

Recent papers in the Markov chain Monte Carlo (MCMC) literature have introduced an alternative decomposition of the posterior for parallel inference [142, 164, 193], which involves the product of so called *subposterior densities* (i.e. posterior densities given a subset of data with an underweighted prior). We apply this new decomposition to a nonconjugate variational inference method called nonparametric variational inference (NVI) [69] to perform low-communication, parallel inference in a general class of models. In particular, we only require weak differentiability conditions on the joint log probability.

The main contribution of our method is that it provides a way to perform embarrassingly parallel inference in data distributed settings for a more-general class of Bayesian models without requiring conditional conjugacy or exponential family assumptions on the model likelihood or prior. In the following sections, we derive the posterior decomposition used by our method, show how we can combine local nonparametric variational approximations to form a variational approximation to the full-data posterior density, and analyze the computational complexity of our algorithms. Finally, we demonstrate our method empirically on a few nonconjugate Bayesian models.

5.3 PRELIMINARIES

We describe existing work on embarrassingly parallel VI with exponential family restrictions, existing work on embarrassingly parallel MCMC, and the difficulties with extending these methods to variational inference in more general, nonconjugate models.

Suppose we have a large set of N i.i.d. data points, $x^N = \{x_1, \ldots, x_N\}$, a likelihood for these data parameterized by $\theta \in \mathbb{R}^d$, written $p(x^N|\theta)$, and a prior density for θ, written $p(\theta)$. We can write the posterior density given all N data points (which we will also refer to as the "full-data" posterior) as

$$p(\theta|x^N) \propto p(\theta)p(x^N|\theta) = p(\theta)\prod_{i=1}^{N} p(x_i|\theta). \tag{38}$$

Now suppose the data x^N is partitioned into M subsets $\{x^{n_1}, \ldots, x^{n_M}\}$ of sizes $n_1, \ldots, n_M$, and distributed over M machines. Recent works in embarrassingly parallel VI [33, 35] have proposed the following solution for inference in this setting. First, in an embarrassingly parallel fashion, compute

$$q_1^*, \ldots, q_M^* = \underset{q_1,\ldots,q_M}{\arg\min} \sum_{m=1}^{M} \mathrm{KL}\left[q_m(\theta)||p(\theta|x^{n_m})\right] \tag{39}$$

where $p(\theta|x^{n_m})$ is the posterior given a subset of data x^{n_m}. Second, form the full-data posterior variational approximation with

$$q^*(\theta) \propto \left(\prod_{m=1}^{M} q_m^*(\theta)\right) / p(\theta)^{M-1}. \tag{40}$$

The justification for this solution is that the full-data posterior can be decomposed as $p(\theta|x^N) \propto \left(\prod_{m=1}^{M} p(\theta|x^{n_m})\right) / p(\theta)^{M-1}$, and further, it can be shown that the above objective retains an important property of the classic (nonparallel) KL objective: if the objective is zero then the full-data approximation $q^*(\theta)$ is equal to the full-data

posterior $p(\theta|x^N)$. I.e., if $\sum_{m=1}^{M} \mathrm{KL}[q_m(\theta)||p(\theta|x^{n_m})] = 0 \implies \mathrm{KL}[q^*(\theta)||p(\theta|x^N)] = 0 \implies q^*(\theta) = p(\theta|x^N)$.

However, this solution has a few major restrictions on the form of the variational approximation and model. Namely, to form q^*, these methods must tractably compute the product of the M variational approximations divided by the prior density (equation (40)). These methods do this by limiting their scope to conditionally conjugate exponential family models, and then using mean field variational methods that restrict the variational approximation to the same exponential family as the prior.

To attempt to extend the scope of models to which embarrassingly parallel VI methods can be applied, we turn to a separate line of work on embarrassingly parallel MCMC methods [142, 164, 193]), which use an alternative decomposition of the posterior distribution. Let the m^{th} *subposterior* density, $p_m(\theta)$, be defined as the posterior given the m^{th} data subset with an underweighted prior, written $p_m(\theta) = p(\theta)^{\frac{1}{M}} p(x^{n_m}|\theta)$. This is defined such that the product of the M subposterior densities is proportional to the full-data posterior, i.e.

$$p_1 \cdots p_M(\theta) \propto p(\theta) \prod_{m=1}^{M} p(x^{n_m}|\theta) \propto p(\theta|x^N). \tag{41}$$

In these methods, a subposterior density estimate $\widehat{p}_m(\theta)$ is learned on each machine (via sampling), and the product of these estimates $\prod_{m=1}^{M} \widehat{p_m}(\theta)$ yields an approximation of the full-data posterior density.

However, we cannot directly apply this new decomposition to typical mean field variational inference approximations (as is done in embarrassingly parallel VI) for the following two reasons:

1. The underweighted prior $p(\theta)^{\frac{1}{M}}$ in the subposterior density may lose conjugacy necessary for the requisite exponential-family-conditionals. Hence, it may not be easy to directly apply these VI methods to approximate the subposterior.
2. Even if we are able to learn a variational approximation for each subposterior, the product of subposterior variational approximations may not have a tractable form that we can analytically compute.

Therefore, to use this alternative decomposition to apply VI to a broader scope of models, we need a family of variational approximations that can be run on general subposterior densities (including those of nonconjugate models) while maintaining a tractable density product that can be analytically computed.

5.4 EMBARRASSINGLY PARALLEL VARIATIONAL INFERENCE IN NONCONJUGATE MODELS

Embarrassingly parallel variational inference (EPVI) in nonconjugate models is a parallel approximate Bayesian inference method for continuous posterior distributions.

It is generally applicable, requiring only that the first two derivatives of the log-joint probability density are computable. For a dataset partitioned over M machines, VI is run in parallel on each machine to approximate the M subposterior densities; afterwards, the local subposterior approximations are combined by computing their product, which approximates the full-data posterior density. Each machine performs variational inference without sharing information, in an embarrassingly parallel manner. We summarize this procedure in Algorithm 6.

Algorithmus 6 : Embarrassingly Parallel Variational Inference in Nonconjugate Models from Neiswanger et al. [143]

Input : Partitioned dataset $\{x^{n_1}, \ldots, x^{n_M}\}$.

Output : Variational approximation $q^*(\theta)$ for the full-data posterior density $p(\theta|x^N)$.

1 **for** $m = 1, \ldots, M$ **do in parallel**

2 Learn a variational approximation $q_m^*(\theta)$ for the m^{th} subposterior $p_m(\theta)$, given data x^{n_m}.

3 Compute product $\prod_{m=1}^{M} q_m^*(\theta)$ of subposterior approximations to yield the full-data variational approximation $q^*(\theta)$.

We illustrate our EPVI procedure for a Bayesian logistic regression model on a toy dataset in Figure 19.

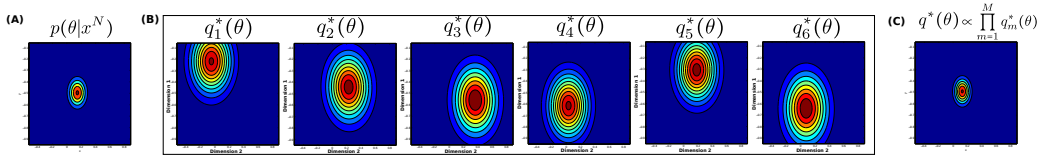


Figure 19: Illustration of our embarrassingly parallel VI method, shown for a Bayesian logistic regression model on a toy dataset. In (a) we show the first two dimensions of the full-data posterior density. In (b) we show the first two dimensions of each of the $M = 6$ subposterior variational approximations after running VI on each subset of data independently. In (c) we show the first two dimensions of the combined product density (formed using the six subposterior variational approximations), which recovers the posterior shown in (a).

5.4.1 *EPVI with Nonparametric Variational Inference*

In a recently proposed method known as nonparametric variational inference (NVI) [69], a posterior approximation is selected from a variational family of densities of the form $q(\theta) = \frac{1}{K} \sum_{k=1}^{K} \mathcal{N}_d(\theta|\mu_k, \sigma_k^2 I_d)$. Some advantages of this method are that it can capture multimodal posterior distributions, can be applied to many nonconjugate models (in fact, the only requirement is that the first two derivatives of the

log joint probability are computable), and has an efficient algorithm to optimize the variational objective. In our case, NVI allows us to perform variational inference on subposterior densities without worrying if they retain the conjugacy necessary to easily apply typical mean-field approximations, and also allows us to develop a method to combine the subposterior variational approximations (i.e. allows us to derive an analytic form for the product of these approximations) to produce a full-data posterior variational approximation. After running this procedure on a subset of data x^{n_m} on a machine m, we can write the inferred variational approximation for the subposterior distribution as

$$q_m^*(\theta) = \frac{1}{K} \sum_{k=1}^{K} \mathcal{N}_d(\theta | \mu_k^{(m)}, \sigma_k^{2^{(m)}} I_d). \tag{42}$$

Due to this choice of q_m^*, we have an analytic form for the product of these densities, $\prod_{m=1}^{M} q_m^*(\theta)$, which gives us a variational approximation for the subposterior density product (and hence for the full-data posterior). In particular, the product of these M mixture-of-Gaussians variational densities gives a (non-uniformly weighted) mixture-of-Gaussians density with K^M components. We can write this product mixture as

$$\begin{aligned} q^*(\theta) \propto \prod_{m=1}^{M} q_m^*(\theta) &= \frac{1}{K^M} \prod_{m=1}^{M} \sum_{k_m=1}^{K} \mathcal{N}_d\left(\theta | \mu_{k_m}^{(m)}, \sigma_{k_m}^{2^{(m)}} I_d\right) \\ &= \sum_{k_1=1}^{K} \cdots \sum_{k_M=1}^{K} w_{k\cdot} \mathcal{N}_d(\theta | \mu_{k\cdot}, \sigma_{k\cdot}^2 I_d) \end{aligned} \tag{43}$$

where we use $k\cdot = (k_1, \ldots, k_M)$ to denote the vector of M subposterior-component-indices (one from each subposterior variational approximation mixture) associated with a given component in this product mixture, and where

$$\sigma_{k\cdot}^2 = \left(\sum_{m=1}^{M} \left(\sigma_{k_m}^{2^{(m)}} \right)^{-1} \right)^{-1} \tag{44}$$

$$\mu_{k\cdot} = \sigma_{k\cdot}^2 I_d \left(\sum_{m=1}^{M} \left(\left(\sigma_{k_m}^{2^{(m)}} \right)^{-1} I_d \right) \mu_{k_m}^{(m)} \right) \tag{45}$$

$$w_{k\cdot} = \frac{\prod_{m=1}^{M} \mathcal{N}_d\left(\mu_{k_m}^{(m)} | \mu_{k\cdot}, \sigma_{k_m}^{2^{(m)}} I_d\right)}{\mathcal{N}_d\left(\mu_{k\cdot} | \mu_{k\cdot}, \sigma_{k\cdot}^2\right)} \tag{46}$$

5.4.2 *Computing the Variational Density Product Mixture*

After learning the optimal local parameters $\{\mu_{k_m}^{(m)}, \sigma_{k_m}^{2^{(m)}}\}_{k_m=1}^{K}$ for each of the $m \in \{1, \ldots, M\}$ subposteriors, we wish to form a variational approximation to the full-data posterior density by taking the product of the M mixtures. However, computing

the parameters and weights for all K^M components in the product mixture becomes infeasible as M grows.

We typically perform Bayesian inference in order to compute expectations with respect to, and explore, the posterior distribution. In practice, one common way to achieve this is to sample from the posterior, and then compute a sample expectation; this is done in both MCMC methods and in VI methods (in the latter case, to compute expectations with respect to a variational approximation after VI has finished runnning [27, 69, 153]). Hence, instead of computing the product mixture (and afterwards, sampling from it to compute expectations), our solution is to bypass this step and directly generate samples from the product mixture. We give a procedure that allows us to compute expectations with respect to the variational approximation in this common sampling manner without requiring us to actually compute the variational approximation.

We give our method for sampling from the full-data variational approximation in Algorithm 7, and then prove that this yields correct samples. The intuitive idea behind our algorithm is the following. To sample from a mixture, one can first sample a component index (proportional to the component weights) and then sample from the chosen mixture component. We therefore need a way to sample product mixture components (proportional to their weights) without first computing all of the K^M component weights. Our solution is to form a Markov chain over the product mixture component indices, and prove that its stationary distribution is a categorical distribution with probability mass values proportional to the product mixture component weights. Hence, at each step in this Markov chain, we can produce a sample from the full variational approximation while only needing to compute a single new product mixture component.

Note that in Algorithm 7, at each step in the Markov chain, we perform two simple steps to sample the next product mixture component: we select a subposterior uniformly at random (line 3), and then re-draw one of its K components uniformly at random (line 5); this specifies a new product mixture component. We then compute the ratio of the weight of this new product mixture component with the previous component's weight (line 7) and accept or reject this proposal (line 8). We then compute the parameters of the sampled component (lines 10-11).

Correctness of Algorithm 7. We prove that Algorithm 7 defines a Markov chain whose stationary distribution is the distribution over the K^M components in the product mixture density.

Theorem 5.4.1. *The procedure given by Algorithm 7 defines a Markov chain whose stationary distribution is the categorical distribution (over K^M categories) with category-probability parameter equal to the vector of product mixture component weights.*

Proof. Note that each of the K^M product mixture components is associated with an M-dimensional vector $k_\cdot = (k_1, \ldots, k_M) \in \{1, \ldots, K\}^M$ (where k_m denotes the index of the m^{th} subposterior's component that contributed a factor to this product mixture component). Hence, instead of sampling an index from a categorical

Algorithmus 7 : Markov chain for sampling variational density product mixture components from Neiswanger et al. [143]

Input : Number of samples R, number of burn-in steps b, learned subposterior variational approximations $\{q_1^*(\theta), \ldots, q_M^*(\theta)\}$.

Output : Parameters $\{\mu_r, \sigma_r^2\}_{r=1}^R$ for the R sampled product mixture components.

1 Draw $k_\cdot = (k_1, \ldots, k_M) \overset{\text{iid}}{\sim} \text{Unif}(\{1, \ldots, K\})$; `/* Initialize Markov chain */`
2 **for** $s = 1, \ldots, b + R$ **do**
3 Draw $m \sim \text{Unif}(\{1, \ldots, M\})$
4 Set $c_\cdot = (c_1, \ldots, c_M) \leftarrow k_\cdot$
5 Draw $c_m \sim \text{Unif}(\{1, \ldots, K\})$
6 Draw $u \sim \text{Unif}([0, 1])$;
7 **if** $u < w_{c_\cdot} / w_{k_\cdot}$ **then**
8 Set $k_\cdot \leftarrow c_\cdot$
9 **if** $s > b$ **then**
10 Set $\mu_{s-b} \leftarrow \mu_{k_\cdot}$; `/* Compute mean of sampled mixture component */`
11 Set $\sigma^2_{s-b} \leftarrow \sigma^2_{k_\cdot}$; `/* Compute var of sampled mixture component */`

distribution, we can equivalently view our task as sampling an M-dimensional vector from a joint distribution over the space $\{1, \ldots, K\}^M$, where each element in this space has a probability mass proportional to its associated product mixture component weight. We can therefore perform Gibbs sampling over this space, where we sample from the conditional distribution over a subposterior component index k_m given all other component indices. To compute and then sample from this conditional distribution, we could iterate over the K possible values of k_m (and compute the component weight of each); however, this could potentially be expensive for large K. Instead, we sample one of the K values for k_m uniformly at random (line 5), and our algorithm becomes a Metropolis-within-Gibbs algorithm [73] where we've used an independent Metropolis proposal [12, 74] (which we achieve by accepting or rejecting, in lines 6-8, the independent Metropolis proposal made in line 5). Note that the dimension m along which we take a Gibbs sampling step is chosen in line 3. Since this Metropolis-within-Gibbs algorithm has been shown to have the correct stationary distribution [73], our proof is complete. □

We describe the complexity of Algorithm 7 in Section 5.4.3. In Section 5.7, we verify that this algorithm achieves the same results as taking expectations after computing the mixture product exactly, while drastically speeding-up performance.

SEQUENTIAL SUBPOSTERIOR SUBSET PRODUCTS. In some cases, it may be simpler to sample from the product mixture in a sequential fashion by sampling from

the product of only a few subposteriors multiple times: we first sample R components from the product of groups of $\tilde{M} < M$ approximations, and then repeat this process on the resulting (uniform) mixtures formed by the sampled components. This continues until samples from only one mixture remain. For example, one could begin by sampling components from the product of all $\frac{M}{2}$ pairs (leaving one subposterior approximation alone if M is odd), thereby forming $\frac{M}{2}$ uniformly weighted mixtures comprised of the sampled components. This process is then repeated—forming pairs and sampling from the pair product mixture—until there are only samples from one product mixture remaining (which are approximate samples from the full-data posterior). This method is potentially advantageous because each intermediate round of product mixture sampling could be done in parallel. However, more samples are potentially required from each intermediate round to generate valid samples from the full variational approximation at the final product. We compare the effectiveness of this method in Section 5.7.

5.4.3 *Method Complexity*

Consider a dataset with N observations, partitioned over M machines. Assume we have approximated each subposterior using NVI with K components, where each component is defined by a d-dimensional parameter. Computing all components of the product mixture exactly requires $O(dMK^M)$ operations. Computing R samples from the product mixture approximation via Algorithm 7 requires $O(dRM)$ operations (assuming a constant number b of burn-in steps). Computing sequential subposterior subset product samples with R samples at each intermediate product requires $O(dRM^2)$ operations overall, but this could be reduced to $O(dR \log M)$ operations on a single machine if each of the $O(\log M)$ rounds of sampling are done in parallel.

Each machine learns and then communicates the optimal variational parameters, which consist of K mean parameter vectors (each in d dimensions), K variance parameter scalars, and K weight parameter scalars. In total, $MK(d+2)$ scalars are communicated throughout the entire procedure.

5.5 LOW COMMUNICATION DISTRIBUTED BLACK BOX VI

Recent methods, known as black box VI methods, aim to allow for VI optimization updates that require minimal analytic derivations, with the goal of providing more automated methods of inference with less work for practitioners. In these black box VI methods, optimization is performed via stochastic gradient descent, where stochastic gradients are computed via Monte Carlo estimates of expectations in the evidence lower bound, using samples drawn from the variational approximation.

We hope to combine our embarrassingly parallel inference methods with black box VI. Recall that methods from embarrassingly parallel inference allow us to draw

samples from the density proportional to the product of subposteriors; this is the combination procedure. We aim to use black box VI together with embarrassingly parallel inference in two ways. First, we can use black box VI to perform variational inference on each subposterior (in the exact same setup as our embarrassingly parallel variational inference procedure). Second, we can incorporate the combination sampling procedure into a black box VI method, which takes the subposterior variational approximations and uses the combination procedure to perform further stochastic gradient updates, in order to refine the combined posterior approximation and return a more accurate variational approximation to the full posterior.

5.5.1 *Background and Motivation*

Black box variational inference (BBVI) is a popular inference method used in probabilistic programming frameworks, since it allows for efficient approximate Bayesian inference with minimal human derivations. We consider the setting where we want to perform inference given a large dataset distributed over multiple machines, in a communication-efficient manner. We provide BBVI methods for this setting, which require very low communication between machines. We first give an embarrassingly parallel BBVI algorithm that yields an approximate solution, and then show how we can perform a post-inference procedure to refine this solution. We also release a probabilistic programming implementation.

There are many cases where a large group of computers are used to collect, store, or process data in a distributed setting. Some examples include weather data gathered by a collection of sensors, web data scraped by a network of computers, and medical data recorded in hospitals throughout a country (which may be private and cannot be moved or pooled).

The goal of this work is to develop automatic Bayesian inference techniques—specifically, methods for black box variational inference—for use in probabilistic programming frameworks, which are effective in this multi-machine/data-distributed setting, with minimal communication needed between machines.

5.5.2 *Preliminaries*

Consider data $\{x_i\}_{i=1}^N$, where $x_i \in \mathbb{R}^p$; a model for the data with probability density function (PDF) $p(x^N|\theta) = \prod_{i=1}^N p(x_i|\theta)$, parameterized by $\theta \in \mathbb{R}^d$; and a prior distribution over θ, with PDF $p(\theta)$. The model and prior define a joint density, written $p(\theta, x^N) = p(\theta)p(x^N|\theta)$. In Bayesian inference, we are interested in the posterior distribution, a conditional of $p(\theta, x^N)$, with PDF $p(\theta|x^N) = p(\theta)p(x^N|\theta)/\int p(\theta)p(x^N|\theta)d\theta$.

BLACK BOX VARIATIONAL INFERENCE. In many cases it is not possible to compute $p(\theta|x^N)$ exactly, and one must resort to approximate inference methods. Varia-

tional inference (VI) is a popular method for computing an approximation to $p(\theta|x^N)$, where one chooses a family of densities $\{q(\theta;\lambda) : \lambda \in \mathcal{L}\}$, and then computes

$$\lambda^* = \arg\min_{\lambda\in\mathcal{L}} \mathrm{KL}\left[q(\theta;\lambda)||p(\theta|x^N)\right], \tag{47}$$

where KL denotes the Kullback-Leibler divergence [44]. Afterwards, the variational approximation $q(\theta;\lambda^*)$ is used as a surrogate for the posterior PDF. To compute λ^*, most VI methods maximize an objective called the evidence lower bound (ELBO), defined as

$$\mathrm{ELBO}(\lambda) = \mathbb{E}_{q(\theta;\lambda)}\left[\log p(\theta, x^N) - \log q(\theta;\lambda)\right]. \tag{48}$$

Black box variational inference (BBVI) aims to run VI in an automatic manner with few model-specific derivations. In BBVI, the ELBO is typically optimized with stochastic gradient descent (SGD), where a Monte Carlo step, which draws samples from the variational approximation, is used to compute stochastic gradients.

DATA-DISTRIBUTED SETTING. Suppose that x^N is partitioned over M machines, i.e. $x^N = \{x^{n_1}, \ldots, x^{n_M}\}$, where x^{n_m} contains n_m data points. Similar to previous work in data-distributed inference [142, 146, 194], we will make use of the *subposterior*, which is a posterior given only a subset of the data, with an underweighted prior. We define the m^{th} subposterior density to be

$$p_m(\theta|x^{n_m}) \propto p(\theta)^{\frac{1}{M}} p(x^{n_m}|\theta). \tag{49}$$

Note that the product of the M subposterior densities is proportional to the full posterior density, i.e. $\prod_{m=1}^{M} p_m(\theta|x^{n_m}) \propto p(\theta|x^N)$. We similarly define the m^{th} local joint density as $p_m(\theta, x^{n_m}) \propto p(\theta)^{\frac{1}{M}} p(x^{n_m}|\theta)$. Strategies for parallel inference in prior work [142, 146, 194] have involved sampling from the subposterior density on each machine, and then using these sets of samples as inputs to a "combination" algorithm that generates samples from the subposterior product.

5.5.3 *Method Overview*

We first present a solution, which is embarrassingly parallel—i.e. each machine can run BBVI independently on a subset of data and afterwards do a single round of communication to yield a solution for the full data. However, we describe how this strategy only approximately solves the VI objective, and derive an algorithm that fixes this issue and optimizes the exact objective.

AN APPROXIMATE EMBARRASSINGLY PARALLEL SOLUTION. Consider the following procedure:

1. On each machine m, in parallel and without communication, use BBVI to compute

$$\begin{aligned}\lambda_m^* &= \arg\max_{\lambda_m} \text{ELBO}_m(\lambda_m) \\ &= \arg\max_{\lambda_m} \mathbb{E}_{q(\theta;\lambda_m)}\left[\log \frac{p_m(\theta, x^{n_m})}{q(\theta;\lambda_m)}\right], \end{aligned} \tag{50}$$

where $\text{ELBO}_m(\lambda_m)$ denotes the ELBO on the m^{th} machine with the m^{th} subset of data. This gives a variational approximation to each subposterior, i.e. $q(\theta;\lambda_m^*) \approx p_m(\theta|x^{n_m})$.

2. Draw L samples from each of the variational approximations, i.e.

$$\{\theta_\ell^m\}_{\ell=1}^L \sim q(\theta;\lambda_m^*), \text{ for } m = 1,\ldots,M. \tag{51}$$

This can be done via the same sampling process used in BBVI (step 1).

3. Use combination methods from embarrassingly parallel MCMC to draw S samples from the product of the M variational approximations, i.e.

$$\begin{aligned}\{\theta_s^*\}_{s=1}^S &= \text{Combine}\left(\{\theta_\ell^1\}_{\ell=1}^L, \ldots, \{\theta_\ell^M\}_{\ell=1}^L\right) \\ &\sim \tilde{q}(\theta;\lambda_{1:M}^*) \propto \prod_{m=1}^M q_m(\theta;\lambda_m^*) \approx p(\theta|x^N). \end{aligned} \tag{52}$$

ISSUES WITH THE ABOVE METHOD. In practice, the above procedure yields samples that appear to approximate the full data posterior $p(\theta|x^N)$. However, it is unclear if the above method is minimizing the KL divergence beween some variational family and the full data posterior (i.e. performs valid VI) and if it will always work. To remedy this, we present a post-inference method, which is run after the above is complete, that improves the inference result and minimizes a correct quantity—the KL divergence between an explicitly defined variational family of densities and $p(\theta|x^N)$. We first define the variational family and then describe the post-inference method.

DEFINITION: *density-product variational family*. Let $\mathcal{Q} = \{q(\theta;\lambda) : \lambda \in \mathcal{L}\}$ be an arbitrary parametric family of densities with parameter $\lambda \in \mathcal{L}$. Let $\Pi_{\mathcal{Q}}$ be a family of densities with parameter tuple $\Lambda = (\lambda_1,\ldots,\lambda_M) \in \mathcal{L}^M = \mathcal{L} \times \ldots \times \mathcal{L}$, consisting of normalized products of M densities from $\mathcal{Q}$, i.e.

$$\Pi_{\mathcal{Q}} = \left\{ q(\theta;\Lambda) \propto \prod_{m=1}^M q_m(\theta;\lambda_m) \ : \ q_m(\theta;\lambda_m) \in \mathcal{Q} m = 1\ldots,M \right\}. \tag{53}$$

We call $\Pi_{\mathcal{Q}}$ a *density product family (DPF) on* $\mathcal{Q}$. Note that in some cases (e.g. for exponential families) $\mathcal{Q}$ is closed under products, and therefore $q(\theta;\Lambda) \in \Pi_{\mathcal{Q}} \implies q(\theta;\Lambda) \in \mathcal{Q}$. For a given $q(\theta;\Lambda)$, let $Z(\Lambda)$ denote its normalizing constant, i.e. $q(\theta;\Lambda) = \frac{1}{Z(\Lambda)} \prod_{m=1}^M q_m(\theta;\lambda_m)$.

OPTIMIZING THE ELBO FOR THE DPF. The goal of VI with a DPF is to find parameters $\Lambda^* = (\lambda_1^*, \ldots, \lambda_M^*)$ that minimize the KL divergence between $q(\theta; \Lambda)$ and $p(\theta|x^N)$, which is typically done by maximizing the ELBO objective. We can write this objective for the variational DPF as

$$\begin{aligned} \Lambda^* = (\lambda_1^*, \ldots, \lambda_M^*) &= \arg\max_{\Lambda \in \mathcal{L}^M} \text{ELBO}(\Lambda) \\ &= \arg\max_{\Lambda \in \mathcal{L}^M} \sum_{m=1}^{M} \mathbb{E}_{q(\theta;\Lambda)} \left[\log \frac{p_m(\theta, x^{n_m})}{q_m(\theta; \lambda_m)} \right] + \log Z(\Lambda). \end{aligned} \tag{54}$$

We have derived black box methods for computing the gradient of (54), where we denote the j^{th} component of this gradient as $\nabla_\Lambda \text{ELBO}(\Lambda)_j$.

POST-INFERENCE METHOD FOR VALID VI SOLUTION. We can then perform the following steps, after the initial round of embarrassingly parallel BBVI, to improve the solution and ensure that we are minimizing the exact KL objective.

1. On each machine j, in parallel and without communication between machines, use BBVI to compute:

$$\lambda_j^* = \arg\max_{\lambda_j} \text{ELBO}(\Lambda)_j. \tag{55}$$

   where we perform optimization using black-box gradients $\nabla_\Lambda \text{ELBO}(\Lambda)_j$.

2. Communicate updated $\{\lambda_j^*\}_{j=1}^M$ to all machines.

3. Repeat until convergence.

Afterwards, we can apply steps 2 and 3 from the embarrassingly parallel procedure given initially to generate samples from the final variational approximation. We find that only a small number of communication rounds are needed for the algorithm to converge on a final result.

5.6 METHOD SCOPE

The algorithms described in this chapter hold for posteriors distributions with twice-differentiable densities in finite-dimensional real spaces. This method may be applied to nonconjugate Bayesian models, and models with multimodal posteriors, with little further restriction on the form of the model and prior distribution.

However, there are certain model types for which our method is not appropriate. These include discrete models, continuous posterior distributions over the simplex, and infinite dimensional models. Furthermore, our method may not be well suited to posteriors with high correlations or different scales between dimensions, and multimodal models suffering from label switching.

5.7 EMPIRICAL STUDY

We demonstrate empirically the ability of our method to significantly speed up VI on nonconjugate models in a distributed setting, while maintaining an accurate posterior approximation. In particular, our experiments aim to show that:

1. We can perform inference in a fraction of the time of nonparallel VI methods as we increase M.
2. We can achieve similar performance as nonparallel VI methods as we increase M.
3. Expectations computed via our product mixture sampling method (Algorithm 7) achieve similar performance as those computed via exact computation of the product mixture.

To demonstrate these, we conduct experimental comparisons with the following strategies:

- **Full-data nonparametric variational inference** (`NVI`)—A (nonparallel) variational inference method designed for use in nonconjugate models, which we run on the full dataset. This method takes as a parameter the number of mixture components K.
- **Subposterior inference on data subsets** (`Subposteriors`)—The subposterior variational approximations, run on subsets of data. This method takes as a parameter the number of mixture components K, and each run returns M of these approximations.
- **Embarrassingly parallel variational inference (exact product)** (`EPVI_exact`)—The method introduced in this chapter, which combines the M subposteriors by computing all components of the product mixture density.
- **Embarrassingly parallel variational inference (mixture product sampling)** (`EPVI_sample`)—The method introduced in this chapter (Algorithm 7), which samples from the product of the M subposterior approximations.
- **Embarrassingly parallel variational inference (sequential subset products)** (`EPVI_subset`)—The method introduced in this chapter, which samples from products of pairs of subposteriors sequentially.

We do not aim to compare the benefits of VI in general in this work, and therefore exclude comparisons against alternative approximate inference methods such as MCMC, expectation propagation, or other deterministic dynamics inference algorithms (such as herding or Bayesian quadrature). To assess the performance of our method, we compute the log-likelihood of held-out test data given our inferred

variational approximation (which we can compute in a consistent manner for many types of models).

Experiments were conducted with a standard cluster system. We obtained subposterior variational approximations by submitting batch jobs to each worker, since these jobs are all independent. We then saved the results to the disk of each worker and transfered them to the same machine, which performed the product mixture sampling algorithms. In each of the following experiments involving timing, we first ran the variational inference optimization procedures until convergence (to provide a time for the `Subposteriors` and `NVI` strategies). Afterwards, we added the (maximum) time required for learning the subposterior approximations, the time needed to transfer the learned parameters to a master machine, and the time required to run the product mixture sampling algorithms (to provide a time for the `EPVI` methods).

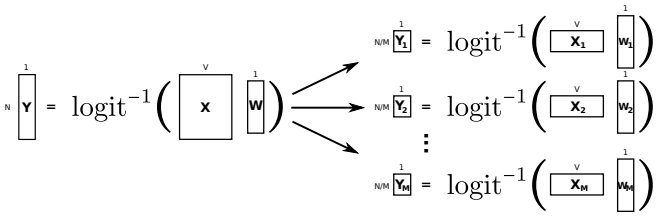


Figure 20: Diagram of the data partitioning scheme for the hierarchical logistic regression model.

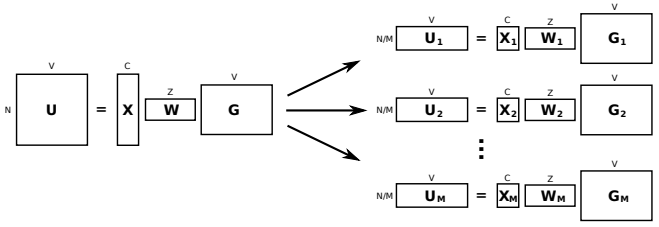


Figure 21: Diagram of the data partitioning scheme for the topographic latent source analysis model.

5.7.1 *Bayesian Generalized Linear Models*

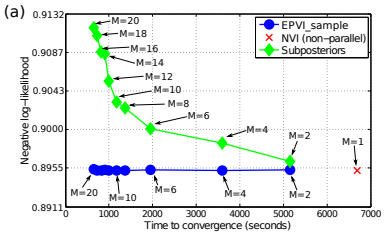
Generalized linear models are widely used for a variety of regression and classification problems. We use a hierarchical Bayesian logistic regression model as a test case in the following experiments. This model places a Gaussian prior on a set of coefficients $\mathbf{w} \in \mathbb{R}^V$ and draws class labels $\mathbf{y} \in \mathbb{R}^N$, conditioned on the product of an observation matrix $\mathbf{X} \in \mathbb{R}^{N \times V}$ and the coefficients, passed through a logistic transform; further, Gamma priors are placed on the variance parameter for each co-

efficient. Notably, this model lacks conditional conjugacy. We write the generative model as

1. Draw global hyperparameter $\alpha \sim \text{Gamma}(a, b)$
2. For $v = 1, \ldots, V$, draw coefficient $w_v \sim \mathcal{N}(0, \alpha^{-1})$
3. For $n = 1, \ldots, N$, draw observation $y_n \sim \text{Bernoulli}\left(\text{logit}^{-1}\left(-\mathbf{w}^\top \mathbf{x}_n\right)\right)$

where $\mathbf{x}_n$ denotes the n^{th} row of $\mathbf{X}$. We partition the data by splitting $\mathbf{X}$ and $\mathbf{y}$ into M disjoint subsets each of size $\frac{N}{M}$, and inferring a variational approximation on each subset. This is illustrated in Figure 20.

Data. We demonstrate our methods on the SUSY particles dataset[1], in which the task is to classify whether or not a given signal (measured by particle detectors in an accelerator) will produce a supersymmetric particle. This dataset has $N = 5,000,000$ observations, of which we hold out 10% for evaluating the test log-likelihood.



(c)

M	Classification Accuracy	M	Classification Accuracy
10	0.78652	110	0.78653
20	0.78652	120	0.78653
30	0.78653	130	0.78654
40	0.78654	140	0.78655
50	0.78654	150	0.78656
60	0.78654	160	0.78657
70	0.78654	170	0.78657
80	0.78654	180	0.78657
90	0.78653	190	0.78656
100	0.78653	200	0.78656

Figure 22: Experimental results for hierarchical Bayesian logistic regression under varying numbers of (a) data-splits M and (b) NVI mixture components K. In (c) we show that the `EPVI_sample` method maintains a consistent classification accuracy over a wide range of M.

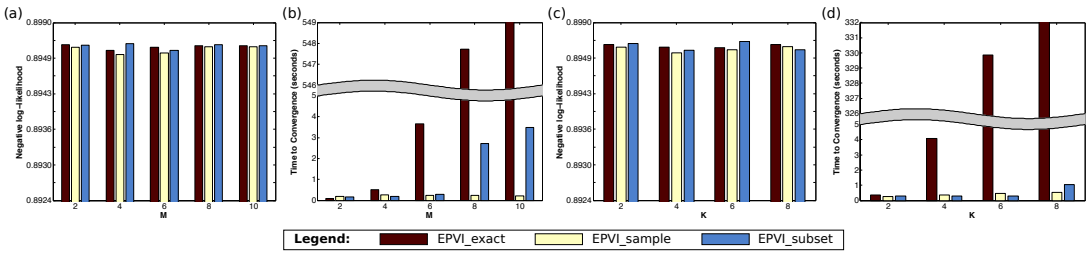


Figure 23: Comparison of the two product mixture sampling methods with the exact product mixture computation under (a)-(b) varying M and (c)-(d) varying K.

Performance under varying M**.** We vary the number of data-splits M from 2 to 20, and record the held-out negative log-likelihood and time taken to converge for each method. For the `Subposteriors` result, we report the maximum time taken to converge and the average negative log-likelihood (over the set of M subposteriors).

1 https://archive.ics.uci.edu/ml/datasets/SUSY

We also record the time taken to converge and the negative log-likelihood for the `NVI` ($M = 1$) standard VI result. The number of mixture components for the NVI part of all methods is fixed at $K = 4$. We plot these results in Figure 22(a), and see that `EPVI_sample` reduces the time to convergence by over an order of magnitude while maintaining nearly the same test negative log-likelihood as `NVI`. In Figure 22(c) we show that performance of the `EPVI_sample` method does not suffer as we increase the number of machines over a greater range, from $M = 10$ to $M = 200$. In this table, to give a more interpretable view of performance, we show classification accuracy (on the held out data) for each M. We see that classification accuracy stays nearly constant at approximately 0.7865 as we increase M throughout this range.

Performance under varying K**.** Next, we vary the number of NVI mixture components K from 2 to 8, and record the held-out negative log-likelihood and time taken to converge for each method. For parallel methods, we fix $M = 10$. We plot these results in Figure 22(b), and see that for all values of K, `EPVI_sample` decreases the time to convergence by nearly tenfold, while maintaining virtually identical test negative log-likelihood values.

Product mixture sampling methods. We also conduct experiments to judge the quality of our two product mixture sampling procedures. We aim to show that our methods yield similar test log-likelihoods as computing expectations via the exact product mixture while greatly decreasing the computation time. We demonstrate this empirically over a range of M and K values. Note that, since we need to compare with the exact product, we restrict this range to values in which we can compute all of the (exponentially-many) product mixture components. For both sampling methods, we fix $R = 500$. Note that we perform the $O(\log(M))$ rounds of `EPVI_subset` sequentially on the machine on which all samples are collected (not in parallel). We plot our results in Figure 23, and see that our sampling methods yield very similar held-out negative log-likelihoods as the exact product over all M (Figure 23(a)) and K (Figure 23(c)) values. We also see that for roughly $M > 6$ (Figure 23(b)) and $K > 4$ (Figure 23(d)), the time needed to compute the exact product increases substantially. Additionally, `EPVI_sample` appears to fare slightly better than `EPVI_subset` in terms of both the test log-likelihood and computation time.

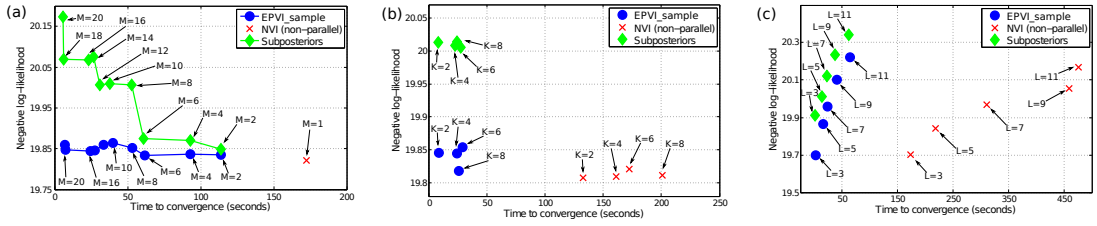


Figure 24: Experimental results for the nonlinear matrix factorization (topographical latent source analysis) model under varying numbers of (a) data-splits M, (b) mixture components K, and (c) latent sources L.

5.7.2 *Nonlinear Matrix Factorization*

We next apply our algorithm to a nonlinear matrix factorization model known as topographic latent source analysis (TLSA) [69]. This model can be viewed as representing an observed matrix as a covariate-dependent superposition of L latent sources. In particular, an observed matrix $\mathbf{U} \in \mathbb{R}^{N \times V}$ is assumed to be drawn conditioned on an observed matrix of covariates $\mathbf{X} \in \mathbb{R}^{N \times C}$, an inferred weight matrix $\mathbf{W} \in \mathbb{R}^{C \times L}$, and a basis matrix $\mathbf{G} \in \mathbb{R}^{L \times V}$ constructed by evaluating a parameterized spatial basis function with parameters $\{\bar{\mathbf{r}}_l, \lambda_l\}$, written $g_{lv} = \exp\{\lambda_l^{-1} \|\mathbf{r}_l - \bar{\mathbf{r}}_l\|^2\}$. Similar to the previous model, this model lacks conditional conjugacy. We can write the full generative process as

1. For latent source $l = 1, \ldots, L$,
   a) Draw hyperparameter $\lambda_l \sim \text{Exponential}(\rho)$
   b) For $d = 1, \ldots, M$, draw $\bar{r}_{ld} \sim \text{Beta}(1, 1)$
   c) For $c = 1, \ldots, C$, draw $w_{cl} \sim \mathcal{N}(0, \sigma_w^2)$
2. For $n = 1, \ldots, N$,
   a) For $v = 1, \ldots, V$, draw observation
   $u_{nv} \sim \mathcal{N}\left(\sum_{c=1}^{C} x_{nc} \sum_{l=1}^{L} w_{cl} g_{lv}, \tau^{-1}\right)$

We partition the data by splitting observed matrices $\mathbf{U}$ and $\mathbf{X}$ into M disjoint subsets each of size $\frac{N}{M}$, and inferring a variational approximation on each subset. This is illustrated in Figure 21.

Data. In the following experiments, we generate $N = 1,000$ observations in $V = 50$ dimensions by choosing hyperparameters $\{\tau = 1,\ \sigma_w^2 = 5,\ \rho = 1\}$, and drawing from the above generative process. We hold out 10% of the data for evaluating the test log-likelihood.

Performance under varying M**,** K**, and** L**.** Similar to the previous model, we first conduct experiments showing held-out negative log-likelihood versus time under varying values for the number of data-splits M and NVI mixture components K. These results are shown in Figure 24(a)-(b). We see that `EPVI_sample` reduces the time to convergence (particularly as the number of subposteriors M increases) while maintaining a similar test negative log-likelihood as `NVI`. We also evaluate the performance of our method under different numbers of latent sources L. We vary L from 2 to 8, and record the held-out negative log-likelihood and time taken to converge, and again see positive results (Figure 24(c)).

5.8 CONCLUSION

In this chapter, we developed an embarrassingly parallel VI algorithm for Bayesian inference in a distributed setting, that does not require models with conditional conjugacy and exponential family assumptions. Unlike existing methods, our strategy uses a decomposition of the full-data posterior involving a product of subposterior densities, which was recently developed in the parallel MCMC literature. We

have shown promising empirical results on nonconjugate models, which illustrate the ability of our method to perform VI in a distributed setting, and provide large speed-ups, while maintaining an accurate posterior approximation.

6

EMBARRASSINGLY PARALLEL INFERENCE IN DEPENDENT MODELS AND QUASI-ERGODIC SETTINGS

6.1 CHAPTER SUMMARY

A number of recent works have proposed "embarrassingly parallel" Markov chain Monte Carlo (MCMC) methods that share the following procedure: data is partitioned among a set of machines, MCMC is run to completion on each machine in parallel without communication, and then the resulting samples are combined to construct samples from the full-data posterior distribution. While some of these methods come with asymptotic guarantees about the correctness of samples, in certain models they can all run into the practical issue of *quasi-ergodicity*, i.e. when the parallel chains become stuck in disparate posterior modes and the sample combination procedures fail. In this chapter, we aim to extend these methods for use in quasi-ergodic settings by initially introducing a small amount of communication between machines, which is annealed to zero at a certain rate, so that the algorithm becomes embarrassingly parallel. Theoretically, we prove that our method generates asymptotically correct samples. Empirically, we demonstrate that our method effectively mitigates the quasi-ergodicity issue on several popular Bayesian models including mixture models, latent Dirichlet allocation, and probabilistic matrix factorization, in which current embarrassingly parallel MCMC methods fail to generate useful samples. We also show results on a large spatiotemporal data analysis task concerning urban transportation, using a novel model for collections of intracity car trips.

6.2 INTRODUCTION

The nonparametric density estimation method studied in [142] retains a key guarantee of classical MCMC: it generates asymptotically exact samples from the posterior distribution. However, even with this theoretical guarantee, the applicability of this method (as well as the others) is greatly limited due to the issue of *quasi-ergodicity*, or poor mixing, in many popular Bayesian models that have highly multimodal posterior distributions. Quasi-ergodicity refers to the problem where an MCMC chain enters a mode, and remains in this mode for the finite duration that sampling is performed, without exploring the full posterior space [65, 130]. For example, such behavior can be seen for MCMC in mixture models, topic models, matrix factorization models, neural networks, and other large Bayesian networks and Markov random fields. Often, samples drawn from around a single mode of the posterior

are still useful in practice, and hence quasi-ergodicity does not pose a major problem to classical (single chain) MCMC methods. However, it can be a major issue for embarrassingly parallel MCMC strategies, as separate chains may become stuck in different posterior modes, causing the accuracy of the final combined samples to be greatly diminished.

In this chapter, we aim to develop a class of embarrassingly parallel MCMC methods that exhibit good performance in these quasi-ergodic settings while retaining their asymptotic guarantees. We accomplish this by adding an initial phase to the parallel sampling procedure where we introduce a small amount of communication between machines to steer each of the parallel chains to a similar region of the posterior space before beginning fully embarrassingly parallel MCMC. While the chains initially undergo periodic synchronization, this is annealed until the procedure transitions into embarrassingly parallel MCMC.

We first define a general framework for embarrassingly parallel MCMC in quasi-ergodic settings in which one specifies a synchronization function, and we prove that if this function satisfies certain criteria and is applied at a certain decreasing frequency, we generate asymptotically correct samples. We then specify a few example synchronization procedures well-suited to different MCMC algorithms.

Unlike embarrassingly parallel MCMC methods that run separate, independent Markov chains on each machine, some existing MCMC parallelization methods aim to parallelize the computation for a *single Markov chain* [84, 94, 147], which we call single-chain parallel MCMC. Single-chain parallel MCMC does not suffer from quasi-ergodicity, at the expense of frequent synchronization. Our method can be viewed as a middle ground between these two extremes. In our procedure, data is partitioned over machines, computation involving only the subset of data is performed on each machine, and machines are synchronized according to the annealing schedule. We show that special cases of our methods can be framed as versions of single-chain parallel MCMC where communication is annealed to zero; this provides speedups as well as theoretical guarantees about the correctness of the samples, two major benefits over existing single-chain parallel MCMC methods.

OUR CONTRIBUTION In this chapter we develop techniques that extend the scope of embarrassingly parallel MCMC to many popular Bayesian models in which quasi-ergodicity occurs. We begin by introducing a general framework that allows us to derive multiple new parallel MCMC algorithms. We then prove that our algorithms inherit the same theoretical guarantees as existing embarrassingly parallel methods (in terms of asymptotically correct samples), under certain conditions, which we quantify. Empirically, we demonstrate the effectiveness of our methods on a few models in which current embarrassingly parallel MCMC methods fail in practice, including mixtures of Gaussians with large numbers of components, latent Dirichlet allocation, and Bayesian probabilistic matrix factorization.

6.3 EMBARRASSINGLY PARALLEL MCMC IN QUASI-ERGODIC SETTINGS

The embarrassingly parallel MCMC algorithm described above performs well when the posterior distribution is likely to be unimodal. However, when the posterior is multimodal, the separate, independent Markov chains can become stuck in different local modes, making it difficult for the final combination stage to work well. The key issue is that the embarrassingly parallel MCMC algorithm does not allow any communication during the MCMC run, making it impossible to steer each of the parallel chains to a similar region of the posterior space. Our general idea is to add a small amount of communication at the beginning of the run and later transit to embarrassingly parallel MCMC through a well-designed communication annealing procedure.

In the following subsections we first provide a general framework for embarrassingly parallel MCMC in quasi-ergodic settings through the use of a `Synch()` function, which dictates how parallel chains are synchronized, and a `SynchSched()` function, which dictates the schedule under which the chains are synchronized. We then give criteria for these functions under which we attain an embarrassingly parallel procedure that yields asymptotically exact samples (proven in Section 6.4). Finally, we specify a few cases of `Synch()` functions on common models and make connections with single-chain parallel MCMC methods.

6.3.1 *General Framework*

There are four main components in the framework: a `Synch()` function, which synchronizes the Markov chains, a `SynchSched()` function, which dictates the frequency of the synchronization, an `MCMC()` function, which samples the next step in a Markov chain for a subposterior density, and a `Combine()` function, which applies one of the existing sample combination procedures (listed in the introduction) to the sets of generated subposterior samples. We denote by $\texttt{Synch}_i^m()$ the synchronization function applied to the m^{th} Markov chain (i.e. the Markov chain on the m^{th} machine) at the i^{th} synchronization instance. We give the framework for embarrassingly parallel MCMC in quasi-ergodic settings in Algorithm 8.

6.3.2 *Criteria for `Synch()` and `SynchSched()` Functions*

Here we give criteria for the `Synch()` and `SynchSched()` functions to allow for a procedure that becomes embarrassingly parallel and yields asymptotically exact samples. First, the synchronization function must have a decreasing influence on each parallel chain, i.e. we must ensure that

$$\lim_{i\to\infty} \texttt{Synch}_i^m(\theta_{1:M}) = \theta_m. \tag{56}$$

Algorithmus 8 : Framework for Quasi-Ergodic Embarrassingly Parallel MCMC from Neiswanger et al. [137]

Input : A partition of data $\{x^{n_1}, \ldots, x^{n_M}\}$, `Synch()` function, `SynchSched()` function, `MCMC()` update function, `Combine()` function, and number of total samples T.

Output : T samples, $\theta^{1:T}$, from the full-data posterior.

$i \leftarrow 1$; `/* Initialize counter for synchronization loop */`
$t \leftarrow 1$; `/* Initialize counter for combined sample index */`
for $m = 1, \ldots, M$, initialize θ_m^0; `/* Initialize Markov chains */`
while $t < T$ **do**
  for $m = 1, \ldots, M$ **do** in parallel
    for $j = 1, \ldots,$ *SynchSched*(i) **do**
      $\theta_m^t \leftarrow$ `MCMC`(θ_m^{t-1}); `/* Take MCMC step for subposterior */`
      $t \leftarrow t + 1$;
  for $m = 1, \ldots, M$, $\theta_m^t \leftarrow$ `Synch`$_i^m(\theta_{1:M}^t)$; `/* Synchronize Markov chains */`
  $i \leftarrow i + 1$;
$\theta^{1:T} \leftarrow$ `Combine`$(\theta_1^{1:T}, \ldots, \theta_M^{1:T})$; `/* Combine subposterior samples */`

In most cases described below, we introduce a parameter γ_i, a function of the synchronization index i, which dictates the influence of the `Synch()` function: when $\gamma_i = 1$, all chains are synchronized to the same point, and when $\gamma_i = 0$, each chain is left unaffected. By picking an appropriate sequence of γ_i terms we can ensure that that our `Synch()` functions satisfy the above criteria. For example, we can set $\gamma_i = \frac{1}{i^k}$ for some $k \geqslant 1$ or $\gamma_i = \max\{0, 1 - ci\}$ for some $c \in [0, 1]$. We prove guarantees about the samples produced under this criteria in Section 6.4. By fixing $\gamma_i = 0$ throughout the entire run, we are left with existing embarrassingly parallel MCMC procedures.

Second, in order to ensure that our procedure becomes embarrassingly parallel, we must anneal the frequency of synchronization. We let `SynchSched(i)` give the number of MCMC steps in between the $(i-1)^{th}$ and i^{th} synchronization. Hence, we require that

$$\lim_{i \to \infty} \texttt{SynchSched}(i) = \infty. \tag{57}$$

For example, we could use the schedule $\texttt{SynchSched}(i) = ci^k$ for some $c, k \geqslant 1$, or keep the schedule at a constant value $c \geqslant 1$ until a condition is met (such when the synchronization influence parameter γ_i goes below a certain thereshold), and then set $\texttt{SynchSched}(i) = \infty$.

6.3.3 *Synchronization Functions*

We describe a few examples of `Synch()` functions, give examples to illustrate their use in practice, and show how certain choices of these functions can be used to generalize existing single-chain parallel MCMC methods.

6.3.3.1 *Synchronization to a Posterior Sample Estimate*

During embarrassingly parallel MCMC, each parallel chain samples from a subposterior density $p_m(\theta)$. Here, we synchronize all chains to an estimated sample from the full-data posterior density $p(\theta|x^N) \propto p_1 \cdots p_M(\theta)$. In existing embarrassingly parallel MCMC strategies, a subposterior sample combination procedure is performed after MCMC is complete, to construct samples from the full-data posterior; here it is performed once to generate a single sample at each synchronization instance.

We denote by $\widehat{\theta}_{p_1 \cdots p_M}$ a draw from an estimate of the product of subposterior densities, i.e.

$$\widehat{\theta}_{p_1 \cdots p_M} \sim \widehat{p_1 \cdots p_M}(\theta) \approx p(\theta|x^N). \tag{58}$$

In order to make the `Synch()` function satisfy the criteria described above, we introduce the γ_i parameter to ensure that the `Synch()` function has a decreasing effect on the current state of the Markov chain. Hence, to synchronize to an estimated posterior sample, we can write the `Synch()` function as

$$\texttt{Synch}_i^m(\theta_{1:M}) = \gamma_i \widehat{\theta}_{p_1 \cdots p_M} + (1 - \gamma_i)\theta_m. \tag{59}$$

EXAMPLE: A POSTERIOR SAMPLE ESTIMATE VIA SAMPLE AVERAGES. Scott et al. [164] provide a combination procedure to construct approximate samples from the full-data posterior distribution, which involves taking the arithmetic mean of subposterior samples. This procedure yields samples from an, in general, biased estimate of the density product function, though it can be carried out with very little computational cost. We can write the synchronization function as

$$\texttt{Synch}_i^m(\theta_{1:M}) = \frac{\gamma_i}{M} \sum_{m=1}^{M} \theta_m + (1 - \gamma_i)\theta_m \tag{60}$$

where the γ_i parameter dictates a weighted average between the estimated posterior sample and the current state of the m^{th} Markov chain θ_m. Note that this function reduces to an arithmetic mean of subposterior samples when $\gamma_i = 1$ and equals the unaffected Markov chain state θ_m when $\gamma_i = 0$.

A more complex subposterior combination procedure involving sample averages is given by [142], which is proven to yield samples from an unbiased estimate of the subposterior density product. The `Sync()` function for this procedure can be similarly formulated by substituting this estimate for $\widehat{\theta}_{p_1 \cdots p_M}$ into Equation 59.

6.3.3.2 *Synchronization with Single-Chain Parallel MCMC*

There exist single-chain parallel methods where data is partitioned over machines, and computation involving only the subset of data is performed on each machine. During sampling, the machines are synchronized at regular intervals [84, 94, 147]. However, many of these methods do not have theoretical guarantees regarding the correctness of the samples [94].

Using our framework, we can derive versions of these parallel single-chain MCMC methods, where we anneal communication (both the influence and frequency of the `Synch()` function) so that they transition to multiple-chain embarrassingly parallel MCMC methods. By doing this, we can both reduce the amount of communication required by these procedures (eventually making them communication-free), and ensure that they achieve theoretically correct sampling.

In many single-chain parallel MCMC procedures, each machine governs a subset of the model parameters and then overwrites this subset on the other machines during synchronization [84, 94, 147]. Some of these methods [94, 147], when used for Gibbs sampling, only perform the Gibbs updates on a given machine for the subset of parameters that the machine governs, while leaving the others fixed; these fixed parameters are only refreshed at each synchronization point.

Suppose we partition the model parameters $\theta \in \mathbb{R}^d$ into M disjoint subsets $\{B_1, \ldots, B_M\}$, i.e. $\theta = (\theta_{B_1}, \ldots, \theta_{B_M})$. We assume the m^{th} machine governs the m^{th} parameter block, and overwrites the values for this block on all other machines during synchronization. The synchronization step for single-chain parallel MCMC methods can therefore be written

$$\mathsf{Synch}_i^m(\theta_{1:M}) = (\theta_{1,B_1}, \theta_{2,B_2}, \ldots, \theta_{M,B_M}) \tag{61}$$

where θ_{m,B_m} denotes the m^{th} parameter block on the m^{th} machine.

To adhere to the above criteria, and convert this into a form that transitions into an embarrassingly parallel algorithm, we rewrite the above update with the γ_i influence parameter. Instead of fully overwriting each parameter block, we now take a weighted average between the values to be overwritten and the current Markov chain states. We can then write the synchronization function as

$$\begin{aligned}\mathsf{Synch}_i^m(\theta_{1:M}) = (&\gamma_i\theta_{1,B_1} + (1-\gamma_i)\theta_{m,B_1}, \ldots, \\ &\gamma_i\theta_{M,B_M} + (1-\gamma_i)\theta_{m,B_M})\end{aligned} \tag{62}$$

EXAMPLE: COLLAPSED GIBBS SAMPLING ON TOPIC MODELS In collapsed Gibbs sampling for latent Dirichlet allocation (LDA) [78], the goal is to infer a posterior distribution over topic assignment variables $z_{a,b}$ for a set of documents $a = 1, \ldots, A$

and words in each document $b = 1, \ldots, B_a$. Gibbs sampling proceeds by iterating through each variable $z_{a,b}$, and sampling from the conditional distribution

$$p(z_{a,b}|z_{-(a,b)}, y, \alpha, \beta) \propto \frac{(c^{-(a,b)}_{z_{a,b},a,1:J} + \alpha_{z_{a,b}}) \cdot (c^{-(a,b)}_{z_{a,b},1:A,y_{a,b}} + \beta_{y_{a,b}})}{c^{-(a,b)}_{z_{a,b},1:A,1:J} + \sum_{j=1}^{J} \beta_j}$$

where $z_{-(a,b)}$ denotes all other assignment variables, y denotes all words in all documents, $c^{-(a,b)}_{k,a,j}$ is a counter that denotes the number of times word j is assigned to topic k in document a excluding the assignment $z_{a,b}$, we've assumed a vocabulary of J words, and we've pre-specified hyperparameter vectors α and β.

Suppose we partition the documents over M machines. For a given machine m, we use $d^{(m)}$ to denote the subset of documents on this machine and $d^{(-m)}$ to denote all other documents. To apply the synchronization procedure described above, we let each machine m govern the subset of assignment variables associated with the documents in $d^{(m)}$, written $z_{d^{(m)}}$. At every synchronization, this machine will overwrite the values of its variables $z^m_{d^{(m)}}$ on all other machines, i.e.

$$\mathsf{Synch}^m_i(z^{1:M}) = \left(z^1_{d^{(1)}}, \ldots, z^M_{d^{(M)}}\right) \tag{63}$$

We use the variation of this procedure mentioned above where, on a machine m, we fix the values of the variables governed by other machines during sampling between synchronization points. We can then derive an expression for the conditional distribution used in Gibbs sampling for our method. At each synchronization, on a given machine m, after updating the assignment variables from all other machines, we can update the associated counter $c^{-(a,b)}_{k,d^{(-m)},j}$ (where the set $d^{(m)}$ in the index denotes the sum over all counters in that set). After incorporating the influence parameter γ_i, we can write an updated Gibbs step for sampling from the conditional distribution as a slightly modified version of the original Gibbs sampling distribution. The formula for the updated Gibbs step can be written as

$$p(z_{a,b}|z_{-(a,b)}, y, \alpha, \beta) \propto \frac{A^* \cdot B^*}{C^*} \tag{64}$$

where

$$\begin{aligned}
A^* &= c^{-(a,b)}_{z_{a,b},a,1:J} + \alpha_{z_{a,b}} \\
B^* &= c^{-(a,b)}_{z_{a,b},d^{(m)},y_{a,b}} + \gamma_i c^{-(a,b)}_{z_{a,b},d^{(-m)},y_{a,b}} + \beta_{y_{a,b}} \\
C^* &= c^{-(a,b)}_{z_{a,b},d^{(m)},1:J} + \gamma_i c^{-(a,b)}_{z_{a,b},d^{(-m)},1:J} + \sum_{j=1}^{J} \beta_j
\end{aligned}$$

6.3.4 *Method Complexity*

We characterize the complexity of MCMC techniques for sampling from the subposterior density on each machine, the complexity of the annealed communication procedures (described in Section 6.3.3) for synchronizing the chains, and the communication complexity (in terms of the total amount of information communicated).

Consider a dataset with N observations, and assume we have partitioned the data onto M machines, each holding $\frac{N}{M}$ observations. Running MCMC on each machine requires $O\left(\frac{N}{M}\right)$ operations per iteration, as opposed to $O(N)$ operations per iteration for typical MCMC methods applied to the full dataset.

For the synchronization procedures, assume we are sampling from posterior densities over d parameters. Synchronization to a posterior sample estimate via sample averaging (Section 6.3.3.1) requires $O(dM)$ operations for each application of the `Synch()` function. Similary, synchronization with single-chain parallel MCMC (Section 6.3.3.2) requires $O(dM)$ operations for each aplication of the `Synch()` function.

Communication in our procedures occurs when each machine sends information during each application of the `Synch()` function. In both of the above-mentioned methods of synchronization (to a posterior sample estimate and with single-chain parallel MCMC), each machine must communicate a single parameter sample. Therefore, in both cases, dM scalars are communicated at each application of the `Synch()` function.

6.4 THEORETICAL GUARANTEES

We now prove that, when the `Synch()` function adheres to the criteria given in Section 6.3.2, our proposed algorithm retains guarantees about the correctness of samples given by existing embarrassingly parallel procedures. In particular, we show that the Markov chain on each machine has the same stationary distribution as that in current embarrassingly parallel methods as $t \to \infty$. Hence, when `Combine()` procedures are applied to samples from each chain, we have the same asymptotic guarantees as current methods. In the following, we use δ to denote the total variation distance, defined as $\delta(P, Q) = \sup_{A \subset S} \|P(A) - Q(A)\|$ for distributions P and Q on a state space S. In the following, we sometimes write this distance equivalently as $\delta(p(\theta), q(\theta))$, where p and q are, respectively, the PDFs of P and Q.

Theorem 6.4.1. *On a given machine m, let $M_1(\theta|\theta')$ denote the transition kernel for an `MCMC` procedure generating samples from subposterior $p_m(\theta)$, where $\theta, \theta' \in S$. Let $M_2^t(\theta|\theta')$ denote the same transition kernel, at iteration t, when the `Synch()` function is applied periodically as in Algorithm 8. If the criteria in Equation 56 is satisfied, then for each $\theta' \in S$,*

$$\lim_{t \to \infty} \delta(M_1(\cdot|\theta') - M_2^t(\cdot|\theta')) = 0. \tag{65}$$

Proof. Let the i^{th} synchronization instance on a given machine m be written $\theta \sim \mathsf{Synch}^i_m(\theta|\theta_{1:M})$. We write t_i to denote the iteration that this synchronization takes place.

For all $\theta' \in S$, we can write the transition kernel pdf as

$$M_2^t(\theta|\theta') = \begin{cases} \mathsf{Synch}^i_m(\theta|\theta_{1:M}) & \text{if } t = t_i \\ M_1(\theta|\theta') & \text{otherwise} \end{cases}$$

For the given synchronization procedures described in Sec. 6.3.3, which use the formulation in terms of influence parameter γ_i, we can write θ^i_s as being drawn from a mixture of M_1 and some secondary distibution S^i_m, i.e. $\theta^i_s \sim \mathsf{Synch}^i_m(\theta|\theta_{1:M}) = (1-\gamma_i)M_1(\theta|\theta') + \gamma_i S^i_m(\theta|\theta_{1:M})$, where again we assume that $\mathsf{Synch}^i_m(\theta|\theta_{1:M})$ denotes the i^{th} synchronization instance on a given machine m, and where the parameter $\gamma_i \to 0$ as $t \to \infty$. We can therefore write the above transition kernel as

$$M_2^t(\theta|\theta') = \begin{cases} (1-\gamma_i)M_1(\theta|\theta') + \gamma_i S^i_m(\theta|\theta_{1:M}) & \text{if } t = t_i \\ M_1(\theta|\theta') & \text{otherwise} \end{cases}$$

This implies that $\lim_{t\to\infty} M_2^t(\cdot|\theta') = M_1(\cdot|\theta')$, i.e. that $M_2^t(\cdot|\theta')$ converges in distribution to $M_1(\cdot|\theta')$. We can then apply Scheffe's theorem [162], which directly gives that $\lim_{t\to\infty} \delta(M_1(\cdot|\theta') - M_2^t(\cdot|\theta')) = 0$. □

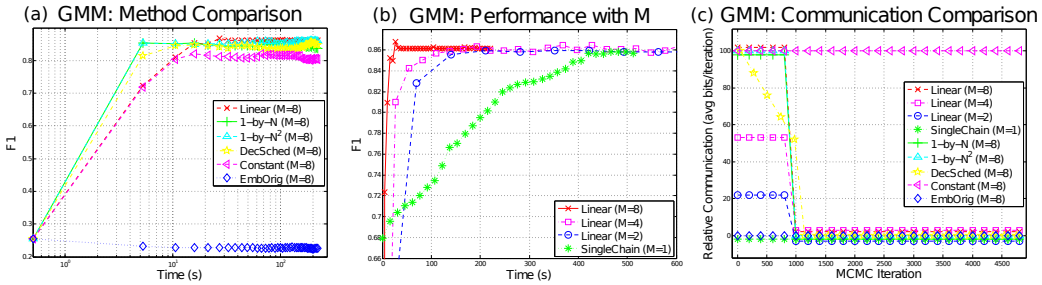


Figure 25: Empirical results for Gaussian mixture models. (a) Performance of our method versus comparison methods. (b) Performance for different numbers of machines $M = 1, 2, 4, 8$. (c) The average communication rate for each method.

Theorem 6.4.2. *Let $M_1(\theta|\theta')$ be the transition kernel for a Markov chain with stationary distribution $\pi(\theta)$ and let $M_2^t(\theta|\theta')$ be the transition kernel for a Markov chain parameterized by t such that for each $\theta' \in S$, $\lim_{t\to\infty} \delta(M_1(\theta|\theta') - M_2^t(\theta|\theta')) = 0$. Then,*

$$\lim_{t\to\infty} \delta\left(\int_{\theta'\in S} \pi(\theta')M_2^t(\theta|\theta') - \pi(\theta)\right) = 0 \tag{66}$$

i.e. in the limit, a stationary distribution exists for M_2 and is equal to $\pi(\theta)$.

Proof. We are given that, for each $\theta' \in S$, $\lim_{t\to\infty} \delta(M_1(\theta|\theta') - M_2^t(\theta|\theta')) = 0 \iff \forall \theta' \in S, \forall \epsilon > 0, \exists t' \text{ s.t. } \forall t \geqslant t', \sup_{\Theta\subset S} \left|\int_{\theta\in\Theta} \left(M_1(\theta|\theta') - M_2^t(\theta|\theta')\right)\right| < \epsilon$.

This implies that, $\forall \epsilon > 0, \exists t'' = t' \text{ s.t. } \forall t \geqslant t''$,

$$
\begin{aligned}
\delta\left(\int_{\theta'\in S} \pi(\theta')M_2^t(\theta|\theta') - \pi(\theta)\right) &= \sup_{\Theta\subset S}\left|\int_{\theta\in\Theta}\left(\pi(\theta) - \int_{\theta'\in S}\pi(\theta')M_2^t(\theta|\theta')\right)\right| \\
&= \sup_{\Theta\subset S}\left|\int_{\theta\in\Theta}\left(\int_{\theta'\in S}\pi(\theta')M_1(\theta|\theta') - \int_{\theta'\in S}\pi(\theta')M_2^t(\theta|\theta')\right)\right| \\
&= \sup_{\Theta\subset S}\left|\int_{\theta\in\Theta}\int_{\theta'\in S}\pi(\theta')\left(M_1(\theta|\theta') - M_2^t(\theta|\theta')\right)\right| \\
&= \sup_{\Theta\subset S}\left|\int_{\theta'\in S}\pi(\theta')\int_{\theta\in\Theta}\left(M_1(\theta|\theta') - M_2^t(\theta|\theta')\right)\right| \\
&\leqslant \sup_{\Theta\subset S}\int_{\theta'\in S}\pi(\theta')\left|\int_{\theta\in\Theta}\left(M_1(\theta|\theta') - M_2^t(\theta|\theta')\right)\right| \\
&\qquad \text{(via triangle inequality)} \\
&\leqslant \int_{\theta'\in S}\pi(\theta')\sup_{\Theta\subset S}\left|\int_{\theta\in\Theta}\left(M_1(\theta|\theta') - M_2^t(\theta|\theta')\right)\right| \\
&\qquad \text{(via Jensen's inequality)} \\
&\leqslant \int_{\theta\in S}\pi(\theta)\epsilon \\
&\qquad \text{(via given hypothesis)} \\
&= \epsilon\int_{\theta\in S}\pi(\theta) = \epsilon
\end{aligned}
$$

Hence $\lim_{t\to\infty}\delta\left(\int_{\theta'\in S}\pi(\theta')M_2^t(\theta|\theta') - \pi(\theta)\right) = 0$. □

6.5 SCOPE OF METHODS AND RELATED WORK

The methods described in this chapter can be directly used in conjunction with previously developed embarrassingly parallel MCMC procedures by performing synchronization to a posterior sample estimate using the `Combine()` function of the existing method (or by using another `Synch()` function that meets the given criteria). Existing embarrassingly parallel MCMC procedures have been derived for posterior distributions over real, finite dimensional parameters such as in Bayesian generalized linear models, nonparametric regression models, and hierarchical models [142, 164, 193, 206], as well as some infinite dimensional nonparametric Bayesian models, such as Gaussian processes [128, 177], and our methods can also be applied in these scenarios.

However, no existing embarrassingly parallel methods have yet been applied to sampling in large graphical models where parallel chains may get stuck in disparate modes, which is the main contribution of our methods. In [142], the authors propose

a way to sample from all modes in small-component mixture models by permuting cluster labels at each step in a Metropolis Hasting sampler in order for each cluster component to explore all modes. This bypasses the problem of quasi-ergodicity, and allows existing embarrassingly parallel methods to be applied here. However, the number of modes arising due to permutations of cluster labels grows factorially with the number of components, and so this strategy cannot scale to more than a modest number of components; further, this method only works for equally weighted mixtures.

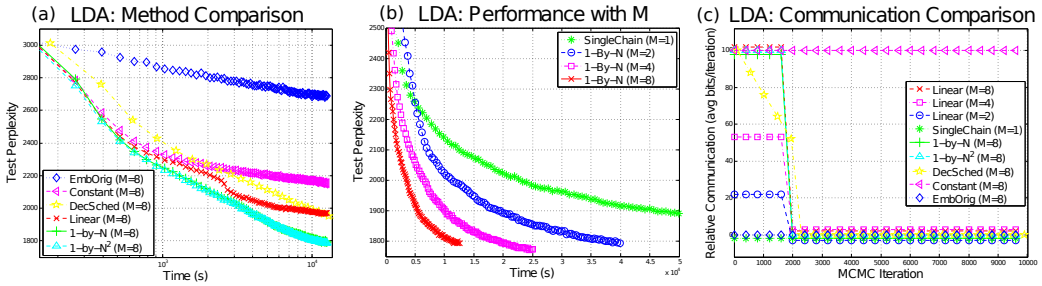


Figure 26: Empirical results for latent Dirichlet allocation. (a) Performance of our method versus comparison methods. (b) Performance for different numbers of machines $M = 1, 2, 4, 8$. (c) The average communication rate for each method.

6.6 EMPIRICAL STUDY

We evalute our method on three popular types of models in machine learning: finite mixture models, topic models, and probabilistic matrix factorization. For each model, we compare fully embarrassingly parallel MCMC (no communication during sampling), synchronous parallel single-chain MCMC (constant levels of communication during sampling), and our annealed communication methods (which go from synchronous to embarrassingly parallel during sampling) under different `Synch()` and `SynchSched()` functions. We also show results for a non-parallel single chain of MCMC, and we show how our methods scale as the number of machines M increases. Specifically, for each of the three models, we have the following comparison methods:

- **EmbOrig**: Fully embarrassingly parallel MCMC, where we fix $\gamma_i = 0$ over all iterations.
- **Linear**: At each synchronization, we decrease γ_i linearly, i.e. we let $\gamma_i = \max\{0, 1 - ci\}$. We let $c = 0.05$, which becomes embarrassingly parallel after the 20^{th} synchronization.
- **1-By-N**: At each synchronization, we decrease γ_i by $\frac{1}{n}$, i.e. we let $\gamma_i = 1 - (i-1)\frac{1}{n}$. After 20 iterations, we fix $\gamma_i = 0$ and begin embarrassingly parallel MCMC.

- **1-By-N^2**: At each synchronization, we decrease γ_i by $\frac{1}{n^2}$, i.e. we let $\gamma_i = 1 - (i-1)\frac{1}{n^2}$. After 20 iterations, we fix $\gamma_i = 0$ and begin embarrassingly parallel MCMC.
- **DecSched**: At each synchronization, we decrease γ_i linearly, i.e. we let $\gamma_i = \max\{0, 1 - 0.05i\}$. We also decrease the frequency of communication until embarassingliy parallel MCMC begins, i.e. we set $\texttt{SynchSched}(i) = \texttt{SynchSched}(1) + (i-1)20$.
- **Constant**: We synchronize at regular intervals without decreasing the influence parameter, i.e. we fix $\gamma_i = 1$ over all iterations.
- **SingleChain**: Typical, single chain MCMC run on the full data, without any parallelism.

For mixture models and probabilistic matrix factorization, we conducted experiments using a standard cluster system. We obtained subposterior samples by submitting batch jobs to each worker since these jobs are all independent. We then saved the results to the disk of each worker and transferred them to the same machine, which performed the final combination. In these experiments, to generate the plots showing performance versus time, we collected all samples generated before a given number of seconds, and added the time taken to transfer the samples and perform the final combination step (note that our final combination procedure used the nonparametric algorithm given in [142]). For latent Dirichlet allocation, we conducted experiments on a multi-core machine with a shared memory architecture. In these experiments, in the plots showing performance versus time, we plot absolute wall-clock time.

GAUSSIAN MIXTURE MODELS (GMM). In a Bayesian Gaussian mixture model (GMM), assume that we have a weighted average of K Gaussian densities, with mean parameters $\mu_1, \ldots, \mu_K$, which generate N observed variables $x_1, \ldots, x_N$. Each observed variable x_i is paired with an associated assignment variable z_i. Further, assume that we have placed a uniform prior with a fixed range (a, b) on each of the mean parameters, and a categorical prior with parameter π on each of the assignment variables. We can write the generative process for this model as

1. Draw $\mu_k \sim \text{Uniform}(a, b)$ for $k = 1, \ldots, K$.
2. For $i = 1, \ldots, N$:
   a) Draw $z_i \sim \text{Cat}(\pi)$
   b) Draw $x_i \sim \mathcal{N}(\mu_{z_i}, \sigma^2)$

We generate a synthetic dataset for this model by drawing $N = 1,000,000$ observations from a $K = 50$ component univariate, uniformly weighted Gaussian mixture. Each component mean was drawn from a Uniform$(0, 10)$ distribution, and each component variance was fixed at $\sigma^2 = 0.01$.

For each of the above-described comparison methods, we used a Metropolis Hastings sampler for our `MCMC` procedure. For the `Synch()` function, we synchronized

to an estimated posterior sample via averaging, as described in Section 6.3.3. For each of our methods, we fix $\texttt{SynchSched}(1) = 500$, and keep this constant until the influence parameter $\gamma_i = 0$, at which point we set $\texttt{SyncSched}(i) = \infty$ (except for the **DecSched** method, which is described above). After completing MCMC with each method, we inferred a mixture component assignment for each observation to get a clustering result.

To evaluate the performance of our methods, we computed the F1 score between the clusterings obtained by each algorithm and the ground truth clustering. Let $\mathcal{P}^g$ denote the set of pairs of observations clustered together in the ground truth, and let $\mathcal{P}^r$ denote the set of pairs of observations clustered together by a given proceedure. The F1 score is then defined to be the harmonic mean between the precision $= |\mathcal{P}^g \cap \mathcal{P}^r|/|\mathcal{P}^r|$ and the recall $= |\mathcal{P}^g \cap \mathcal{P}^r|/|\mathcal{P}^g|$. This definition of F1 score is invariant to permutations of the observations [200].

The results for this model are shown in Figure 25. Figure 25(a) shows the F1 score versus time for each of our competing procedures, with the number of machines fixed at $M = 8$. We see that fully embarrassingly parallel MCMC (**EmbOrig**) fails to generate useful samples (and yields an F1 score that only decreases after initialization), while the methods proposed in this chapter (and in particular the **1-By-N**^{2} method) give the best performance. Figure 25(b) shows that our methods yield improved performance as the number of machines is increased, for a fixed amount of data (demonstrated in this plot with the **Linear** synchronization method). Figure 25(c) shows the average number of bits used in communication by each of the methods at each point in time.

LATENT DIRICHLET ALLOCATION (LDA). In latent Dirichlet allocation (LDA), assume that we have K topic vectors, $\phi_1, \ldots, \phi_K$ (each a J-dimensional vector that sums to 1, where we've assumed a vocabulary of J words), from which we generate a set of documents indexed $a = 1, \ldots, A$ and words in each document w_b, $b = 1, \ldots, B_a$ (using the notation from Section 6.3.3.2). Each document a is associated with a topic allocation vector θ_a (a K-dimensional vector that sums to 1), and a topic assignment variable z_b for each word $b = 1, \ldots, B_a$ in that document. We place a multinomial prior on each topic vector, and a Dirichlet prior on each document's topic allocation vector. We can write the generative process for this model as

1. Draw $\phi_k \sim \text{Mult}(\beta)$ for $k = 1, \ldots, K$
2. For $a = 1, \ldots, A$:
   a) Draw $\theta_a \sim \text{Dir}(\alpha)$
   b) For $b = 1, \ldots, B_a$:
      i. Draw $z_b \sim \text{Cat}(\theta)$
      ii. Draw $w_b \sim \text{Cat}(\phi_{z_b})$

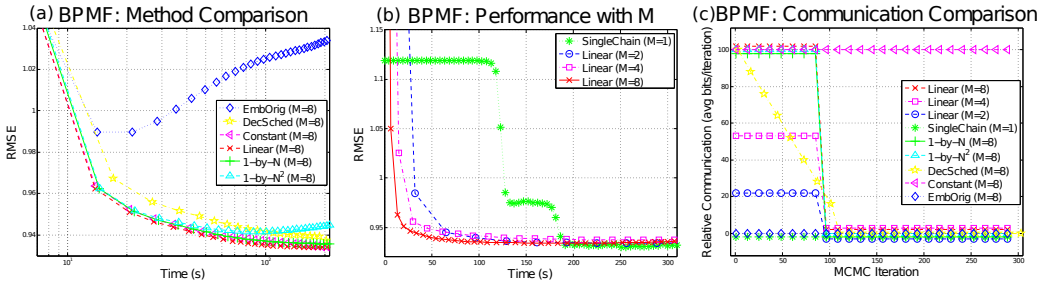


Figure 27: Empirical results for Bayesian probabilistic matrix factorization. (a) Performance of our method versus comparison methods. (b) Performance for different numbers of machines $M = 1, 2, 4, 8$. (c) The average communication rate for each method.

We demonstrate our methods for LDA on the NIPS dataset[1], which has approximately $2,000,000$ tokens (i.e. assignment variables z_i, over all documents) and a vocabulary of $12,000$ words. Here, we use collapsed Gibbs sampling for the `MCMC` procedure, where we use the `Synch()` function given in the LDA example in Section 6.3.3.2. For each of our methods, we fix `SynchSched(1)` $= 100$, and keep this constant until the influence parameter $\gamma_i = 0$, upon which we set `SyncSched`$(i) = \infty$ (except for the **DecSched** method, which is described above). For the **Constant** comparison method, this procedure is very similar to the approximate distributed LDA (AD-LDA) algorithm given by [147]. We ran each of the comparison methods on 90% of the data and then computed the perplexity on the final held-out 10% of the data, using the formula in [147].

The results for this model are shown in Figure 26(a)-(c). Notably, we see in Figure 26(a) that we can achieve a lower test perplexity when we reduce the communication of AD-LDA to achieve embarrassingly parallel sampling. We can therefore generate more-accurate samples in less time (and require much less synchronization). We also see improved performance as the number of cores is increased in Figure 26(b).

BAYESIAN PROBABILISTIC MATRIX FACTORIZATION (BPMF). We next demonstrate our methods for Bayesian probabilistic matrix factorization (BPMF) [160]. Suppose we have N users and M movies, where R_{ij} is the rating given by user i for movie j. Let U_i and V_j, respectively, be user-specific and movie-specific latent feature vectors. Further, suppose we have user hyperparameters $\Theta_U = \{\mu_U, \Lambda_U\}$ (for the prior on each U_i) and movie hyperparameters $\Theta_V = \{\mu_V, \Lambda_V\}$ (for the prior on each V_j), each of which is also given a prior with parameters $\Theta_0 = \{\mu_0, \beta_0, W_0, \nu_0\}$. We can write the full generative process as

1. Draw $\Theta_U \sim \mathcal{N}(\mu_U|\mu_0, (\beta_0\Lambda_U)^{-1})\mathcal{W}(\Lambda_U|W_0, \nu_0)$
2. Draw $\Theta_V \sim \mathcal{N}(\mu_V|\mu_0, (\beta_0\Lambda_V)^{-1})\mathcal{W}(\Lambda_V|W_0, \nu_0)$

1 https://archive.ics.uci.edu/ml/datasets/Bag+of+Words

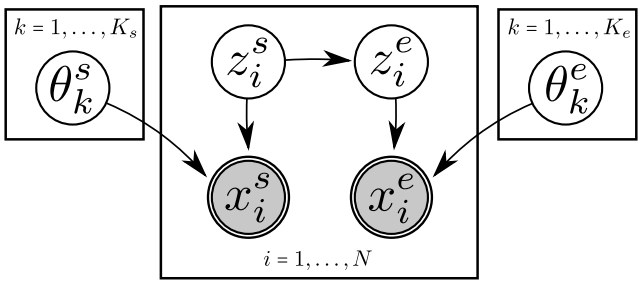


Figure 28: Graphical model for the hidden segments mixture model (HSMM) for taxi records.

3. For $i = 1, \ldots, N$:
   a) Draw $U_i \sim \mathcal{N}(\mu_U, \Lambda_U^{-1})$
4. For $j = 1, \ldots, M$:
   a) Draw $V_j \sim \mathcal{N}(\mu_V, \Lambda_V^{-1})$
5. For $i = 1, \ldots, N$ and $j = 1, \ldots, M$:
   a) Draw $R_{ij} \sim \mathcal{N}(U_i^\top V_j, \alpha^{-1})$

where $\mathcal{W}(W_0, \nu_0)$ denotes a Wishart distribution with ν_0 degrees of freedom and a matrix scale parameter W_0.

We apply our model to a subset of the Netflix dataset[2], which contains $1,000,000$ (user, movie, rating) triples. For the `MCMC` procedure, we applied a Gibbs sampler[†], and for the `Synch()` procedure, we synchronized to an estimated posterior sample via averaging. To evaluate performance, we compute the root mean squared error (RMSE), defined in [160], on a held-out test set of 10% of the data, for each of the competing methods.

The results for this model are shown in Figure 27(a)-(c). We again see in Figure 27(a) that typical embarrassingly parallel MCMC (without any initial synchronization) shows poor results (where the RMSE begins to increase significantly), while the methods presented in this chapter are able to continually decrease the RMSE as sampling continues.

6.7 ANALYZING NEW YORK CITY TAXI RECORDS

Our final experiment involves a large-scale exploratory analysis of spatiotemporal transportation-flow data, which allows users to flexibly analyze traffic patterns at arbitrarily chosen portions of, for example, the day, week, and year. To achieve this analysis, we design a graphical model for this data and use it in a large scale distributed inference experiment.

2 http://www.cs.toronto.edu/ rsalakhu/BPMF.html

Our data consisted of New York City taxi records from the year 2015 [3]. Each taxi record consists of a time and location for both pickup (i.e. start of the trip) and dropoff (i.e. end of the trip.). There are roughly 14 million taxi trips each month and 170 million total trips in 2015. Taxi locations range over the five New York City Boroughs, as well as some nearby locations in New Jersey (and, in rare cases, other states).

In this experiment we aim to demonstate the ability of our method to allow for flexible analysis of spatiotemporal data. In particular, we aim to show that our embarrassingly parallel MCMC methods can allow for analysis of arbitrarily chosen subsets of a large datset *after* inference has already been completed. As an example use case, consider the taxi records datset. This is a time-varying dataset, and we can use this temporal component to split up the data into groups. In this case, we split our data into fine grained groups of taxi trips with pickup times occuring in the same hour (for example, in the Month of February of 2015 alone, this yields 672 groups of data). We can then perform MCMC on each subset (i.e. each hour) of data. In previous experiments, we have been interested in combining all locally inferred models. Here, we are interested in taking any subset of models (i.e. any arbitrary group of hours) and selectively combining those. By doing this, we can yield an inference result on queries such as the flow of traffic during weekdays versus weekends, or in morning rush hours versus evening rush hours; we can also aim to analyze traffic patterns on days with certain, for example, weather events (i.e. hours where it was sunny versus stormy) or in the presence of other covariates.

We next describe our model. Denote the i^{th} taxi trip pickup location as x_i^s and dropoff location as x_i^e. In this model, we assume that pickup and dropoff locations tend to cluster at common locations in the city; we will refer to these clusters as "hubs" or "transportation hubs". Our model includes an independent set of hubs for both pickup locations and dropoff locations. Let z_i^s be an assignment variable for pickup location x_i^s, which labels the identity of the associated pickup hub for the i^{th} record, and let z_i^e be a similar assignment variable for dropoff x_i^e. Further, assume there are K_s pickup hubs (i.e. $z_i^s \in \{1, \ldots, K_s\}$) and K_e dropoff hubs (i.e. $z_i^e \in \{1, \ldots, K_e\}$). Additionally, let θ_k^s denote the parameters of the emission distribution f_s for the k^{th} pickup hub (i.e. parameters of the distribution f_s that generates the x_i^s assigned to pickup hub k) and let θ_k^e denote the parameters of the emission distribution f_e for the k^{th} dropoff hub (i.e. parameters of the distribution f_e that generates the x_i^e associated with pickup hub k). Finally, assume there is an appropriately normalized transition matrix T that dictates the transition, or relationship, between a taxi record's pickup hub and its dropoff hub; more specifically, the entry T_{k_1,k_2} in this matrix will dictate the probability of transitioning to dropoff hub k_2 from pickup hub k_1.

3 http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml

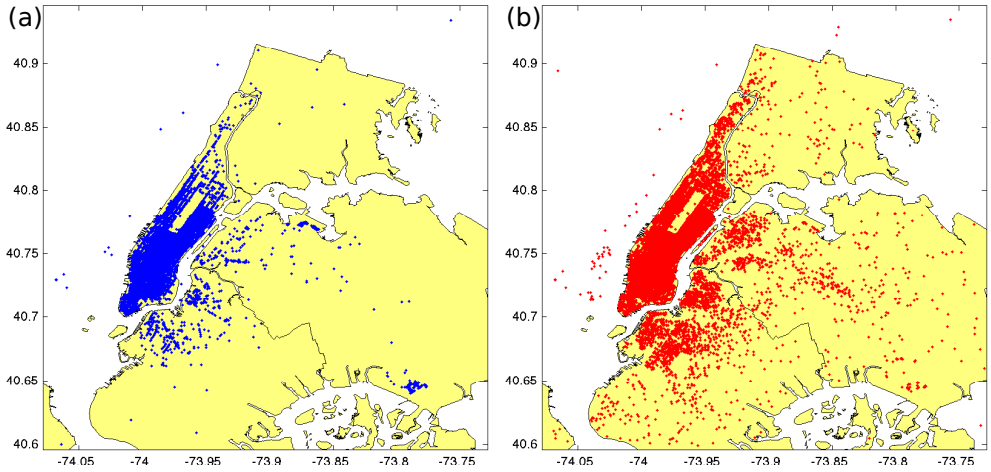


Figure 29: One hour of (a) taxi pickup locations in blue, and (b) taxi dropoff locations in red.

The generative process of our model, which we call a hidden segments mixture model (HSMM) for taxi records $i = 1, \ldots, N$, can be written as following:

$$\begin{aligned}
\theta^s_k &\sim g_s(\alpha_s), \text{ for } k = \{1, \ldots, K\} \\
\theta^e_k &\sim g_e(\alpha_e), \text{ for } k = \{1, \ldots, K\} \\
z^s_i &\sim \text{Uniform}(\{1, \ldots, K\}) \\
z^e_i &\sim \text{Categorical}(T_{z^s_i,:}) \\
x^s_i &\sim f_s\left(\theta^s_{z^s_i}\right) \\
x^e_i &\sim f_e\left(\theta^s_{z^s_i}\right)
\end{aligned}$$

where g_s and g_e are, respectively, prior distributions over pickup hub and dropoff hub parameters (and are parameterized by α_s and α_e) and $T_{k,:}$ denotes the k^{th} row of transition matrix T. In the following experiments, we will assume very simple emission distributions for both taxi pickups and dropoffs: we choose f_s and f_e to be Gaussian, and hub parameter priors g_s and g_e to be Normal-Wishart. Furthermore, we fix $K = 30$ through all experiments; this value was chosen via visual inspection of inference results on a subset of the data. A graphical depiction of this model is drawn in Figure 28. When performing the combination procedure in this model, we use the parametric combination strategy (Section 3.1), which yields better results given the highly multimodal posterior density landscape in this latent variable model.

We plot one hour of this data in Figure 29. Plot (a) shows all pickup locations in blue, and plot (b) shows all dropoff locations in red. There are slightly over 30,000 taxi records in this hour alone. On individual groups of data (i.e. over individual hours), we perform inference in this model using Gibbs sampling. We run each

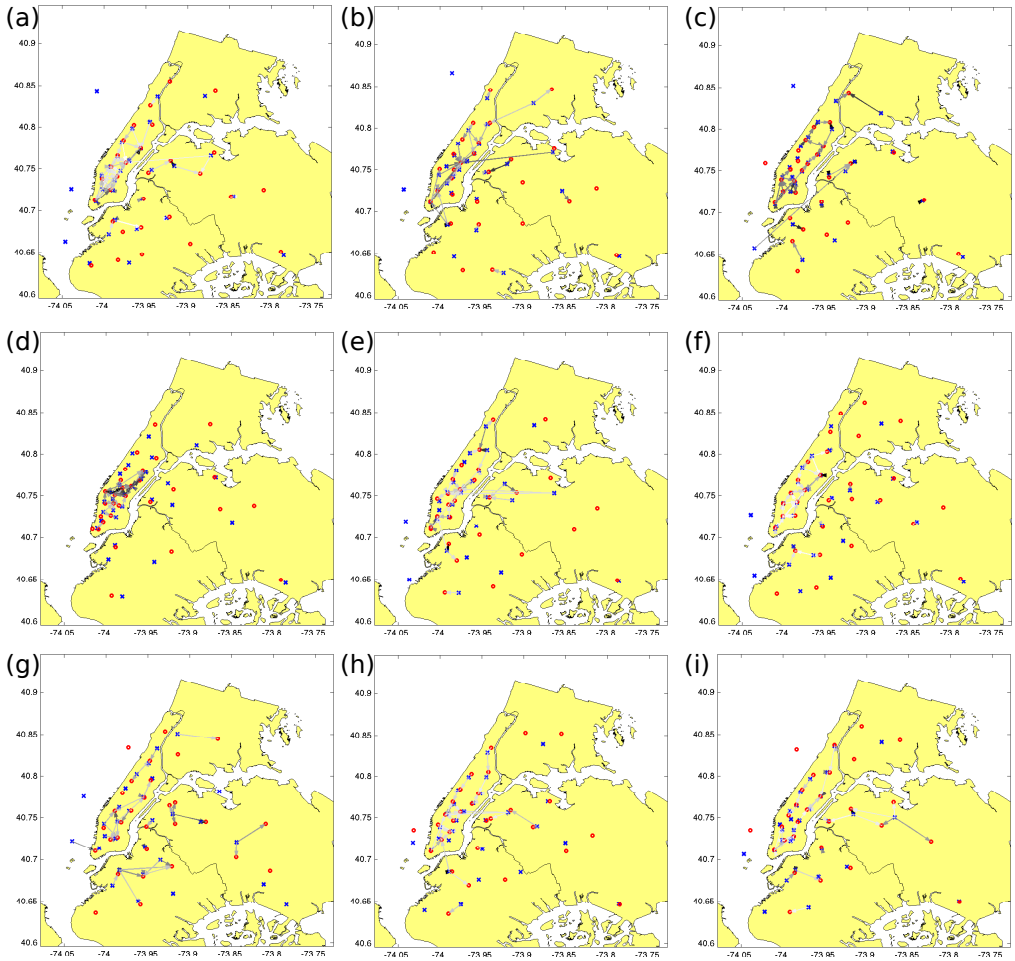


Figure 30: Combined inference results for the HSMM applied to one month of taxi data, showing taxi traffic patterns at different times of the day and week. Each plot shows the inferred pickup hubs (blue crosses), dropoff hubs (red circles), and highest-weighted transitions between pickup and dropoff hubs (directed edges denote the largest 40 elements of T, where a darker edge corresponds with a stronger weight). See text for details on individual plots.

of these chains of MCMC for 50,000 steps, and then thin and randomly permute the resulting samples. Our Gibbs sampling implementation takes on the order of one hour to yield these samples. We also run our combination algorithm for 50,000 steps; in the following visualizations, we plot the parameters yielded by the final step of this algorithm (we are therefore plotting an approximate point estimate, as it is difficult to visualize a posterior distribution over the parameters of this model).

In this set of experiments, we combine selective subsets of the inferred local (hourly) results to analyze taxi traffic flow; in particular, we determine pickup hubs, dropoff hubs, and transition probabilities between the hubs, at different times of the

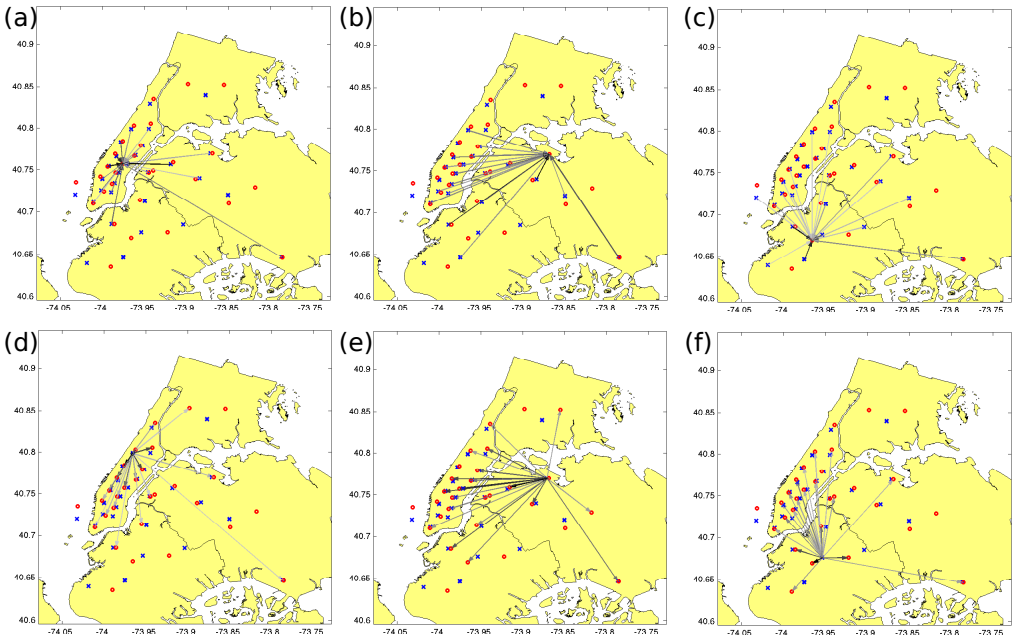


Figure 31: Combined inference results for the HSMM applied to one month of taxi data, showing taxi traffic patterns either to or from a selected individual hub. Each plot shows the inferred pickup hubs (blue crosses), dropoff hubs (red circles), and highest-weighted transitions for a single pickup or dropoff hub (directed edges denote the largest 20 elements of the row or column of T associated with this pickup or dropoff hub, where a darker edge corresponds with a stronger weight). The top row of plots shows the distribution over pickup hubs for three dropoff hubs (one in Manhattan, one in Queens, and one in Brooklyn), while the bottom row of plots shows the distribution over dropoff hubs for three pickup hubs (in three similar locations). See text for more details on individual plots.

day and week, over the course of the month of February 2015. We show results in Figure 30. In each plot, we overlay the found pickup hubs (blue crosses) and dropoff hubs (red circles) on top of the outlines of the five New York City boroughs (note that some of the hubs are positioned outside of the boroughs in New Jersey). To show the transition probabilities between the hubs, we plot arrows between hubs for the 40 highest weighted entries of the transition matrix T; these arrows are colored based on their weights, with a darker arrow corresponding to a stronger transition between hubs. As an example, we show an inference result for a single hour (17:00-18:00 on Thursday, February 2015) in Figure 30 (d); Note that this result is yielded directly from local inference, without any futher combination.

In Figure 30, plot (a) shows the inference result of combining all possible hours (i.e. the model posterior over the full dataset). We see here that the hubs are fairly evenly distributed over Manhattan, Brooklyn, and Queens (with a concentration along the East and West side of Manhattan), and a small collection of pickup hubs

in New Jersey. We furthermore see that the transitions between hubs concentrate in Manhattan and East Queens, but are all fairly homogeneous (without a large pattern of specific highly weighted transitions). Plots (b) and (c) show the traffic of pattern during morning rush hour (06:00-09:00) and evening rush hour (16:00-19:00), respectively. In (b), we see a strong concentration of traffic going to midtown Manhattan (particularly, midtown east) and the financial district (South Manhattan). In these morning hours, there is relatively little taxi traffic in Brooklyn and Queens (though we do see a major flow from a hub associated with LaGuardia airport, in North Queens, to midtown Manhattan). In (c), during the evening hours, we see major traffic to and between Greenwich Village and East Village in Manhattan, and to the West and East sides of cental park; we also see increased activity in both Queens and Brooklyn. Plots (e), (f), (h), and (i) aim to elicit the differences in traffic patterns between weekdays and weekends. In particular, plots (e) and (f) show combined model results during weekdays (i.e. Mondays through Fridays), where (e) shows results for the 09:00 hour only, and (f) shows results taken over all hours in the day. Likewise, plots (h) and (i) show combined model results during weekends (i.e. Saturday and Sunday), where (h) shows results taken for the 09:00 hour only, and (i) shows results taken over all hours in the day. We see in weekdays, there is a much higher concentration of taxi traffic in Manhattan (and looking at the 09:00 hour, we see a particular concentration in Midtown Manhattan and East Queens near the border of Manhattan), while in weekends, there is a much higher concentration around the border of central park, throughout Queens, and in Brooklyn. Finally, in plot (g), we show taxi traffic results combined only over late-night hours (00:00-04:00). In this plot, we see far more traffic in small clusters throughout Brooklyn, Queens, and the Bronx, and a cluster near Greenwich Village and East Village in Manhattan.

We can also use the inferred model results to show the transportation patterns for a selected individual hub. For example, we might select the hub closest to the LaGuardia airport in North Queens, and want to see the distribution over hubs that people take the taxi to (from LaGuardia) or the distribution over hubs that people take the taxi from (to LaGuardia). We can get these types of results for any of our inferred hubs. In Figure 31, we show such results for a few selected hubs. Plot (a) shows the distribution over pickup hubs for a single dropoff hub in midtown Manhattan. We see that most taxi trips are from neighboring hubs in Manhattan, and from LaGuardia airport in North Queens and John F. Kennedy airport in Southeast Queens. Similarly, in Plot (b), we show the distribution over dropoff hubs for a single pickup hub in Manhattan, and see a similar transportation pattern. In plots (b) and (e) we show the same type of results for a dropoff hub and pickup hub in North Queens, and in plots (c) and (f) we show the same type of results for a dropoff hub and pickup hub in central Brooklyn. Of note is that plots (b) and (e) correspond to the dropoff and pickup hubs closest to LaGuardia airport in queens; we see that the highest weighted pickup hubs traveling to to this airport are in Queens and Brooklyn

(including the hub located at John F. Kennedy Airport), while the highest weighted dropoff hubs traveling from this airport are in West Queens and Manhattan.

6.8 CONCLUSION

Current embarrassingly parallel MCMC approaches allow for parallel sampling with very low communication costs, but are not able to be applied in quasi-ergodic settings, which includes many popular machine learning models. In this chapter we proposed a generic framework to extend embarrassingly parallel techniques to these settings by introducing an initial communication phase that steers parallel MCMC chains to a similar region of the posterior space. The amount of communication goes to zero as the chains converge, and the procedure transitions to an embarrassingly parallel sampler. We used our general framework to derive multiple new parallel MCMC algorithms and showed how special cases of these algorithms are similar to existing single-chain MCMC methods, with the benefits of less communication and stronger theoretical guarantees regarding the correctness of samples. Additionally, we proved that our proposed method retains the asymptotic convergence guarantees of existing methods, and we demonstrated good empirical performance on multiple applications, including clustering with mixture models, topic modeling, matrix factorization, and analysis of a large urban transportation dataset.

Part III

INCORPORATING STRUCTURE

7

PRIOR SWAPPING

7.1 CHAPTER SUMMARY

While Bayesian methods are praised for their ability to incorporate useful prior knowledge, in practice, convenient priors that allow for computationally cheap or tractable inference are commonly used. In this chapter, we investigate the following question:

> *For a given model, is it possible to compute an inference result with any convenient false prior, and afterwards, given any target prior of interest, quickly transform this result into the target posterior?*

A potential solution is to use importance sampling (IS). However, we demonstrate that IS will fail for many choices of the target prior, depending on its parametric form and similarity to the false prior. Instead, we propose prior swapping, a method that leverages the pre-inferred false posterior to efficiently generate accurate posterior samples under arbitrary target priors. Prior swapping lets us apply less-costly inference algorithms to certain models, and incorporate new or updated prior information "post-inference". We give theoretical guarantees about our method, and demonstrate it empirically on a number of models and priors.

7.2 INTRODUCTION

There are many cases in Bayesian modeling where a certain choice of prior distribution allows for computationally simple or tractable inference. For example,

- Conjugate priors yield posteriors with a known parametric form and therefore allow for non-iterative, exact inference [48].
- Certain priors yield models with tractable conditional or marginal distributions, which allows efficient approximate inference algorithms to be applied (e.g. Gibbs sampling [170], sampling in collapsed models [181], or mean-field variational methods [192]).
- Simple parametric priors allow for computationally cheap density queries, maximization, and sampling, which can reduce costs in iterative inference algorithms (e.g. Metropolis-Hastings [125], gradient-based MCMC [134], or sequential Monte Carlo [53]).

For these reasons, one might hope to infer a result under a convenient-but-unrealistic prior, and afterwards, attempt to correct the result. More generally, given an infer-

ence result (under a convenient prior or otherwise), one might wish to incorporate updated prior information, or see a result under different prior assumptions, without having to re-run a costly inference algorithm.

This leads to the main question of this chapter: for a given model, is it possible to use any convenient false prior to infer a false posterior, and afterwards, given any target prior of interest, efficiently and accurately infer the associated target posterior?

One potential strategy involves sampling from the false posterior and reweighting these samples via importance sampling (IS). However, depending on the chosen target prior—both its parametric form and similarity to the false prior—the resulting inference can be inaccurate due to high or infinite variance IS estimates (demonstrated in Sec. 7.3.1).

We instead aim to devise a method that yields accurate inferences for arbitrary target priors. Furthermore, like IS, we want to make use of the pre-inferred false posterior, without simply running standard inference algorithms on the target posterior. Note that most standard inference algorithms are iterative and *data-dependent*: parameter updates at each iteration involve data, and the computational cost or quality of each update depends on the amount of data used. Hence, running inference algorithms directly on the target posterior can be costly (especially given a large amount of data or many target priors of interest) and defeats the purpose of using a convenient false prior.

In this chapter, we propose *prior swapping*, an iterative, *data-independent* method for generating accurate posterior samples under arbitrary target priors. Prior swapping uses the pre-inferred false posterior to perform efficient updates that do not depend on the data, and thus proceeds very quickly. We therefore advocate breaking difficult inference problems into two easier steps: first, do inference using the most computationally convenient prior for a given model, and then, for all future priors of interest, use prior swapping.

In the following sections, we demonstrate the pitfalls of using IS, describe the proposed prior swapping methods for different types of false posterior inference results (e.g. exact or approximate density functions, or samples) and give theoretical guarantees for these methods. Finally, we show empirical results on heavy-tailed and sparsity priors in Bayesian generalized linear models, and relational priors over components in mixture and topic models.

7.3 METHODOLOGY

Suppose we have a dataset of n vectors $x^n = \{x_1, \ldots, x_n\}$, $x_i \in \mathbb{R}^p$, and we have chosen a family of models with the likelihood function $L(\theta|x^n) = p(x^n|\theta)$, parameterized by $\theta \in \mathbb{R}^d$. Suppose we have a prior distribution over the space of model parameters θ, with probability density function (PDF) $\pi(\theta)$. The likelihood and prior define a joint model with PDF $p(\theta, x^n) = \pi(\theta)L(\theta|x^n)$. In Bayesian inference, we are

interested in computing the posterior (conditional) distribution of this joint model, with PDF

$$p(\theta|x^n) = \frac{\pi(\theta)L(\theta|x^n)}{\int \pi(\theta)L(\theta|x^n)\,d\theta}. \tag{67}$$

Suppose we've chosen a different prior distribution $\pi_f(\theta)$, which we refer to as a *false prior* (while we refer to $\pi(\theta)$ as the *target prior*). We can now define a new posterior

$$p_f(\theta|x^n) = \frac{\pi_f(\theta)L(\theta|x^n)}{\int \pi_f(\theta)L(\theta)|x^n)\,d\theta} \tag{68}$$

which we refer to as a *false posterior*.

We are interested in the following task: given a false posterior inference result (i.e. samples from $p_f(\theta|x^n)$, or some exact or approximate PDF), choose an arbitrary target prior $\pi(\theta)$ and efficiently sample from the associated target posterior $p(\theta|x^n)$—or, more generally, compute an expectation $\mu_h = \mathbb{E}_p[h(\theta)]$ for some test function $h(\theta)$ with respect to the target posterior.

7.3.1 *Importance Sampling and Prior Sensitivity*

We begin by describing an initial strategy, and existing work in a related task known as prior sensitivity analysis.

Suppose we have T false posterior samples $\{\tilde{\theta}_t\}_{t=1}^T \sim p_f(\theta|x^n)$. In importance sampling (IS), samples from an importance distribution are used to estimate the expectation of a test function with respect to a target distribution. A straightforward idea is to use the false posterior as an importance distribution, and compute the IS estimate

$$\hat{\mu}_h^{IS} = \sum_{t=1}^{T} w(\tilde{\theta}_t)h(\tilde{\theta}_t) \tag{69}$$

where the weight function $w(\theta) \propto \frac{p(\theta|x^n)}{p_f(\theta|x^n)} \propto \frac{\pi(\theta)}{\pi_f(\theta)}$, and the T weights are normalized to sum to one.

IS-based methods have been developed for the task of *prior sensitivity analysis* (PSA). In PSA, the goal is to determine how the posterior varies over a sequence of priors (e.g. over a parameterized family of priors $\pi(\theta;\gamma_i)$, $i = 0, 1, \ldots$). Existing work has proposed inferring a single posterior under prior $\pi(\theta;\gamma_0)$, and then using IS methods to infer further posteriors in the sequence [19, 30, 79].

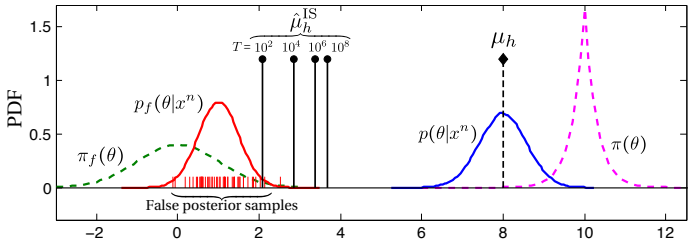
This strategy is effective when subsequent priors are similar enough, but breaks down when two priors are sufficiently dissimilar, or are from ill-matched parametric families, which we illustrate in an example below.

Note that, in general for IS, as $T \to \infty$, $\hat{\mu}_h^{IS} \to \mu_h$ almost surely. However, IS estimates can still fail in practice if $\hat{\mu}_h^{IS}$ has high or infinite variance. If so, the variance

of the weights $w(\tilde{\theta}_t)$ will be large (a problem often referred to as weight degeneracy), which can lead to inaccurate estimates. In our case, the variance of $\hat{\mu}_h^{\text{IS}}$ is only finite if

$$\mathbb{E}_{p_f}\left[h(\theta)^2 \frac{\pi(\theta)^2}{\pi_f(\theta)^2}\right] \propto \mathbb{E}_{p}\left[h(\theta)^2 \frac{\pi(\theta)}{\pi_f(\theta)}\right] < \infty. \tag{70}$$

For a broad class of h, this is satisfied if there exists $M \in \mathbb{R}$ such that $\frac{\pi(\theta)}{\pi_f(\theta)} < M, \forall\theta$ [71]. Given some pre-inferred $p_f(\theta|x^n)$ with false prior $\pi_f(\theta)$, the accuracy of IS thus depends on the target prior of interest. For example, if $\pi(\theta)$ has heavier tails than $\pi_f(\theta)$, the variance of $\hat{\mu}_h^{\text{IS}}$ will be infinite for many h. Intuitively, we expect the variance to be higher for π that are more dissimilar to π_f.



T	$\hat{\mu}_h^{\text{IS}}$	$\lvert\hat{\mu}_h^{\text{IS}} - \mu_h\rvert$	Wall time (s)
10^2	2.068	5.922	4.423
10^4	2.847	5.142	10.419
10^6	3.375	4.614	120.069
10^8	3.674	4.315	11889.733

IS estimate $\hat{\mu}_h^{\text{IS}}$ using $\{\tilde{\theta}_t\}_{t=1}^T \sim p_f(\theta|x^n)$.
(True value: $\mu_h = \mathbb{E}_p[h(\theta)] = 7.9892$.)

Figure 32: Importance sampling with false posterior samples. As the number of samples T grows, the difference between the IS estimate $\hat{\mu}_h^{\text{IS}}$ and the true value μ_h decreases increasingly slowly. The difference remains large even when $T = 10^8$. See text for analysis.

We show a concrete example of this in Fig. 32. Consider a normal model for data $x^n \sim \mathcal{N}(\theta, 1)$, with a standard normal false prior $\pi_f(\theta) = \mathcal{N}(\theta|0, 1)$. This yields a closed-form false posterior (due to the conjugate π_f), which is also normal. Suppose we'd like to estimate the posterior expectation under a Laplace target prior, with mean 10 and variance 1, for test function $h(\theta) = \theta$ (i.e. an estimate of the target posterior mean). We draw T false posterior samples $\{\tilde{\theta}_t\}_{t=1}^T \sim p_f(\theta|x^n)$, compute weights $w(\tilde{\theta}_t)$ and IS estimate $\hat{\mu}_h^{\text{IS}}$, and compare it with the true expectation μ_h.

We see in Fig. 32 that $|\mu_h - \hat{\mu}_h^{\text{IS}}|$ slows significantly as T increases, and maintains a high error even as T is made very large. We can analyze this issue theoretically. Suppose we want $|\mu_h - \hat{\mu}_h^{\text{IS}}| < \delta$. Since we know $p_f(\theta|x^n)$ is normal, we can compute a lower bound on the number of false posterior samples T that would be needed for the expected estimate to be within δ of μ_h. Namely, if $p_f(\theta|x^n) = \mathcal{N}(\theta|m, s^2)$, in order for $|\mu_h - \mathbb{E}_{p_f}[\hat{\mu}_h^{\text{IS}}]| < \delta$, we'd need

$$T \geqslant \exp\left\{\frac{1}{2s^2}(|\mu_h - m| - \delta)^2\right\}.$$

In the example in Fig. 32, we have $m = 1$, $s^2 = 0.25$, and $\mu_h = 7.9892$. Hence, for $|\mu_h - \mathbb{E}_{p_f}[\hat{\mu}_h^{\text{IS}}]| < 1$, we'd need $T > 10^{31}$ samples (see Sec. 7.6 for full details of this analysis). Note that this bound actually has nothing to do with the parametric

form of $\pi(\theta)$—it is based solely on the normal false posterior, and its distance to the target posterior mean μ_h. However, even if this distance was small, the importance estimate would still have infinite variance due to the Laplace target prior. Further, note that the situation can significantly worsen in higher dimensions, or if the false posterior has a lower variance.

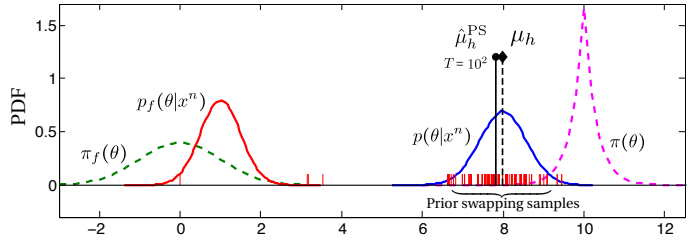
7.3.2 *Prior Swapping*

We'd like a method that will work well even when false and target priors $\pi_f(\theta)$ and $\pi(\theta)$ are significantly different, or are from different parametric families, with performance that does not worsen (in accuracy nor computational complexity) as the priors are made more dissimilar.

Redoing inference for each new target posterior can be very costly, especially when the data size n is large, because the per-iteration cost of most standard inference algorithms scales with n, and many iterations may be needed for accurate inference. This includes both MCMC and sequential monte carlo (SMC) algorithms (i.e. repeated-IS-methods that infer a sequence of distributions). In SMC, the per-iteration cost still scales with n, and the variance estimates can still be infinite if subsequent distributions are ill-matched.

Instead, we aim to leverage the inferred false posterior to more-efficiently compute any future target posterior. We begin by defining a *prior swap density* $p_s(\theta)$. Suppose for now that a false posterior inference algorithm has returned a density function $\tilde{p}_f(\theta)$ (we will give more details on $\tilde{p}_f$ later; assume for now that it is either equal to $p_f(\theta|x^n)$ or approximates it). We then define the prior swap density as

$$p_s(\theta) \propto \frac{\tilde{p}_f(\theta)\pi(\theta)}{\pi_f(\theta)}. \tag{71}$$



T	$\hat{\mu}_h^{\mathrm{PS}}$	$\lvert\hat{\mu}_h^{\mathrm{PS}} - \mu_h\rvert$	Wall time (s)
10^2	7.823	0.166	0.039
10^3	8.021	0.032	0.381
10^4	7.979	0.010	3.808
10^5	7.996	0.007	38.027

Sample estimate $\hat{\mu}_h^{\mathrm{PS}}$ using $\{\theta_t\}_{t=1}^T \sim p_s(\theta)$.
(True value: $\mu_h = \mathbb{E}_p[h(\theta)] = 7.9892$.)

Figure 33: Using prior swapping to compute estimate $\hat{\mu}_h^{\mathrm{PS}}$ by drawing samples $\{\theta_t\}_{t=1}^T \sim p_s(\theta)$.

Note that if $\tilde{p}_f(\theta) = p_f(\theta|x^n)$, then $p_s(\theta) = p(\theta|x^n)$. However, depending on how we represent $\tilde{p}_f(\theta)$, $p_s(\theta)$ can have a much simpler analytic representation than $p(\theta|x^n)$, which is typically defined via a likelihood function (i.e. a function of the data) and causes inference algorithms to have costs that scale with the data size n. Specifically, we will only use low-complexity $\tilde{p}_f(\theta)$ that can be evaluated in constant time with respect to the data size n.

Our general strategy is to use $p_s(\theta)$ as a surrogate for $p(\theta|x^n)$ in standard MCMC or optimization procedures, to yield data-independent algorithms with constant cost per iteration. Intuitively, the likelihood information is captured by the false posterior—we make use of this instead of the likelihood function, which is costly to evaluate.

More concretely, at each iteration in standard inference algorithms, we must evaluate a data-dependent function associated with the posterior density. For example, we evaluate a function proportional to $p(\theta|x^n)$ in Metropolis-Hastings (MH) [125], and $\nabla_\theta \log p(\theta|x^n)$ in gradient-based MCMC methods (such as Langevin dynamics (LD) [158] and Hamiltonian Monte Carlo (HMC) [134]) and in optimization procedures that yield a MAP point estimate. In prior swapping, we instead evaluate $p_s(\theta)$ in MH, or $\nabla_\theta \log p_s(\theta)$ in LD, HMC, or gradient optimization to a MAP estimate (see Sec. 7.7 for algorithm pseudocode). Here, each iteration only requires evaluating a few simple analytic expressions, and thus has $O(1)$ complexity with respect to data size.

We demonstrate prior swapping on our previous example (using a normal false prior and Laplace target prior) in Fig. 33, where we have a closed-form (normal PDF) $\tilde{p}_f(\theta)$. To do prior swapping, we run a Metropolis-Hastings algorithm on the target density $p_s(\theta)$. Note that drawing each sample in this Markov chain does not involve the data x^n, and can be done in constant time with respect to n (which we can see by viewing the wall time for different T). In Fig. 33, we draw T samples $\{\theta_t\}_{t=1}^T \sim p_s(\theta)$, compute a sample estimate $\hat{\mu}_h^{PS} = \frac{1}{T}\sum_{t=1}^T \theta_t$, and compare it with the true value μ_h. We see that $\hat{\mu}_h^{PS}$ converges to μ_h after a relatively small number of samples T.

7.3.3 *Prior Swapping with False Posterior Samples*

The previous method is only applicable if our false posterior inference result is a PDF $\tilde{p}_f(\theta)$ (such as in closed-form inference or variational approximations). Here, we develop prior swapping methods for the setting where we only have access to samples $\{\tilde{\theta}_t\}_{t=1}^{T_f} \sim p_f(\theta|x^n)$. We propose the following procedure:

1. Use $\{\tilde{\theta}_t\}_{t=1}^{T_f}$ to form an estimate $\tilde{p}_f(\theta) \approx p_f(\theta|x^n)$.
2. Sample from $p_s(\theta) \propto \frac{\pi(\theta)\tilde{p}_f(\theta)}{\pi_f(\theta)}$ with prior swapping, as before.

Note that, in general, $p_s(\theta)$ only approximates $p(\theta|x^n)$. As a final step, after sampling from $p_s(\theta)$, we can:

3. Apply a correction to samples from $p_s(\theta)$.

We will describe two methods for applying a correction to p_s samples—one involving importance sampling, and one involving semiparametric density estimation. Additionally, we will discuss forms for $\tilde{p}_f(\theta)$, guarantees about these forms, and how to optimize the choice of $\tilde{p}_f(\theta)$. In particular, we will argue why (in constrast to the initial IS strategy) these methods do not fail when $p(\theta|x^n)$ and $p_f(\theta|x^n)$ are very dissimilar or have ill-matching parametric forms.

PRIOR SWAP IMPORTANCE SAMPLING. Our first proposal for applying a correction to prior swap samples involves IS: after estimating some $\tilde{p}_f(\theta)$, and sampling $\{\theta_t\}_{t=1}^T \sim p_s(\theta)$, we can treat $\{\theta_t\}_{t=1}^T$ as importance samples, and compute the IS estimate

$$\hat{\mu}_h^{\text{PSis}} = \sum_{t=1}^{T} w(\theta_t) h(\theta_t) \tag{72}$$

where the weight function is now

$$w(\theta) \propto \frac{p(\theta|x^n)}{p_s(\theta)} \propto \frac{p_f(\theta|x^n)}{\tilde{p}_f(\theta)} \tag{73}$$

and the weights are normalized so that $\sum_{t=1}^T w(\theta_t) = 1$.

The key difference between this and the previous IS strategy is the weight function. Recall that, previously, an accurate estimate depended on the similarity between $\pi(\theta)$ and $\pi_f(\theta)$; both the distance to and parametric form of $\pi(\theta)$ could produce high or infinite variance estimates. This was an issue because we wanted the procedure to work well for any $\pi(\theta)$. Now, however, the performance depends on the similarity between $\tilde{p}_f(\theta)$ and $p_f(\theta|x^n)$—and by using the false posterior samples, we can estimate a $\tilde{p}_f(\theta)$ that well approximates $p_f(\theta|x^n)$. Additionally, we can prove that certain choices of $\tilde{p}_f(\theta)$ *guarantee* a finite variance IS estimate. Note that the variance of $\hat{\mu}_h^{\text{PSis}}$ is only finite if

$$\mathbb{E}_{p_f}\left[h(\theta)^2 \frac{p_f(\theta|x^n)^2}{\tilde{p}_f(\theta)^2}\right] \propto \mathbb{E}_p\left[h(\theta)^2 \frac{p_f(\theta|x^n)}{\tilde{p}_f(\theta)}\right] < \infty.$$

To bound this, it is sufficient to show that there exists $M \in \mathbb{R}$ such that $\frac{p_f(\theta|x^n)}{\tilde{p}_f(\theta)} < M$ for all θ (assuming a test function $h(\theta)$ with finite variance) [71]. To satisfy this condition, we will propose a certain parametric family $\tilde{p}_f^\alpha(\theta)$. Note that, to maintain a prior swapping procedure with $O(1)$ cost, we want a $\tilde{p}_f^\alpha(\theta)$ that can be evaluated in constant time. In general, a $\tilde{p}_f^\alpha(\theta)$ with fewer terms will yield a faster procedure. With these in mind, we propose the following family of densities.

Definition. For a parameter $\alpha = (\alpha_1, \ldots, \alpha_k)$, $\alpha_j \in \mathbb{R}^p$, $k > 0$, let density $\tilde{p}_f^\alpha(\theta)$ satisfy

$$\tilde{p}_f^\alpha(\theta) \propto \pi_f(\theta) \prod_{j=1}^{k} p(\alpha_j|\theta)^{n/k} \tag{74}$$

where $p(\alpha_j|\theta)$ denotes the model conditional PDF.

The number of terms in $\tilde{p}_f^\alpha(\theta)$ (and cost to evaluate) is determined by the parameter k. Note that this family is inspired by the true form of the false posterior $p_f(\theta|x^n)$. However, $\tilde{p}_f^\alpha(\theta)$ has constant-time evaluation, and we can estimate its parameter α using samples $\{\tilde{\theta}_t\}_{t=1}^{T_f} \sim p_f(\theta|x^n)$. Furthermore, we have the following guarantees.

Theorem 7.3.1. *For any* $\alpha = (\alpha_1, \ldots, \alpha_k) \subset \mathbb{R}^p$ *and* $k > 0$ *let* $\tilde{p}_f^{\alpha}(\theta)$ *be defined as in Eq. (74). Then, there exists* $M > 0$ *such that* $\frac{p_f(\theta|x^n)}{\tilde{p}_f^{\alpha}(\theta)} < M$, *for all* $\theta \in \mathbb{R}^d$.

Corollary 7.3.1.1. *For* $\{\theta_t\}_{t=1}^T \sim p_s^{\alpha}(\theta) \propto \frac{\tilde{p}_f^{\alpha}(\theta)\pi(\theta)}{\pi_f(\theta)}$, $w(\theta_t) = \frac{p_f(\theta_t|x^n)}{\tilde{p}_f^{\alpha}(\theta_t)} \left(\sum_{r=1}^T \frac{p_f(\theta_r|x^n)}{\tilde{p}_f^{\alpha}(\theta_r)}\right)^{-1}$, *and test function that satisfies* $Var_p[h(\theta)] < \infty$, *the variance of IS estimate* $\hat{\mu}_h^{PSis} = \sum_{t=1}^T h(\theta_t)w(\theta_t)$ *is finite.*

Proofs for these theorems are given in Sec. 7.8.

Note that we do not know the normalization constant for $\tilde{p}_f^{\alpha}(\theta)$. This is not an issue for its use in prior swapping, since we only need access to a function proportional to $p_s^{\alpha}(\theta) \propto \tilde{p}_f^{\alpha}(\theta)\pi(\theta)\pi_f(\theta)^{-1}$ in most MCMC algorithms. However, we still need to estimate α, which *is* an issue because the unknown normalization constant is a function of α. Fortunately, we can use the method of *score matching* [93] to estimate α given a density such as $\tilde{p}_f^{\alpha}(\theta)$ with unknown normalization constant.

Once we have found an optimal parameter α^*, we draw samples from $p_s^{\alpha^*}(\theta) \propto \tilde{p}_f^{\alpha^*}(\theta)\pi(\theta)\pi_f(\theta)^{-1}$, compute weights for these samples (Eq. (73)), and compute the IS estimate $\hat{\mu}_h^{PSis}$. We give pseudocode for the full prior swap importance sampling procedure in Alg. 9.

Algorithmus 9 : Prior Swap Importance Sampling from Neiswanger et al. [145]

Input : False posterior samples $\{\tilde{\theta}_t\}_{t=1}^{T_f} \sim p_f(\theta|x^n)$.
Output : IS estimate $\hat{\mu}_h^{PSis}$.

1 Score matching: estimate α^* using $\{\tilde{\theta}_t\}_{t=1}^{T_f}$.
2 Prior swapping: sample $\{\theta_t\}_{t=1}^T \sim p_s^{\alpha^*}(\theta) \propto \frac{\tilde{p}_f^{\alpha^*}(\theta)\pi(\theta)}{\pi_f(\theta)}$.
3 Importance sampling: compute $\hat{\mu}_h^{PSis} = \sum_{t=1}^T h(\theta_t)w(\theta_t)$.

SEMIPARAMETRIC PRIOR SWAPPING. In the previous method, we chose a parametric form for $\tilde{p}_f^{\alpha}(\theta)$; in general, even the optimal α will yield an inexact approximation to $p_f(\theta|x^n)$. Here, we aim to incorporate methods that return an increasingly exact estimate $\tilde{p}_f(\theta)$ when given more false posterior samples $\{\tilde{\theta}_t\}_{t=1}^{T_f}$.

One idea is to use a nonparametric kernel density estimate $\tilde{p}_f^{np}(\theta)$ and plug this into $p_s^{np}(\theta) \propto \tilde{p}_f^{np}(\theta)\pi(\theta)\pi_f(\theta)^{-1}$. However, nonparametric density estimates can yield inaccurate density tails and fare badly in high dimensions. To help mitigate these problems, we turn to a semiparametric estimate, which begins with a parametric estimate, and adjusts it as samples are generated. In particular, we use a density estimate that can be viewed as the product of a parametric density estimate and a nonparametric correction function [83]. This density estimate is consistent as the number of samples $T_f \to \infty$. Instead of (or in addition to) correcting prior swap samples with importance sampling, we can correct them by updating the nonparametric correction function as we continue to generate false posterior samples.

Given T_f samples $\{\tilde{\theta}_t\}_{t=1}^{T_f} \sim p_f(\theta|x^n)$, we write the semiparametric false posterior estimate as

$$\tilde{p}_f^{sp}(\theta) = \frac{1}{T_f}\sum_{t=1}^{T_f}\left[\frac{1}{b^d}K\left(\frac{\|\theta-\tilde{\theta}_t\|}{b}\right)\frac{\tilde{p}_f^{\alpha}(\theta)}{\tilde{p}_f^{\alpha}(\tilde{\theta}_t)}\right], \tag{75}$$

where K denotes a probability density kernel, with bandwidth b, where $b \to 0$ as $T_f \to \infty$ (see [196] for details on probability density kernels and bandwidth selection). The semiparametric prior swap density is then

$$\begin{aligned} p_s^{sp}(\theta) \propto & \frac{\tilde{p}_f^{sp}(\theta)\pi(\theta)}{\pi_f(\theta)} = \frac{1}{T_f}\sum_{t=1}^{T_f}\frac{K\left(\frac{\|\theta-\tilde{\theta}_t\|}{b}\right)\tilde{p}_f^{\alpha}(\theta)\pi(\theta)}{\tilde{p}_f^{\alpha}(\tilde{\theta}_t)\pi_f(\theta)b^d} \\ & \propto [p_s^{\alpha}(\theta)]\left[\frac{1}{T_f}\sum_{t=1}^{T_f}\frac{K\left(\frac{\|\theta-\tilde{\theta}_t\|}{b}\right)}{\tilde{p}_f^{\alpha}(\tilde{\theta}_t)}\right]. \end{aligned} \tag{76}$$

Hence, the prior swap density $p_s^{sp}(\theta)$ is proportional to the product of two densities: the parametric prior swap density $p_s^{\alpha}(\theta)$, and a correction density. To estimate expectations with respect to $p_s^{sp}(\theta)$, we can follow Alg. 9 as before, but replace the weight function in the final IS estimate with

$$w(\theta) \propto \frac{p_s^{sp}(\theta)}{p_s^{\alpha}(\theta)} \propto \frac{1}{T_f}\sum_{t=1}^{T_f}\frac{K\left(\frac{\|\theta-\tilde{\theta}_t\|}{b}\right)}{\tilde{p}_f^{\alpha}(\tilde{\theta}_t)}. \tag{77}$$

One advantage of this strategy is that computing the weights doesn't require the data—it thus has constant cost with respect to data size n (though its cost does increase with the number of false posterior samples T_f). Additionally, as in importance sampling, we can prove that this procedure yields an exact estimate of $\mathbb{E}[h(\theta)]$, asymptotically, as $T_f \to \infty$ (and we can provide an explicit bound on the rate at which $p_s^{sp}(\theta)$ converges to $p(\theta|x^n)$). We do this by showing that $p_s^{sp}(\theta)$ is consistent for $p(\theta|x^n)$.

Theorem 7.3.2. *Given false posterior samples* $\{\tilde{\theta}_t\}_{t=1}^{T_f} \sim p_f(\theta|x^n)$ *and* $b \asymp T_f^{-1/(4+d)}$, *the estimator* p_s^{sp} *is consistent for* $p(\theta|x^n)$, *i.e. its mean-squared error satisfies*

$$\sup_{p(\theta|x^n)} \mathbb{E}\left[\int (p_s^{sp}(\theta) - p(\theta|x^n))^2 \, d\theta\right] < \frac{c}{T_f^{4/(4+d)}}$$

for some $c > 0$ *and* $0 < b \leqslant 1$.

The proof for this theorem is given in Sec. 7.8.

7.4 EMPIRICAL RESULTS

We show empirical results on Bayesian generalized linear models (including linear and logistic regression) with sparsity and heavy tailed priors, and on latent factor models (including mixture models and topic models) with relational priors over factors (e.g. diversity-encouraging, agglomerate-encouraging, etc.). We aim to demonstrate empirically that prior swapping efficiently yields correct samples and, in some cases, allows us to apply certain inference algorithms to more-complex models than was previously possible. In the following experiments, we will refer to the following procedures:

- **Target posterior inference:** some standard inference algorithm (e.g. MCMC) run on $p(\theta|x^n)$.
- **False posterior inference:** some standard inference algorithm run on $p_f(\theta|x^n)$.
- **False posterior IS:** IS using samples from $p_f(\theta|x^n)$.
- **Prior swap exact:** prior swapping with closed-form $\tilde{p}_f(\theta) = p_f(\theta|x^n)$.
- **Prior swap parametric:** prior swapping with parametric $\tilde{p}_f^{\alpha}(\theta)$ given by Eq. (74).
- **Prior swap IS:** correcting samples from $\tilde{p}_f^{\alpha}(\theta)$ with IS.
- **Prior swap semiparametric:** correcting samples from $\tilde{p}_f^{\alpha}(\theta)$ with the semiparametric estimate IS procedure.

To assess performance, we choose a test function $h(\theta)$, and compute the Euclidean distance between $\mu_h = \mathbb{E}_p[h(\theta)]$ and some estimate $\hat{\mu}_h$ returned by a procedure. We denote this performance metric by *posterior error* $= \|\mu_h - \hat{\mu}_h\|_2$. Since μ_h is typically not available analytically, we run a single chain of MCMC on the target posterior for one million steps, and use these samples as ground truth to compute μ_h. For timing plots, to assess error of a method at a given time point, we collect samples drawn before this time point, remove the first quarter as burn in, and add the time it takes to compute any of the corrections.

7.4.1 *Sparsity Inducing and Heavy Tailed Priors in Bayesian Generalized Linear Models*

Sparsity-encouraging regularizers have gained a high level of popularity over the past decade due to their ability to produce models with greater interpretability and parsimony. For example, the L_1 norm has been used to induce sparsity with great effect [183], and has been shown to be equivalent to a mean-zero independent Laplace prior [165, 183]. In a Bayesian setting, inference given a sparsity prior can be difficult, and often requires a computationally intensive method (such as MH or HMC) or posterior approximations (e.g. expectation propagation [126]) that make factorization or parametric assumptions [70, 165]. We propose a cheap yet accurate solution: first get an inference result with a more-tractable prior (such as a normal prior),

and then use prior swapping to quickly convert the result to the posterior given a sparsity prior.

Our first set of experiments are on Bayesian linear regression models, which we can write as $y_i = X_i\theta + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma^2)$, $\theta \sim \pi$, $i = 1,...,n$. For π, we compute results on Laplace, Student's t, and VerySparse (with PDF $\text{VerySparse}(\sigma) = \prod_{i=1}^{d} \frac{1}{2\sigma} \exp\{-|\theta_i|^{0.4}/\sigma\}$ [165]) priors. Here, a normal π_f is conjugate and allows for exact false posterior inference. Our second set of experiments are on Bayesian logistic regression models, which we write as $y_i \sim \text{Bern}(p_i)$, $p_i = \text{logistic}(X_i\theta)$, $\theta \sim \pi$, $i = 1,...,n$. which we will pair with both heavy tailed priors and a hierarchical target prior $\pi = \mathcal{N}(0, \alpha^{-1}I)$, $\alpha \sim \text{Gamma}(\gamma, 1)$. For these experiments, we also use a normal π_f. However, this false prior is no longer conjugate, and so we use MCMC to sample from $p_f(\theta|x^n)$.

For linear regression, we use the YearPredictionMSD data set[1], ($n = 515345$, $d = 90$), in which regression is used to predict the year associated with a a song, and for logistic regression we use the MiniBooNE particle identification data set[2], ($n = 130065$, $d = 50$), in which binary classification is used to distinguish particles.

In Fig. 34, we compare prior swapping and IS methods, in order to show that the prior swapping procedures yield accurate posterior estimates, and to compare their speeds of convergence. We plot posterior error vs. wall time for each method's estimate of the posterior mean $\mathbb{E}_p[h(\theta)] = \mathbb{E}_p[\theta]$ for two sparsity target priors (Laplace and VerySparse), for both linear and logistic regression. In linear regression (only), since the normal conjugate π_f allows us to compute a closed form $p_f(\theta|x^n)$, we can run the *prior swap exact* method, where $\tilde{p}_f(\theta) = p_f(\theta|x^n)$. However, we can also sample from $p_f(\theta|x^n)$ to compute $\tilde{p}_f^{\alpha^*}(\theta)$, and therefore compare methods such as *prior swap parametric* and the two correction methods. In logistic regression, we do not have a closed form $p_f(\theta|x^n)$; here, we only compare the methods that make use of samples from $p_f(\theta|x^n)$. In Fig. 34, we see that the prior swapping methods (particularly prior swap IS) quickly converge to nearly zero posterior error. Additionally, in linear regression, we see that prior swap parametric, using $\tilde{p}_f(\theta) = \tilde{p}_f^{\alpha^*}(\theta)$, yields similar posterior error as prior swap exact, which uses $\tilde{p}_f(\theta) = p(\theta|x^n)$.

In Fig. 35, we show how prior swapping can be used for fast inference in Bayesian linear models with sparsity or heavy-tailed priors. We plot the time needed to first compute the false posterior (via exact inference) and then run prior swapping (via the MH procedure) on some target posterior, and compare this with the MH algorithm run directly on the target posterior. In (a) and (b) we show convergence plots and see that prior swapping performs faster inference (by a few orders of magnitude) than direct MH. In plot (b) we reduce the variance of the target prior; while this hurts the accuracy of false posterior IS, prior swapping still quickly converges to zero error. In (c) we show 1-d density marginals as we increase the prior sparsity, and in (d) we show prior swapping results for various sparsity priors.

1 https://archive.ics.uci.edu/ml/datasets/YearPredictionMSD
2 https://archive.ics.uci.edu/ml/datasets/MiniBooNE+particle+identification

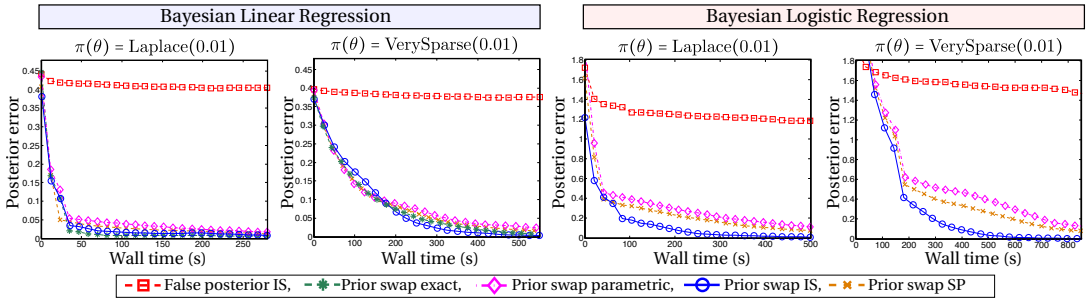


Figure 34: Comparison of prior swapping and IS methods for Bayesian linear and logistic regression under Laplace and VerySparse target priors. The prior swapping methods (particularly prior swap exact and prior swap IS) quickly converge to low posterior errors.

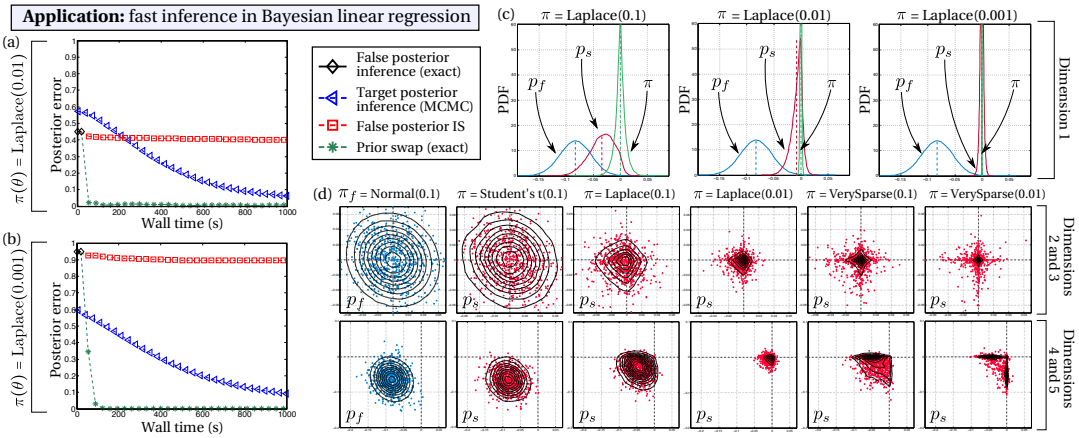


Figure 35: Prior swapping for fast inference in Bayesian linear models with sparsity and heavy-tailed priors: (a-b) Convergence plots showing that prior swapping performs accurate inference faster than the comparison methods and is robust to changing π. (c) Inferred 1-d density marginals when prior sparsity is increased. (d) Prior swapping results for a variety of different sparsity priors.

In Sec. 7.5, we also include results on logistic regression with the hierarchical target prior, as well as results for synthetic data where we are able to compare timing and posterior error as we tune n and d.

7.4.2 *Priors over Factors in Latent Variable Models*

Many latent variable models in machine learning—such as mixture models, topic models, probabilistic matrix factorization, and others—involve a set of latent factors (e.g. components or topics). Often, we'd like to use priors that encourage interesting behaviors among the factors. For example, we might want dissimilar factors through a diversity-promoting prior [107, 205] or for the factors to show some sort of sparsity

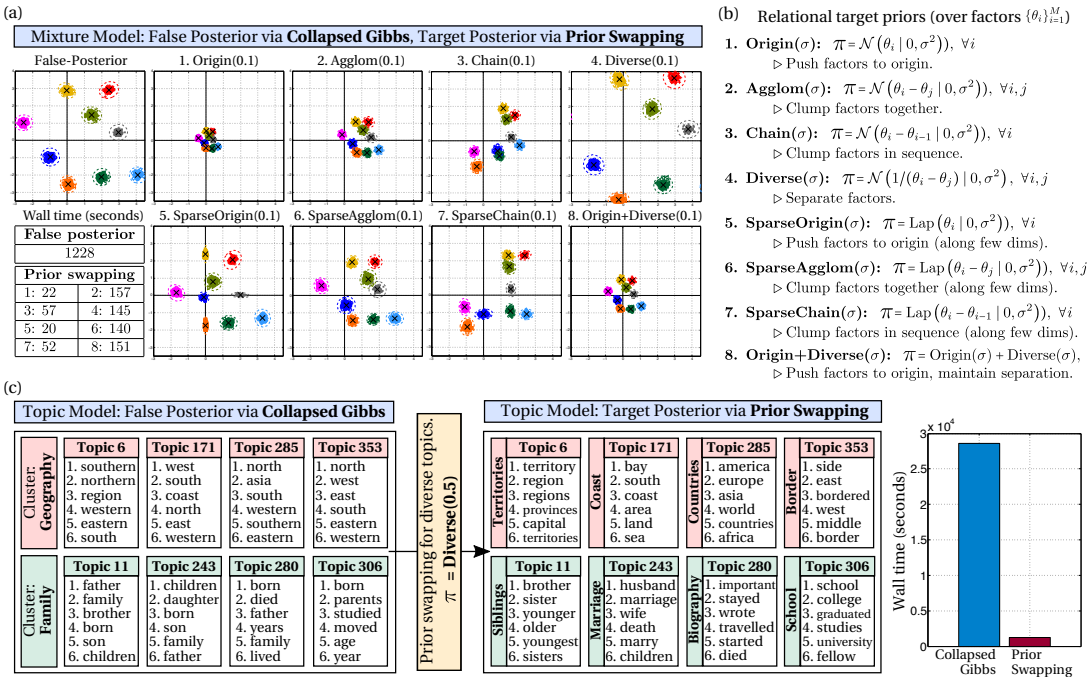


Figure 36: Latent factor models: (a) Prior swapping results for relational target priors (defined in (b)) over components in a mixture model. (c) Prior swapping with a diversity-promoting target prior on an LDA topic model (Simple English Wikipedia corpus) to separate redundant topic clusters; the top 6 words per topic are shown. In (a, c) we show wall times for the initial inference and prior swapping.

pattern [101, 122]. Inference in such models is often computationally expensive or designed on a case-by-case basis [101, 205].

However, when conjugate priors are placed over the factor parameters, collapsed Gibbs sampling can be applied. In this method, the factor parameters are integrated out, leaving only a subset of variables; on these, the conditional distributions can be computed analytically, which allows for Gibbs sampling over these variables. Afterwards, samples of the collapsed factor parameters can be computed. Hence, we propose the following strategy: first, assign a prior for the factor parameters that allows for collapsed Gibbs sampling; afterwards, reconstruct the factor samples and apply prior swapping for more complex relational priors over the factors. We can thus perform convenient inference in the collapsed model, yet apply more-sophisticated priors to variables in the uncollapsed model.

We first show results on a Gaussian mixture model (GMM), written $x_i \sim \mathcal{N}(\mu_{z_i}, \Sigma_{z_i})$, $z_i \sim \text{Dir}(\alpha)$, $\{\mu_m\}_{m=1}^M \sim \pi$, $i = 1,...,n$. Using a normal π_f over $\{\mu_m\}_{m=1}^M$ allows for collapsed Gibbs sampling. We also show results on a topic model (latent Dirichlet allocation (LDA) [28]) for text data (for the form of this model, see [28, 191]). Here, using a Dirichlet π_f over topics allows for collapsed Gibbs sampling. For mixture

models, we generate synthetic data from the above model (n=10,000, d=2, M=9), and for topic models, we use the Simple English Wikipedia[3] corpus (n=27,443 documents, vocab=10,192 words), and set M=400 topics.

In Fig. 36, we show results for mixture and topic models. In (a) we show inferred posteriors over GMM components for a number of relational target priors, which we define in (b). In (c), we apply the diversity-promoting target prior to LDA, to separate redundant topics. Here, we show two topic clusters ("geography" and "family") in $p_f(\theta|x^n)$, which are separated into distinct, yet thematically-similar, topics after prior swapping. In (a) and (c) we also show wall times of the inference methods.

7.5 FURTHER EMPIRICAL RESULTS

Here we show further empirical results on a logistic regression model with hierarchical target prior given by $\pi = \mathcal{N}(0, \alpha^{-1}I)$, $\alpha \sim \text{Gamma}(\gamma, 1)$. We use synthetic data so that we are able to compare the timing and posterior error of different methods as we tune n and d.

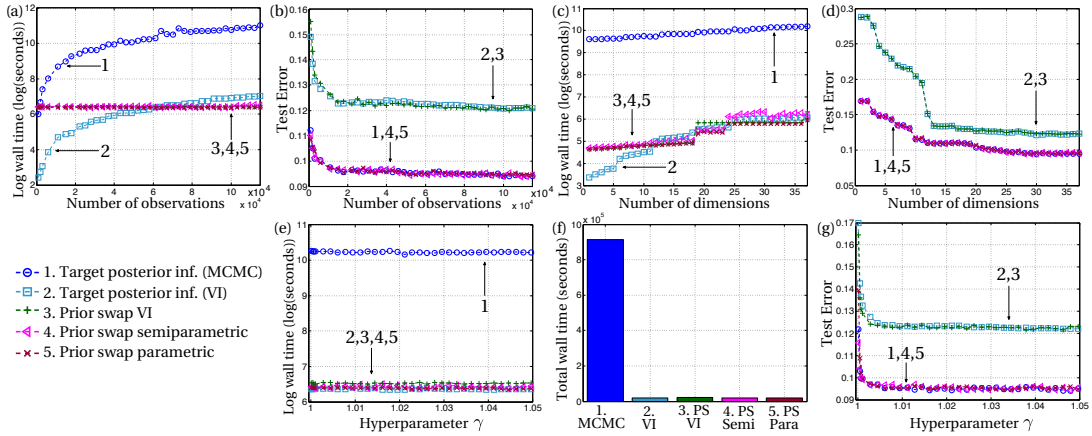


Figure 37: Bayesian hierarchical logistic regression: (a-b) Wall time and test error comparisons for varying data size n. As n is increased, wall time remains constant for prior swapping but grows for standard inference methods. (c-d) Wall time and test error comparisons for varying model dimensionality d. (e-g) Wall time and test error comparisons for inferences on a set of prior hyperparameters $\gamma \in [1, 1.05]$. Here, a single false posterior $\tilde{p}_f(\theta)$ (computed at $\gamma = 1.025$) is used for prior swapping on all other hyperparameters.

In this experiment, we assume that we are given samples from a false posterior $p_f(\theta|x^n)$, and we want to most-efficiently compute the target posterior under prior $\pi(\theta)$. In addition to the prior swapping methods, we can run standard iterative inference algorithms, such as MCMC or variational inference (VI), on the target posterior (initializing them, for example, at the false posterior mode) as comparisons. The

3 https://simple.wikipedia.org/

following experiments aim to show that, once the data size n grows large enough, prior swapping methods become more efficient than standard inference algorithms. They also aim to show that the held-out test error of prior swapping matches that of these standard inference algorithms. In these experiments, we also add a prior swap method called *prior swapping VI*; this method involves making a VI approximation to $p_f(\theta|x^n)$, and using it for $\tilde{p}_f(\theta)$. Prior swapping VI allows us to see whether the test error is similar to standard VI inference algorithms, which compute some approximation to the posterior. Finally, we show results over a range of target prior hyperparameter values γ to show that prior swapping maintains accuracy (i.e. has a similar error as standard inference algorithms) over the full range.

We show results in Fig. 37. In (a) and (b) we vary the number of observations (n=10-120,000) and see that prior swapping has a constant wall time while the wall times of both MCMC and VI increase with n. In (b) we see that the prior swapping methods achieve the same test error as the standard inference methods. In (c) and (d) we vary the number of dimensions (d=1-40). In this case, all methods have increasing wall time, and again the test errors match. In (e), (f), and (g), we vary the prior hyperparameter (γ=1-1.05). For prior swapping, we infer a single $\tilde{p}_f(\theta)$ (using $\gamma = 1.025$) with both MCMC and VI applied to $p_f(\theta|x^n)$, and compute *all other* hyperparameter results using this $\tilde{p}_f(\theta)$. This demonstrates that prior swapping can quickly infer correct results over a range of hyperparameters. Here, the prior swapping semiparametric method matches the test error of MCMC slightly better than the parametric method.

7.6 DETAILS ON THE IS EXAMPLE

Here we provide details on the IS example (for a normal π_f and Laplace π) given in Sec. 7.3.1.

We made the following statement: if $p_f(\theta|x^n) = \mathcal{N}(\theta|m, s^2)$, in order for $|\mu_h - \mathbb{E}_{p_f}[\hat{\mu}_h^{IS}]| < \delta$, we need

$$T \geqslant \exp\left\{\frac{1}{2s^2}(|\mu_h - m| - \delta)^2\right\}.$$

To show this, we first give an upper bound on the expected value of the maximum of T zero-mean s^2-variance Gaussian random variables. Let $\{\tilde{\theta}_t\}_{t=1}^T \sim g$, where $g(\theta) = \mathcal{N}(\theta|0, s^2)$, and let $Z = \max_t\{\tilde{\theta}_t\}_{t=1}^T$. Then, for some $b > 0$,

$$\exp\{b\mathbb{E}_g[Z]\} \leqslant \mathbb{E}_g[\exp\{bZ\}] = \mathbb{E}_g\left[\max_t\left\{\exp\{b\tilde{\theta}_t\}\right\}_{t=1}^T\right] \leqslant \sum_{t=1}^T \mathbb{E}_g\left[\exp\{b\tilde{\theta}_t\}\right] = T\exp\{b^2s^2/2\},$$

where the first inequality is due to Jensen's inequality, and the final equality is due to the definition of a Gaussian moment generating function. The above implies that

$$\mathbb{E}_g[Z] \leqslant \frac{\log T}{b} + \frac{bs^2}{2}.$$

Setting $b = \sqrt{\frac{2}{s^2}\log T}$, we have that

$$\mathbb{E}_g\left[\max_t\{\tilde{\theta}_t\}_{t=1}^T\right] = \mathbb{E}_g[Z] \leqslant s\sqrt{2\log T}.$$

However, note that for all $\{\tilde{\theta}_t\}_{t=1}^T$, and weights $\{w(\tilde{\theta}_t)\}_{t=1}^T$ (such that $\sum_{t=1}^T w(\tilde{\theta}_t) = 1$), the IS estimate $\hat{\mu}_h^{IS}$ for $h(\theta) = \theta$ must be less than or equal to $\max_t\{\tilde{\theta}_t\}_{t=1}^T$ (since the weighted average of $\{\tilde{\theta}_t\}_{t=1}^T$ cannot be larger than the maximum of this set). Therefore,

$$\mathbb{E}_g\left[\hat{\mu}_h^{IS}\right] \leqslant \mathbb{E}_g\left[\max_t\{\tilde{\theta}_t\}_{t=1}^T\right] \leqslant s\sqrt{2\log T},$$

and equivalently

$$T \geqslant \exp\left\{\frac{1}{2s^2}\mathbb{E}_g\left[\hat{\mu}_h^{IS}\right]^2\right\}.$$

In our example, we wanted the expected estimate to be within δ of μ_h, i.e. we wanted $|\mu_h - \mathbb{E}_g[\hat{\mu}_h^{IS}]| < \delta \iff \delta - \mu_h \leqslant \mathbb{E}_g[\hat{\mu}_h^{IS}] \leqslant \mu_h + \delta$, and therefore,

$$T \geqslant \exp\left\{\frac{1}{2s^2}\mathbb{E}_g\left[\hat{\mu}_h^{IS}\right]^2\right\} \geqslant \exp\left\{\frac{1}{2s^2}(\delta - \mu_h)^2\right\}.$$

Finally, notice that the original statement involved samples $\{\tilde{\theta}_t\}_{t=1}^T \sim p_f(\theta|x^n) = \mathcal{N}(m, s^2)$ (instead of from $g = \mathcal{N}(0, s^2)$). But this is equivalent to setting $p_f(\theta|x^n) = g(\theta)$, and shifting our goal so that we want $\delta - |\mu_h - m| \leqslant \mathbb{E}_{p_f}[\hat{\mu}_h^{IS}] \leqslant |\mu_h - m| + \delta$. This gives us the desired bound:

$$T \geqslant \exp\left\{\frac{1}{2s^2}\mathbb{E}_{p_f}\left[\hat{\mu}_h^{IS}\right]^2\right\} \geqslant \exp\left\{\frac{1}{2s^2}(\delta - |\mu_h - m|)^2\right\}.$$

7.7 PRIOR SWAPPING PSEUDOCODE (FOR A FALSE POSTERIOR PDF INFERENCE RESULT $\tilde{p}_f(\theta)$)

Here we give pseudocode for the prior swapping procedure, given some false posterior PDF inference result $\tilde{p}_f(\theta)$, using the prior swap functions $p_s(\theta) \propto \frac{\tilde{p}_f(\theta)\pi(\theta)}{\pi_f(\theta)}$ and $\nabla_\theta \log p_s(\theta) \propto \nabla_\theta \log \tilde{p}_f(\theta) + \nabla_\theta \log \pi(\theta) - \nabla_\theta \log \pi_f(\theta)$, as described in Sec. 7.3.2.

In Alg. 10, we show prior swapping via the Metropolis-Hastings algorithm, which makes repeated use of $p_s(\theta)$. In Alg. 11 we show prior swapping via Hamiltonian Monte Carlo, which makes repeated use of $\nabla_\theta \log p_s(\theta)$. A special case of Alg. 11, which occurs when we set the number of simulation steps to $L = 1$ (in line 6), is prior swapping via Langevin dynamics.

Algorithmus 10 : Prior swapping via Metropolis-Hastings from Neiswanger et al. [145]

Input : Prior swap function $p_s(\theta)$, and proposal q.

Output : Samples $\{\theta_t\}_{t=1}^{T} \sim p_s(\theta)$ as $T \to \infty$.

Initialize θ_0. ▷ Initialize Markov chain.
for $t = 1, \ldots, T$ **do**
  Draw $\theta_s \sim q(\theta_s \mid \theta_{t-1})$. ▷ Propose new sample.
  Draw $u \sim \text{Unif}([0, 1])$.
  if $u < \min\left\{1, \frac{p_s(\theta_s)q(\theta_t|\theta_s)}{p_s(\theta_t)q(\theta_s|\theta_t)}\right\}$ **then**
    Set $\theta_t \leftarrow \theta_s$. ▷ Accept proposed sample.
  else
    Set $\theta_t \leftarrow \theta_{t-1}$. ▷ Reject proposed sample.

Algorithmus 11 : Prior swapping via Hamiltonian Monte Carlo from Neiswanger et al. [145]

Input : Prior swap function $p_s(\theta)$, its gradient-log $\nabla_\theta \log p_s(\theta)$, and step-size ϵ.

Output : Samples $\{\theta_t\}_{t=1}^{T} \sim p_s(\theta)$ as $T \to \infty$.

Initialize θ_0. ▷ Initialize Markov chain.
for $t = 1, \ldots, T$ **do**
  Draw $r_t \sim \mathcal{N}(0, I)$.
  Set $(\widetilde{\theta}_0, \widetilde{r}_0) \leftarrow (\theta_{t-1}, r_{t-1})$
  Set $\widetilde{r}_0 \leftarrow \widetilde{r}_0 + \frac{\epsilon}{2}\nabla_\theta \log p_s(\widetilde{\theta}_0)$. ▷ Propose new sample (next 4 lines).
  for $l = 1, \ldots, L$ **do**
    Set $\widetilde{\theta}_l \leftarrow \widetilde{\theta}_{l-1} + \epsilon\widetilde{r}_{l-1}$.
    Set $\widetilde{r}_l \leftarrow \widetilde{r}_{l-1} + \epsilon\nabla_\theta \log p_s(\widetilde{\theta}_l)$.
  Set $\widetilde{r}_L \leftarrow \widetilde{r}_L + \frac{\epsilon}{2}\nabla_\theta \log p_s(\widetilde{\theta}_L)$.
  Draw $u \sim \text{Unif}([0, 1])$.
  if $u < \min\left\{1, \frac{p_s(\widetilde{\theta}_L)\widetilde{r}_L^\top \widetilde{r}_L}{p_s(\theta_{t-1})r_{t-1}^\top r_{t-1}}\right\}$ **then**
    Set $\theta_t \leftarrow \widehat{\theta}_L$. ▷ Accept proposed sample.
  else
    Set $\theta_t \leftarrow \theta_{t-1}$. ▷ Reject proposed sample.

7.8 PROOFS OF THEORETICAL GUARANTEES

Here, we prove the theorems stated in Sec. 7.3.3.

Throughout this analysis, we assume that we have T samples $\{\tilde{\theta}_t\}_{t=1}^{T_f} \subset \mathcal{X} \subset \mathbb{R}^d$ from the false-posterior $p_f(\theta|x^n)$, and that $b \in \mathbb{R}_+$ denotes the bandwidth of our semiparametric false-posterior density estimator $\tilde{p}_f^{sp}(\theta)$. Let Hölder class $\Sigma(2, L)$ on $\mathcal{X}$ be defined as the set of all $\ell = \lfloor 2 \rfloor$ times differentiable functions $f : \mathcal{X} \to \mathbb{R}$ whose derivative $f^{(l)}$ satisfies

$$|f^{(\ell)}(\theta) - f^{(\ell)}(\theta')| \leqslant L \left|\theta - \theta'\right|^{2-\ell} \quad \text{for all } \theta, \theta' \in \mathcal{X}.$$

Let the class of densities $\mathcal{P}(2, L)$ be

$$\mathcal{P}(2, L) = \left\{ f \in \Sigma(2, L) \;\middle|\; f \geqslant 0, \int f(\theta) d\theta = 1 \right\}.$$

Let data $x^n = \{x_1, \ldots, x_n\} \subset \mathcal{Y} \subset \mathbb{R}^p$, let $\mathcal{Z} \subset \mathcal{Y}$ be any set such that $x^n \subset \mathcal{Z}$, and let $\mathcal{F}_{\mathcal{Z}}(L)$ denote the set of densities $p : \mathcal{Y} \to \mathbb{R}$ that satisfy

$$|\log p(x) - \log p(x')| \leqslant L|x - x'|, \quad \text{for all } x, x' \in \mathcal{Z}.$$

In the following theorems, we assume that the false-posterior density $p_f(\theta|x^n)$ is bounded, i.e. that there exists some $B > 0$ such that $p_f(\theta|x^n) \leqslant B$ for all $\theta \in \mathbb{R}^d$; that the prior swap density $p_s(\theta) \in \mathcal{P}(2, L)$; and that the model family $p(x^n|\theta) \in \mathcal{F}_{\mathcal{Z}}(L)$ for some $\mathcal{Z}$.

Theorem 2.1. *For any $\alpha = (\alpha_1, \ldots, \alpha_k) \subset \mathbb{R}^p$ and $k > 0$ let $\tilde{p}_f^{\alpha}(\theta)$ be defined as in Eq. (74). Then, there exists $M > 0$ such that $\frac{p_f(\theta|x^n)}{\tilde{p}_f^{\alpha}(\theta)} < M$, for all $\theta \in \mathbb{R}^d$.*

Proof. To prove that there exists $M > 0$ such that $\frac{p_f(\theta|x^n)}{\tilde{p}_f^{\alpha}(\theta)} < M$, note that the false posterior can be written

$$p_f(\theta|x^n) = \frac{1}{Z_1} \pi_f(\theta) \prod_{i=1}^{n} L(\theta|x_i) = \frac{1}{Z_1} \pi_f(\theta) \prod_{i=1}^{n} p(x_i|\theta),$$

and the parametric estimate $\tilde{p}_f^{\alpha}(\theta)$ is defined to be

$$\tilde{p}_f^{\alpha}(\theta) = \frac{1}{Z_2} \pi_f(\theta) \prod_{j=1}^{k} p(\alpha_j|\theta)^{n/k}.$$

Let $d = \max_{i,j} |x_i - \alpha_j|$. For any $i \in \{1, \ldots, n\}$, $j \in \{1, \ldots, k\}$,

$$|\log p(x_i|\theta) - \log p(\alpha_j|\theta)| \leqslant Ld \implies \left|\log \frac{p(x_i|\theta)}{p(\alpha_j|\theta)}\right| \leqslant Ld,$$

and

$$\exp\left\{\log \frac{p(x_i|\theta)}{p(\alpha_j|\theta)}\right\} \leqslant \exp\left\{\left|\log \frac{p(x_i|\theta)}{p(\alpha_j|\theta)}\right|\right\} \leqslant \exp\{Ld\} \implies \frac{p(x_i|\theta)}{p(\alpha_j|\theta)} \leqslant \exp\{Ld\}.$$

Therefore

$$\frac{p_f(\theta|x^n)}{\tilde{p}_f^{\alpha}(\theta)} \leqslant \frac{Z_2}{Z_1} \frac{\prod_{i=1}^{n} p(x_i|\theta)}{\prod_{j=1}^{k} p(\alpha_j|\theta)^{n/k}} \leqslant \frac{Z_2}{Z_1} \exp\{nLd\} = M.$$

□

Corollary 2.1.1. *For* $\{\theta_t\}_{t=1}^{T} \sim p_s^{\alpha}(\theta) \propto \frac{\tilde{p}_f^{\alpha}(\theta)\pi(\theta)}{\pi_f(\theta)}$, $w(\theta_t) = \frac{p_f(\theta_t|x^n)}{\tilde{p}_f^{\alpha}(\theta_t)} \left(\sum_{r=1}^{T} \frac{p_f(\theta_r|x^n)}{\tilde{p}_f^{\alpha}(\theta_r)}\right)^{-1}$, *and test function that satisfies* $Var_p[h(\theta)] < \infty$, *the variance of IS estimate* $\hat{\mu}_h^{PSis} = \sum_{t=1}^{T} h(\theta_t) w(\theta_t)$ *is finite.*

Proof. This follows directly from the sufficient conditions for finite variance IS estimates given by [71], which we have proved are satisfied for $\hat{\mu}_h^{PSis}$ in Theorem 2.1. □

Theorem 2.2. *Given false posterior samples* $\{\tilde{\theta}_t\}_{t=1}^{T_f} \sim p_f(\theta|x^n)$ *and* $b \asymp T_f^{-1/(4+d)}$, *the estimator* p_s^{sp} *is consistent for* $p(\theta|x^n)$, *i.e. its mean-squared error satisfies*

$$\sup_{p(\theta|x^n)\in\mathcal{P}(2,L)} \mathbb{E}\left[\int (p_s^{sp}(\theta) - p(\theta|x^n))^2 \, d\theta\right] < \frac{c}{T_f^{4/(4+d)}}$$

for some $c > 0$ *and* $0 < b \leqslant 1$.

Proof. To prove mean-square consistency of our semiparametric prior swap density estimator p_s^{sp}, we give a bound on the mean-squared error (MSE), and show that it tends to zero as we increase the number of samples T_f drawn from the false-posterior. To prove this, we bound the bias and variance of the estimator, and use this to bound the MSE. In the following, to avoid cluttering notation, we will drop the subscript p_f in $\mathbb{E}_{p_f}[\cdot]$.

We first bound the bias of our semiparametric prior swap estimator. For any $p(\theta|x^n) \in \mathcal{P}(2, L)$, we can write the bias as

$$\begin{aligned}
|\mathbb{E}[p_s^{sp}(\theta)] - p(\theta|x^n)| &= c_1 \left|\mathbb{E}\left[\tilde{p}_f^{sp}(\theta)\frac{\pi(\theta)}{\pi_f(\theta)}\right] - p_f(\theta|x^n)\frac{\pi(\theta)}{\pi_f(\theta)}\right| \\
&= c_2 \left|\frac{\pi(\theta)}{\pi_f(\theta)}\mathbb{E}\left[\tilde{p}_f^{sp}(\theta)\right] - p_f(\theta|x^n)\right| \\
&= c_3 \left|\mathbb{E}\left[\tilde{p}_f^{sp}(\theta)\right] - p_f(\theta|x^n)\right| \\
&\leqslant c h^2
\end{aligned}$$

for some $c > 0$, where we have used the fact that $\left|\mathbb{E}\left[\tilde{p}_f^{sp}(\theta)\right] - p_f(\theta|x^n)\right| \leqslant \tilde{c}h^2$ for some $\tilde{c} > 0$ (given in [83, 196]).

We next bound the variance of our semiparametric prior swap estimator. For any $p(\theta|x^n) \in \mathcal{P}(2, L)$, we can write the variance of our estimator as

$$\begin{aligned}\text{Var}\left[p_s^{sp}(\theta)\right] &= c_1 \text{Var}\left[\tilde{p}_f^{sp}(\theta)\frac{\pi(\theta)}{\pi_f(\theta)}\right] \\ &= \frac{\pi(\theta)^2}{\pi_f(\theta)^2}\text{Var}\left[\tilde{p}_f^{sp}(\theta)\right] \\ &\leqslant \frac{c}{T_f h^d}\end{aligned}$$

for some $c > 0$, where we have used the facts that $\text{Var}\left[\tilde{p}_f^{sp}(\theta)\right] \leqslant \frac{c}{Th^d}$ for some $c > 0$ and $\mathbb{E}\left[\tilde{p}_f^{sp}(\theta)\right]^2 \leqslant \tilde{c}$ for some $\tilde{c} > 0$ (given in [83, 196]). Next, we will use these two results to bound the mean-squared error of our semiparametric prior swap estimator, which shows that it is mean-square consistent.

We can write the mean-squared error as the sum of the variance and the bias-squared, and therefore,

$$\begin{aligned}\mathbb{E}\left[\int (p_s^{sp}(\theta) - p(\theta|x^n))^2 \, d\theta\right] &\leqslant c_1 h^2 + \frac{c_2}{Th^d} \\ &= \frac{c}{T_f^{4/(4+d)}}\end{aligned}$$

for some $c > 0$, using the fact that $h \asymp T_f^{-1/(4+d)}$. □

7.9 PRIOR SWAPPING AND EMBARRASSINGLY PARALLEL INFERENCE

Embarassingly parallel inference methods (Chapters 4-6) are post-inference methods that allow for partial inference results to be computed on subsets of data and afterwards efficiently combined to yield full inference results for the full data. In this section, we discuss cases where prior swapping can be used to improve algorithms for embarrassingly parallel inference. Specifically, prior swapping allows for the number of partitions to be unknown at inference time when performing distributed inference with embarrassingly parallel methods, and allows these methods to be used for sequential inference in online or streaming data settings (such as in model-based sequential decision making and optimization).

7.9.1 *Prior Swapping for Embarrassingly Parallel Inference with an Unknown Number of Partitions*

Recall that the goal of embarrassingly parallel inference is to compute a local posterior on each machine m, given a subset of data x^{n_m}. Previously, we described how to compute a subposterior density $p_m(\theta) \propto p(\theta)^{\frac{1}{M}} p(x^{n_m}|\theta)$, and then compute (or

sample from) an estimate of the product of all M subposterior densities, which is proportional to the full data posterior density.

In the settings for embarrassingly parallel MCMC (chapter 4) and VI (chapter 5), we assumed that we know the number of data partitions M before subposterior inference was carried out on each partition. This allowed for the prior to appropriately underweighted during inference on each subposterior, so that the combination procedures could yield a valid posterior result. However, this comes with a few disadvantages:

- One needs to know or decide the number of partitions of the data in advance of inference.
- It is not possible to add new data (such as unexpected additional observations) without redoing subposterior inferences.
- In practice, too many data partitions could lead to computational issues if the prior becomes too underweighted.

There is another, more subtle, disadvantage of requiring us to know the number of partitions M in advance. When we carry out subposterior inference, we are computing a quantity on each machine that doesn't represent something meaningful, such as the local posterior uncertainty given the subset of data x^{n_m}, and is dependent on the total number of machines M. It would be advantageous if this intermediate computed quantity instead had some intrinsic meaning or use—for example, if it provided some information about the uncertainty on the local subset of data—and wasn't just a temporary quantity used only to compute uncertainty for the full dataset. Furthermore, as a consequence of this, we must combine all M machines to compute the full posterior, and cannot compute some meaningful quantity on any subset of machines.

Here's the major issue: suppose we did infer the correct local posterior (meaning the actual posterior given the local subset of data) on each machine,

$$p(\theta|x^{n_m}) \propto \pi(\theta)p(x^{n_m}|\theta). \tag{78}$$

Taking the product of an arbitrary number of these local posteriors yields a combined posterior with too much prior weight; for example, combining all M of these local posteriors would yield

$$\prod_{m=1}^{M} p(\theta|x^{n_m}) \propto \pi(\theta)^M p(x^n|\theta). \tag{79}$$

To allow for correct local posteriors to be computed, and yet still apply the product density combination methods developed earlier, we need to find a way to mitigate this overweighted prior. One potential option is to apply methods from prior swapping to swap out the overweighted prior and swap in the correct prior for the full posterior (or for any abitrary subset of machines).

For example, suppose we computed the (correct) local posterior $p(\theta|x^{n_m})$ on M machines and performed combination procedures from Chapters 4-5 to yield samples from the density with PDF proportional to $\pi(\theta)^M p(x^n|\theta)$. We can consider this the false posterior density $\tilde{p}_f(\theta)$, where we've assumed an implicit false prior

$$\pi_f(\theta) \propto \pi(\theta)^M. \tag{80}$$

We can then take $\pi(\theta)$ to be the target prior density, and perform prior swapping to "divide out" the extra $\pi(\theta)^{M-1}$ prior mass incurred when we performed embarrassingly parallel inference. This would entail running an inference algorithm on the prior swap PDF

$$p_s(\theta) \propto \frac{\tilde{p}_f(\theta)\pi(\theta)}{\pi_f(\theta)} \propto \frac{\tilde{p}_f(\theta)}{\pi(\theta)^{M-1}}. \tag{81}$$

The combination of embarrassingly parallel inference and prior swapping allows us to compute the correct local posterior on a given subset of data, and then use these inference results in embarrassingly parallel inference procedures to compute the correct full data posterior distribution.

We can also use prior swapping to help in other ways. For example, we can do local inference with computationally convenient priors (such as priors that allow for easier, analytic, or more precise inferences), run methods from embarrassingly parallel inference to combine these local posteriors, and then use prior swapping to convert the final combined posterior into the correct result (under the desired prior).

7.9.2 *Prior Swapping for Embarrassingly Parallel Inference in a Sequential Setting*

In a sequential setting, we assume that we have a sequence of time steps $t = 1, \ldots, T$, where we begin with dataset x_0 and then append new data x_t to this set at each time step. At each time step t, let $\tilde{x}_t$ be the full set of data, i.e. $\tilde{x}_t = \tilde{x}_{t-1} \cup x_t$. The goal is to compute the posterior distribution given the data set at or before step t, for each $t = 1, \ldots, T$. This involves computing

$$p(\theta|\tilde{x}_t) = p(\theta|\tilde{x}_{t-1} \cup x_t) \propto \pi(\theta)p(\tilde{x}_{t-1}|\theta)p(x_t|\theta), \tag{82}$$

where we've assumed independent observations given parameter θ.

Suppose we have computed the posterior at time $t-1$ given all data up to this point, written $p(\theta|\tilde{x}_{t-1})$. Suppose that at time step t we now encounter the new dataset x_t, and compute the local posterior

$$p(\theta|x_t) \propto \pi(\theta)p(x_t|\theta). \tag{83}$$

We'd like to be able to use the previous time step's full posterior, $p(\theta|\tilde{x}_{t-1})$, along with the current time step's local posterior $p(\theta|x_t)$ to compute the current time step's

full posterior, $p(\theta|\tilde{x}_t)$. At each time step, we would thus only need to spend computation on the local posterior (which is cheaper, due to the small amount of data), and reuse the full inference result from the previous time step.

However, combining the previous time step's full posterior $p(\theta|\tilde{x}_{t-1})$ with the current time step's local posterior $p(\theta|x_t)$ naively via embarrassingly parallel inference yields

$$p(\theta|\tilde{x}_{t-1})p(\theta|x_t) \propto \pi(\theta)^2 p(\tilde{x}_t|\theta). \tag{84}$$

This is proportional to the current time step's full posterior with an overweighted prior. We could therefore apply prior swapping here to remove the extra $\pi(\theta)$ prior mass, where we take the false posterior to be $\tilde{p}_f(\theta) \propto p(\theta|\tilde{x}_{t-1})p(\theta|x_t)$. The prior swap density would therefore be

$$p_s(\theta) \propto \frac{\tilde{p}_f(\theta)\pi(\theta)}{\pi_f(\theta)} \propto \frac{p(\theta|\tilde{x}_{t-1})p(\theta|x_t)}{\pi(\theta)}. \tag{85}$$

In practice, at each time step of sequential inference, we could first apply embarrassingly parallel inference methods to infer (e.g. approximate the PDF of or generate samples from) $\tilde{p}_f(\theta) \propto p(\theta|\tilde{x}_{t-1})p(\theta|x_t)$, and then run a second (cheap) inference algorithm on the prior swap density $p_s(\theta)$, to compute the time step's full posterior $p(\theta|\tilde{x}_t)$.

7.10 CONCLUSION

Given some false posterior inference result, and an arbitrary target prior, we have studied methods to accurately compute the associated target posterior (or expectations with respect to it), and to do this efficiently by leveraging the pre-inferred result. We have argued and shown empirically that this strategy is effective even when the false and target posteriors are quite dissimilar. We believe that prior swapping methods show promise to allow a wider range of (and possibly less-costly) inference alorithms to be applied to certain models, and to allow updated or new prior information to be more-easily incorporated into models without re-incurring the full costs of standard inference algorithms. We have also shown how prior swapping can be used to aid in embarrassingly parallel inference: it can both allow for distributed inference given an unknown number of partitions M, and sequential inference that iteratively reuses previous inference results.

8

PROBO: VERSATILE BAYESIAN OPTIMIZATION USING ANY PROBABILISTIC PROGRAMMING LANGUAGE

8.1 CHAPTER SUMMARY

Optimizing an expensive-to-query function is a common task in science and engineering, where it is beneficial to keep the number of queries to a minimum. A popular strategy is Bayesian optimization (BO), which leverages probabilistic models for this task. Most BO today uses Gaussian processes (GPs), or a few other surrogate models. However, there is a broad set of Bayesian modeling techniques that could be used to capture complex systems and reduce the number of queries in BO. Probabilistic programming languages (PPLs) are modern tools that allow for flexible model definition, prior specification, model composition, and automatic inference. In this chapter, we develop ProBO, a BO procedure that uses only standard operations common to most PPLs. This allows a user to drop in a model built with an arbitrary PPL and use it directly in BO. We describe acquisition functions for ProBO, and strategies for efficiently optimizing these functions given complex models or costly inference procedures. Using existing PPLs, we implement new models to aid in a few challenging optimization settings, and demonstrate these on model hyperparameter and architecture search tasks.

8.2 INTRODUCTION

Bayesian optimization (BO) is a popular method for zeroth-order optimization of an unknown ("black box") system. A BO procedure iteratively queries the system to yield a set of input/output data points, computes the posterior of a Bayesian model given these data, and optimizes an acquisition function defined on this posterior in order to determine the next point to query.

BO involves performing inference and optimization to choose each point to query, which can incur a greater computational cost than simpler stategies, but may be ultimately beneficial in settings where queries are expensive. Specifically, if BO can reach a good optimization objective in fewer iterations than simpler methods, it may be effective in cases where the expense of queries far outweighs the extra cost of BO. Some examples of this are in science and engineering, where a query could involve synthesizing and measuring the properties of a material, collecting metrics from an industrial process, or training a large machine learning model, which can be expensive in cost, time, or human labor.

The most common model used in BO is the *Gaussian process* (GP), for which we can compute many popular acquisition functions. There has also been some work deriving BO procedures for other flexible models including random forests [92] and neural networks [175]. In this chapter, we argue that more-sophisticated models that better capture the details of a system can help reduce the number of iterations needed in BO, and allow for BO to be effectively used in custom and complex settings. For example, systems may have complex noise [96, 166], yield multiple types of observations [60], depend on covariates [105], have interrelated subsystems [180], and more. To accurately capture these systems, we may want to design custom models using a broader library of Bayesian tools and techniques. For example, we may want to compose models—such as GPs, latent factor (e.g. mixture) models, deep Bayesian networks, hierarchical regression models—in various ways, and use them in BO.

Probabilistic programming languages (PPLs) are modern tools for specifying Bayesian models and performing inference. They allow for easy incorporation of prior knowledge and model structure, composition of models, quick deployment, and automatic inference, often in the form of samples from or variational approximations to a posterior distribution. PPLs may be used to specify and run inference in a variety of models, such as graphical models, GPs, deep Bayesian models, hierarchical models, and implicit (simulator-based) models, to name a few [7, 21, 38, 50, 117, 121, 127, 161, 186, 202].

We would like to be able to build an arbitrary model with any PPL and then automatically carry out BO with this model. However, this comes with a few challenges. In BO with GPs, we have the posterior in closed-form, and use this when computing and optimizing acquisition functions. PPLs, however, use a variety of approximate inference procedures, which can be costly to run and yield different posterior representations (e.g. samples [38, 202], variational approximations [21, 186], implicit models [91, 187], or amortized distributions [112, 155]). We need a method that can compute and optimize acquisition functions automatically, given the variety of representations, and efficiently, making judicious use of PPL procedures.

Towards this end, we develop ProBO, a BO system for PPL models, which computes and optimizes acquisition functions via operations that can be implemented in a broad variety of PPLs. This system comprises algorithms that cache and use these operations efficiently, which allows it to be used in practice given complex models and expensive inference procedures. The overall goal of ProBO is to allow a custom model written in an arbitrary PPL to be "dropped in" and immediately used in BO.

This chapter has two main contributions: (1) We present ProBO, a system for versatile Bayesian optimization using models from any PPL. (2) We describe optimization settings that are difficult for standard BO methods and models, and then use PPLs to implement new models for these settings, which are dropped into ProBO and show good optimization performance. Our open source release of ProBO is available at `https://github.com/willieneis/ProBO`.

8.3 RELATED WORK

A few prior works make connections between PPLs and BO. BOPP [152] describes a BO method for marginal maximum a posteriori (MMAP) estimates of latent variables in a probabilistic program. This work relates BO and PPLs, but differs from us in that the goal of BOPP is to use BO (with GP models) to help estimate latent variables in a given PPL, while we focus on using PPLs to build new surrogate models for BO.

BOAT [45] provides a custom PPL involving composed Gaussian process models with parametric mean functions, for use in BO. For these models, exact inference can be performed and the expected improvement acquisition directly used. This work has similar goals as us, though we instead aim to provide a system that can be applied to models from *any* existing PPL (not constrained to a certain family of GP models), and specifically with PPLs that use approximate inference algorithms where we cannot compute acquisition functions in standard ways.

8.4 PROBO

We first describe a general abstraction for PPLs, and use this abstraction to define ProBO and present algorithms for computing a few acquisition functions. We then show how to efficiently optimize these acquisition functions.

8.4.1 *Abstraction for Probabilistic Programs*

Suppose we are modeling a system which, given an input $x \in \mathcal{X}$, yields observations $y \in \mathcal{Y}$, written $y \sim s(x)$. Let $f : \mathcal{Y} \to \mathbb{R}$ be an objective function that maps observations y to real values. Observing the system n times at different inputs yields a dataset $\mathcal{D}_n = \{(x_i, y_i)\}_{i=1}^n$. Suppose we have a Bayesian model for $\mathcal{D}_n$, with likelihood $p(\mathcal{D}_n|z) = \prod_{i=1}^n p(y_i|z; x_i)$, where $z \in \mathcal{Z}$ are latent variables. We define the joint model PDF to be $p(\mathcal{D}_n, z) = p(z)p(\mathcal{D}_n|z)$, where $p(z)$ is the PDF of the prior on z. The posterior PDF is then $p(z|\mathcal{D}_n) = p(\mathcal{D}_n, z)/\int p(\mathcal{D}_n, z)dz$.

Our abstraction assumes three basic PPL operations:

1. `inf`$(\mathcal{D})$: given data $\mathcal{D}$, this runs an inference algorithm and returns an object `post`, which is a PPL-dependent representation of the posterior distribution.
2. `post`(s): given a seed $s \in \mathbb{Z}^+$, this returns a sample from the posterior distribution.
3. `gen`(x, z, s): given an input $x \in \mathcal{X}$, a latent variable $z \in \mathcal{Z}$, and a seed $s \in \mathbb{Z}^+$, this returns a sample from the generative distribution $p(y|z; x)$.

Note that `post` and `gen` are deterministic, i.e. for a fixed seed s, `post`/`gen` produce the same output each time they are called.

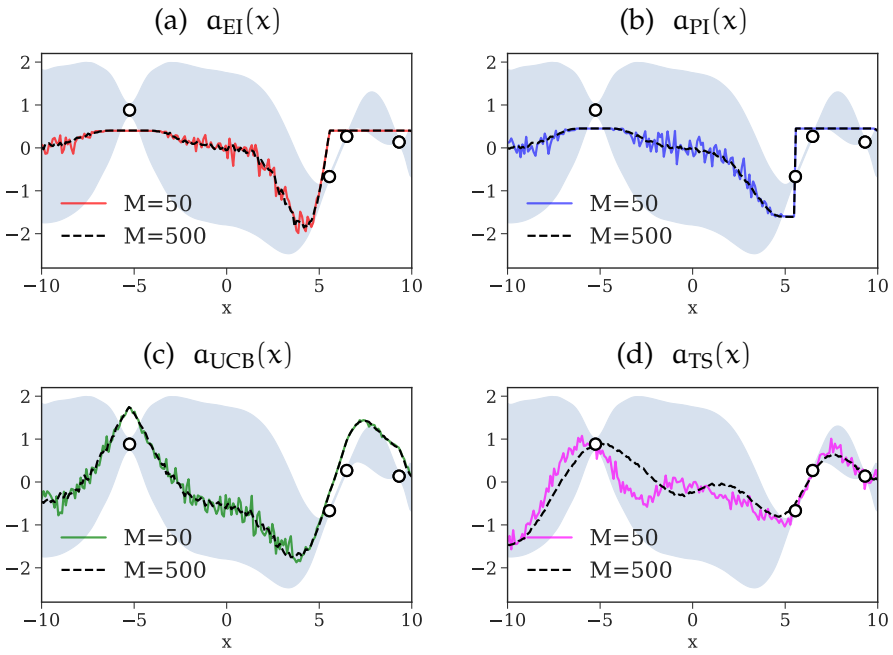


Figure 38: Visualizations of PPL acquisition functions $a(x)$ given in Algs. 13-16 for use in ProBO. In each plot, the data and posterior predictive distribution are shown, and $a(x)$ is given for two fidelities: $M = 50$ (solid color line) and $M = 500$ (dashed black line).

Scope.

This abstraction applies to a number of PPLs, which use a variety of inference strategies and compute different representations for the posterior. For example, in PPLs using Markov chain Monte Carlo (MCMC) or sequential Monte Carlo (SMC) algorithms [38, 161, 202], `inf` computes a set of posterior samples and `post` draws uniformly from this set. For PPLs using variational inference (VI), implicit models, or exact inference methods [21, 77, 186], `inf` computes the parameters of a distribution, and `post` draws a sample from this distribution. In amortized or compiled inference methods [100, 112, 155], `inf` trains or calls a pretrained model that maps observations to a posterior or proposal distribution, and `post` samples from this.

8.4.2 *Main Procedure*

Recall that we use PPLs to model a system s, which yields observations $y \sim s(x)$ given a query x, and where $f(y) : \mathcal{Y} \to \mathbb{R}$ denotes the objective value that we want to optimize. The goal of ProBO is to return $x^* = \arg\min_{x \in \mathcal{X}} \mathbb{E}_{y \sim s(x)}[f(y)]$. We give ProBO in Alg. 12. Each iteration consists of four steps: call an inference procedure `inf`, select an input x by optimizing an acquisition function a that calls `post` and `gen`, observe the system at x, and add new observations to the dataset.

Algorithmus 12 : ProBO($\mathcal{D}_0$, `inf`, `gen`) from Neiswanger et al. [139]

Input : Inputs listed here.
Output : Outputs listed here.

1 **for** $n = 1, \ldots, N$ **do**
2 $\quad$ `post` $\leftarrow$ `inf`$(\mathcal{D}_{n-1})$ ▷ Run inference algorithm to compute `post`
3 $\quad$ $x_n \leftarrow \arg\min_{x \in \mathcal{X}} a(x, \texttt{post}, \texttt{gen})$ ▷ Optimize acquisition using `post` and `gen`
4 $\quad$ $y_n \sim s(x_n)$ ▷ Observe system at x_n
5 $\quad$ $\mathcal{D}_n \leftarrow \mathcal{D}_{n-1} \cup (x_n, y_n)$ ▷ Add new observations to dataset
6 Return $\mathcal{D}_N$.

Note that acquisition optimization (line 3) involves only `post` and `gen`, while `inf` is called separately beforehand. We discuss the computational benefits of this, and the cost of ProBO, in more detail in Sec. 8.4.4. Also note that many systems can have extra observations $y \in \mathcal{Y}$, in addition to the objective value, that provide information which aids optimization [11, 180, 204]. For this reason, our formalism explicitly separates system observations y from their objective values $f(y)$. We show examples of models that take advantage of extra observations in Sec. 8.5.

In the following sections, we develop algorithms for computing acquisition functions $a(x)$ automatically using only `post` and `gen` operations, without requiring any model specific derivation. We refer to these as *PPL acquisition functions*.

8.4.3 *PPL Acquisition Functions via `post` and `gen`*

In ProBO (Alg. 12), we denote the PPL acquisition function with $a(x, \texttt{post}, \texttt{gen})$. Each PPL acquisition algorithm includes a parameter M, which represents the fidelity of the approximation quality of a. We will describe an adaptive method for choosing M during acquisition optimization in Sec. 8.4.5. Below, we will make use of the posterior predictive distribution, which is defined to be $p(y|\mathcal{D}_n; x) = \mathbb{E}_{p(z|\mathcal{D}_n)}[p(y|z; x)]$.

There are a number of popular acquisition functions used commonly in Bayesian optimization, such as expected improvement (EI) [129], probability of improvement (PI) [106], GP upper confidence bound (UCB) [176], and Thompson sampling (TS) [182]. Here, we propose a few simple acquisition estimates that can be computed with `post` and `gen`. Specifically, we give algorithms for EI (Alg. 13), PI (Alg. 14), UCB (Alg. 15), and TS (Alg. 16) acquistion strategies, though similar algorithms could be used for other acquisitions involving expectations or statistics of either $p(y|D_n; x)$ or $p(y|z; x)$.

We now describe the PPL acquisition functions given in Alg. 13-16 in more detail, and discuss the approximations given by each. Namely, we show that these yield versions of exact acquisition functions as $M \to \infty$. These algorithms are related to

Monte Carlo acquisition function estimates for GP models [80, 81, 174], which have been developed for specific acquisition functions.

Expected Improvement (EI), Alg. 13

Let $\mathcal{D}$ be the data at a given iteration of ProBO. In our setting, the expected improvement (EI) acquisition function will return the expected improvement that querying the system at $x \in \mathcal{X}$ will have over the minimal observed objective value, $f_{\min} = \min_{y \in \mathcal{D}} f(y)$. We can write the exact EI acquisition function as

$$a^*_{\text{EI}}(x) = \int \mathbb{1}\{f(y) \leqslant f_{\min}\}(f_{\min} - f(y))\, p(y|\mathcal{D}; x)\, dy. \tag{86}$$

In Alg. 13, for a sequence of steps $m = 1, \ldots, M$, we draw $z_m \sim p(z|\mathcal{D})$ and $y_m \sim p(y|z_m; x)$ via `post` and `gen`, and then compute $\lambda(y_{1:M}) = \sum_{m=1}^{M} \mathbb{1}[f(y_m) \leqslant f_{\min}](f_{\min} - f(y_m))$. Marginally, y_m is drawn from the posterior predictive distribution, i.e. $y_m \sim \mathbb{E}_{p(z|\mathcal{D})}[p(y|z; x)] = p(y|\mathcal{D}; x)$. Therefore, as the number of calls M to `post` and `gen` grows, $a_{\text{EI}}(x) \to a^*_{\text{EI}}(x)$ (up to a multiplicative constant) at a rate of $O(\sqrt{M})$.

Algorithmus 13 : PPL EI acquisition, $a_{\text{EI}}(x, \texttt{post}, \texttt{gen})$ from Neiswanger et al. [139]

Input : Inputs listed here.
Output : Outputs listed here.
for $m = 1, \ldots, M$ **do**
  $z_m \leftarrow \texttt{post}(s_m)$
  $y_m \leftarrow \texttt{gen}(x, z_m, s_m)$
$f_{\min} \leftarrow \min_{y \in \mathcal{D}} f(y)$
Return $\sum_{m=1}^{M} \mathbb{1}[f(y_m) \leqslant f_{\min}](f_{\min} - f(y_m))$

Algorithmus 14 : PPL PI acquisition, $a_{\text{PI}}(x, \texttt{post}, \texttt{gen})$ from Neiswanger et al. [139]

Input : Inputs listed here.
Output : Outputs listed here.
for $m = 1, \ldots, M$ **do**
  $z_m \leftarrow \texttt{post}(s_m)$
  $y_m \leftarrow \texttt{gen}(x, z_m, s_m)$
$f_{\min} \leftarrow \min_{y \in \mathcal{D}} f(y)$
Return $\sum_{m=1}^{M} \mathbb{1}[f(y_m) \leqslant f_{\min}]$

Probability of Improvement (PI), Alg. 14

In our setting, the probability of improvement (PI) acquisition function will return the probability that observing the system at query $x \in \mathcal{X}$ will improve upon the

Algorithmus 15 : PPL UCB acquisition, $a_{\text{UCB}}(x, \texttt{post}, \texttt{gen})$ from Neiswanger et al. [139]

Input : Inputs listed here.
Output : Outputs listed here.

for $m = 1, \ldots, M$ **do**
  $z_m \leftarrow \texttt{post}(s_m)$
  $y_m \leftarrow \texttt{gen}(x, z_m, s_m)$
Return $\widehat{\text{LCB}}\left(f(y_m)_{m=1}^M\right)$ ▷ See text for details

Algorithmus 16 : PPL TS acquisition, $a_{\text{TS}}(x, \texttt{post}, \texttt{gen})$ from Neiswanger et al. [139]

Input : Inputs listed here.
Output : Outputs listed here.

$z \leftarrow \texttt{post}(s_1)$
for $m = 1, \ldots, M$ **do**
  $y_m \leftarrow \texttt{gen}(x, z, s_m)$
Return $\sum_{m=1}^M f(y_m)$

minimally observed objective value, $f_{\min} = \min_{y \in \mathcal{D}} f(y)$. We can write the exact PI acquisition function as

$$a_{\text{PI}}^*(x) = \int \mathbb{1}\{f(y) \leqslant f_{\min}\} p(y|\mathcal{D}; x)\, dy. \tag{87}$$

In Alg. 14, for a sequence of steps $m = 1, \ldots, M$, we draw $z_m \sim p(z|\mathcal{D})$, $y_m \sim p(y|z_m; x)$ via `post` and `gen`, and then compute $\lambda(y_{1:M}) = \sum_{m=1}^M \mathbb{1}[f(y_m) \leqslant f_{\min}]$. As before, y_m is drawn (marginally) from the posterior predictive distribution. Therefore, as the number of calls M to `post` and `gen` grows, $a_{\text{PI}}(x) \to a_{\text{PI}}^*(x)$ (up to a multiplicative constant) at a rate of $O(\sqrt{M})$.

Upper Confident Bound (UCB), Alg. 15

We propose an algorithm based on the principle of optimization under uncertainty (OUU), which aims to compute a lower confidence bound for $p(f(y)|\mathcal{D}; x)$, which we denote by $\text{LCB}[p(f(y)|\mathcal{D}_n; x)]$. In Alg. 15, we use an estimate of this, $\widehat{\text{LCB}}(f(y_m)_{m=1}^M)$. Note that we use a lower confidence bound since we are performing minimization, though we denote our acquisition function with the more commonly used title UCB. Two simples strategies for estimating this LCB are

1. *Empirical quantiles:* Order $f(y_m)_{m=1}^M$ into $f_{(1)} \leqslant \ldots \leqslant f_{(M)}$, and return $f_{(b)}$ if $b \in \mathbb{Z}$, or else return $\frac{1}{2}(f_{(\lfloor b \rfloor)} + f_{(\lfloor b \rfloor + 1)})$, where $b \in [0, M+1]$ is a tradeoff parameter.

2. *Parametric assumption:* As an example, if we model $p(f(y)|\mathcal{D}_n; x) = \mathcal{N}(f(y)|\mu, \sigma^2)$, we can compute $\hat{\mu} = \frac{1}{M}\sum_{m=1}^{M} f(y_m)$ and $\hat{\sigma}^2 = \frac{1}{M-1}\sum_{m=1}^{M}(f(y_m) - \hat{\mu})^2$, and return $\hat{\mu} - \beta\hat{\sigma}^2$, where $\beta > 0$ is a trade-off parameter.

The first proposed estimate (empirical quantiles) is a consistent estimator, though may yield worse performance in practice than the second proposed estimate in cases where we can approximate $p(f(y_m)|\mathcal{D}; x)$ with some parametric form.

Thompson Sampling (TS), Alg. 16

Thompson sampling (TS) proposes proxy values for unknown model variables by drawing a posterior sample, and then performs optimization as if this sample were the true model variables, and returns the result. In Alg. 16, we provide an acquisition function to carry out a TS strategy in ProBO , using `post` and `gen`. At one given iteration of BO, a specified seed is used so that each call to $a_{TS}(x)$ produces the same posterior latent variable sample $\tilde{z} \sim p(z|\mathcal{D})$ via `post`. After, `gen` is called repeatedly to produce $y_m \sim p(y|\tilde{z}; x)$ for $m = 1, \ldots, M$, and the objective values of these are averaged to yield $\lambda(y_{1:M}) = \sum_{m=1}^{M} f(y_m)$. Here, each $f(y_m) \sim \mathbb{E}_{p(y|\tilde{z};x)}[f(y)]$. Optimizing this acquisition function serves as a proxy for optimizing our model given the true model variables, using posterior sample $\tilde{z}$ in place of unknown model variables.

In Summary

As $M \to \infty$, for constants c_1, c_2, c_3, and c_4,

$$a_{EI}(x, \texttt{post}, \texttt{gen}) \to c_1 \int \mathbb{1}\left\{f(y) \leqslant \min_{y' \in \mathcal{D}} f(y')\right\}\left(\min_{y' \in \mathcal{D}} f(y') - f(y)\right) p(y|\mathcal{D}; x)\,dy \tag{88}$$

$$a_{PI}(x, \texttt{post}, \texttt{gen}) \to c_2 \int \mathbb{1}\left\{f(y) \leqslant \min_{y' \in \mathcal{D}} f(y')\right\} p(y|\mathcal{D}; x)\,dy \tag{89}$$

$$a_{UCB}(x, \texttt{post}, \texttt{gen}) \to c_3\ \text{LCB}\,[p(f(y)|\mathcal{D}; x)] \tag{90}$$

$$a_{TS}(x, \texttt{post}, \texttt{gen}) \to c_4 \int f(y)\ p(y|\tilde{z}; x)\,dy, \quad \text{for } \tilde{z} \sim p(z|\mathcal{D}), \tag{91}$$

We visualize Alg. 13-16 in Fig. 38 (a)-(d), showing $M \in \{50, 500\}$.

8.4.4 *Computational Considerations*

In ProBO, we run a PPL's inference procedure when we call `inf`, which has a cost dependent on the underlying inference algorithm. For example, most MCMC methods have complexity $O(n)$ per iteration [14]. However, ProBO only runs `inf` once per query; acquisition optimization, which may be run hundreds of times per query, instead uses only `post` and `gen`. For many PPL models, `post` and `gen` can be implemented cheaply. For example, `post` often involves drawing from a pool of samples

or from a known distribution, and gen often involves sampling from a fixed-length sequence of known distributions and transformations, both of which typically have $O(1)$ complexity. However, for some models, gen can involve running a more costly simulation. For these cases, we provide acquisition optimization algorithms that use post and gen efficiently in Sec. 8.4.5.

8.4.5 *Efficient Optimization of PPL Acquisition Functions*

In ProBO, we must optimize over the acquisition algorithms defined in the previous section, i.e. compute $x_n = \arg\min_{x \in \mathcal{X}} a(x, \texttt{post}, \texttt{gen})$. Note that post and gen are not in general analytically differentiable, so in contrast with [201], we cannot optimize $a(x)$ with gradient-based methods. We therefore explore strategies for efficient zeroth-order optimization.

In Alg. 13-16, M denotes the number of times post and gen are called in an evaluation of $a(x)$. As seen in Fig. 38, a small M will return a noisy estimate of $a(x)$, while a large M will return a more-accurate estimate. However, for some PPLs, the post and/or gen operations can be costly (e.g. if gen involves a complex simulation [120, 185]), and we'd like to minimize the number of times they are called.

This is a special case of a multi-fidelty optimization problem [58], with fidelity parameter M. Unlike typical multi-fidelity settings, our goal is to reduce the number of calls to post and gen for a single x only, via modifying the acquisition function $a(x, \texttt{post}, \texttt{gen})$. This way, we can drop in any off-the-shelf optimizer that makes calls to a. Suppose we have F fidelities ranging from a small number of samples M_{min} to a large number M_{max}, i.e. $M_{\text{min}} = M_1 < \ldots < M_F = M_{\text{max}}$. Intuitively, when calling $a(x, \texttt{post}, \texttt{gen})$ on a given x, we'd like to use a small M if $a(x)$ is far from the minimal value $a(x^*)$, and a larger M if $a(x)$ is close to $a(x^*)$.

We propose the following procedure: Suppose a_{min} is the minimum value of a seen so far during optimization (for any x). For a given fidelity M_f (starting with f=1), we compute a lower confidence bound (LCB) for the sampling distribution of $a(x, \texttt{post}, \texttt{gen})$ with M_f calls to post and gen. We can do this via the bootstrap method [54] along with the LCB estimates described in Sec. 8.4.3. If this LCB is below a_{min}, it remains plausible that the acquisition function minimum is at x, and we repeat these steps at fidelity M_{f+1}. After reaching a fidelity f^* where the LCB is above a_{min} (or upon reaching the highest fidelity $f^* = F$), we return the estimate $a(x, \texttt{post}, \texttt{gen})$ with M_{f^*} calls. We give this procedure in Alg. 17.

In Alg. 18 we use notation $\lambda(y_{1:M})$ to denote the final operation (last line) in one of Algs. 13-16 (e.g. $\lambda_a(y_{1:M}) = \sum_{m=1}^{M} \mathbb{1}[f(y_m) \leqslant f_{\text{min}}]$ in the case of PI). As a simple example, we could run a two-fidelity algorithm, with $M \in \{M_1, M_2\}$, where $M_1 \ll M_2$. For a given x, a_{MF} would first call post and gen M_1 times, and compute the LCB with the bootstrap. If the LCB is greater than a_{min}, it would return an $a(x, \texttt{post}, \texttt{gen})$ with the M_1 calls; if not, it would return it with M_2 calls. Near

Algorithmus 17 : Multi-fidelity a, $a_{MF}(x, \texttt{post}, \texttt{gen})$ from Neiswanger et al. [139]

Input : Inputs listed here.
Output : Outputs listed here.

$a_{min} \leftarrow$ Min value of a seen so far
$\ell = -\infty$, $f = 1$
while $\ell \leqslant a_{min}$ **do**
  $\ell \leftarrow$ LCB-bootstrap $(\texttt{post}, \texttt{gen}, M_f)$
  $f \leftarrow f + 1$
Return $a(x, \texttt{post}, \texttt{gen})$ using $M = M_f$

Algorithmus 18 : LCB-bootstrap$(\texttt{post}, \texttt{gen}, M_f)$ from Neiswanger et al. [139]

Input : Inputs listed here.
Output : Outputs listed here.

$y_{1:M_f} \leftarrow$ Call `post` and `gen` M_f times
for $j = 1, \ldots, B$ **do**
  $\tilde{y}_{1:M_f} \leftarrow$ Resample$(y_{1:M_f})$
  $a_j \leftarrow \lambda(\tilde{y}_{1:M_f})$ ▷ See text for details
Return LCB $(a_{1:B})$

optima, this will make $M_1 + M_2$ calls to `post` and `gen`, and will make M_1 calls otherwise.

One can apply any derivative-free (zeroth-order) global optimization procedure that iteratively calls a_{MF}. In general, we can replace the optimization step in ProBO (Alg. 12, line 3) with $x_n \leftarrow \arg\min_{x \in \mathcal{X}} a_{MF}(x)$, for each of the PPL acquisition functions described in Sec. 8.4.3. In Sec. 8.5.4, we provide experimental results for this method, showing favorable performance relative to high fidelity acquisition functions, as well as reduced calls to `post` and `gen`.

8.5 EXAMPLES AND EXPERIMENTS

We provide examples of models to aid in complex optimization scenarios, implement these models with PPLs, and show empirical results. Our main goals are to demonstrate that we can plug models built with various PPLs into ProBO, and use these to improve BO performance (i.e. reduce the number of iterations) when compared with standard methods and models. We also aim to verify that our acquisition functions and extensions (e.g. multi-fidelity a_{MF}) perform well in practice.

PPL Implementations:

We implement models with Stan [38] and Edward [186], which (respectively) make use of the No U-Turn Sampler [87] (a form of Hamiltonian Monte Carlo) and black box variational inference [153]. We also use George [7] and GPy [77] for GP comparisons.

8.5.1 *BO with State Observations*

Setting:

Some systems exhibit unique behavior in different regions of the input space $\mathcal{X}$ based on some underlying state. We often do not know these state regions apriori, but can observe the state of an x when it is queried. Two examples of this, for computational systems, are:

- Timeouts or failures: there may be regions where queries fail or time out. We can observe if a query has a "pass" or "fail" state [60, 63, 108].
- Resource usage regions: queries can have distinct resource usage patterns. We can observe this pattern for a query, and use it to assign a state [6, 45].

Assume that for each query $x \in \mathcal{X}$, $s(x)$ returns a $y \in \mathcal{Y} = \mathbb{R} \times \mathbb{Z}^+$, with two types of information: an objective value y_0 and an state observation y_1 indicating the region assignment. We take the objective function to be $f(y) = y_0$.

Model:

Instead of using a single black box model for the entire input space $\mathcal{X}$ we provide a model that infers the distinct regions and learns a model for each. For the case of two states, we can write the generative model for this as: $c \sim \text{Bernoulli}(\cdot|C(x))$, $y \sim cM_1(\cdot|x) + (1-c)M_2(\cdot|x)$, where M_1 and M_2 are models for $y|x$ (e.g. GP regression models) and C is a classification model (e.g. a Bayesian NN) that models the probability of M_1. We refer to this model as a *switching model*. This model could be extended to more states. We show inference in this model in Fig. 39 (d). Comparing this with a a GP (Fig. 39 (c)), we see that GP hyperparameter estimates can be negatively impacted due to the nonsmooth landscape around region boundaries.

Empirical Results:

We demonstrate the switching model on the task of neural network architecture and hyperparameter search [97, 212] with timeouts, where in each query we train a network and return accuracy on a held out validation set. However, training must finish within a given time threshold, or it *times out*, and instead returns a preset (low accuracy) value. We optimize over multi-layer perceptron (MLP) neural networks, where we represent each query as a vector $x \in \mathbb{R}^4$, where x = (number of layers,

layer width, learning rate, and batch size). We train and validate each network on the Pima Indians Diabetes Dataset [171]. Whenever training has not converged within 60 seconds, the system times out and returns a fixed accuracy value of 30%. We use a GP regression model for M_1, and a Gaussian model (with mean and variance latent variables) for M_2. We compare ProBO with this switching model against standard BO using GPs, plotting the maximum validation accuracy found vs iteration n, averaged over 10 trials, in Fig. 39 (e)-(f).

8.5.2 *Robust Models for Contaminated BO*

Setting:

We may want to optimize a system that periodically yields "contaminated observations," i.e. outliers drawn from a second noise distribution. Examples of this are queries involving unstable simulations [116], or faulty computer systems [163]. This is similar to the setting of Huber's ϵ-contamination model [90], and we refer to this as *contaminated BO*. The contaminating distribution may have some dependence on input $\mathcal{X}$ (e.g. may be more prevalent in a window around the optimum value x^*). Note that this differs from Sec. 8.5.1 because we do not have access to state observations, and the noise distributions are not in exclusive regions of $\mathcal{X}$. To perform accurate BO in this setting, we need models that are robust to the contamination noise.

Model:

We develop a *denoising model*, which infers (and ignores) contaminated data points. Given a system model M_s and contamination model M_c we write our denoising model as $y \sim w_s M_s(\cdot|z_s; x) + w_c M_c(\cdot|z_c; x)$, where $z_s, z_c \sim \text{Prior}(\cdot)$, and $w_s, w_c \sim \text{Prior}(\cdot|x)$ (and where Prior denotes some appropriate prior density). This is a mixture where weights (w_s, w_c) can depend on input x. We show inference in this model in Fig. 40 (c)-(d).

Empirical Results:

We show experimental results for ProBO on a synthetic optimization task. This allows us to know the true optimal value x^* and objective $f(s(x^*))$, which may be difficult to judge in real settings (given the contaminations), and to show results under different contamination levels. For an $x \in \mathbb{R}^d$, with probability $1-p$ we query the function $f(x) = \|x\|_2 - \frac{1}{d}\sum_{i=1}^{d}\cos(x_i)$, which has a minimum value of $f(x^*) = -1$ at $x^* = 0_d$, and with probability p, we receive a contaminated value with distribution $f(x) \sim \text{Unif}([f_{max}/10, f_{max}])$, where f_{max} is $\max_{x\in\mathcal{X}} f(x)$. We compare ProBO using a *denoising GP* model with standard BO using GPs. We show results for both a low contamination setting ($p = .01$) and a high contamination setting ($p = .33$), in Fig. 40 (e)-(h), where we plot the minimal found value (under the

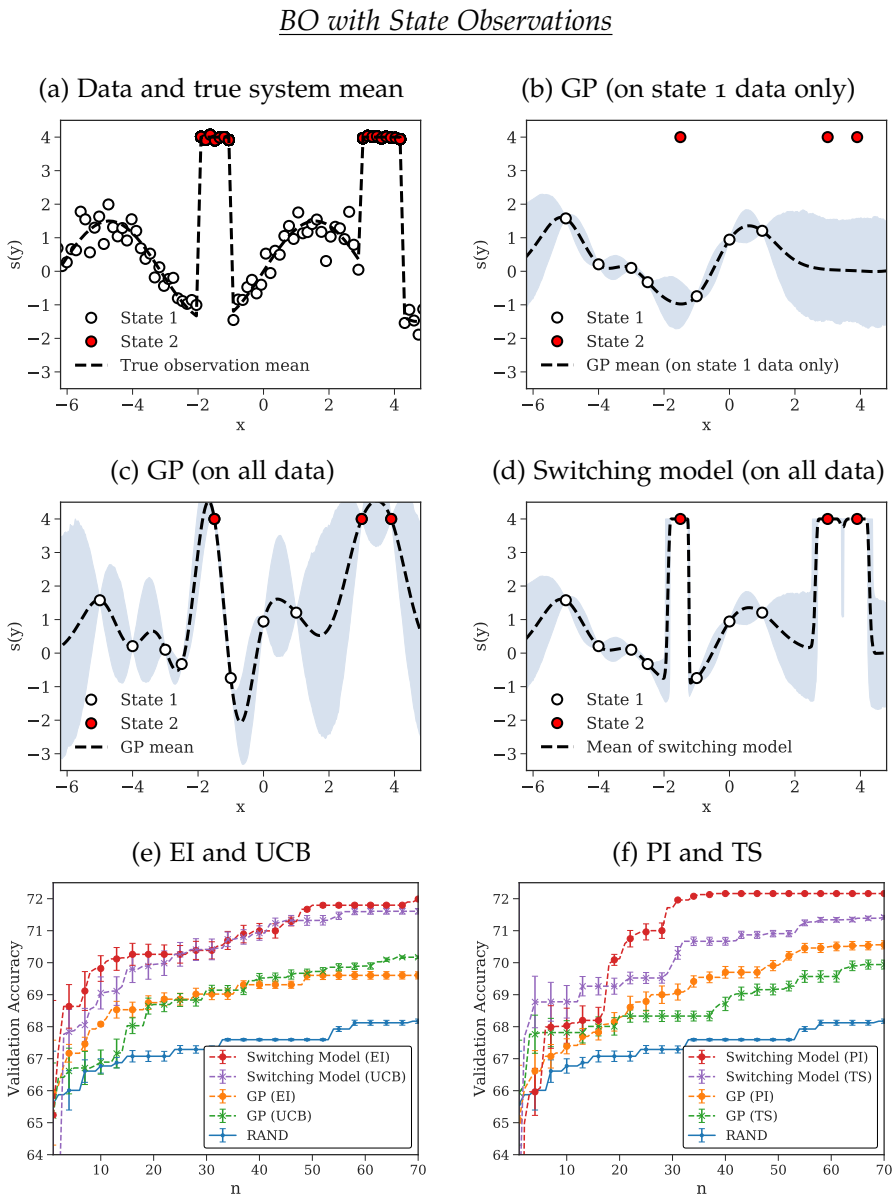


Figure 39: BO with state observations (Sec. 8.5.1). We show (a) the true system, and inference results on (b) a GP model fit on state 1 data only, (c) the same GP model fit on all data, where hyperparameter estimates are badly influenced, and (d) our switching model fit on all data. In (e)-(f) we show results on the task of neural network architecture and hyperparameter search with timeouts, comparing ProBO using a switching model to BO using GPs. Curves are averaged over 10 trials, and error bars represent one standard error.

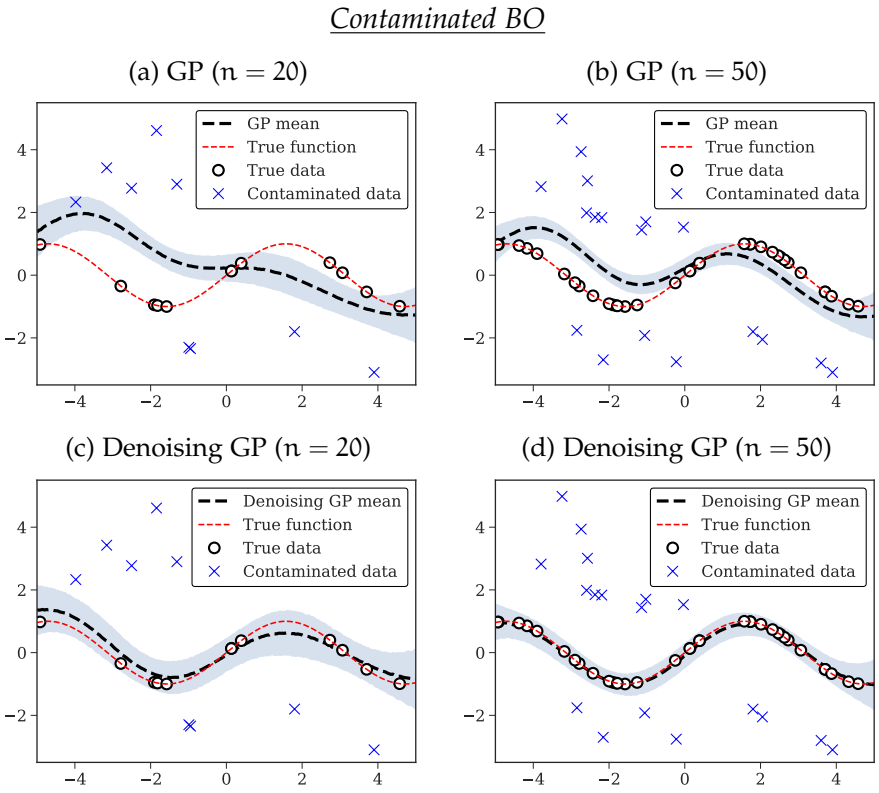


Figure 40: Contaminated BO (Sec. 8.5.2). We show (a) inference in a GP with $n = 20$ and (b) $n = 50$, and (c) inference in a denoising GP with $n = 20$ and (d) $n = 50$.

noncontaminated model) $f_{min} = \min_{t \leqslant n} f(y_t)$ vs iteration n, averaged over 10 trials. In the low contamination setting, both models converge to a near-optimal value and perform similarly, while in the high contamination setting, ProBO with denoising GP converges to a near optimal value while standard BO with GPs does not.

8.5.3 *BO with Prior Structure on the Objective Function*

Setting:

In some cases, we have prior knowledge about properties of the objective function, such as trends with respect to $x \in \mathcal{X}$. For example, consider the task of tuning hyperparameters of a machine learning model, where the hyperparameters correspond with model complexity. For datasets of moderate size, there are often two distinct phases as model complexity grows: a phase where the model underfits, where increasing modeling complexity reduces error on a held-out validation set; and a phase where the model overfits, where validation error increases with respect to model complexity. We can design a model that leverages trends such as these.

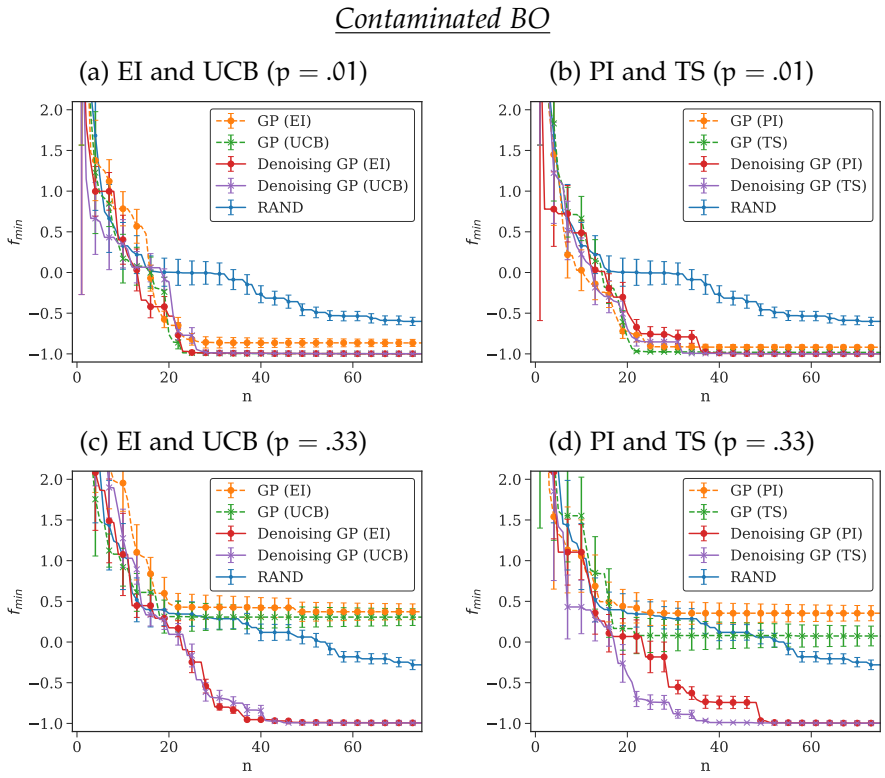


Figure 41: Contaminated BO (Sec. 8.5.2). We show (e)-(f), for low corruption ($p = .01$), ProBO using denoising GPs is competitive with standard BO using GP models. In (g)-(h), for higher corruption ($p = .33$), ProBO converges to the optimal value while standard BO does not, even as n grows. Curves are averaged over 10 trials, and error bars represent one standard error.

Model:

We design a model for tuning model complexity, which we refer to as a basin model. Let $y \sim \mathcal{N}(R(x - \mu; a, b) + c, \sigma^2)$ where $R(x; a, b) = a^T \text{ReLU}(x) + b^T \text{ReLU}(-x)$, with priors on parameters $\mu \in \mathbb{R}_d$, $a, b \in \mathbb{R}^+_d$, $c \in \mathbb{R}$, and $\sigma^2 > 0$. This model captures the inflection point with variable μ, and uses variables a and b to model the slope of the optimization landscape above and below (respectively) μ. We give a one dimensional view of validation error data from an example where x corresponds to neural network layer width, and show inference with a basin model for this data in Fig. 42 (b).

Empirical Results:

In this experiment, we optimize over the number of units (i.e. layer width) of the hidden layers in a four layer MLP trained on the Wisconsin Breast Cancer Diagnosis dataset [23]. We compare ProBO using a basin model, with standard BO using a GP. We see in Fig 42 (c)-(d) that ProBO with the basin model can significantly outperform

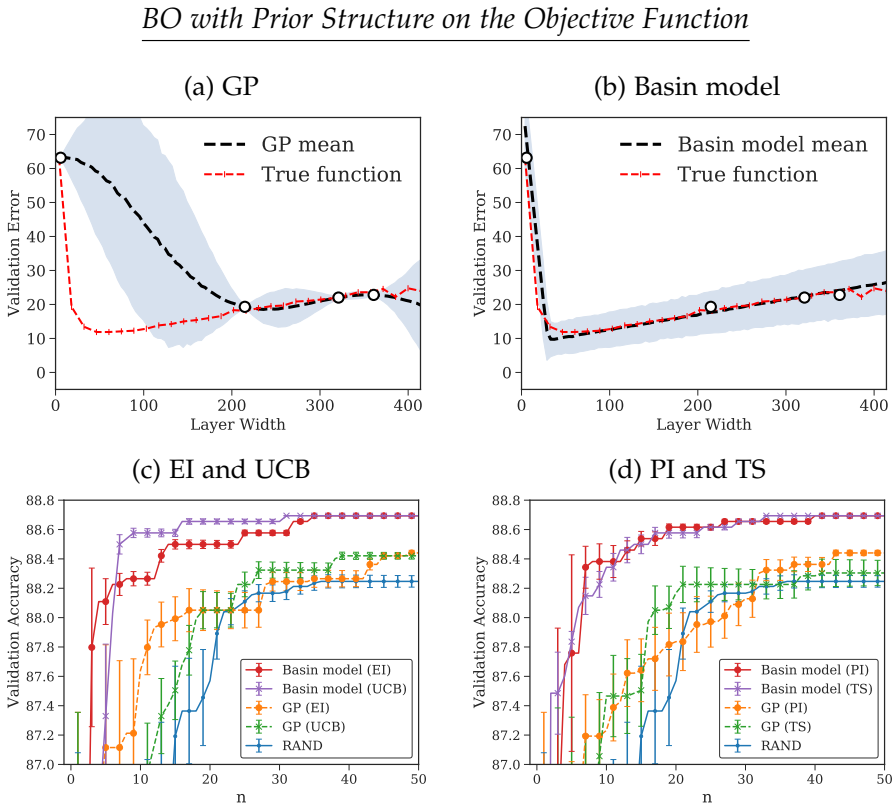
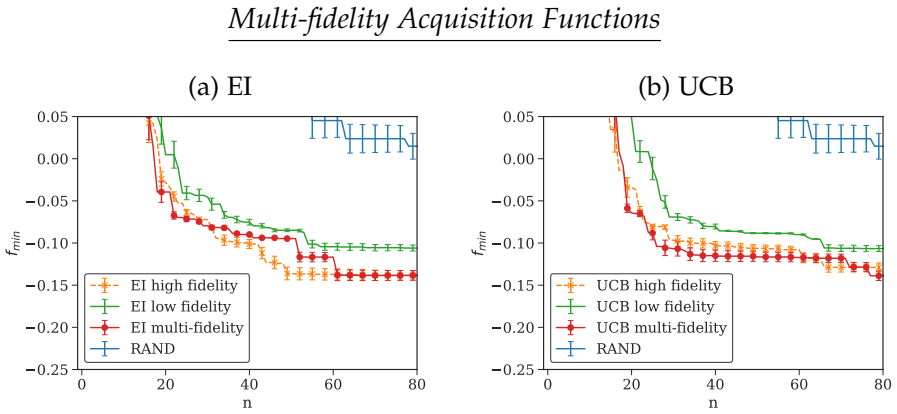


Figure 42: Basin model for overfitting (Sec. 8.5.3). We plot validation accuracy vs layer width for a small dataset, and show inference in (a) a GP and (b) our basin model. In (c-d) we show results of model complexity hyperparameter tuning experiments, comparing ProBO using a basin model with BO using GPs. Curves are averaged over 10 trials, and error bars represent one standard error.

standard BO with GPs. In this optimization task, the landscape around the inflection point (of under to over fitting) can be very steep, which may hurt the performance of GP models. In contrast, the basin model can capture this shape and quickly identify the inflection point via inferences about μ.

8.5.4 *Multi-fidelity Acquisition Optimization*

We empirically assess our multi-fidelity acquisition function optimization algorithm (Sec. 8.4.5). Our goal is to demonstrate that increasing the fidelity M in black box acquisitions can yield better performance in ProBO, and that our multi-fidelity method (Alg. 17) maintains the performance of the high-fidelity acquisitions while reducing the number of calls to `post` and `gen`. We perform an experiment in a two-fidelity setting, where $M \in \{10, 1000\}$, and we apply α_{MF} to EI and UCB, using a GP model and the (non-corrupted) synthetic system described in Sec. 8.5.2. Results are shown in



(c) Calls to `gen`

PPL acquisition method $a(x)$	Avg. number `gen`/$a(x)$
EI high-fidelity	1000
EI multi-fidelity	347.89
EI low-fidelity	10
UCB high-fidelity	1000
UCB multi-fidelity	324.65
UCB low-fidelity	10

Figure 43: Results on α_{MF} experiments (Sec. 8.5.4), showing (a)-(b) ProBO using α_{MF} (Alg. 17) vs using a fixed high-fidelity α ($M = 1000$) and a fixed low-fidelity α ($M = 10$). Here, α_{MF} performs competitively with the high-fidelity α, while low fidelity α performs worse. In (c) we show the average number of `post/gen` calls per evaluation of α. We see that the α_{MF} reduces the number of calls. Curves are averaged over 10 trials, and error bars represent one standard error.

Fig. 43 (a)-(c), where we compare high-fidelity α ($M = 1000$), low-fidelity α ($M = 10$), and multi-fidelity α_{MF}, for EI and UCB acquisitions. For both, the high-fidelity and multi-fidelity methods show comparable performance, while the low-fidelity method performs worse. We also see in Fig. 43 (c) that the multi-fidelity method reduces the number of calls to `post/gen` by a factor of 3, on average, relative to the high fidelity method.

8.5.5 *Structured Models for Multi-task and Contextual BO, and Model Ensembles*

We may want to optimize multiple systems jointly, where there is some known relation between the systems. In some instances, we have a finite set of systems (multi-task BO) and in some cases systems are each indexed by a context vector $c \in \mathbb{R}^d$ (contextual BO). We develop a model that can incorporate prior structure about the relationship among these systems. Our model warps a latent model based on

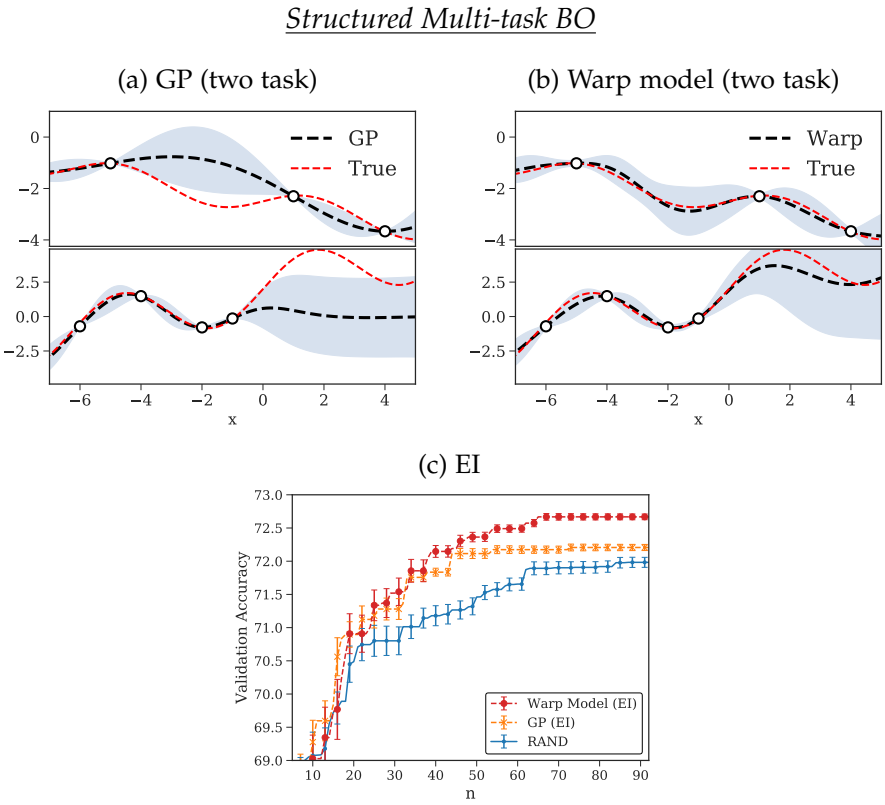


Figure 44: Structured multi-task BO (Sec. 8.5.5). We show (a) independent GPs and (b) our warp model, in a two-task setting (task one on top, task two on bottom). In (c) we show results for structured multi-task BO on a neural network hyperparameter search problem (details in Sec. 8.6). Curves are averaged over 10 trials, and error bars represent one standard error.

context/task-specific parameters, so we call this a warp model. We show inference in this model in Fig. 44 (b). In Sec. 8.6 we define this model, and describe experimental results shown in Fig. 44 (c).

Alternatively, we may have multiple models that capture different aspects of a system, or we may want to incorporate information given by, for instance, a parametric model (e.g. a model with a specific trend, shape, or specialty for a subset of the data) into a nonparametric model (e.g. a GP, which is highly flexible, but has fewer assumptions). To incorporate multiple sources of information or bring in side information, we want a valid way to create ensembles of multiple PPL models. We develop strategies to combine the posterior predictive densities of multiple PPL models, using only our three PPL operations. We describe this in Sec. 8.7.

8.6 STRUCTURED MODELS FOR MULTI-TASK AND CONTEXTUAL BO

In this section we provide details about our models for multi-task and contextual BO, described in Sec. 8.5.5 and shown in Fig. 44. Many prior methods have been proposed for for multi-task BO [180] and contextual BO [105], though these often focus on new acquisition strategies for GP models. Here we propose a model for the case where we have some structured prior information about the relation between systems, or some parametric relationship that we want to incorporate into our model.

We first consider the multi-task setting and then extend this to the contextual setting. Suppose that we have T tasks (i.e. subsystems to optimize) with data subsets $\mathcal{D} = \{\mathcal{D}_1, \ldots, \mathcal{D}_T\}$, where $\mathcal{D}_t$ has data $\{x_{t,i}, y_{t,i}\}_{i=1}^{n_t}$, and where n_t denotes the number of observations in the t^{th} task. For each $(x_{t,i}, y_{t,i})$ pair within $\mathcal{D}$, we have a latent variable $z_{t,i} \in \mathcal{Z}$. Additionally, for each task t, we have task-specific latent "warp" variables, denoted w_t.

Given these, we define our warp model to be

1. For $t = 1, \ldots, T$:
   a) $w_t \sim \text{Prior}(w)$
   b) For $i = 1, \ldots, n_t$:
      i. $z_{t,i} \sim p(z|x_{t,i})$
      ii. $y_{t,i} \sim p(y|z_{t,i}, x_{t,i}, w_t)$

We call $p(z|x_{t,i})$ our latent model, and $p(y|z_{t,i}, x_{t,i}, w_t)$ our warp model, which is parameterized by warp variables w_t. Intuitively, we can think of the variables $z_{t,i}$ as latent "unwarped" versions of observations $y_{t,i}$, all in a single task. Likewise, we can intuitively think of observed variables $y_{t,i}$ as "warped versions of $z_{t,i}$", where w_t dictates the warping for each subset of data $\mathcal{D}_t$.

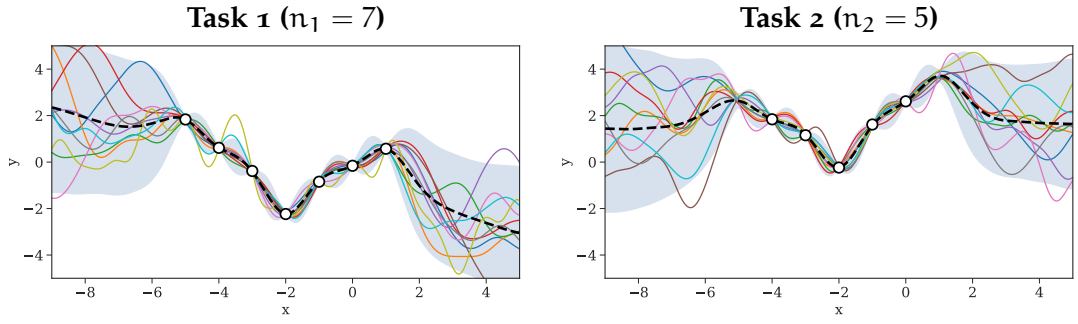


Figure 45: Warp model inference on tasks one (left) and two (right), where $n_1 = 7$ and $n_2 = 5$. This warp model assumes a linear warp with respect to both the latent variables z and inputs x. Posterior mean, posterior samples, and posterior predictive distribution are shown.

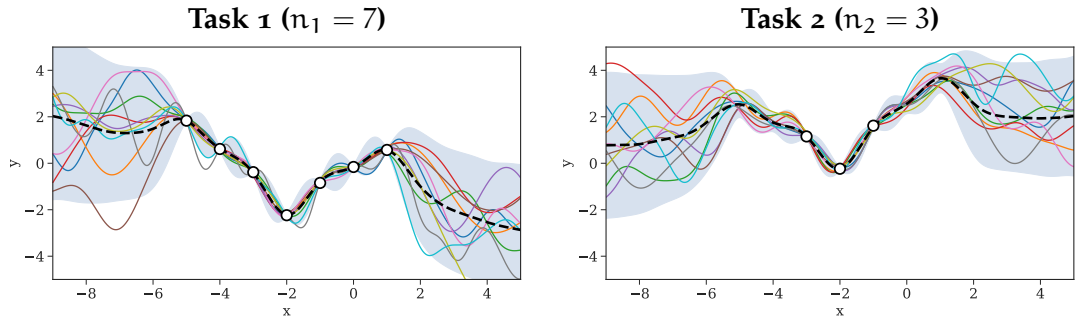


Figure 46: Warp model inference on tasks one (left) and two (right), where $n_1 = 7$ and n_2 is reduced to $n_2 = 3$. This warp model assumes a linear warp with respect to both the latent variables z and inputs x. Posterior mean, posterior samples, and posterior predictive distribution are shown. Here, we see more uncertainty around the two removed points in task two, relative to Fig. 45.

We now give a concrete instantiation of this model. Let the latent model be $p(z|x_{t,i}) = \mathcal{GP}(\mu(x), k(x, x'))$, i.e. we put a Gaussian process prior on the latent variables z. For a given task, let the warping model for y be a linear function of both z and x (with some added noise), where warping parameters w are parameters of this linear model, i.e. $y_{t,i} \sim w_0 + w_1 x_{t,i} + w_2 z_{t,i} + \epsilon$.

Intuitively, this model assumes that there is a latent GP, which is warped via a linear model of both the GP output z and input x to yield observations y for a given task (and that there is a separate warp for each task). We illustrate this model in Fig. 45 and 46 (where $n_1 = 7$ in both, $n_2 = 5$ in the former, and $n_2 = 3$ in the latter). As we remove points x in task two, we see more uncertainty in the posterior predictive distribution at these points.

We can also extend this warp model for use in a contextual optimization setting, where we want to jointly optimize over a set of systems each indexed by a context vector $c \in \mathbb{R}^d$. In practice, we observe the context c_i for input $x_i \in \mathcal{X}$, and therefore perform inference on a dataset $\mathcal{D}_n = \{x_i, c_i, y_i\}_{i=1}^n$.

To allow for this, we simply let our warp model also depend on c, i.e. let the warp model be $p(y|z_{t,i}, x_{t,i}, w_t, c_{t,i})$. Intuitively, this model assumes that there is a single latent system, which is warped by various factors (e.g. the context variables) to produce observations.

8.6.1 *Empirical Results*

Here we describe the empirical results shown in Fig. 44 (c). We aim to perform the neural architecture and hyperparameter search task from Sec. 8.5.1, but for two different settings, each with a unique preset batch size. Based on prior observations, we believe that the validation accuracy of both systems at a given query x can be accurately modeled as a linear transformation of some common latent system, and

we apply the warp model described above. We compare ProBO using this warp model with a single GP model over the full space of tasks and inputs. We show results in Fig. 44 (c), where we plot the best validation accuracy found *over both tasks* vs iteration n. Both methods use the EI acquisition function, and we compare these against a baseline that chooses queries uniformly at random. Here, ProBO with the warp model is able to find a query with a better maximum validation accuracy, relative to standard BO with GP model.

8.7 ENSEMBLES OF PPL MODELS WITHIN PROBO

We may have multiple models that capture different aspects of a system, or we may want to incorporate information given by, for instance, a parametric PPL model (e.g. a model with a specific trend, shape, or specialty for a subset of the data) into a nonparametric PPL model (e.g. a GP, which is flexible, but has fewer assumptions).

To incorporate multiple sources of information or bring in side information, we want a valid way to create ensembles of multiple models. Here, we develop a method to combine the posterior predictive densities of multiple PPL models, using only our three PPL operations. Our procedure constructs a model similar to a product of experts model [82], and we call our strategy a Bayesian product of experts (BPoE). This model can then be used in our ProBO framework.

As an example, we show an ensemble of two models, $\mathcal{M}_1$ and $\mathcal{M}_2$, though this could be extended to an arbitrarily large group of models. Assume $\mathcal{M}_1$ and $\mathcal{M}_2$ are both plausible models for a dataset $\mathcal{D}_n = \{(x_i, y_i)\}_{i=1}^n$.

Let $\mathcal{M}_1$ have likelihood $p_1(\mathcal{D}_n|z_1) = \prod_{i=1}^n p_1(y_i|z_1; x_i)$, where $z_1 \in \mathcal{Z}_1$ are latent variables with prior $p_1(z_1)$. We define the joint model PDF for $\mathcal{M}_1$ to be $p_1(\mathcal{D}_n, z_1) = p_1(z_1) \prod_{i=1}^n p_1(y_i|z_1; x_i)$. The posterior (conditional) PDF for $\mathcal{M}_1$ can then be written $p_1(z_1|\mathcal{D}_n) = p_1(\mathcal{D}_n, z_1)/p_1(\mathcal{D}_n)$. We can write the posterior predictive PDF for $\mathcal{M}_1$ as

$$p_1(y|\mathcal{D}_n; x) = \mathbb{E}_{p_1(z_1|\mathcal{D}_n)}\left[p_1(y|z_1; x)\right]. \tag{92}$$

Similarly, let $\mathcal{M}_2$ have likelihood $p_2(\mathcal{D}_n|z_2) = \prod_{i=1}^n p_2(y_i|z_2; x_i)$, where $z_2 \in \mathcal{Z}_2$ are latent variables with prior PDF $p_2(z_2)$. We define the joint model PDF for $\mathcal{M}_2$ to be $p_2(\mathcal{D}_n, z_2) = p_2(z_2) \prod_{i=1}^n p_2(y_i|z_2; x_i)$, the posterior (conditional) PDF to be $p_2(z_2|\mathcal{D}_n) = p_2(\mathcal{D}_n, z_2)/p_2(\mathcal{D}_n)$, and the posterior predictive PDF to be

$$p_2(y|\mathcal{D}_n; x) = \mathbb{E}_{p_2(z_2|\mathcal{D}_n)}\left[p_2(y|z_2; x)\right]. \tag{93}$$

Note that $z_1 \in \mathcal{Z}_1$ and $z_2 \in \mathcal{Z}_2$ need not be in the same space nor related.

Given models $\mathcal{M}_1$ and $\mathcal{M}_2$, we define the Bayesian Product of Experts (BPoE) ensemble model, $\mathcal{M}_e$, with latent variables $z = (z_1, z_2) \in \mathcal{Z}_1 \times \mathcal{Z}_2$, to be the model with posterior predictive density

$$p(y|\mathcal{D}_n; x) \propto p_1(y|\mathcal{D}_n; x) p_2(y|\mathcal{D}_n; x). \tag{94}$$

The posterior predictive PDF for the BPoE ensemble model $\mathcal{M}_e$ is proportional to the product of the posterior predictive PDFs of the constituent models $\mathcal{M}_1$ and $\mathcal{M}_2$. Note that this uses the product of expert assumption [82] on y, which intuitively means that $p(y|\mathcal{D}_n; x)$ is high where both $p_1(y|\mathcal{D}_n; x)$ and $p_2(y|\mathcal{D}_n; x)$ agree (i.e. an "and" operation). Intuitively, this model gives a stronger posterior belief over y in regions where both models have consensus, and weaker posterior belief over y in regions given by only one (or neither) of the models.

Given this model, we need an algorithm for computing and using the posterior predictive for $\mathcal{M}_e$ within the ProBO framework. In our acquisition algorithms, we use `gen` to generate samples from predictive distributions. We can integrate these with combination algorithms from the embarrassingly parallel MCMC literature [142, 145, 194] to develop an algorithm that generates samples from the posterior predictive of the ensemble model $\mathcal{M}_e$ and uses these in a new acquisition algorithm. We give this procedure in Alg. 19, which introduces the `ensemble` operation. Note that `ensemble` takes as input two operations `gen1` and `gen2` (assumed to be from two PPL models), as well as two sets of M posterior samples z_1^M and z_2^M (assumed to come from calls to `post1` and `post2` from the two PPL models). Also note that in Alg. 19 we've used $\text{Combine}(y_1, y_2)$ to denote a combination algorithm, which we detail in Sec. 8.7.2.

We can now swap the `ensemble` operation in for the `gen` operation in Algs. 13-16. Note that the BPoE allows us to easily ensemble models written in different PPLs. For example, a hierarchical regression model written in Stan [38] using Hamiltonian Monte Carlo for inference could be combined with a deep Bayesian neural network written in Pyro [21] using variational inference and with a GP written in GPy [77] using exact inference.

Algorithmus 19 : PPL model ensemble with BPoE, `ensemble`$(x, $`gen1`$, $`gen2`$, z_1^M, z_2^M)$ from Neiswanger et al. [139]

Input : Inputs listed here.
Output : Outputs listed here.

for $m = 1, \ldots, M$ **do**
  $s_1, s_2 \sim \text{Unif}(\{1, \ldots, M\})$
  $\tilde{y}_{1,m} \leftarrow$ `gen1`(x, z_{1,s_1}, s_1)
  $\tilde{y}_{2,m} \leftarrow$ `gen2`(x, z_{2,s_2}, s_2)
$y_{1:M} \leftarrow \text{Combine}(\tilde{y}_{1,M}, \tilde{y}_{2,M})$
Return $y_{1:M}$.

8.7.1 *Example: Combining Phase-Shift and GP Models.*

We describe an example and illustrate it in Fig. 47. Suppose we expect a few phase shifts in our input space $\mathcal{X}$, which partition $\mathcal{X}$ into regions with uniform output. We

can model this system with $y \sim \mathcal{N}(y | \sum_{k=1}^{K} \text{logistic}(x; m_k, s_k, \mu_k) + b_k, \sigma^2)$, where latent variables $m_{1:K}$, $s_{1:K}$, $\mu_{1:K}$, and $b_{1:K}$ are assigned appropriate priors, and where $\text{logistic}(x; m, s, \mu) = \frac{m}{1+\exp(-s(x-\mu))}$. This model may accurately describe general trends in the system, but it may be ultimately misspecified, and underfit as the number of observations n grows.

Alternatively, we could model this system as a black box using a Gaussian process. The GP posterior predictive may converge to the correct landscape given enough data, but it is nonparametric, and does not encode our assumptions.

We can use the BPoE model to combine both the phase shift and GP models. We see in Fig. 47 that when $n = 2$ (first row), the BPoE model resembles the phase shift model, but when $n = 50$ (second row), it more closely resembles the true landscape modeled by the GP.

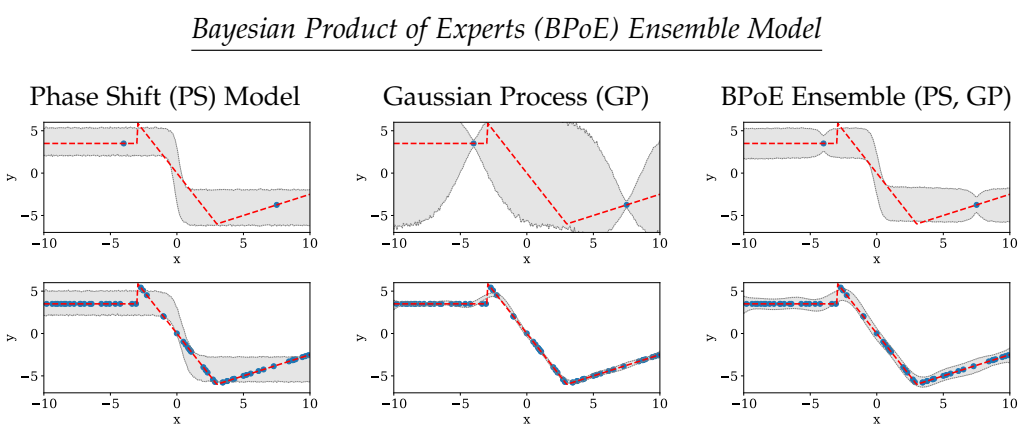


Figure 47: Visualization of the Bayesian product of experts (BPoE) ensemble model (column three) of a phase shift (PS) model (column one), defined in Sec. 8.7.1, and a GP (column two). In the first row ($n = 2$), when n is small, the BPoE ensemble more closely resembles the PS model. In the second row ($n = 50$), when n is larger, the BPoE ensemble more closely resembles the GP model, and both accurately reflect the true landscape (red dashed line). In all figures, the posterior predictive is shown in gray.

8.7.2 *Combination Algorithms for the* `ensemble` *Operation (Alg. 19)*

We make use of combination algorithms from the embarrassingly parallel MCMC literature [142, 145, 194], to define the `ensemble` operation (Alg. 19) for use in applying the ProBO framework to a BPoE model. We describe these combination algorithms here in more detail.

For convenience, we describe these methods for two Bayesian models, $\mathcal{M}_1$ and $\mathcal{M}_2$, though these methods apply similarly to an abitrarily large set of models.

The goal of these combination methods is to combine a set of M samples $y_{1,1:M} \sim p_1(y|\mathcal{D}_n; x)$ from the posterior predictive distribution of a model $\mathcal{M}_1$, with a disjoint

set of M samples $y_{2,1:M} \sim p_2(y|\mathcal{D}_n; x)$ from the posterior predictive distribution of a model $\mathcal{M}_2$, to produce samples

$$y_{3,1:M} \sim p(y|\mathcal{D}_n; x) \propto p_1(y|\mathcal{D}_n; x) p_2(y|\mathcal{D}_n; x), \tag{95}$$

where $p(y|\mathcal{D}_n; x)$ denotes the posterior predictive distribution of a BPoE ensemble model $\mathcal{M}_e$, with constituent models $\mathcal{M}_1$ and $\mathcal{M}_2$.

We use the notation Combine$(y_{1,1:M}, y_{2,1:M})$ to denote a combination algorithm. We give a combination algorithm in Alg. 20 for our setting based on a combination algorithm presented in [142].

Algorithmus 20 : Combine sample sets, Combine$(y_{1,1:M}, y_{2,1:M})$ from Neiswanger et al. [139]

Input : Inputs listed here.
Output : Outputs listed here.

$t_1, t_2 \overset{\text{iid}}{\sim} \text{Unif}(\{1, \ldots, M\})$
for $i = 1, \ldots, M$ **do**
  $c_1, c_2 \overset{\text{iid}}{\sim} \text{Unif}(\{1, \ldots, M\})$
  $u \sim \text{Unif}([0, 1])$
  if $u > \frac{w_{(c_1,c_2)}}{w_{(t_1,t_2)}}$ **then**
    $t_1 \leftarrow c_1$
    $t_2 \leftarrow c_2$
  $y_{3,i} \sim \mathcal{N}\left(\bar{y}_{(t_1,t_2)}, \frac{i^{-1/2}}{2}\right)$
Return $y_{3,1:M}$.

We must define a couple of terms used in Alg. 20. The mean output $\bar{y}_{(t_1,t_2)}$, for indices $t_1, t_2 \in \{1, \ldots, M\}$, is defined to be

$$\bar{y}_{(t_1,t_2)} = \frac{1}{2}\left(y_{1,t_1} + y_{2,t_2}\right), \tag{96}$$

and weights $w_{(t_1,t_2)}$ (alternatively, $w_{(c_1,c_2)}$), for indices $t_1, t_2 \in \{1, \ldots, M\}$, are defined to be

$$w_{(t_1,t_2)} = \mathcal{N}\left(y_{1,t_1}|\bar{y}_{(t_1,t_2)}, i^{-1/2}\right) \mathcal{N}\left(y_{2,t_2}|\bar{y}_{(t_1,t_2)}, i^{-1/2}\right). \tag{97}$$

Note that this Combine$(y_{1,1:M}, y_{2,1:M})$ algorithm (Alg. 20) holds for sample sets from two arbitrary posterior predictive distributions $p_1(y|\mathcal{D}_n; x)$ and $p_2(y|\mathcal{D}_n; x)$, without any parametric assumptions such as Gaussianity.

8.8 CONCLUSION

In this chapter we presented ProBO, a system for performing Bayesian optimization using models from any probabilistic programming language. We developed algorithms to compute acquisition functions using common PPL operations (without

requiring model-specific derivations), and showed how to efficiently optimize these functions. We presented a few models for challenging optimization scenarios, and we demonstrated promising empirical results on the tasks of BO with state observations, contaminated BO, BO with prior structure on the objective function, and structured multi-task BO, where we were able to drop-in models from existing PPL implementations.

Part IV

CONCLUSIONS AND FUTURE DIRECTIONS

9 CONCLUSION

9.1 SUMMARY

In this thesis, we began by summarizing advantages and relevant use cases for probabilistic modeling, discussed a few broad classes of models and inference algorithms, and introduced new probabilistic models for text, network, and visual data. We then described post-inference methods for scalable computation in distributed and sequential data settings, efficient incorporation of and inference with complex prior information, and automated use of diverse inference results in downstream applications, where we focused on the task of model-based sequential decision making and optimization.

When developing methods in this thesis, we aimed to target the following challenges and problem settings.

- **Big data.** Large datasets can increase the cost of many common approximate inference procedures, such as Markov chain Monte Carlo (MCMC) and variational inference (VI), and may increase the time needed to produce a valid posterior approximation in probabilistic models.
- **Distributed data.** Data may be collected, processed, or stored by a collection of agents. For example, data might be split onto multiple machines due to its large size, or kept separate due to privacy concerns. In addition to distributed data, we may also work in a streaming data setting, where data is processed by multiple machines but never stored.
- **Multiple inferences.** We may wish to carry out multiple related inferences in an efficient manner. An example of this is sequential inference, where we wish to perform inference over multiple time steps, where data is added at each time step. Another example is the task of prior sensitivity analysis, where we wish to carry out multiple inferences to see how results change under different prior assumptions.
- **Incorporating structure.** It may be challenging to incorporate complex structure into probabilistic modeling and inference procedures. For example, it may be more costly to perform inference in models with complex prior structure (or restrict the types of inference algorithms that can be applied), and it may be difficult to use structured models with complex posterior representations in downstream applications.

- **Automating procedures.** We would like to have procedures or software frameworks that allow for efficient and automatic inference in the above problem settings, in a wide range of probabilistic models, and that allow for a wide range of inference algorithms. We also would like to be able to automatically use the variety of possible inference results effectively in downstream applications.

In Part 1 of this thesis, we developed probabilistic models for text, network, and video data, and derived approximate sampling-based inference algorithms for these models. In Chapter 2, we introduced the latent random offset (LRO) model for link prediction in citation networks, and described an optimization strategy to learn MAP point estimates for latent variables in this model. In Chapter 3, we introduced the dependent Dirichlet process mixture of objects (DDPMO) for unsupervised detection-free tracking and object modeling in videos, and described online inference algorithms for this model using sequential Monte Carlo (SMC) and particle Markov chain Monte Carlo (PMCMC) strategies.

In Part 2, we presented algorithms for scalable approximate inference on big data and in data distributed settings. In Chapter 4, we introduced methods for embarrassingly parallel Markov chain Monte Carlo (EP-MCMC), and described algorithms for performing parallel inference using nonparametric and semiparametric density estimation. In Chapter 5, we introduced methods for embarrassingly parallel variational inference (EP-VI), and for low-communication black box variational inference (BBVI). In Chapter 6, we described methods for embarrassingly parallel inference in quasi-ergodic settings, where parallel MCMC chains might only explore a single mode of the posterior, and in dependent models or models with local latent variables.

In Part 3, we focused on methods that allow for the incorporation of structure: either prior structure in models, or model structure in model-based sequential decision making and optimization procedures. In Chapter 7, we introduced methods for prior swapping that aim to allow for efficient incorporation of potentially complex prior information. These methods also allow for efficient prior sensitivity analysis, where we can more-quickly assess how our inference results change given a set of different prior assumptions, and provide benefits to help improve embarrassingly parallel inference when the number of data partitions is unknown and for use in a sequential inference setting. In Chapter 8, we introduced methods to help allow arbitrary probabilistic models, defined via probabilistic programming languages (PPLs), to be used in Bayesian optimization and other sequential decision making procedures, and developed a system (ProBO) to carry this out in a more automated fashion.

9.2 FUTURE RESEARCH DIRECTIONS

In this section, we describe a few extensions to the work performed in this thesis, which may make for interesting future research directions.

9.2.1 *Embarrassingly Parallel Inference for Additional Latent Variable Types*

Embarrassingly parallel inference methods involve applying combination algorithms to the results of inference procedures, such as MCMC or VI algorithms. The combination algorithms presented in this thesis focused on continuous, unbounded latent variables, such as those in $\mathbb{R}^d$ space. One potential future direction is to extend these parallel inference strategies to latent variables in discrete, integral, constrained continuous, and other spaces. For example, in some popular latent variable models, we infer posterior distributions that are defined over some d-dimensional simplex. We'd like to develop procedures that allow us to use embarrassingly parallel inference methods in these model settings.

9.2.2 *Randomized Algorithms for Minimizing Combination Communication in Embarrassingly Parallel Inference*

While many of the embarrassingly parallel inference methods discussed in this thesis provide strategies to combine the local posterior approximations, there has been little focus on developing efficient methods for disseminating these local representations throughout a collection of machines or distributed agents. In the future, we'd like to use tools from randomized information dissemination (e.g. network coding gossip) protocols to spread the local inference results throughout a collection of machines in a scalable manner as the number of machines in the collection grows. We also hope to integrate the different combination algorithms into this dissemination procedure. Overall, we hope that this strategy can reduce both the amount of information that must be communicated to perform the combination step in embarrassingly parallel inference, as well as the amount of total computation needed for the combinations, all while making the combination procedure more robust to machine or communication failures.

9.2.3 *Incorporating Test Functions of Interest into Embarrassingly Parallel Inference*

A chief goal of sample-based inference is to compute an approximation of the posterior expectation of a test function h, i.e. $\mathbb{E}_{p(\theta|x^n)}[h(\theta)]$, using a Monte Carlo estimate. If we know, in advance of inference, information about the test function of interest, we can incorporate this information into our embarrassingly parallel inference methods in order to improve the quality of these methods for the given test function. For example, one possible strategy is to use importance sampling methods to compute

weights for each sample from the combined full data posterior estimate, which allows for weighted-sample-based estimates of expectations with respect to this full posterior. We hope to also take this into account during the subposterior inference phase, and develop procedures that more quickly converge to accurate estimates for a given test function.

9.2.4 *Local Posterior Revisions*

We have many procedures that make or leverage approximations to either the full posterior or some local posterior. The performances of many of the post-inference methods described in this thesis are determined by the quality of these posterior approximations. There are many cases where a posterior inference algorithm will return some approximate posterior, and it may be beneficial to develop methods that can characterize, and then improve, the quality of these returned posterior inference results. For example, these methods could aim to improve the quality of the posterior approximation in some local region of the parameter space. In some cases, improving the posterior approximation in certain local regions can yield large improvements on the performance of the post-inference methods. One instance of this is in embarrassingly parallel inference methods, where improving the approximation accuracy for each subposterior in the region where the subposterior product has high mass can greatly increase the accuracy of the subsequent combination procedure. Furthermore, we aim to develop these methods with a focus on scalable computation, developing procedures that can be run in parallel, and with minimal communication in a distributed environment.

9.2.5 *VI-Specific Prior Swapping*

In this thesis, we have described how to run prior swapping procedures (from Chapter 7) on any false posterior PDF, which can include a false posterior approximation computed via variational inference. In this case, the false posterior approximation is a member of a variational family that minimizes the KL divergence to the false posterior PDF. Suppose we run our prior swapping procedure on this false posterior approximation, which yields some target posterior approximation. We would like to guarantee that, given this procedure, the resulting approximate target posterior also minimizes some distance (ideally, the KL divergence as is typically used in variational inference) between the prior swapping approximate posterior result and the target posterior.

9.2.6 *Simulator-based Models in Sequential Decision Making and Optimization*

Universal probabilistic programming languages [112, 121, 202] allow for the development of models defined by arbitrary forward simulators, and aim to provide auto-

matic inference for these models. They comprise a very broad class of models, and have the potential to incorporate sophisticated custom-built simulations of a broad array of systems. We think there may be the potential for running ProBO [138] (Chapter 8) with models involving complex simulators of real world phenomena, and that this simulation-guided Bayesian optimization is an interesting avenue for future work.

9.2.7 *Implementation and Software Release*

PROBABILISTIC PROGRAMMING IMPLEMENTATION We plan to implement post-inference methods for efficient and scalable inference, such as embarrassingly parallel inference and prior swapping, in probabilistic programming frameworks. These frameworks aim to allow for Bayesian inference in an automatic fashion, with very little derivation on the part of the user. Examples of frameworks that allow for automated Bayesian inference include Stan [38], Edward [186], PyMC3 [161], and many others. Typically, the user must only specify the form of the model and the type of inference algorithm they wish to use. Many of these frameworks have predefined routines for MCMC and VI (and stochastic gradient versions of these algorithms). We aim to implement our post-inference methods in these probabilistic programming frameworks, using the predefined and black box inference routines, to allow for automatic deployment of our methods.

EMBARRASSINGLY PARALLEL BAYES We are developing a package for embarrassingly parallel inference and prior swapping as open source software, and it will be available at `https://github.com/willieneis/embarrassingly-parallel-bayes`.

PROBO We are developing ProBO as open source software, and it will be available at `https://github.com/willieneis/ProBO`.

BIBLIOGRAPHY

[1] Lada A Adamic and Eytan Adar. "Friends and neighbors on the web." In: *Social networks* 25.3 (2003), pp. 211–230.

[2] Alekh Agarwal and John C Duchi. "Distributed delayed stochastic optimization." In: *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*. IEEE. 2012, pp. 5451–5452.

[3] Sungjin Ahn, Anoop Korattikara, and Max Welling. "Bayesian Posterior Sampling via Stochastic Gradient Fisher Scoring." In: *Proceedings of the 29th International Conference on Machine Learning*. 2012, pp. 1591–1598.

[4] Edoardo M Airoldi, David M Blei, Stephen E Fienberg, and Eric P Xing. "Mixed membership stochastic blockmodels." In: *The Journal of Machine Learning Research* 9 (2008), pp. 1981–2014.

[5] A. Alahi, L. Jacques, Y. Boursier, and P. Vandergheynst. "Sparsity-driven people localization algorithm: Evaluation in crowded scenes environments." In: *Performance Evaluation of Tracking and Surveillance (PETS-Winter), 2009 Twelfth IEEE International Workshop on*. IEEE. 2009, pp. 1–8.

[6] Omid Alipourfard, Hongqiang Harry Liu, Jianshu Chen, Shivaram Venkataraman, Minlan Yu, and Ming Zhang. "Cherrypick: Adaptively unearthing the best cloud configurations for big data analytics." In: *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*. 2017, pp. 469–482.

[7] S. Ambikasaran, D. Foreman-Mackey, L. Greengard, D. W. Hogg, and M. O'Neil. "Fast Direct Methods for Gaussian Processes and the Analysis of NASA Kepler Mission Data." In: *arXiv preprint arXiv:1403.6015* (Mar. 2014).

[8] Christophe Andrieu, Arnaud Doucet, and Roman Holenstein. "Particle markov chain monte carlo methods." In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 72.3 (2010), pp. 269–342.

[9] Charles E Antoniak. "Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems." In: *The annals of statistics* (1974), pp. 1152–1174.

[10] D. Arsic, A. Lyutskanov, G. Rigoll, and B. Kwolek. "Multi camera person tracking applying a graph-cuts based foreground segmentation in a homography framework." In: *Performance Evaluation of Tracking and Surveillance (PETS-Winter), 2009 Twelfth IEEE International Workshop on*. IEEE. 2009, pp. 1–8.

[11] Raul Astudillo and P Frazier. "Multi-attribute bayesian optimization under utility uncertainty." In: *Proceedings of the NIPS Workshop on Bayesian Optimization*. 2017.

[12] Yves F Atchadé and François Perron. "Improving on the independent Metropolis-Hastings algorithm." In: *Statistica Sinica* 15.1 (2005), pp. 3–18.

[13] Boris Babenko, Ming-Hsuan Yang, and Serge Belongie. "Robust object tracking with online multiple instance learning." In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 33.8 (2011), pp. 1619–1632.

[14] Rémi Bardenet, Arnaud Doucet, and Chris Holmes. "On Markov chain Monte Carlo methods for tall data." In: *The Journal of Machine Learning Research* 18.1 (2017), pp. 1515–1557.

[15] John L Barron, David J Fleet, and SS Beauchemin. "Performance of optical flow techniques." In: *International journal of computer vision* 12.1 (1994), pp. 43–77.

[16] Matthew James Beal. "Variational algorithms for approximate Bayesian inference." PhD thesis. University of London, 2003.

[17] J. Berclaz, F. Fleuret, and P. Fua. "Multiple object tracking using flow linear programming." In: *Performance Evaluation of Tracking and Surveillance (PETS-Winter), 2009 Twelfth IEEE International Workshop on*. IEEE. 2009, pp. 1–8.

[18] D. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1999.

[19] Julian Besag, Peter Green, David Higdon, and Kerrie Mengersen. "Bayesian computation and stochastic systems." In: *Statistical science* (1995), pp. 3–41.

[20] Steven Bethard and Dan Jurafsky. "Who should I cite: learning literature search models from citation behavior." In: *International Conference on Information and Knowledge Management*. 2010, pp. 609–618.

[21] 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." In: *Journal of Machine Learning Research* (2018).

[22] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[23] CL Blake and CJ Merz. *UCI repository of machine learning databases. Irvine, CA: University of California, Department of Information and Computer Science*. 1998.

[24] D. Blei. "Probabilistic Topic Models." In: *Communications of the ACM* 55.4 (2012), pp. 77–84.

[25] D. Blei and J. Lafferty. "Topic Models." In: *Text Mining: Theory and Applications*. Ed. by A. Srivastava and M. Sahami. Taylor and Francis, 2009.

[26] D. Blei, A. Ng, and M. Jordan. "Latent Dirichlet Allocation." In: *Journal of Machine Learning Research* 3 (Jan. 2003), pp. 993–1022.

[27] David M Blei and Michael I Jordan. "Variational methods for the Dirichlet process." In: *Proceedings of the twenty-first international conference on Machine learning*. ACM. 2004, p. 12.

[28] David M Blei, Andrew Y Ng, and Michael I Jordan. "Latent dirichlet allocation." In: *The Journal of Machine Learning Research* 3 (2003), pp. 993–1022.

[29] D.S. Bolme, Y.M. Lui, BA Draper, and JR Beveridge. "Simple real-time human detection using a single correlation filter." In: *Performance Evaluation of Tracking and Surveillance (PETS-Winter), 2009 Twelfth IEEE International Workshop on*. IEEE. 2009, pp. 1–8.

[30] Luke Bornn, Arnaud Doucet, and Raphael Gottardo. "An efficient computational approach for prior sensitivity analysis and cross-validation." In: *Canadian Journal of Statistics* 38.1 (2010), pp. 47–64.

[31] Michael D Breitenstein, Fabian Reichlin, Bastian Leibe, Esther Koller-Meier, and Luc Van Gool. "Markovian tracking-by-detection from a single, uncalibrated camera." In: (2009).

[32] 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." In: *arXiv preprint arXiv:1012.2599* (2010).

[33] Tamara Broderick, Nicholas Boyd, Andre Wibisono, Ashia C Wilson, and Michael Jordan. "Streaming Variational Bayes." In: *Advances in Neural Information Processing Systems*. 2013, pp. 1727–1735.

[34] Thomas Brox and Jitendra Malik. "Large displacement optical flow: descriptor matching in variational motion estimation." In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 33.3 (2011), pp. 500–513.

[35] Trevor Campbell and Jonathan How. "Approximate Decentralized Bayesian Inference." In: 2014.

[36] F. Caron, M. Davy, and A. Doucet. "Generalized Polya Urn for Time-varying Dirichlet Process Mixtures." In: *23rd Conference on Uncertainty in Artificial Intelligence (UAI'2007), Vancouver, Canada, July 2007*. 2007.

[37] François Caron, Willie Neiswanger, Frank Wood, Arnaud Doucet, and Manuel Davy. "Generalized Pólya urn for time-varying Pitman-Yor processes." In: *Journal of Machine Learning Research (JMLR)* 18.27 (2017).

[38] Bob Carpenter, Andrew Gelman, Matt Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus A Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell. "Stan: a probabilistic programming language." In: *Journal of Statistical Software* (2015).

[39] Allison June-Barlow Chaney and David M. Blei. "Visualizing Topic Models." In: *The International AAAI Conference on Weblogs and Social Media*. 2012.

[40] J. Chang and D. Blei. "Relational Topic Models for Document Networks." In: *Artificial Intelligence and Statistics*. 2009.

[41] J. Chang, J. Boyd-Graber, C. Wang, S. Gerrish, and D. Blei. "Reading Tea Leaves: How Humans Interpret Topic Models." In: *Advances in Neural Information Processing Systems (NIPS)*. 2009.

[42] Anil M Cheriyadat and Richard J Radke. "Non-negative matrix factorization of partial track data for motion segmentation." In: *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE. 2009, pp. 865–872.

[43] D. Conte, P. Foggia, G. Percannella, and M. Vento. "Performance evaluation of a people tracking system on pets2009 database." In: *Advanced Video and Signal Based Surveillance (AVSS), 2010 Seventh IEEE International Conference on*. IEEE. 2010, pp. 119–126.

[44] Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

[45] Valentin Dalibard, Michael Schaarschmidt, and Eiko Yoneki. "BOAT: Building auto-tuners with structured Bayesian optimization." In: *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee. 2017, pp. 479–488.

[46] Jeffrey Dean and Sanjay Ghemawat. "MapReduce: simplified data processing on large clusters." In: *Communications of the ACM* 51.1 (2008), pp. 107–113.

[47] Ian Dewancker, Michael McCourt, Scott Clark, Patrick Hayes, Alexandra Johnson, and George Ke. "A Stratified Analysis of Bayesian Optimization Methods." In: *arXiv preprint arXiv:1603.09441* (2016).

[48] Persi Diaconis, Donald Ylvisaker, et al. "Conjugate priors for exponential families." In: *The Annals of statistics* 7.2 (1979), pp. 269–281.

[49] Jean Diebolt and Christian P Robert. "Estimation of finite mixture distributions through Bayesian sampling." In: *Journal of the Royal Statistical Society. Series B (Methodological)* (1994), pp. 363–375.

[50] Joshua V Dillon, Ian Langmore, Dustin Tran, Eugene Brevdo, Srinivas Vasudevan, Dave Moore, Brian Patton, Alex Alemi, Matt Hoffman, and Rif A Saurous. "TensorFlow Distributions." In: *arXiv preprint arXiv:1711.10604* (2017).

[51] D. Doermann and D. Mihalcik. "Tools and techniques for video performance evaluation." In: *Pattern Recognition, 2000. Proceedings. 15th International Conference on*. Vol. 4. IEEE. 2000, pp. 167–170.

[52] Randal Douc and Olivier Cappé. "Comparison of resampling schemes for particle filtering." In: *Image and Signal Processing and Analysis, 2005. ISPA 2005. Proceedings of the 4th International Symposium on*. IEEE. 2005, pp. 64–69.

[53] Arnaud Doucet, Simon Godsill, and Christophe Andrieu. "On sequential Monte Carlo sampling methods for Bayesian filtering." In: *Statistics and computing* 10.3 (2000), pp. 197–208.

[54] Bradley Efron. "Bootstrap methods: another look at the jackknife." In: *Breakthroughs in statistics*. Springer, 1992, pp. 569–593.

[55] A. Ellis and J. Ferryman. "PETS2010 and PETS2009 Evaluation of Results Using Individual Ground Truthed Single Views." In: *Advanced Video and Signal Based Surveillance (AVSS), 2010 Seventh IEEE International Conference on*. IEEE. 2010, pp. 135–142.

[56] Rémi Emonet, Jagannadan Varadarajan, and J-M Odobez. "Extracting and locating temporal motifs in video scenes using a hierarchical non parametric Bayesian model." In: *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE. 2011, pp. 3233–3240.

[57] Shai Fine, Yoram Singer, and Naftali Tishby. "The hierarchical hidden Markov model: Analysis and applications." In: *Machine learning* 32.1 (1998), pp. 41–62.

[58] Alexander IJ Forrester, András Sóbester, and Andy J Keane. "Multi-fidelity optimization via surrogate modelling." In: *Proceedings of the royal society of london a: mathematical, physical and engineering sciences*. Vol. 463. 2088. The Royal Society. 2007, pp. 3251–3269.

[59] Katerina Fragkiadaki and Jianbo Shi. "Detection free tracking: Exploiting motion and topology for segmenting and tracking under entanglement." In: *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE. 2011, pp. 2073–2080.

[60] Jacob R Gardner, Matt J Kusner, Zhixiang Eddie Xu, Kilian Q Weinberger, and John P Cunningham. "Bayesian Optimization with Inequality Constraints." In: *ICML*. 2014, pp. 937–945.

[61] Jan Gasthaus. *Spike Sorting Using Time-Varying Dirichlet Process Mixture Models*. 2008.

[62] W. Ge and R.T. Collins. "Evaluation of sampling-based pedestrian detection for crowd counting." In: *Performance Evaluation of Tracking and Surveillance (PETS-Winter), 2009 Twelfth IEEE International Workshop on*. IEEE. 2009, pp. 1–7.

[63] Michael A Gelbart, Jasper Snoek, and Ryan P Adams. "Bayesian optimization with unknown constraints." In: *arXiv preprint arXiv:1403.5607* (2014).

[64] A. Gelman. *Bayesian data analysis*. CRC press, 2004.

[65] A. Gelman, J. Carlin, H. Stern, and D. Rubin. *Bayesian Data Analysis, Second Edition*. 2nd. London: Chapman & Hall, 2003.

[66] Andrew Gelman, Aleks Jakulin, Maria Grazia Pittau, Yu-Sung Su, et al. "A weakly informative default prior distribution for logistic and other regression models." In: *The Annals of Applied Statistics* 2.4 (2008), pp. 1360–1383.

[67] Alexander Genkin, David D Lewis, and David Madigan. "Large-scale Bayesian logistic regression for text categorization." In: *Technometrics* 49.3 (2007), pp. 291–304.

[68] Samuel J Gershman and David M Blei. "A tutorial on Bayesian nonparametric models." In: *Journal of Mathematical Psychology* 56.1 (2012), pp. 1–12.

[69] Samuel Gershman, Matt Hoffman, and David M Blei. "Nonparametric variational inference." In: *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*. 2012, pp. 663–670.

[70] Sebastian Gerwinn, Jakob H Macke, and Matthias Bethge. "Bayesian inference for generalized linear models for spiking neurons." In: *Frontiers in Computational Neuroscience* 4.12 (2010).

[71] John Geweke. "Bayesian inference in econometric models using Monte Carlo integration." In: *Econometrica: Journal of the Econometric Society* (1989), pp. 1317–1339.

[72] Zoubin Ghahramani. "An introduction to hidden Markov models and Bayesian networks." In: *Hidden Markov models: applications in computer vision*. World Scientific, 2001, pp. 9–41.

[73] Wally R Gilks, NG Best, and KKC Tan. "Adaptive rejection Metropolis sampling within Gibbs sampling." In: *Applied Statistics* (1995), pp. 455–472.

[74] Paolo Giordani and Robert Kohn. "Adaptive independent Metropolis–Hastings by fast estimation of mixtures of normals." In: *Journal of Computational and Graphical Statistics* 19.2 (2010), pp. 243–259.

[75] Daniel Golovin, Benjamin Solnik, Subhodeep Moitra, Greg Kochanski, John Karro, and D Sculley. "Google vizier: A service for black-box optimization." In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. 2017, pp. 1487–1495.

[76] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative adversarial nets." In: *Advances in neural information processing systems*. 2014, pp. 2672–2680.

[77] GPy. *GPy: A Gaussian process framework in python*. http://github.com/SheffieldML/GPy. 2012.

[78] Thomas L Griffiths and Mark Steyvers. "Finding scientific topics." In: *Proceedings of the National academy of Sciences of the United States of America* 101.Suppl 1 (2004), pp. 5228–5235.

[79] W Keith Hastings. "Monte Carlo sampling methods using Markov chains and their applications." In: *Biometrika* 57.1 (1970), pp. 97–109.

[80] Philipp Hennig and Christian J Schuler. "Entropy search for information-efficient global optimization." In: *Journal of Machine Learning Research* 13.Jun (2012), pp. 1809–1837.

[81] 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." In: (2015).

[82] Geoffrey E Hinton. "Training products of experts by minimizing contrastive divergence." In: *Neural computation* 14.8 (2002), pp. 1771–1800.

[83] Nils Lid Hjort and Ingrid K Glad. "Nonparametric density estimation with a parametric start." In: *The Annals of Statistics* (1995), pp. 882–904.

[84] Qirong Ho, James Cipar, Henggang Cui, Seunghak Lee, Jin Kyu Kim, Phillip B. Gibbons, Gregory R. Ganger, Garth Gibson, and Eric P. Xing. "More Effective Distributed ML via a Stale Synchronous Parallel Parameter Server." In: *Advances in Neural Information Processing Systems*. 2013.

[85] Qirong Ho, Jacob Eisenstein, and Eric P Xing. "Document hierarchies from text and links." In: *Proceedings of the 21st international conference on World Wide Web*. ACM. 2012, pp. 739–748.

[86] M. Hoffman, D. Blei, C. Wang, and J. Paisley. "Stochastic Variational Inference." In: *ArXiv e-prints* (June 2012). arXiv: 1206.7051 [stat.ML].

[87] Matthew D Hoffman and Andrew Gelman. "The no-U-turn sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo." In: *arXiv preprint arXiv:1111.4246* (2011).

[88] Liangjie Hong and Brian D Davison. "Empirical study of topic modeling in twitter." In: *Proceedings of the first workshop on social media analytics*. ACM. 2010, pp. 80–88.

[89] Yifan Hu, Yehuda Koren, and Chris Volinsky. "Collaborative Filtering for Implicit Feedback Datasets." In: *IEEE International Conference on Data Mining*. 2008.

[90] Peter J Huber. "Robust estimation of a location parameter." In: *Breakthroughs in statistics*. Springer, 1992, pp. 492–518.

[91] Ferenc Huszár. "Variational inference using implicit distributions." In: *arXiv preprint arXiv:1702.08235* (2017).

[92] Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. "Sequential model-based optimization for general algorithm configuration." In: *International Conference on Learning and Intelligent Optimization*. Springer. 2011, pp. 507–523.

[93] Aapo Hyvärinen. "Estimation of non-normalized statistical models by score matching." In: *Journal of Machine Learning Research* 6.Apr (2005), pp. 695–709.

[94] Matthew Johnson, James Saunderson, and Alan Willsky. "Analyzing Hogwild Parallel Gaussian Gibbs Sampling." In: *Advances in Neural Information Processing Systems*. 2013, pp. 2715–2723.

[95] Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. "An introduction to variational methods for graphical models." In: *Machine learning* 37.2 (1999), pp. 183–233.

[96] Pasi Jylänki, Jarno Vanhatalo, and Aki Vehtari. "Robust Gaussian process regression with a Student-t likelihood." In: *Journal of Machine Learning Research* 12.Nov (2011), pp. 3227–3257.

[97] Kirthevasan Kandasamy, Willie Neiswanger, Jeff Schneider, Barnabas Poczos, and Eric Xing. "Neural architecture search with bayesian optimisation and optimal transport." In: *arXiv preprint arXiv:1802.07191* (2018).

[98] R. Kasturi, D. Goldgof, P. Soundararajan, V. Manohar, J. Garofolo, R. Bowers, M. Boonstra, V. Korzhova, and J. Zhang. "Framework for performance evaluation of face, text, and vehicle detection and tracking in video: Data, metrics, and protocol." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2008), pp. 319–336.

[99] Leo Katz. "A new status index derived from sociometric analysis." In: *Psychometrika* 18.1 (1953), pp. 39–43.

[100] Diederik P Kingma and Max Welling. "Auto-encoding variational bayes." In: *arXiv preprint arXiv:1312.6114* (2013).

[101] David Knowles and Zoubin Ghahramani. "Nonparametric Bayesian sparse factor models with application to gene expression modeling." In: *The Annals of Applied Statistics* (2011), pp. 1534–1552.

[102] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[103] Anoop Korattikara, Yutian Chen, and Max Welling. "Austerity in MCMC Land: Cutting the Metropolis-Hastings Budget." In: *arXiv preprint arXiv:1304.5299* (2013).

[104] Yehuda Koren, Robert Bell, and Chris Volinsky. "Matrix factorization techniques for recommender systems." In: *Computer* 42.8 (2009), pp. 30–37.

[105] Andreas Krause and Cheng S Ong. "Contextual gaussian process bandit optimization." In: *Advances in Neural Information Processing Systems*. 2011, pp. 2447–2455.

[106] Harold J Kushner. "A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise." In: *Journal of Basic Engineering* 86.1 (1964), pp. 97–106.

[107] James T Kwok and Ryan P Adams. "Priors for diversity in generative latent variable models." In: *Advances in Neural Information Processing Systems*. 2012, pp. 2996–3004.

[108] Remi Lam and Karen Willcox. "Lookahead Bayesian optimization with inequality constraints." In: *Advances in Neural Information Processing Systems*. 2017, pp. 1890–1900.

[109] John Langford, Alex J Smola, and Martin Zinkevich. "Slow learners are fast." In: *Advances in Neural Information Processing Systems*. 2009.

[110] Kathryn Blackmond Laskey and James W Myers. "Population Markov chain Monte Carlo." In: *Machine Learning* 50.1-2 (2003), pp. 175–196.

[111] Lucien Le Cam. "Asymptotic methods in statistical decision theory." In: *New York* (1986).

[112] Tuan Anh Le, Atilim Gunes Baydin, and Frank Wood. "Inference compilation and universal probabilistic programming." In: *arXiv preprint arXiv:1610.09900* (2016).

[113] Dennis V Lindley and Adrian FM Smith. "Bayes estimates for the linear model." In: *Journal of the Royal Statistical Society: Series B (Methodological)* 34.1 (1972), pp. 1–18.

[114] Yan Liu, Alexandru Niculescu-Mizil, and Wojciech Gryc. "Topic-link LDA: joint models of topic and author community." In: *International Conference on Machine Learning*. ACM. 2009, pp. 665–672.

[115] Yucheng Low, Joseph E Gonzalez, Aapo Kyrola, Danny Bickson, Carlos E Guestrin, and Joseph Hellerstein. "Graphlab: A new framework for parallel machine learning." In: *arXiv preprint arXiv:1408.2041* (2014).

[116] DD Lucas, R Klein, J Tannahill, D Ivanova, S Brandon, D Domyancic, and Y Zhang. "Failure analysis of parameter-induced simulation crashes in climate models." In: *Geoscientific Model Development* 6.4 (2013), pp. 1157–1171.

[117] David J Lunn, Andrew Thomas, Nicky Best, and David Spiegelhalter. "WinBUGS-a Bayesian modelling framework: concepts, structure, and extensibility." In: *Statistics and computing* 10.4 (2000), pp. 325–337.

[118] Steven N MacEachern. "Dependent dirichlet processes." In: *Unpublished manuscript, Department of Statistics, The Ohio State University* (2000).

[119] David JC MacKay. "Probable networks and plausible predictions—a review of practical Bayesian methods for supervised neural networks." In: *Network: computation in neural systems* 6.3 (1995), pp. 469–505.

[120] Vikash K Mansinghka, Tejas D Kulkarni, Yura N Perov, and Josh Tenenbaum. "Approximate bayesian image interpretation using generative probabilistic graphics programs." In: *Advances in Neural Information Processing Systems*. 2013, pp. 1520–1528.

[121] Vikash Mansinghka, Daniel Selsam, and Yura Perov. "Venture: a higher-order probabilistic programming platform with programmable inference." In: *arXiv preprint arXiv:1404.0099* (2014).

[122] Vinicius Diniz Mayrink, Joseph Edward Lucas, et al. "Sparse latent factor models with interactions: Analysis of gene expression data." In: *The Annals of Applied Statistics* 7.2 (2013), pp. 799–822.

[123] Viveka Mayya, Willie Neiswanger, Ricardo Medina, Chris H Wiggins, and Michael L Dustin. "Integrative analysis of T cell motility from multi-channel microscopy data using TIAM." In: *Journal of immunological methods* 416 (2015), pp. 84–93.

[124] Peter Meer. "Kernel-based object tracking." In: *IEEE Transactions on pattern analysis and machine intelligence* 25.5 (2003).

[125] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. "Equation of state calculations by fast computing machines." In: *The journal of chemical physics* 21.6 (1953), pp. 1087–1092.

[126] Thomas P Minka. "Expectation propagation for approximate Bayesian inference." In: *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc. 2001, pp. 362–369.

[127] Tom Minka. "Infer. NET 2.5." In: *http://research. microsoft. com/infernet* (2012).

[128] Stanislav Minsker, Sanvesh Srivastava, Lizhen Lin, and David B Dunson. "Robust and scalable Bayes via a median of subset posterior measures." In: *arXiv preprint arXiv:1403.2660* (2014).

[129] Jonas Močkus. "On Bayesian methods for seeking the extremum." In: *Optimization Techniques IFIP Technical Conference*. Springer. 1975, pp. 400–404.

[130] Lawrence Murray. "Distributed Markov chain Monte Carlo." In: *Proceedings of Neural Information Processing Systems Workshop on Learning on Cores, Clusters and Clouds*. Vol. 11. 2010.

[131] Ramesh Nallapati and William Cohen. "Link-PLSA-LDA: A new unsupervised model for topics and influence of blogs." In: *International Conference for Weblogs and Social Media*. 2008.

[132] Ramesh Nallapati, William Cohen, and John Lafferty. "Parallelized variational EM for latent Dirichlet allocation: An experimental evaluation of speed and scalability." In: *Data Mining Workshops, 2007. ICDM Workshops 2007. Seventh IEEE International Conference on*. IEEE. 2007, pp. 349–354.

[133] Ramesh Nallapati, Daniel A Mcfarland, and Christopher D Manning. "Topicflow model: Unsupervised learning of topic-specific influences of hyperlinked documents." In: *International Conference on Artificial Intelligence and Statistics*. 2011, pp. 543–551.

[134] R Neal. "MCMC Using Hamiltonian Dynamics." In: *Handbook of Markov Chain Monte Carlo* (2011), pp. 113–162.

[135] Radford M Neal. *Bayesian learning for neural networks*. Vol. 118. Springer Science & Business Media, 2012.

[136] Willie Neiswanger. *Embarrassingly Parallel MCMC for Fast and Flexible Analysis of Spatiotemporal Data*. Tech. rep. 2016.

[137] Willie Neiswanger, Avinava Dubey, Chong Wang, and Eric Xing. *Embarrassingly Parallel MCMC in Quasi-ergodic Settings*. Tech. rep. 2016.

[138] Willie Neiswanger, Kirthevasan Kandasamy, Barnabas Poczos, Jeff Schneider, and Eric Xing. "Probo: a framework for using probabilistic programming in bayesian optimization." In: *arXiv preprint arXiv:1901.11515* (2019).

[139] Willie Neiswanger, Kirthevasan Kandasamy, Barnabas Poczos, Jeff Schneider, and Eric P. Xing. "ProBO: Versatile Bayesian Optimization Using Any Probabilistic Programming Language." In: *arXiv preprint arXiv:1901.11515* (2019).

[140] Willie Neiswanger, Xue Liu, and Eric Xing. "Low Communication Distributed Black Box VI." In: *The International Conference on Probabilistic Programming*. 2018.

[141] Willie Neiswanger, Chong Wang, Qirong Ho, and Eric P Xing. "Modeling citation networks using latent random offsets." In: *Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence (UAI 2014)*. 2014.

[142] Willie Neiswanger, Chong Wang, and Eric Xing. "Asymptotically Exact, Embarrassingly Parallel MCMC." In: *Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence (UAI 2014)*. 2014.

[143] Willie Neiswanger, Chong Wang, and Eric Xing. "Embarrassingly Parallel Variational Inference in Nonconjugate Models." In: *arXiv preprint arXiv:1510.04163*. 2015.

[144] Willie Neiswanger, Frank Wood, and Eric P Xing. "The dependent dirichlet process mixture of objects for detection-free tracking and object modeling." In: *The Seventeenth International Conference on Artificial Intelligence and Statistics (AISTATS 2014)*. 2014.

[145] Willie Neiswanger and Eric Xing. "Post-inference prior swapping." In: *Proceedings of the 34th International Conference on Machine Learning—Volume 70 (ICML 2017)*. 2017, pp. 2594–2602.

[146] Christopher Nemeth, Chris Sherlock, et al. "Merging MCMC subposteriors through Gaussian-process approximations." In: *Bayesian Analysis* 13.2 (2018), pp. 507–530.

[147] David Newman, Arthur Asuncion, Padhraic Smyth, and Max Welling. "Distributed algorithms for topic models." In: *The Journal of Machine Learning Research* 10 (2009), pp. 1801–1828.

[148] Mark EJ Newman. "Modularity and community structure in networks." In: *Proceedings of the National Academy of Sciences* 103.23 (2006), pp. 8577–8582.

[149] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. "Smooth sensitivity and sampling in private data analysis." In: *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*. ACM. 2007, pp. 75–84.

[150] Junier Oliva, Barnabás Póczos, and Jeff Schneider. "Distribution to distribution regression." In: *Proceedings of The 30th International Conference on Machine Learning*. 2013, pp. 1049–1057.

[151] Sam Patterson and Yee Whye Teh. "Stochastic Gradient Riemannian Langevin Dynamics on the Probability Simplex." In: *Advances in Neural Information Processing Systems*. 2013.

[152] Tom Rainforth, Tuan Anh Le, Jan-Willem van de Meent, Michael A Osborne, and Frank Wood. "Bayesian optimization for probabilistic programs." In: *Advances in Neural Information Processing Systems*. 2016, pp. 280–288.

[153] Rajesh Ranganath, Sean Gerrish, and David Blei. "Black box variational inference." In: *Artificial Intelligence and Statistics*. 2014, pp. 814–822.

[154] Danilo Jimenez Rezende and Shakir Mohamed. "Variational inference with normalizing flows." In: *arXiv preprint arXiv:1505.05770* (2015).

[155] Daniel Ritchie, Paul Horsfall, and Noah D Goodman. "Deep amortized inference for probabilistic programs." In: *arXiv preprint arXiv:1610.05735* (2016).

[156] Peter E Rossi and Greg M Allenby. "Bayesian statistics and marketing." In: *Marketing Science* 22.3 (2003), pp. 304–328.

[157] Peter E Rossi, Greg M Allenby, and Rob McCulloch. *Bayesian statistics and marketing*. John Wiley & Sons, 2012.

[158] PJ Rossky, JD Doll, and HL Friedman. "Brownian dynamics as smart Monte Carlo simulation." In: *The Journal of Chemical Physics* 69.10 (1978), pp. 4628–4633.

[159] Sam Roweis and Zoubin Ghahramani. "A unifying review of linear Gaussian models." In: *Neural computation* 11.2 (1999), pp. 305–345.

[160] Ruslan Salakhutdinov and Andriy Mnih. "Bayesian probabilistic matrix factorization using Markov chain Monte Carlo." In: *Proceedings of the 25th international conference on Machine learning*. ACM. 2008, pp. 880–887.

[161] John Salvatier, Thomas V Wiecki, and Christopher Fonnesbeck. "Probabilistic programming in Python using PyMC3." In: *PeerJ Computer Science* 2 (2016), e55.

[162] Henry Scheffé et al. "A useful convergence theorem for probability distributions." In: *The Annals of Mathematical Statistics* 18.3 (1947), pp. 434–438.

[163] Bianca Schroeder and Garth Gibson. "A large-scale study of failures in high-performance computing systems." In: *IEEE transactions on Dependable and Secure Computing* 7.4 (2009), pp. 337–350.

[164] Steven L. Scott, Alexander W. Blocker, and Fernando V. Bonassi. "Bayes and Big Data: The Consensus Monte Carlo Algorithm." In: *Bayes 250*. 2013.

[165] Matthias W Seeger. "Bayesian inference and optimal design for the sparse linear model." In: *The Journal of Machine Learning Research* 9 (2008), pp. 759–813.

[166] Amar Shah, Andrew Wilson, and Zoubin Ghahramani. "Student-t processes as alternatives to Gaussian processes." In: *Artificial Intelligence and Statistics*. 2014, pp. 877–885.

[167] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. "Taking the human out of the loop: A review of bayesian optimization." In: *Proceedings of the IEEE* 104.1 (2016), pp. 148–175.

[168] Jianbo Shi and Jitendra Malik. "Normalized cuts and image segmentation." In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 22.8 (2000), pp. 888–905.

[169] Taeshik Shon and Jongsub Moon. "A hybrid machine learning approach to network anomaly detection." In: *Information Sciences* 177.18 (2007), pp. 3799–3821.

[170] Adrian FM Smith and Gareth O Roberts. "Bayesian computation via the Gibbs sampler and related Markov chain Monte Carlo methods." In: *Journal of the Royal Statistical Society. Series B (Methodological)* (1993), pp. 3–23.

[171] Jack W Smith, JE Everhart, WC Dickson, WC Knowler, and RS Johannes. "Using the ADAP learning algorithm to forecast the onset of diabetes mellitus." In: *Proceedings of the Annual Symposium on Computer Application in Medical Care*. American Medical Informatics Association. 1988, p. 261.

[172] Alexander Smola and Shravan Narayanamurthy. "An architecture for parallel topic models." In: *Proc. VLDB Endow.* 3.1-2 (Sept. 2010), pp. 703–710. ISSN: 2150-8097. URL: http://dl.acm.org/citation.cfm?id=1920841.1920931.

[173] Alexander Smola and Shravan Narayanamurthy. "An architecture for parallel topic models." In: *Proceedings of the VLDB Endowment* 3.1-2 (2010), pp. 703–710.

[174] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. "Practical bayesian optimization of machine learning algorithms." In: *Advances in neural information processing systems*. 2012, pp. 2951–2959.

[175] 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*. 2015, pp. 2171–2180.

[176] Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. "Gaussian process optimization in the bandit setting: No regret and experimental design." In: *arXiv preprint arXiv:0912.3995* (2009).

[177] Sanvesh Srivastava, Volkan Cevher, Quoc Tran-Dinh, and David B Dunson. "WASP: Scalable Bayes via barycenters of subset posteriors." In: *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*. 2015, pp. 912–920.

[178] Chris Stauffer and W. Eric L. Grimson. "Learning patterns of activity using real-time tracking." In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 22.8 (2000), pp. 747–757.

[179] Rebecca C Steorts, Matt Barnes, and Willie Neiswanger. "Performance Bounds for Graphical Record Linkage." In: *arXiv preprint arXiv:1703.02679* (2017).

[180] Kevin Swersky, Jasper Snoek, and Ryan P Adams. "Multi-task bayesian optimization." In: *Advances in neural information processing systems*. 2013, pp. 2004–2012.

[181] Yee W Teh, David Newman, and Max Welling. "A collapsed variational Bayesian inference algorithm for latent Dirichlet allocation." In: *Advances in neural information processing systems*. 2006, pp. 1353–1360.

[182] William R Thompson. "On the likelihood that one unknown probability exceeds another in view of the evidence of two samples." In: *Biometrika* 25.3/4 (1933), pp. 285–294.

[183] Robert Tibshirani. "Regression shrinkage and selection via the lasso." In: *Journal of the Royal Statistical Society. Series B (Methodological)* (1996), pp. 267–288.

[184] Carlo Tomasi and Takeo Kanade. *Detection and tracking of point features*. School of Computer Science, Carnegie Mellon Univ., 1991.

[185] Tina Toni, David Welch, Natalja Strelkowa, Andreas Ipsen, and Michael PH Stumpf. "Approximate Bayesian computation scheme for parameter inference and model selection in dynamical systems." In: *Journal of the Royal Society Interface* 6.31 (2008), pp. 187–202.

[186] Dustin Tran, Alp Kucukelbir, Adji B. Dieng, Maja Rudolph, Dawen Liang, and David M. Blei. "Edward: A library for probabilistic modeling, inference, and criticism." In: *arXiv preprint arXiv:1610.09787* (2016).

[187] Dustin Tran, Rajesh Ranganath, and David Blei. "Hierarchical implicit models and likelihood-free variational inference." In: *Advances in Neural Information Processing Systems*. 2017, pp. 5523–5533.

[188] Pavan Turaga, Rama Chellappa, Venkatramana S Subrahmanian, and Octavian Udrea. "Machine recognition of human activities: A survey." In: *Circuits and Systems for Video Technology, IEEE Transactions on* 18.11 (2008), pp. 1473–1488.

[189] Harini Veeraraghavan, Paul Schrater, and Nikos Papanikolopoulos. "Robust target detection and tracking through integration of motion, color, and geometry." In: *Computer Vision and Image Understanding* 103.2 (2006), pp. 121–138.

[190] Chong Wang and David Blei. "Collaborative topic modeling for recommending scientific articles." In: *ACM International Conference on Knowledge Discovery and Data Mining (KDD)*. 2011.

[191] Chong Wang and David M Blei. "Collaborative topic modeling for recommending scientific articles." In: *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2011, pp. 448–456.

[192] Chong Wang and David M Blei. "Variational inference in nonconjugate models." In: *The Journal of Machine Learning Research* 14.1 (2013), pp. 1005–1031.

[193] Xiangyu Wang and David B Dunson. "Parallel MCMC via Weierstrass Sampler." In: *arXiv preprint arXiv:1312.4605* (2013).

[194] Xiangyu Wang, Fangjian Guo, Katherine A Heller, and David B Dunson. "Parallelizing MCMC with Random Partition Trees." In: *arXiv preprint arXiv:1506.03164* (2015).

[195] Xiaogang Wang, Xiaoxu Ma, and Eric Grimson. "Unsupervised activity perception by hierarchical bayesian models." In: *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*. IEEE. 2007, pp. 1–8.

[196] Larry Wasserman. *All of nonparametric statistics*. Springer Science & Business Media, 2006.

[197] X. Wei and B. Croft. "LDA-Based Document Models for Ad-hoc Retrieval." In: *SIGIR*. 2006.

[198] Max Welling and Yee W Teh. "Bayesian learning via stochastic gradient Langevin dynamics." In: *Proceedings of the 28th International Conference on Machine Learning*. 2011, pp. 681–688.

[199] Darren J Wilkinson. "Parallel Bayesian computation." In: *Statistics Textbooks and Monographs* 184 (2006), p. 477.

[200] Sinead Williamson, Avinava Dubey, and Eric P Xing. "Parallel Markov chain Monte Carlo for Nonparametric Mixture Models." In: *Proceedings of the 30th International Conference on Machine Learning*. 2013, pp. 98–106.

[201] James T Wilson, Frank Hutter, and Marc Peter Deisenroth. "Maximizing acquisition functions for Bayesian optimization." In: *arXiv preprint arXiv:1805.10196* (2018).

[202] Frank Wood, Jan Willem van de Meent, and Vikash Mansinghka. "A New Approach to Probabilistic Programming Inference." In: *Proceedings of the 17th International conference on Artificial Intelligence and Statistics*. 2014, pp. 1024–1032.

[203] Bo Wu and Ram Nevatia. "Cluster boosted tree classifier for multi-view, multi-pose object detection." In: *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*. IEEE. 2007, pp. 1–8.

[204] Jian Wu, Matthias Poloczek, Andrew G Wilson, and Peter Frazier. "Bayesian optimization with gradients." In: *Advances in Neural Information Processing Systems*. 2017, pp. 5267–5278.

[205] Pengtao Xie, Jun Zhu, and Eric Xing. "Diversity-Promoting Bayesian Learning of Latent Variable Models." In: *Proceedings of the 33st International Conference on Machine Learning (ICML-16)*. 2016.

[206] Minjie Xu, Balaji Lakshminarayanan, Yee Whye Teh, Jun Zhu, and Bo Zhang. "Distributed Bayesian posterior sampling via moment sharing." In: *Advances in Neural Information Processing Systems*. 2014, pp. 3356–3364.

[207] Feng Yan, Ningyi Xu, and Yuan Qi. "Parallel inference for latent dirichlet allocation on graphics processing units." In: *Advances in Neural Information Processing Systems*. 2009, pp. 2134–2142.

[208] J. Yang, PA Vela, Z. Shi, and J. Teizer. "Probabilistic multiple people tracking through complex situations." In: *11th IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*. 2009.

[209] Alper Yilmaz, Omar Javed, and Mubarak Shah. "Object tracking: A survey." In: *Acm Computing Surveys (CSUR)* 38.4 (2006), p. 13.

[210] Ke Zhai, Jordan Boyd-Graber, Nima Asadi, and Mohamad L Alkhouja. "Mr. LDA: A flexible large scale topic modeling package using variational inference in mapreduce." In: *Proceedings of the 21st international conference on World Wide Web*. ACM. 2012, pp. 879–888.

[211] Wayne Xin Zhao, Jing Jiang, Jianshu Weng, Jing He, Ee-Peng Lim, Hongfei Yan, and Xiaoming Li. "Comparing twitter and traditional media using topic models." In: *European Conference on Information Retrieval*. Springer. 2011, pp. 338–349.

[212] Barret Zoph and Quoc V Le. "Neural architecture search with reinforcement learning." In: *arXiv preprint arXiv:1611.01578* (2016).