Machine Learning: Social Values, Data Efficiency, and Beyond Prediction

Travis Dick

CMU-CS-19-113
May 2019

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

Thesis Committee:
Maria-Florina Balcan (Chair)
Tom Mitchell
Ariel Procaccia
Yishay Mansour (Tel Aviv University)

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

This research was sponsored by Schmidt Sciences and the National Science Foundation under grant numbers CCF-1422910, CCF-1535967, and IIS-161874. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

Keywords: Data-efficient Machine Learning, Data-driven Algorithm Configuration, Online Piecewise Lipschitz Optimization, Differential Privacy, Fairness in Machine Learning, Envy-freeness

Abstract

In this thesis, we build on the theory and practice of Machine Learning to accommodate several modern requirements of learning systems. In particular, we focus on new requirements stemming from three distinct sources: making the best use of available data, applying learning tools to problems beyond standard prediction, and incorporating social values. Each of these themes provides an exciting research direction for the practice and theory of Machine Learning.

Learning from Mostly Unlabeled Data in Multi-class Settings. Large scale multi-class learning tasks with an abundance of unlabeled data are ubiquitous in modern Machine Learning. For example, an in-home assistive robot needs to learn to recognize common household objects, familiar faces, facial expressions, gestures, and so on in order to be useful. Such a robot can acquire large amounts of unlabeled training data simply by observing its surroundings, but it would be prohibitively time consuming to ask its owner to annotate any large portion of this data. More generally, in many modern learning problems we often have easy and cheap access to large quantities of unlabeled training data but obtaining high-quality labeled examples is relatively expensive. The first chapter of my thesis focuses on theory for large-scale multi-class learning with limited labeled data. We begin by assuming that a given supervised learning algorithm would succeed at the learning task if it had access to labeled data. Then we use the implicit assumptions made by that algorithm to show that different label-efficient algorithms will also succeed.

Machine Learning Beyond Standard Prediction Problems. While most machine learning focuses on learning to make predictions, there are important learning problems where the output of the learner is not a prediction rule. We focus on data-driven algorithm configuration, where the goal is to find the best algorithm parameters for a specific application domain. A recent exciting line of work has considered data-driven algorithm configuration in the statistical setting, where each application domain is modeled as a distribution over problem instances. In this work, we extend the theoretical foundations of this field to accommodate two new learning settings: the online setting where problems are chosen by an adversary and arrive one at a time, and the private setting, where each problem instance contains sensitive information that should not be released. Algorithm configuration problems often reduce to the maximization of a collection of piecewise Lipschitz functions. Unfortunately, the discontinuities of these functions lead to worst-case impossibility results for both the online and private settings. We introduce a novel condition called dispersion that, when satisfied, allows for meaningful regret bounds and utility guarantees in these settings. Next, we show that dispersion is satisfied for many data-driven algorithm configuration problems under very mild assumptions. Finally, we uncover additional structure in data-driven algorithm configuration problems enabling efficient semi-bandit feedback, leading to very efficient configuration procedures with strong regret bounds.

Social Values for Machine Learning Systems. Machine learning systems are becoming central to the infrastructure of our society. The wide-spread use of such systems creates exciting possibilities for profoundly positive impacts in our lives though improvements to, for example, medicine, communication, and transportation. Since these systems are often not explicitly designed with social values in mind, there is a risk that their adoption could result in undesired outcomes such as privacy violations or unfair treatment of individuals. My thesis develops principled techniques for incorporating two social values into machine learning algorithms: privacy and fairness. We import the fairness notion of envy-freeness from fair division into machine learning and consider whether it is possible to learn envy-free classifiers. We also develop general tools for differentially private optimization of piecewise Lipschitz functions, a problem that arises naturally in several learning settings, including applications beyond standard prediction problems.

Acknowledgments

First, I would like to thank my advisor, Nina Balcan. Throughout my five years at CMU, Nina has been an inspiration and incredibly supportive of my research. Her expertise and deep intuition across a wide range of topics made it easy to find interesting and fruitful research directions. In addition to her intellectual support, Nina also helped me in many other ways. For example, my presentation and writing skills have been dramatically improved thanks to her mentorship. She also helped me to make many connections in the research community, both at CMU and more broadly. I thank Nina for her generosity during my PhD—without her, this thesis would not have been possible.

I am incredibly thankful for the interesting discussions and insightful comments given by the other members of my thesis committee: Yishay Mansour, Tom Mitchell, and Ariel Procaccia. I could not have asked for a more exciting committee to share my work with.

I thank all of my other wonderful collaborators: Kareem Amin, Misha Khodak, Alex Kulesza, Manuel Lang, Mu Li, Yingyu Liang, Andrés Muñoz Medina, Wenlong Mou, Ritesh Noothigattu, Wesley Pegden, Krishna Pillutla, Tuomas Sandholm, Dravyansh Sharma, Alex Smola, Sergei Vassilvitskii, Ellen Vitercik, Colin White, and Hongyang Zhang. I enjoyed working with and learning from all of you. I also want to thank the computer science community at CMU more broadly, and especially Deb Cavlovich, Catherine Copetas, and Amy Protos, for being extremely helpful.

I am also thankful for my advisors and friends from the University of Alberta that helped set me on my path towards academia. In particular, I would like to thank Rich Sutton, András György, Csaba Szepesvári, Martin Jagersand, Martin Müller, and the "Concord Crew" (Roshan Shariff, Clint Pahl, and honorary member Adel Lari).

Finally, I would like to thank my family for their endless support and encouragement. My parents Barry and Linda have been profoundly supportive and are always excited to hear about what I am working on, even at 3am on the first night of a visit! My sisters Stephanie and Kristen are some of my closest friends and never hesitate to give advice or support. Finally, my niece and nephew, Niah and Nathan, make every visit brighter and make me hopeful for the future. Thank you all so much.

Contents

1 Introduction **1**

2 Label Efficient Learning by Exploiting Multi-class Output Codes **4**
2.1 Introduction 4
2.2 Related Work 6
2.3 Preliminaries 7
2.4 Error Correcting Output Codes 7
2.5 One-Versus-All on the Unit Ball 12
2.6 The Boundary Features Condition 18
2.7 Extensions to the Agnostic Setting 24
2.8 Conclusion and Discussion 24

3 Online and Private Algorithm Configuration **26**
3.1 Private and Online Algorithm Configuration from Dispersion 27
3.1.1 Introduction 27
3.1.2 Dispersion Condition 32
3.1.3 Online Optimization 34
3.1.4 Differentially Private Optimization 36
3.1.5 Dispersion in application-specific algorithm selection 38
3.1.6 Generalization guarantees for distributional learning 40
3.1.7 Conclusion 41
3.2 Semi-bandit Optimization in the Dispersed Setting 41
3.2.1 Introduction 41
3.2.2 Semi-bandit Optimization of Piecewise Lispchitz Functions 45
3.2.3 General Tools for Verifying Dispersion 48
3.2.4 Online Algorithm Selection with Semi-bandit Feedback 49

4 Data-driven Algorithm Configuration and Metric Learning for Clustering **55**
4.1 Introduction 55
4.2 Learning Clustering Metrics 56
4.3 Learning Merge Functions 59
4.4 Efficient Algorithm Selection 60
4.4.1 Optimizing the Merge Function 61
4.4.2 Optimizing the Metric 64
4.5 Experiments 66

5 A New Approach to Individual Fairness: Envy-free Classification **72**
5.1 Introduction . . . 72
5.1.1 Our Results . . . 74
5.1.2 Related Work . . . 74
5.2 The Model . . . 75
5.2.1 Envy-Freeness . . . 75
5.2.2 Optimization and Learning . . . 76
5.3 Arbitrary Classifiers . . . 76
5.4 Low-Complexity Families of Classifiers . . . 77
5.4.1 Natarajan Dimension Primer . . . 78
5.4.2 Main Result . . . 78

A Appendix for Chapter 2 **93**
A.1 Appendix for Error Correcting Output Codes . . . 93
A.2 Appendix For One-vs-all on the Unit Ball . . . 95
A.3 Appendix for Boundary Features Condition . . . 96

B Appendix for Chapter 3 **98**
B.1 Appendix for Section 3.1 . . . 98
B.1.1 Generic lemmas for dispersion . . . 98
B.1.2 Properties of κ-bounded distributions . . . 101
B.1.3 Efficient sampling . . . 103
B.1.4 Proofs for online learning (Section 3.1.3) . . . 106
B.1.5 Proofs for differential privacy (Section 3.1.4) . . . 119
B.1.6 Proofs for algorithm configuration (Section 3.1.5) . . . 124
B.1.7 Proofs for distributional learning (Section 3.1.6) . . . 139
B.1.8 Discretization-based algorithm . . . 140
B.2 Appendix for Section 3.2 . . . 141
B.2.1 Online Optimization . . . 141
B.2.2 Dispersion Tools . . . 147
B.2.3 Appendix for Applications . . . 150
B.2.4 Transformations of Bounded Densities . . . 153
B.2.5 Discretization-based Algorithm . . . 155

C Appendix for Chapter 4 **158**

D Appendix for Chapter 5 **160**

List of Figures

2.1 An example problem satisfying Assumption 2.2 and the projected density q when the density p is uniform on K. 12
2.2 An example of the boundary features problem. The arrows indicate the positive side of the linear functions. 18
2.3 Examples of half-balls that would be included (green) or excluded (red) by the plane detection algorithm. 19

3.1 The dashed and solid lines correspond to two partitionings of the rectangle. Each of the displayed balls is either not split, split by one partition, or split by both. 33

4.1 A two dimensional dataset with well separated clusters that become interleaved when projecting onto either the x or y axes. 57
4.2 An example clustering instance where both single and complete linkage produce high-cost clusterings, but a mixture of the two merge functions leads to zero cost. 60
4.3 An example of the execution tree of $\mathcal{A}_{\text{merge}}(\mathrm{D}_{\text{min}}, \mathrm{D}_{\text{max}})$ for a clustering instance with 4 points. We also show the clustering of the points produced by the sequence of merges associated with each node in the tree. Each colored rectangle represents a cluster (and for clarity we also show the two children of each cluster). The leaves show the two possible cluster trees that can be output by any algorithm from the family $\mathcal{A}_{\text{merge}}(\mathrm{D}_{\text{min}}, \mathrm{D}_{\text{max}})$ for this instance. 62
4.4 Empirical loss for interpolating between single and complete linkage as well as average and complete linkage on 1000 randomly sampled tasks from the MNIST subsets distribution. 70
4.5 Empirical loss for interpolating between single and complete linkage as well as average and complete linkage on 1000 randomly sampled tasks from the CIFAR-10 Subsets distribution. 70
4.6 Empirical loss for interpolating between single and complete linkage as well as average and complete linkage on 1000 randomly sampled tasks from the rings and disks distribution. 70
4.7 Empirical loss for interpolating between single and complete linkage as well as average and complete linkage on 1000 randomly sampled tasks from the Omniglot Subsets distribution. 71
4.8 Empirical loss for interpolating between the neural network feature embedding metric and stroke metric for Omniglot data. $\beta = 0$ corresponds to the stroke distance, while $\beta = 1$ corresponds to the neural network embedding. 71

B.1 A graph of the 2-linear function ϕ_2. 131
B.2 Relationship between the binary search intervals $[a, b]$ and $[c, d]$ and the true interval $[\rho^*_{\min}, \rho^*_{\max}]$ on which $\mathcal{A}(x, \rho')$ outputs y_ρ. 152

D.1 Illustration of $\mathcal{X}$ and an example utility function u for $d = 2$. Red shows preference for 1, blue shows preference for 0, and darker shades correspond to more intense preference. (The gradients are rectangular to match the L_∞ norm, so, strangely enough, the misleading X pattern is an optical illusion.) 163

Chapter 1

Introduction

In this thesis we extend the practice and theory of machine learning to accommodate several new requirements. We focus on three themes, each stemming from the application of machine learning to modern real-world scenarios. First we look at making the best use of the types of data that are readily available. Next, we apply machine learning tools to problems beyond standard prediction. Finally, we consider incorporating social values, like privacy and fairness, into machine learning systems.

Data Efficiency in Multi-class Learning. In many real-world classification problems, unlabeled data is cheap and readily available, while obtaining high-quality labels is time consuming or expensive. Machine learning systems should make the best use of the cheap and abundantly available unlabeled data to minimize the need for the more expensive labeled data. This has been the focus of the fields of semi-supervised and active learning. In Chapter 2, we explore conditions for large-scale multi-class learning under which unlabeled data provably reduces the labeled sample complexity of learning. Our results argue that if the underlying learning task is one for which a given supervised learning algorithm would succeed at learning, given a fully labeled dataset, then much more label-efficient algorithms will also succeed. We obtain these results by carefully examining the implicit assumptions made by specific classes of supervised learning algorithms, showing that when they succeed it implies geometric structure in the underlying learning problem that can be exploited in semi-supervised and active learning settings.

Machine Learning Beyond Prediction. Most machine learning systems learn to make predictions from data. For example, in medical settings, the learner may receive a training dataset consisting of medical feature representations describing patients, together with an expert diagnosis for each patient. Then the learner's goal is to identify patterns in the expert diagnoses so that it can predict what the diagnosis will be for new unseen patients. A key requirement is that the learned prediction rules generalize, in the sense that they make accurate predictions on new unseen data.

There are many other important learning problems where the output of the learner is not a prediction rule. An alternate setting that we consider in Chapter 3 is data-driven algorithm configuration. Our goal is to find the best parameter settings for a parameterized algorithm when it is used for problem instances arising in some specific target application. The parameter-tuning procedure learns about the specific application of the algorithm by observing a collection of example problem instances from that application domain. For example, if our goal is to learn good parameters for a clustering algorithm, then the training data could consist of clustering instances together with target clusterings

for each instance. Then the learner's goal is to find parameters that produce clusterings with the best possible agreement with the target clusterings. In data-driven algorithm configuration, we care about finding parameters that will continue to perform well on future instances.

While there are many similarities between learning to predict and data-driven algorithm configuration, there are also a number of important differences that need to be overcome. For example, the output of an algorithm is often a volatile function of its parameters, and very small changes to the parameters can lead to a cascade of differences in the output. Chapter 3 explores online and private optimization of piecewise Lipschitz functions, an optimization problem that captures many of the challenges of online data-driven algorithm configuration. In particular, in Section 3.1, we introduce a condition on sequences of piecewise Lipschitz functions, called dispersion, that measures how concentrated their discontinuities are. We show that for sufficiently dispersed functions, both online and private optimization are possible. We also show that many interesting algorithm configuration problems satisfy dispersion under mild and realistic assumptions, allowing us to provide online configuration procedures for these settings. In Section 3.2, we uncover additional structure present in many algorithm configuration problems and use it to design even more efficient online configuration procedures. Intuitively, when tuning the parameters of algorithms, we need to run the algorithm with different parameter settings to learn how well they perform. We show that running the algorithm with a single parameter setting often reveals the performance of a whole range of similar parameter settings, essentially at no additional computational cost. Leveraging this extra information leads to more efficient optimization procedures.

In Chapter 4 we study the algorithmic aspects of data-driven algorithm configuration for linkage based clustering. Rather than working in the online setting, where the learner faces an adversarially chosen sequence of clustering instances, we give results for the distributional setting. Here, each clustering application domain is modeled as a distribution over problem instances, the algorithm configuration procedure obtains an i.i.d. set of training instances, and their goal is to find the algorithm with highest expected performance. We provide sample complexity guarantees for learning the best metric to use when clustering, and design efficient algorithms for finding the empirically optimal algorithm for a set of training instances. We conduct experiments on a number of clustering application distributions showing that algorithm selection can dramatically improve performance.

Social Values in Machine Learning. When machine learning systems interact with society, we expect them to uphold our social values. For example, if a model is trained on data containing sensitive information about individuals, it should not be possible to learn about specific individuals in the training data by inspecting either its predictions or its parameters. Similarly, when machine-learned models are used to make consequential decisions, such as whether to release a defendant on bail, or whether to grant an individual a loan, we want guarantees that the learned models uphold our notions of fair treatment. Since models trained to maximize predictive accuracy may not inherently have these properties, an important research direction is to design principled techniques for explicitly incorporating social values into learning algorithms.

In Chapter 5, we import *envy-freeness* from fair-division as a notion of fairness in machine learning. Envy-freeness is particularly well-suited to situations where we are learning to assign one of many outcomes to individuals who have heterogeneous preferences for those outcomes. This is in contrast with much of the fairness literature, which focuses on binary classification settings with one desirable outcome and one undesirable. Intuitively, a classifier that assigns individuals to outcomes is envy-free if no individual would prefer to change their assignment to that of someone else. Our cur-

rent results focus on the generalizability of envy-freeness, showing that as long as the learner restricts itself to a set of relatively low-complexity classifiers, classifiers that are envy-free on a large enough sample will remain approximately envy-free on the underlying distribution.

The project on piecewise Lipschitz optimization discussed in Chapter 3 also connects with the theme of incorporating social values into machine learning systems. Our results show how to optimize a collection of piecewise Lipschitz functions while at the same time providing very strong privacy guarantees when those functions encode sensitive information about individuals.

Chapter 2

Label Efficient Learning by Exploiting Multi-class Output Codes

2.1 Introduction

Motivation: Large scale multi-class learning problems with an abundance of unlabeled data are ubiquitous in modern machine learning. For example, an in-home assistive robot needs to learn to recognize common household objects, familiar faces, facial expressions, gestures, and so on in order to be useful. Such a robot can acquire large amounts of unlabeled training data simply by observing its surroundings, but it would be prohibitively time consuming (and frustrating) to ask its owner to annotate any significant portion of this raw data. More generally, in many modern learning problems we often have easy and cheap access to large quantities of unlabeled training data (e.g., on the internet) but obtaining high-quality labeled examples is relatively expensive. More examples include text understanding, recommendation systems, or wearable computing [136, 138, 137, 107]. The scarcity of labeled data is especially pronounced in problems with many classes, since supervised learning algorithms typically require labeled examples from every class. In such settings, algorithms should strive to make the best use of unlabeled data in order to minimize the need for expensive labeled examples.

Overview: We approach label-efficient learning by making the implicit assumptions of popular multi-class learning algorithms explicit and showing that they can also be exploited when learning from limited labeled data. We focus on a family of techniques called *output codes* that work by decomposing a given multi-class problem into a collection of binary classification tasks [108, 57, 97, 29]. The novelty of our results is to show that the existence of various low-error output codes constrains the distribution of unlabeled data in ways that can be exploited to reduce the label complexity of learning. We consider both the consistent setting, where the output code achieves zero error, and the agnostic setting, where the goal is to compete with the best output code. The most well known output code technique is one-vs-all learning, where we learn one binary classifier for distinguishing each class from the union of the rest. When output codes are successful at learning from labeled data, it often implies geometric structure in the underlying problem. For example, if it is possible to learn an accurate one-vs-all classifier with linear separators, it implies that no three classes can be collinear, since then it would be impossible for a single linear separator to distinguish the middle class from the union of the others. In this work exploit this implicitly assumed structure to design label-efficient al-

gorithms for the commonly assumed cases of one-vs-all and error correcting output codes, as well as a novel boundary features condition that captures the intuition that every bit of the codewords should be significant.

Our results: Before discussing our results, we briefly review the output code methodology. For a problem with L classes, a domain expert designs a code matrix $C \in \{\pm 1\}^{L \times m}$ where each column partitions the classes into two meaningful groups. The number of columns m is chosen by the domain expert. For example, when recognizing household objects we could use the following true/false questions to define the partitions: "is it made of wood?", "is it sharp?", "does it have legs?", "should I sit on it?", and so on. Each row of the code matrix describes one of the classes in terms of these partitions (or semantic features). For example, the class "table" could be described by the vector $(+1, -1, +1, -1)$, which is called the class' codeword. Once the code matrix has been designed, we train an output code by learning a binary classifier for each of the binary partitions (e.g., predicting whether an object is made of wood or not). To predict the class of a new example, we predict its codeword in $\{\pm 1\}^m$ and output the class with the nearest codeword under the Hamming distance. Two popular special cases of output codes are one-vs-all learning, where C is the identity matrix (with -1 in the off-diagonal entries), and error correcting output codes, where the Hamming distance between the codewords is large.

In each of our results we assume that there exists a consistent or low-error linear output code classifier and we impose constraints on the code matrix and the distribution that generates the data. We present algorithms and analysis techniques for a wide range of different conditions on the code matrix and data distribution to showcase the variety of implicit structures that can be exploited. For the code matrix, we consider the case when the codewords are well-separated (i.e., the output code is error correcting), the case of one-vs-all (where the code matrix is the identity), and a natural boundary features condition. These conditions can loosely be compared in terms of the Hamming distance between codewords. In the case of error correcting output codes, the distance between codewords is large (at least $d + 1$ when the data is d-dimensional), in one-vs-all the distance is always exactly 2, and finally in the boundary features condition the distance can be as small as 1. In the latter cases, the lower Hamming distance requirement is balanced by other structure in the code matrix. For the distribution, we either assume that the data density function satisfies a thick level set condition or that the density is upper and lower bounded on its support. Both regularity conditions are used to ensure that the geometric structure implied by the consistent output code will be recoverable based on a sample of data.

Error correcting output codes: We first showcase how to exploit the implicit structure assumed by the commonly used and natural case of linear output codes where the Hamming distance between codewords is large. In practice, output codes are designed to have this property in order to be robust to prediction errors for the binary classification tasks [57]. We suppose that the output code makes at most β errors when predicting codewords and has codewords with Hamming distance at least $2\beta + d + 1$ in a d-dimensional problem. The key insight is that when the code words are well separated, this implies that points belonging to different classes must be geometrically separated as well. This suggests that tight clusters of data will be label-homogeneous, so we should be able to learn an accurate classifier using only a small number of label queries per cluster. The main technical challenge is to show that our clustering algorithm will not produce too many clusters (in order to keep the label complexity controlled), and that with high probability, a new sample from the distribution

will have the same label as its nearest cluster. We show that when the data density satisfies a thick-level set condition (requiring that its level sets do not have bridges or cusps that are too thin), then a single-linkage clustering algorithm can be used to recover a small number of label-homogeneous clusters.

One-vs-all: Next, we consider the classic one-vs-all setting for data in the unit ball. This is an interesting setting because of the popularity of one-vs-all classification and because it significantly relaxes the assumption that the codewords are well separated (in a one-vs-all classifier, the Hamming distance between codewords is exactly 2). The main challenge in this setting is that there need not be a margin between classes and a simple single-linkage style clustering might group multiple classes into the same cluster. To overcome this challenge, we show that the classes are probabilistically separated in the following sense: after projecting onto the surface of the unit ball, the level sets of the projected density are label-homogeneous. Equivalently, the high-density regions belonging to different classes must be separated by low-density regions. We exploit this structure by estimating the connected components of the ϵ level set using a robust single-linkage clustering algorithm.

The boundary features condition: Finally, we introduce an interesting and natural condition on the code matrix capturing the intuition that every binary learning task should be significant. This condition has the weakest separation requirement, allowing the codewords to have a Hamming distance of only 1. This setting is our most challenging, since it allows for the classes to be very well connected to one another, which prevents clustering or level set estimation from being used to find a small number of label-homogeneous clusters. Nevertheless, we show that the implicit geometric structure implied by the output code can be exploited to learn using a small number of label queries. In this case, rather than clustering the unlabeled sample, we apply a novel hyperplane-detection algorithm that uses the *absence* of data to learn local information about the boundaries between classes. We then use the implicit structure of the output code to extend these local boundaries into a globally accurate prediction rule.

Agnostic Setting: Finally, we show that our results for the error correcting, one-vs-all, and boundary features cases can all be extended to an agnostic learning setting, where we do not assume that there exists a consistent output code classifier.

Our results show an interesting trend: when linear output codes are able to learn from labeled data, it is possible to exploit the same underlying structure in the problem to learn using a small number of label requests. Our results hold under several natural assumptions on the output code and general conditions on the data distribution, and employ both clustering and hyperplane detection strategies to reduce the label complexity of learning.

2.2 Related Work

Reduction to binary classification is one of the most widely used techniques in applied machine learning for attacking multi-class problems. Indeed, the one-vs-all, one-vs-one, and the error correcting output code approaches [57] all follow this structure [108, 97, 29, 50, 1].

There is no prior work providing error bounds for output codes using unlabeled data and interaction. There has been a long line of work for providing provable bounds for semi-supervised

learning [11, 9, 31, 42] and active learning [12, 51, 10, 79]. These works provide bounds on the benefits of unlabeled data and interaction for significantly different semi-supervised and active learning methods that are based different assumptions, often focusing on binary classification, thus the results are largely incomparable. Another line of recent work considers the multi-class setting and uses unlabeled data to consistently estimate the risk of classifiers when the data is generated from a known family of models [59, 7, 8]. Their results do not immediately imply learning algorithms and they consider generative assumptions, while in contrast our work explicitly designs learning algorithms under commonly used discriminative assumptions.

Another work related to ours is that of Balcan et al. [14], where labels are recovered from unlabeled data. The main tool that they use, in order to recover the labels, is the assumption that there are multiple views and an underlying ontology that are known, and restrict the possible labeling. Similarly, Steinhardt and Liang [131] show how to use the method of moments to estimate the risk of a model from unlabeled data under the assumption that the data has three independent views. Our work is more widely applicable, since it applies when we have only a single view.

The output-code formalism is also used by Palatucci et al. [116] for the purpose of zero shot learning. They demonstrate that it is possible to exploit the semantic relationships encoded in the code matrix to learn a classifier from labeled data that can predict accurately even classes that *did not appear in the training set*. These techniques make very similar assumptions to our work but require that the code matrix C is known and the problem that they solve is different.

2.3 Preliminaries

We consider multiclass learning problems over an instance space $\mathcal{X} \subset \mathbb{R}^d$ where each point is labeled by $f^* : \mathcal{X} \to \{1, \ldots, k\}$ to one out of k classes and the probability of observing each outcome $x \in \mathcal{X}$ is determined by a data distribution $\mathcal{P}$ on $\mathcal{X}$. The density function of $\mathcal{P}$ is denoted by $p : \mathcal{X} \to [0, \infty)$. In all of our results we assume that there exists a consistent (but unknown) linear output-code classifier defined by a code matrix $C \in \{\pm 1\}^{L \times m}$ and m linear separators $h_1, \ldots, h_m$. We denote class i's code word by C_i and define $h(x) = (\mathrm{sign}(h_1(x)), \ldots, \mathrm{sign}(h_m(x)))$ to be the predicted code word for point x. We let $d_{\mathrm{Ham}}(c, c')$ denote the Hamming distance between any codewords $c, c' \in \{\pm 1\}^m$. Finally, to simplify notation, we assume that the diameter of $\mathcal{X}$ is at most 1.

Our goal is to learn a hypothesis $\hat{f} : \mathcal{X} \to \{1, \ldots, k\}$ minimizing $\mathrm{err}(\hat{f}) = \Pr_{X \sim \mathcal{P}}(\hat{f}(x) \neq f(x))$ from an unlabeled sample drawn from the data distribution $\mathcal{P}$ together with a small set of actively queried labeled examples.

Finally, we use the following notation throughout this chapter: For any set A in a metric space $(\mathcal{X}, d)$, the σ-interior of A is the set $\mathrm{int}_\sigma(A) = \{x \in A : B(x, \sigma) \subset A\}$, where $B(x, \sigma)$ denotes the ball of radius σ about x. We use the notation $\tilde{O}(\cdot)$ to suppress logarithmic terms.

2.4 Error Correcting Output Codes

We first consider the implicit structure when there exists a consistent linear *error correcting* output code classifier:

Assumption 2.1. There exists a code matrix $C \in \{\pm 1\}^{L \times m}$ and linear functions $h_1, \ldots, h_m$ such that: (1) there exists $\beta \geq 0$ such that any point x from class y satisfies $d_{\mathrm{Ham}}(h(x), C_y) \leq \beta$, (2)

The Hamming distance between the codewords of C is at least $2\beta + d + 1$; and (3) at most d of the separators $h_1, \ldots, h_m$ intersect at any point.

Part (1) of this condition is a bound on the number of linear separators that can make a mistake when the output code predicts the codeword of a new example, part (2) formalizes the requirement of having well separated codewords, and part (3) requires that the hyperplanes be in general position, which is a very mild condition that can be satisfied by adding an arbitrarily small perturbation to the linear separators.

Despite being very natural, Assumption 2.1 conveniently implies that there exists a distance $g > 0$ such that any points that f^* assigns to different classes must be at least distance g apart. To see this, fix any pair of points x and x' with $f^*(x) \neq f^*(x')$. By the triangle inequality, we have that $d_{\text{Ham}}(h(x), h(x')) \geq d + 1$, implying that the line segment $[x, x']$ crosses at least $d + 1$ of the linear separators. Since only d linear separators can intersect at a point, the line segment must have non-zero length. Applying this argument to the closest pair of points between all pairs of classes and taking the minimum length gives the result. A formal proof is given Section A.1 of the Appendix.

Lemma 2.1. *Under Assumption 2.1, there exists $g > 0$ such that if points x and x' belong to different classes, then $\|x - x'\| > g$.*

Lemma 2.1 suggests that we should be able to reduce the label complexity of learning by clustering the data and querying the label of each cluster, since nearby points must belong to the same class. If we use a single-linkage style clustering algorithm that merges clusters whenever their distance is smaller than g, we are guaranteed that the clusters will be label-homogeneous, and therefore we can recover nearly all of the labels by querying one label from the largest clusters. See Algorithm 1 for pseudocode.

Algorithm 1 Single-linkage learning.

Input: Sample $S = \{x_1, \ldots, x_n\}$, radius $r_c > 0$, target error $\epsilon > 0$

1. Let $\{\hat{A}_1\}_{i=1}^N$ be the connected components of the graph G with vertex set S and an edge between x_i and x_j if $\|x_i - x_j\| \leq r_c$.
2. In decreasing order of size, query the label of each $\hat{A}_i$ until $\leq \frac{\epsilon}{4} n$ points belong to unlabeled clusters.
3. Output $\hat{f}(x) =$ label of nearest labeled cluster to x.

In order to get a meaningful reduction in label complexity, we need to ensure that when we cluster a sample of data, most of the samples will belong to a small number of clusters. For this purpose, we borrow the following very general and interesting thick level set condition from Steinwart [132]: a density function p has C-thick level sets if there exists a level $\lambda_0 > 0$ and a radius $\sigma_0 > 0$ such that for every level $\lambda \leq \lambda_0$ and radius $\sigma < \sigma_0$, (1) the σ-interior of $\{p \geq \lambda\}$ is non-empty and (2) every point in $\{p \geq \lambda\}$ is at most distance $C\sigma$ from the σ-interior. This condition elegantly characterizes a large family of distributions for which single-linkage style clustering algorithms succeed at recovering the high-density clusters and only rules out distributions whose level sets have bridges or cusps that are too thin. The thickness parameter C measures how pointed the boundary of the level sets of p can be. For example, in $\mathbb{R}^d$ if the level set of p is a ball then $C = 1$, while if the level set is a cube, then $C = \sqrt{d}$ (as a result of the "pointy" corners of the cube).

Using the thick level set condition to guarantee that our clustering algorithm will not subdivide the high-density clusters of p, we obtain the following result for Algorithm 1

Theorem 2.1. *Suppose that Assumption 2.1 holds and that the data distribution has C-thick level sets. For any target error $\epsilon > 0$, let N be the number of connected components of $\{p \geq \epsilon/(2\,\mathrm{Vol}(K))\}$. With probability at least $1-\delta$, running Algorithm 1 with parameter $r_c < g$ on an unlabeled sample of size $n = \tilde{O}(\frac{1}{\epsilon^2}((4C)^{2d}d^{d+1}/r_c^{2d} + N))$ will query at most N labels and output a classifier with error at most ϵ.*

Proof. For convenience, define $\sigma = r_c/(4C)$ and $\lambda = \epsilon/(2\,\mathrm{Vol}(K))$. Using a standard VC-bound [141] together with the fact that balls have VC-dimension $d+1$, for $n = O((4C)^{2d}d^{d+1}/(\epsilon^2 r_c^{2d}))$ guarantees that with probability at least $1-\delta/2$ the following holds simultaneously for every center $x \in \mathbb{R}^d$ and radius $r \geq 0$:

$$\left| \frac{|B(x,r) \cap S|}{n} - \mathcal{P}(B(x,r)) \right| \leq \frac{1}{2}\lambda\sigma^d v_d, \tag{2.1}$$

where v_d denotes the volume of the unit ball in $\mathbb{R}^d$. Assume that this high probability event occurs.

We first show that the sample S forms a $2C\sigma$-covering of the set $\{p \geq \lambda\}$; that is, for every $x \in \{p \geq \lambda\}$ we have $d(x,S) \leq 2C\sigma$. Let x be any point in $\{p \geq \lambda\}$. Since p has C-thick level sets, we know that there exists a point $y \in \mathrm{int}_\sigma(\{p \geq \lambda\})$ such that $\|x-y\| \leq C\sigma$. Moreover, the ball $B(y,\sigma)$ is contained in $\{p \geq \lambda\}$, which implies that it has probability mass at least $\lambda\sigma^d v_d$ and by (2.1) we have that $|B(y,r) \cap S|/n \geq \frac{1}{2}\lambda\sigma^d v_d > 0$, so there must exist a point $z \in S \cap B(y,\sigma)$. Now we have that $d(x,S) \leq \|x-z\| \leq \|x-y\| + \|y-z\| \leq C\sigma + \sigma \leq 2C\sigma$, where the final inequality follows from the fact that $C \geq 1$.

Now let $A_1, \ldots, A_N$ be the N connected components of $\{p \geq \lambda\}$. We will argue that for each $i \in [N]$, there exists a unique cluster output by step 1 of the algorithm, say $\hat{A}_i$, such that $\hat{A}_i$ contains $A_i \cap S$ and for any point $x \in A_i$, the closest output cluster is $\hat{A}_i$.

To see that $\hat{A}_i$ contains $A_i \cap S$, consider any pair of points x and x' in $A_i \cap S$. Since A_i is connected, we know there is a path $\pi : [0,1] \to A_i$ such that $\pi(0) = x$ and $\pi(1) = x'$. Since the sample set $\mathcal{X}$ is a $2C\sigma$ covering of $\{p \geq \lambda\}$, it is also a $2C\sigma$-covering of A_i, which implies that we can find a sequence of points $y_1, \ldots, y_M \in \mathcal{X}$ (possibly with repetition) such that the path π passes through the balls $B(y_1, 2C\sigma), \ldots, B(y_M, 2C\sigma)$ in order. Since consecutive balls must touch at the point that the path π crosses from one ball to the next, we know that $\|y_i - y_{i+1}\| \leq 4C\sigma = r_c$, and therefore the path $x \to y_1 \to \cdots \to y_M \to x'$ is a path in the graph G connecting x and x'.

Now consider any point $x \in A_i$. We argued above that there exists a sample point $z \in \hat{A}_i$ that was within distance $2C\sigma$ from x. Now let z^* be the closest sample in $\mathcal{X}$ to x. Then we know that $\|x-z^*\| \leq \|x-z\| \leq 2C\sigma$. By the triangle inequality, we have that $d\|z-z^*\| \leq \|z-x\| + \|x-z^*\| \leq 4C\sigma \leq r_c$, and therefore z and z^* are connected in the graph G. Since z belongs to $\hat{A}_i$, it follows that z^* does too, and therefore the closest cluster to x is $\hat{A}_i$.

It remains to bound the error of the resulting classification rule. Since there is a margin of width $g > 0$ separating the classes, we know that every connected component of $\{p \geq \lambda\}$ must contain points belonging to exactly one class. Moreover, since we ran the algorithm with connection radius $r_c < g$, we know that the clusters output by step 1 will contain points belonging to exactly one class. It follows that if we query the label of any point in the cluster $\hat{A}_i$ then the algorithm will not error on any test point in A_i. Say that one of the connected components A_i is labeled if we query the label of the corresponding cluster $\hat{A}_i$.

Applying Hoeffding's inequality and the union bound to all possible 2^N unions of the sets A_1, $\ldots$, A_N, our value of n guarantees that with probability at least $1-\delta/2$, the following holds simul-

taneously for all subsets of indices $I \subset [N]$:

$$\left| \left| S \cap \left(\bigcup_{i \in I} A_i \right) \right| / n - P\left(\bigcup_{i \in I} A_i \right) \right| \leq \frac{\epsilon}{4}.$$

Since the algorithm queries labels until at most $\frac{\epsilon}{4}n$ points belong to unlabeled clusters, we know that the number of samples belonging to the unlabeled A_i sets is at most $\frac{\epsilon}{4}n$. By the above uniform convergence, it follows that their total probability mass is at most $\epsilon/2$. Finally, since the algorithm only errors on test points in $\{p \leq \lambda\}$, which has probability mass at most $\epsilon/2$ or on unlabeled A_i sets, the error of the resulting classifier is at most ϵ. □

The exponential dependence on the dimension in Theorem 2.1 is needed to ensure the sample S will be a fine covering of the level set of p w.h.p., which guarantees that Algorithm 1 will not subdivide its connected components into smaller clusters. When the data has low intrinsic dimensionality, the unlabeled sample complexity is only exponential in the intrinsic dimension. The following result shows that under the common assumption that the distribution is a doubling measure, then the unlabeled sample complexity is exponential only in the doubling dimension. Recall that a probability measure $\mathcal{P}$ is said to have doubling dimension D if for every point x in the support of $\mathcal{P}$ and every radius $r > 0$, we have that $\mathcal{P}(B(x, 2r)) \leq 2^D \mathcal{P}(B(x, r))$ (see, for example, [52]).

Theorem 2.2. *Suppose that Assumption 2.1 holds the data distribution P has doubling dimension D, and the support of P has N connected components. With probability at least $1 - \delta$, running Algorithm 1 with parameter $r_c < g$ on a sample of size $n = \tilde{O}\left(d/r_c^{2D} + N/\epsilon^2\right)$ will query at most N labels and have error at most ϵ.*

Proof. Let x be any point in the support of $\mathcal{P}$. Since we assumed that the diameter of $\mathcal{X}$ is 1, we know that $\mathcal{X} \subset B(x, 1)$ and therefore $\mathcal{P}(B(x, 1)) = 1$. Applying the doubling condition $\lg(r)$ times, it follows that for any radius $r > 0$ we have that $\mathcal{P}(B(x, r)) \geq r^{-D}$.

As in the proof of Theorem 2.1, for our choice of n the following holds with probability at least $1 - \delta/2$ uniformly for every center x in $\mathcal{X}$ and radius $r \geq 0$:

$$\left| |B(x, r) \cap S|/n - \mathcal{P}(B(x, r)) \right| \leq \frac{1}{2} r^{-D}.$$

Assume this high probability event occurs. Since every ball of radius r centered at a point in the support of $\mathcal{P}$ has mass at least r^{-D}, each such ball must contain at least one sample point and it follows that the sample S forms an r^{-D}-covering of the support of $\mathcal{P}$.

The rest of the proof now follows identically the proof of Theorem 2.1 with the $A_1, \ldots, A_N$ sets being the connected components of the support of $\mathcal{P}$, since each connected component must be label-homogeneous. □

The unlabeled sample complexity in Theorem 2.1 depends on the gap g between classes because we must have $r_c < g$. Such a scale parameter must appear in our results, since Assumption 2.1 is scale-invariant, yet our algorithm exploits scale-dependent geometric properties of the problem. If we have a conservatively small estimate $\hat{g} \leq g$, then the conclusion of Theorem 2.1 and Theorem 2.2 continue to hold if the connection radius and unlabeled sample complexity are set using the estimate $\hat{g}$. Nevertheless, in some cases we may not have an estimate of g, making it difficult to to apply Algorithm 1. The following result shows that if we have an estimate of the number of high-density

clusters, and these clusters have roughly balanced probability mass, then we are still able to take advantage of the geometric structure even when the distance g is unknown. The idea is to construct a hierarchical clustering of S using single linkage, and then to use a small number of label queries to find a good pruning.

Algorithm 2 Hierarchical single-linkage learning.

Input: Sample $S = \{x_1, \ldots, x_n\}$, $t \in \mathbb{N}$.

1. Let T be the hierarchical clustering of S obtained by single-linkage.
2. Query the labels of a random subset of S of size t.
3. Let $\{\hat{B}_i\}_{i=1}^{M}$ be the coarsest pruning of T such that each $\hat{B}_i$ contains labels from one class.
4. Output $\hat{f}(x) =$ label of nearest $\hat{B}_i$ to x.

Theorem 2.3. *Suppose Assumption 2.1 holds and the density p has C-thick level sets. For any $0 < \epsilon \leq 1/2$, suppose that $\{A_i\}_{i=1}^{N}$ are the connected components of $\{p \geq \epsilon/(2\,\mathrm{Vol}(K))\}$ and for some $\alpha \geq 1$ we have $P(A_i) \leq \alpha P(A_j)$ for all i, j. With probability $\geq 1 - \delta$, running Algorithm 2 with $t = \tilde{O}(\alpha N)$ on an unlabeled sample of size $n = \tilde{O}(\frac{1}{\epsilon^2}(C^{2d}d^{d+1}/g^{2d} + N))$ will have error $\leq \epsilon$.*

Proof. Define $\lambda = \epsilon/(2\,\mathrm{Vol}(K))$ and let $A_1, \ldots, A_N$ be the connected components of $\{p \geq \lambda\}$. Suppose that each A_i set has probability mass at least γ. Under the assumption that the probability mass of the largest A_i is at most α times the mass of the smallest, we have that $\gamma \geq (1-\epsilon)/(\alpha N)$, but the result holds for any arbitrary lower bound γ.

Since we query the labels of points without replacement, the set of labeled examples is an iid sample from the data density p. Whenever $m \geq \frac{2}{\gamma}\ln\frac{2N}{\delta}$, with probability at least $1 - \delta/2$, every set A_i will contain at least one labeled example, since they each have probability mass at least γ. Assume this high probability event holds.

Let g be the margin between classes that is guaranteed by Lemma 2.1. Whenever samples $x, x' \in S$ have $\|x - x'\| \leq g$, they must belong to the same cluster $\hat{B}_i$. Applying an identical covering-style argument as in Theorem 2.1, we have that with probability at least $1 - \delta/2$, for every A_i set there is a cluster, say $\hat{B}_i$, such that:

1. All samples in $A_i \cap S$ are contained in $\hat{B}_i$.
2. For every $x \in A_i$, the nearest cluster to x is $\hat{B}_i$.

Since every A_i set contains at least one labeled example, it follows that whenever two of these high-density clusters belong to different classes, they will contain differently labeled points and therefore will not have been merged by Algorithm 2. It follows that the label of $\hat{B}_i$ must agree with the label of A_i. At this point, the error analysis follows identically as in Theorem 2.1. □

In Section 2.7 we describe a meta-argument that can be used to extend our results into the agnostic setting, where we no longer require that the output code is consistent. Details for the error correcting case are given in Section A.1.

In this section we showed that when there exists linear error correcting correcting output code with low error, then it is possible to reduce the label complexity of learning to the number of high-density clusters, which are the connected components of $\{p \geq \epsilon\}$. The label-complexity of our algorithms is always linear in the number of high density clusters, while the worst-case unlabeled complexity of our algorithms is exponential in the dimension (or intrinsic dimension).

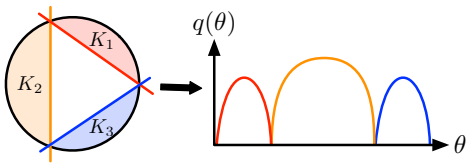


Figure 2.1: An example problem satisfying Assumption 2.2 and the projected density q when the density p is uniform on K.

2.5 One-Versus-All on the Unit Ball

In this section we show that even when the codewords are not well separated, we can still exploit the implicit structure of output codes to reduce the label complexity of learning by clustering the data. Specifically, we consider the implicit structure of a linear one-vs-all classifier over the unit ball:

Assumption 2.2. The instance space $\mathcal{X}$ there exist L linear separators $h_1, \ldots, h_L$ such that: (1) point x belongs to class i iff $h_i(x) > 0$, and (2) for all i, $h_i(x) = w_i^\top x - b_i$ with $\|w_i\| = 1$ and $b_i \geq b_{\min} > 0$.

See Figure 2.1 for an example problem satisfying this condition. Since a one-vs-all classifier is an output code where the code matrix is the identity, the Hamming distance between any pair of codewords is exactly 2. Therefore, in this setting we do not have a result similar to Lemma 2.1 to ensure that the classes are geometrically separated. Instead, we exploit the one-vs-all structure to show the classes are probabilistically separated and employ a robust clustering algorithm.

As before, we study this problem under a mild constraint on the data distribution. For each class i denote the set of points in class i by $K_i = \{x : \|x\| \leq 1, h_i(x) > 0\}$ and $K = \bigcup_{i=1}^{L} K_i$. In this section, we assume that the density p is supported on K with upper and lower bounds:

Assumption 2.3. There exist constants $0 < c_{\text{lb}} \leq c_{\text{ub}}$ s.t. for $x \in K$ we have $c_{\text{lb}} \leq p(x) \leq c_{\text{ub}}$ and otherwise $p(x) = 0$.

This distributional constraint is quite general: it only requires that we will not observe examples for which the one-vs-all classifier would be confused (i.e., where none of its linear separators claim the point) and that the density does not take extreme values. When K is compact, every continuous density supported on K satisfies Assumption 2.3.

Our algorithm for this setting first projects the data onto the unit sphere $\mathcal{S}^{d-1} = \{x \in \mathbb{R}^d : \|x\| = 1\}$ and then applies a robust clustering algorithm to the projected data. The projection does not introduce any errors, since the label of an example is independent of its distance to the origin. This is because each linear separator carves out a spherical cap for its class, and no two class caps overlap. Since we assume that no class contains the origin, it follows that an examples label depends only on its projection to the sphere. We show that projecting to the sphere has the useful property that the projected density goes to zero at the boundary of the classes, which suggests that we can use a robust single-linkage style clustering algorithm to find label-homoegeneous clusters. Algorithm 3 gives pseudocode, using the notation $\theta(u, v) = \arccos(u^\top v)$ for the angle between u and v and $V^d(r)$ is the probability that a uniformly random sample from $\mathcal{S}^{d-1}$ lands in a given spherical cap of angular radius r.

Our first result characterizes the density of the projected data (defined relative to the uniform distribution on $\mathcal{S}^{d-1}$).

Algorithm 3 Robust single-linkage learning.

Input: Sample $S = \{x_1, \dots, x_n\}$, radius $r_c > 0$.

1. Define $r_a = r_c/2$ and $\tau = \frac{c_{\mathrm{lb}}}{2c_{\mathrm{ub}}} V^d(r_a)\epsilon$.
2. Let $v_i = \frac{x_i}{\|x_i\|}$ be the projection of x_i to the sphere.
3. Mark v_i active if $|\{v_j : \theta(v_i, v_j) \le r_a\}| \ge \tau n$ and inactive otherwise for $i \in [n]$.
4. Let $\hat{A}_1, \dots, \hat{A}_N$ be the connected components of the graph G whose vertices are the active v_i with an edge between v_i and v_j if $\theta(v_i, v_j) < r_c$.
5. In decreasing order of size, query the label of each $\hat{A}_i$ until $\le \frac{\epsilon}{4} n$ points belong to unlabeled clusters.
6. Output $\hat{f}(x) =$ label of nearest cluster to $x/\|x\|$.

Lemma 2.2. *Suppose Assumptions 2.2 and 2.3 hold and let $q : \mathcal{S}^{d-1} \to [0, \infty)$ be the density function of the data projected onto the unit sphere. Then $q_{\mathrm{lb}}(v) \le q(v) \le q_{\mathrm{ub}}(v)$, where*

$$q_{\mathrm{lb}}(v) = \begin{cases} c_{\mathrm{lb}} dv_d (1 - (b_i / w_i^\top v)^d) & \text{if } v \in K_i \\ 0 & \text{otherwise}, \end{cases}$$

and $q_{\mathrm{ub}}(v) = c_{\mathrm{ub}}/c_{\mathrm{lb}} \cdot q_{\mathrm{lb}}(v)$, where v_d is the volume of the unit ball in d dimensions.

Proof. Let $X \sim p$ be and set $V = X/\|X\|_2$ so that V is a sample from q. For any set $A \subset \mathcal{S}^{d-1}$, we know that $\Pr(V \in A) = \Pr(X \in \mathrm{cone}(A))$, where $\mathrm{cone}(A) = \{rv : r > 0, v \in A\}$, which gives

$$\Pr(V \in A) = \Pr(X \in \mathrm{cone}(A)) = \int_{x \in \mathrm{cone}(A)} p(x)\, dx = \int_{v \in A} dv_d \int_{r=0}^{\infty} p(rv) r^{d-1}\, dr\, d\mu_\circ(v),$$

where the last inequality follows by a change of variables x to (r, v) where $r = \|x\|_2$ and $v = x/\|x\|_2$. The term r^{d-1} is the determinant of the Jacobian of the change of variables, and the term dv_d, which is the surface area of $\mathcal{S}^{d-1}$, appears since $\mu_\circ$ is normalized so that $\mu_\circ(\mathcal{S}^{d-1}) = 1$. From this, it follows that the density function q can be written as

$$q(v) = dv_d \int_{r=0}^{\infty} p(rv) r^{d-1}\, dr, \tag{2.2}$$

since integrating this function over any set A gives the probability that V will land in A. From our assumptions on p, we know that

$$p(rv) \ge \sum_{i=1}^{L} \mathbb{I}\{rv \in K_i\} c_{\mathrm{lb}}.$$

Moreover, we can rewrite the indicator as $\mathbb{I}\{rv \in K_i\} = \mathbb{I}\{\frac{b_i}{w_i^\top v} < r \le 1\}$. Substituting this into

(2.2) gives

$$\begin{aligned} q(v) &\geq \sum_{i=1}^{L} c_{\mathrm{lb}} dv_d \int_{r=0}^{\infty} \mathbb{I}\Big\{ \frac{b_i}{w_i^\top v} < r \leq 1 \Big\} r^{d-1}\, dr \\ &= \sum_{i=1}^{L} \mathbb{I}\{v \in K_i\} c_{\mathrm{lb}} dv_d \int_{r=b_i/(w_i^\top v)}^{1} r^{d-1}\, dr \\ &= \sum_{i=1}^{L} \mathbb{I}\{v \in K_i\} c_{\mathrm{lb}} dv_d (1 - b_i/(w_i^\top v)^d) \\ &= q_{\mathrm{lb}}(v) \end{aligned}$$

Note that the indicator $\mathbb{I}\{v \in K_i\}$ appears in line 2 because the integral is only non-zero when $b_i/(w_i^\top v) < 1$, which is exactly the condition that $v \in K_i$. The upper bound on q follows by an identical argument using the upper bound on $p(rv)$. □

Both bounds are defined piece-wise with one piece for each class. Restricted to class i, both the $q_{\mathrm{lb}}(v)$ and $q_{\mathrm{ub}}(v)$ are decreasing functions of $\theta(w_i, v)$, which implies that their λ-level sets are spherical caps. Therefore, each class contributes one large connected component to the level set of q that is roughly a spherical cap centered at the point w_i and the density of q goes to zero at the boundary of each class. Our main result is as follows:

Theorem 2.4. *Suppose Assumptions 2.2 and 2.3 hold and that f^* is consistent. There exists an r_c satisfying $r_c = \Omega(\epsilon c_{\mathrm{lb}}/(c_{\mathrm{ub}}^2 b_{\min}))$ such that with probability at least $1-\delta$, running Algorithm 3 with parameter r_c on an unlabeled sample of size $n = \tilde{O}((c_{\mathrm{ub}}^4 d/(\epsilon^2 c_{\mathrm{lb}}^2 b_{\min}^2))^d)$ will query at most L labels and output a classifier with error at most ϵ.*

Note that if the scale parameter $b_{\min}$ is unknown, the conclusion of Theorem 2.4 continues to hold if the connection radius r_c and unlabeled sample complexity n are set using a conservatively small estimate $\hat{b}_{\min}$ satisfying $\hat{b}_{\min} \leq b_{\min}$. This comes at the cost of an increased unlabeled sample complexity.

Before proving Theorem 2.4, we develop some general results for the robust linkage clustering algorithm. More generally, Algorithm 3 can be applied in any metric space $(\mathcal{X}, d)$ by replacing θ with the distance metric d and suitable settings for the internal parameters r_a and τ. For the robust linkage approach to have low error, each class should have one large connected component in the graph G constructed by the algorithm so that: (1) with high probability a new point in class i will be nearest to that largest component, and (2) the large components of different classes are separated. Intuitively, G will have these properties if each positive region K_i has a connected high-density inner region A_i covering most of its probability mass and when it is rare to observe a point that is close to two or more classes. This notion is formalized below.

Let S be any set in $\mathcal{X}$. We say that a path $\pi : [0,1] \to \mathcal{X}$ *crosses* S if the path starts and ends in different connected components of the complement of S in $\mathcal{X}$ and we say that the *width* of S is the length of the shortest path that crosses S.

Definition 2.1. The sets $A_1, \ldots, A_k$ are (r_c, r_a, τ, γ)-clusterable under probability P if there exists a separating set S of width at least r_c such that: (1) Each A_i is connected; (2) If $x \in \mathcal{X}$ satisfies $d(x, A_i) \leq r_c/3$ then $\Pr_{X\sim P}(X \in B(x, r_a)) > \tau + \gamma$; (3) If $x \in A_i$ then $\Pr_{X\sim P}(X \in$

$B(x, r_c/3)) > \gamma$; (4) Every path from A_i to A_j crosses S; and (5) If $x \in S$ then $\Pr_{X \sim P}(X \in B(x, r_a)) < \tau - \gamma$.

Note that typically there must be a gap between the set A_i and the set S in order to satisfy the probability requirements (i.e., the set S will be smaller than $\mathcal{X} - \bigcup_{i=1}^{L} A_i$). The first three properties ensure that each set A_i will have one large connected component and the remaining two properties ensure that these connected components will be disconnected. Following an analysis similar to that of the cluster tree algorithm of Chaudhuri and Dasgupta [44] gives the following result.

Lemma 2.3. *Suppose that the sets $A_1, \ldots, A_N$ are (r_c, r_a, τ, γ)-clusterable with respect to distribution P. For any failure probability $\delta > 0$, let G be the graph constructed by Algorithm 3 run on a sample S of size $O(\frac{1}{\gamma^2}(D + \ln \frac{1}{\delta})$, where D is the VC-dimension of balls in $(\mathcal{X}, d)$, with parameters and r_c, r_a, and τ. Define $K_i = \{x \in S : d(x, A_i) \leq r_c/3\}$ for each $i \in [N]$. With probability at least $1 - \delta$, the graph G has the following properties:*

1. **Complete:** *For each i, all samples in K_i are active and included in the graph G.*
2. **Separated:** *For any $i \neq j$, there is no path in G from K_i to K_j.*
3. **Connected:** *For every i, the set K_i is connected in G.*
4. **Extendible:** *For any point $x \in A_i$, the nearest connected component of G to x contains K_i.*

Proof. The proof technique used here follows a similar argument as Chaudhuri and Dasgupta [44].

We use a standard VC bound [141] to relate the probability constraints in the clusterability definition to the empirical measure $\hat{P}$. For our value of n we have

$$\Pr\big(\sup_{x,r} \big|\hat{P}(B(x,r)) - P(B(x,r))\big| > \gamma\big) < \delta.$$

This implies that with probability at least $1 - \delta$ for all points x we have: (1) if $d(x, A_i) \leq \frac{r_c}{3}$ for any i then $\hat{P}(B(x, r_a)) > \tau$; (2) if $x \in S$ then $\hat{P}(B(x, r_a)) < \tau$; and (3) if $x \in A_i$ for any i then $\hat{P}(B(x, \frac{r_c}{3})) > 0$. We now use these facts to prove that the graph G has the completeness, separation, and connectedness properties.

Completeness follows from the fact that every sample $x \in \hat{K}_i$ is within distance $r_c/3$ of A_i and therefore $\hat{P}(B(x, r_a)) > \tau$.

To show separation, first observe that every sample $z \in S$ that belongs to S will be marked as inactive, since $\hat{P}(B(z, r_a)) < \tau$. Now let $x \in \hat{K}_i$ and $x' \in \hat{K}_j$ for $i \neq j$. Since the graph G does not contain any samples in the set S, any path in G from x to x' must have one edge that crosses S. Since the width of S is at least r_c, this edge would not be included in the graph G, and therefore G does not include a path from x to x'.

To show connectedness, let x and x' be any pair of samples in $\hat{K}_i$ and let v and v' be their nearest points in A_i, respectively. By definition of $\hat{K}_i$, we know that $d(x, v) < r_c/3$ and $d(x', v') < r_c/3$. Since A_i is a connected set, there is a path $\pi : [0, 1] \to A_i$ in A_i starting at v and ending at v'. Cover the path π with a sequence of points $z_1, \ldots, z_k$ such that $d(z_j, z_{j+1}) < r_c/3$ for all j and the path π is covered by the balls $B(z_j, r_c/3)$. Further, choose $z_1 = v$ and $z_k = v'$. Since each point z_j belongs to A_i, the empirical probability mass of the ball $B(z_j, r_c/3)$ is non-zero, which implies that it must contain at least one sample point, say $y_j \in S$. We may take $y_1 = x$ and $y_k = x'$. Since every sample $y_1, \ldots, y_k$ is within distance $r_c/3$ of A_i, they are all active and included in the graph G. Moreover,

since $d(y_j, y_{j+1}) < r_c$, we have that the path $x = y_1 \to \cdots \to y_k = x'$ is a path connecting x and x' in G, as required.

Finally to show extensibility, let $x \in A_i$ be any point. By the uniform convergence for balls, $P(x, r_c/3)$ has non-zero empirical probability mass and therefore contains at least one active sample, say z. Since z is within distance $r_c/3$ of A_i, it belongs to the set K_i. Now let z^* be the closest active sample to x. We must have $d(x, z^*) \le d(x, z) \le r_c/3$ and it follows that $d(z, z^*) \le d(z, x) + d(x, z^*) \le 2r_c/3 < r_c$. Therefore, z^* also belongs to K_i, as required. □

We now prove Theorem 2.4 by combining Lemmas 2.2 and 2.3:

Proof of Theorem 2.4. For each class $i \in [L]$, define $A_i = \{q_{\text{ub}}^{(i)} \ge \epsilon\}$. We will show that the sets $A_1, \ldots, A_L$ are (r_c, r_a, γ, τ)-clusterable for appropriate choices of the parameters. Then Lemma 2.3 will guarantee that with high probability, the clustering produced by Algorithm 3 will approximate the connected components of the ϵ-level of $\{q_{\text{ub}} \ge \epsilon\}$.

Recall that for each class $i \in [L]$, the sets $\{q_{\text{ub}}^{(i)} \ge \epsilon\}$ and $\{q_{\text{lb}}^{(i)} \ge \epsilon\}$ are spherical caps. To simplify notation, let $C(u, r) = \{v \in \mathcal{S}^{d-1} : \theta(v, u) \le r\}$ denote the spherical cap of angular radius r centered at u. Let $\rho_{\text{ub}}^{(i)}(\lambda) = \arccos(b_i(1 - \lambda/(c_{\text{ub}} d v_d))^{-1/d})$ denote the angular radius of $\{q_{\text{ub}}^{(i)} \ge \epsilon\}$, so that $\{q_{\text{ub}}^{(i)} \ge \epsilon\} = C(w_i, \rho_{\text{ub}}^{(i)}(\epsilon))$, and $\rho_{\text{lb}}^{(i)}(\lambda)$, defined similarly, be the angular radius of $\{q_{\text{lb}}^{(i)} \ge \epsilon\}$. Define $\tilde{\epsilon} = \frac{c_{\text{lb}}}{c_{\text{ub}}}\epsilon$ and suppose for the moment that we can find an activation radius $r_a > 0$ small enough so that the following inequalities hold for all classes $i = 1, \ldots, L$:

$$\frac{5}{3} r_a \le \rho_{\text{lb}}^{(i)}\left(\frac{3\tilde{\epsilon}}{4}\right) - \rho_{\text{ub}}^{(i)}(\epsilon) \qquad \text{and} \qquad 2r_a \le \rho_{\text{ub}}^{(i)}(0) - \rho_{\text{ub}}^{(i)}\left(\frac{\tilde{\epsilon}}{4}\right).$$

Given such an activation radius, we will show that the sets $A_1, \ldots, A_L$ are (r_c, r_a, τ, γ)-clusterable with $r_c = 2r_a$, $\tau = \frac{\tilde{\epsilon} V^d(r_a)}{2}$, and $\gamma = \frac{\tilde{\epsilon} V^d(r_c/3)}{4}$ and the separating set is $S = \{v \in \mathcal{S}^{d-1} : \theta(v, w_i) \ge \rho_{\text{ub}}^{(i)}(0) - r_a \text{ for all } i\}$:

1. *Connection:* Each A_i set is a spherical cap and therefore connected.

2. *High-density near A_i:* Let $v \in \mathcal{S}^{d-1}$ be such that $\theta(v, A_i) < r_c/3$ and let $u \in C(v, r_a)$ be any point in the spherical cap of angular radius r_a about v. By the triangle inequality, we know that $\theta(w_i, u) \le \theta(w_i, v) + \theta(v, u) \le \rho_{\text{ub}}^{(i)}(\epsilon) + \frac{5}{3} r_a \le \rho_{\text{lb}}^{(i)}(\frac{3\tilde{\epsilon}}{4})$. This implies that $q(u) \ge \frac{3\tilde{\epsilon}}{4}$ for all points in $C(v, r_a)$ and therefore $\Pr_{V \sim q}(V \in C(v, r_a)) \ge \frac{4\tilde{\epsilon}}{3} V^d(r_a) \ge \tau + \gamma$.

3. *High-density inside A_i:* Now let $v \in A_i$. Since $r_c/3 < r_a$, the above arguments show that $q(u) \ge \frac{4\tilde{\epsilon}}{3}$ for all points $u \in C(v, r_c/3)$ and therefore $\Pr_{V \sim q}(V \in C(v, r_c/3)) \ge \frac{4\tilde{\epsilon}}{3} V^d(r_c/3) \ge \gamma$.

4. *Separation by the set S:* For each class i, the set S contains the annulus $\{v \in \mathcal{S}^{d-1} : \rho_{\text{ub}}^{(i)}(0) - r_a \le \theta(w_i, v) \le \rho_{\text{ub}}^{(i)}(0)\}$ which has width r_a. Any path from one A_i to another A_j must cross two such annuli, each of width r_a, so the length of the path crossing S is at least $2r_a = r_c$.

5. *Low density inside S:* Finally, let v be any point in the set S and let $u \in C(v, r_a)$. For any class i, the reverse triangle inequality gives that $\theta(w_i, v) \ge \theta(v, w_i) - \theta(u, w_i) \ge \rho_{\text{ub}}^{(i)}(0) - 2r_a \ge \rho_{\text{ub}}^{(i)}(\frac{\tilde{\epsilon}}{4})$. Since this is true for all classes i, we have $q(v) \le \frac{\tilde{\epsilon}}{4}$ and therefore $\Pr_{V \sim q}(V \in C(v, r_a)) \le \frac{\tilde{\epsilon}}{4} V^d(r_a) \le \tau - \gamma$.

It follows that the sets $A_1, \ldots, A_L$ are (r_c, r_a, τ, γ)-clusterable and it only remains to find an activation radius r_a that satisfies the above inequalities. Since the robust linkage algorithm needs to estimate the probability mass of balls to within error $\gamma = \frac{\tilde{\epsilon} V^d (2r_a/3)}{4}$, we want this activation radius to be not too small.

Taking the first order Taylor expansion of the $\rho_{\text{lb}}^{(i)}$ and $\rho_{\text{ub}}^{(i)}$ functions, we have:

$$\rho_{\text{lb}}^{(i)}(\lambda) = \arccos(b_i) - \frac{b_i}{\sqrt{1-b_i^2}} \frac{1}{c_{\text{lb}} d v_d} \lambda + O(\lambda^2)$$

$$\rho_{\text{ub}}^{(i)}(\lambda) = \arccos(b_i) - \frac{b_i}{\sqrt{1-b_i^2}} \frac{1}{c_{\text{ub}} d v_d} \lambda + O(\lambda^2),$$

as $\lambda \to 0$. Therefore, we have that

$$\rho_{\text{lb}}^{(i)}(3\tilde{\epsilon}/4) - \rho_{\text{ub}}^{(i)}(\epsilon) = \frac{1}{4 d v_d c_{\text{ub}}} \cdot \frac{b_i}{\sqrt{1-b_i^2}} \epsilon + O(\epsilon^2)$$

and

$$\rho_{\text{ub}}^{(i)}(0) - \rho_{\text{ub}}^{(i)}(\tilde{\epsilon}/4) = \frac{c_{\text{lb}}}{4 d v_d c_{\text{ub}}^2} \cdot \frac{b_i}{\sqrt{1-b_i^2}} \epsilon + O(\epsilon^2),$$

which shows that it is sufficient to set $r_a = \frac{3 c_{\text{lb}}}{20 d v_d c_{\text{ub}}^2} \cdot \frac{b_i}{\sqrt{1-b_i^2}} \epsilon + O(\epsilon^2) = \Omega(\frac{c_{\text{lb}}}{c_{\text{ub}}^2} b_{\min} \epsilon)$ as $\epsilon \to 0$ and it follows that $n = O(\frac{1}{\gamma^2}(d + \ln \frac{1}{\delta})) = \tilde{O}((c_{\text{ub}}^4 d / (\epsilon^2 c_{\text{lb}}^2 b_{\min}^2))^d)$.

Finally, we show that the algorithm correctly recovers the labels of the large clusters. For $n = \tilde{O}(L/\epsilon^2)$, we have that with probability at least $1 - \delta$ the following holds simultaneously for all 2^L subsets $I \subset [L]$: $\left|\hat{P}(A_I) - P(A_I)\right| \leq \epsilon/4$, where $A_I = \bigcup_{i \in I} A_i$. Since all samples in A_I are marked as active (by Lemma 2.3), this implies that all but at most $\frac{\epsilon}{4} n$ of the active points will belong to the A_i sets. It follows that if the algorithm queries the labels of the L largest clusters, they will also contain all but $\frac{\epsilon}{4} n$ active samples.

On the other hand, whenever we query the label of one of the A_i sets, we know that we will correctly classify every test point belonging to A_i, so the error of the resulting classifier is at most the probability mass of $\{q_{\text{ub}} \leq \epsilon\}$ together with the probability mass of the A_i sets for which we did not query the label. Since the unqueried A_i sets have empirical probability mass at most $\epsilon/4$ and we have uniform convergence for all unions of A_i sets to within error $\epsilon/4$, it follows that the total probability mass of the unlabeled A_i sets is at most $\epsilon/2$ and it follows that the error of the resulting classifier is at most ϵ. □

There are two main differences between the sample complexity of Theorem 2.4 and the results from Section 2.4. First, the unlabeled sample complexity now has an ϵ^{-2d} dependence, rather than only ϵ^{-2}. This is because the distance between the connected components of $\{p \geq \epsilon\}$ goes to zero (in the worst case) as $\epsilon \to 0$, so our algorithm must be able to detect low-density regions of small width. In contrast, Lemma 2.1 allowed us to establish a non-diminishing gap $g > 0$ between the classes when the codewords were well separated. On the other hand, the label complexity in this setting is better, scaling with L instead of N, since we are able to establish that each class will have one very large cluster containing nearly all of its data.

Theorem A.2 in the appendix gives an analysis of Algorithm 3 in the agnostic setting of Section 2.7.

2.6 The Boundary Features Condition

Finally, in this section we introduce a novel condition on the code matrix called the boundary features condition that captures the intuition that every binary classification task should be significant. Assumption 2.4 formalizes this intuition.

Assumption 2.4. There exists a code matrix $C \in \{\pm 1\}^{L \times m}$, linear functions $h_1, \ldots, h_m$, and a scale parameter $R > 0$ so that: (1) for any point x in class y, we have $h(x) = C_y$; (2) for each h_j, there exists a class i such that negating the j^{th} entry of C_i produces a codeword C_i' not in C and there exists a point x on the hyperplane $h_j = 0$ such that every point in $B(x, R)$ has either code word C_i or C_i'; and (3) any pair of points $x, x' \in \mathcal{X}$ such that $h(x)$ and $h(x')$ are not codewords in C and $h(x) \neq h(x')$ must have $\|x - x'\| \geq R$.

Part (1) of this assumption requires that the output code classifier is consistent, part (2) is a condition that guarantees every linear separator h_j separates at least one class i from a region of space that does not belong to any class, and part (3) requires that points with codewords not in the code matrix must either have the same codeword or be separated by distance R. Part (3) allows us to simplify our algorithm and analysis and is trivially satisfied in cases where all points in $\mathcal{X}$ that do not belong to any class have the same codeword, as is the case for one-vs-all classification and the problem in Figure 2.2.

Problems in this setting are more challenging than those of the previous sections because they may not be amenable to clustering-based learning strategies. Whenever the Hamming distance between a pair of codewords is only 1, this implies that one of the linear separators h_j forms a shared boundary between the classes, and therefore these classes may be connected by a large and high-density region. Instead, Assumption 2.4 guarantees that for every linear separator h_j, there is some ball $B(x, R)$ centered on h_j that is half-contained in the set of points belonging to some class i and the other half belongs to the set of points that do not belong to any class. Therefore, by looking for hyperplanes that locally separate sample data from empty regions of space, we can recover the linear separator h_j from the local *absence* of data. Define $K_i = \{x \in \mathcal{X} : h(x) = C_i\}$ to be the set of points that belong to class i and $K = \bigcup_{i=1}^{L} K_i$. Under the condition that the density p is supported on K and is upper and lower bounded, we exploit this structure in an algorithm that directly learns the linear separators $h_1, \ldots, h_m$.

Our hyperplane detection algorithm works by searching for balls of radius r whose centers are sample points such that one half of the ball contains very few samples. If a half-ball contains very few sample points then it must be mostly disjoint from the set K. But since its center point belongs to the set K, this means that the hyperplane defining the half-ball is a good approximation to at least one of the true hyperplanes. See Figure 2.3 for examples of half-balls that would pass and fail this test. The

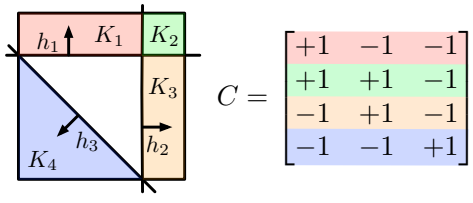


Figure 2.2: An example of the boundary features problem. The arrows indicate the positive side of the linear functions.

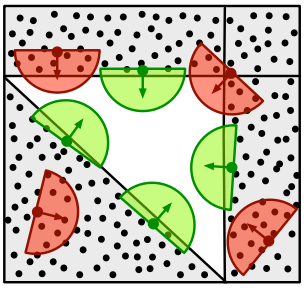


Figure 2.3: Examples of half-balls that would be included (green) or excluded (red) by the plane detection algorithm.

collection H of hyperplanes produced in this way partition the space into cells. Our algorithm queries the labels of the cells containing the most sample points and classifies test points based on the label of their cell in the partition (and if the label is unknown, we output a random label). Pseudocode is given in Algorithm 4 using the following notation: for any center $x \in \mathcal{X}$, radius $r \geq 0$, and direction $w \in \mathcal{S}^{d-1}$, let $B^{1/2}(x, r, w) = \{y \in B(x, r) : w^\top (y - x) > 0\}$ and define $p^{1/2}(r) = \frac{1}{2} c_{\text{lb}} r^d v_d$.

Algorithm 4 Plane-detection algorithm.

Input: Sample $S = \{x_1, \ldots, x_n\}$, $r > 0$, $\tau > 0$.

1. Initialize set of candidate hyperplanes $H = \emptyset$.
2. For all samples $\hat{x} \in S$ with $B(\hat{x}, r) \subset \mathcal{X}$:
   (a) Let $\hat{w} = \operatorname{argmin}_{w \in \mathcal{S}^{d-1}} |B^{1/2}(\hat{x}, r, w) \cap S|$.
   (b) If $|B^{1/2}(\hat{x}, r, \hat{w}) \cap S|/n < \tau$, add $(\hat{x}, \hat{w})$ to H.
3. Let $\{\hat{C}_i\}_{i=1}^N$ be the partitioning of $\mathcal{X}$ induced by H.
4. Query the label of the L cells with the most samples.
5. Output $\hat{f}(x)$ = label of C_i containing x.

Each candidate hyperplane produced by Algorithm 4 is associated with a half-ball that caused it to be included in H. In fact, we can think of the pairs $(\hat{x}, \hat{w})$ in H as either encoding the linear function $\hat{h}(x) = w^\top (x - \hat{x})$ or the half-ball $B^{1/2}(\hat{x}, r, \hat{w})$, where r is the scale parameter of the algorithm. Most of our arguments will deal with the half-balls directly, so we adopt the second interpretation. The analysis of Algorithm 4 has two main steps. First, we show that the face of every half-ball in the set H is a good approximation to at least one of the true hyperplanes, and that every true hyperplane is well approximated by the face of at least one half-ball in H. Second, using the fact that the half-balls in H are good approximations to the true hyperplanes, we argue that the output classifier will only be inconsistent with the true classification rule in a small margin around each of the true linear separators. Then the error of the classification rule is easily bounded by bounding the probability mass of these margins.

To measure the approximation quality, we say that the half-ball $B^{1/2} = B^{1/2}(\hat{x}, r, \hat{w})$ is an α-approximation to the linear function h if $\Pr_{X \sim B^{1/2}}(\text{sign}(h(X)) = \text{sign}(h(\hat{x}))) \leq \alpha$, where $\Pr_{X \sim B^{1/2}}$ denotes the probability when X is sampled uniformly from the half-ball $B^{1/2}$. The motivation for this definition is as follows: given any point $\hat{x} \in \mathcal{X}$, the half-ball $B^{1/2}(\hat{x}, r, \hat{w})$ will be an α-approximation to h_i only if $\hat{x}$ is on one side of the decision surface of h_i and all but an α-fraction of the half-ball's volume is on the other side. Intuitively, this means that the face of the half-ball must

approximate the decision surface of the function h_i.

The following Lemma shows that when Algorithm 4 is run with appropriate parameters and on a large enough sample drawn from the data distribution, then with high probability the algorithm will include at least one half-ball in H α-approximating each true hyperplane h_i and every half-ball in H will be an α-approximation to at least one true hyperplane. Recall that $p^{1/2}(r) = \frac{1}{2}c_{\text{lb}}r^d v_d$ is a lower bound on the probability mass of a half-ball of radius r contained in the set K.

Lemma 2.4. *Fix any $\alpha > 0$ and confidence parameter $\delta > 0$. Let H be the set of half-balls produced by Algorithm 4 when run with parameters $r = R/2$ and $\tau = \frac{1}{2}\alpha p^{1/2}(r)$ on a sample of size $n = O(\frac{1}{\gamma^2}(\ln^2 \frac{d}{\gamma} + \ln \frac{1}{\delta}))$ where $\gamma = \frac{2}{5}\tau = \frac{1}{5}\alpha p^{1/2}(r)$. Then with probability at least $1-\delta$, every half-ball in H will be an α-approximation to at least one true hyperplane h_i, and every true hyperplane h_i will be α-approximated by at least one half-ball in H.*

Proof. Since the VC-dimension of both balls and half-spaces in $\mathbb{R}^d$ is $d+1$, the VC-dimension of the set of intersections of balls and up to two half-spaces is $O(d \ln d)$. Therefore, by a standard VC-bound [141], if we see an iid sample S of size $n = O(\frac{1}{\gamma^2}(\ln^2 \frac{d}{\gamma} + \ln \frac{1}{\delta}))$, then with probability at least $1-\delta$ the empirical measure of any ball intersected with up to two half-spaces will be within γ of its true probability mass. In other words, the fraction of the sample set S that lands in any ball intersected with up to two half-spaces will be within γ of the probability that a sample X drawn from $\mathcal{P}$ will land in the same set. For the remainder of the proof, assume that this high-probabilty event holds.

First, we show that every half-ball in the set H is an α-approximation to at least one true hyperplane. Suppose otherwise, then there is a half-ball $B^{1/2} = B^{1/2}(\hat{x}, r, \hat{w})$ with $(\hat{x}, \hat{w}) \in H$ that is not an α approximation to any true hyperplane h_i. The center $\hat{x}$ of the half-ball must belong to the positive region K, since it is one of the sample points. If the half-ball $B^{1/2}$ is contained entirely in the set K, then the probability that a new sample X drawn from $\mathcal{P}$ will land in the half-ball $B^{1/2}$ is $p^{1/2}(r)$ and therefore the fraction of samples that landed in the half-ball is at least $p^{1/2}(R/2) - \gamma$. But since $p^{1/2}(r) - \gamma \geq \frac{4}{5}\alpha p^{1/2}(r) > \tau$, this contradicts the half-ball being included in the set H. Otherwise, the half-ball contains at least one point y that does not belong to the set K (i.e., it does not belong to any class). Since $\hat{x}$ is in the set K, there is at least one true hyperplane h_i that separates $\hat{x}$ from y. Since $r = R/2 < R$, every other point y' in the half-ball that does not belong to any class must have the same code word as y (since, by assumption, points outside of K that do not belong to any class must have the same code word when they are closer than R), and therefore must be on the same side of h_i as y. It follows that all points in the half-ball on the same side of h_i as $\hat{x}$ (i.e., those points for which the sign of h_i matches the sign of $h_i(\hat{x})$) belong to the set K. But, since the half-ball is not an α-approximation to h_i, this implies that at least an α fraction of the half-ball's volume must belong to the set K. Therefore, the probability that a new sample x drawn from the data distribution p belongs to the half-ball can be lower bounded as follows:

$$\Pr_{x \sim p}(x \in B^{1/2}) \geq c_{\text{lb}} \operatorname{Vol}(B^{1/2} \cap K) = c_{\text{lb}} \operatorname{Vol}(B^{1/2}) \frac{\operatorname{Vol}(B^{1/2} \cap K)}{\operatorname{Vol}(B^{1/2})} \geq \alpha p^{1/2}(r).$$

By the uniform convergence argument, the fraction of the samples in S contained in the half-ball $B^{1/2}$ is at least $\alpha p^{1/2}(r) - \gamma > \tau$, which contradicts the half-ball being in H. In either case we arrived at a contradiction and it follows that every half-ball in H is an α-approximation to at least one true hyperplane h_i.

Finally, we show that the set H will contain at least one half-ball that is an α-approximation to each true hyperplane h_i. Fix any true hyperplane h_i. By assumption, there is a class ℓ and a point x_0 on the decision surface of h_i so that one half-ball of $B(x_0, R)$ with face h_i is is contained in $K_\ell \subset K$ and the other half-ball is disjoint from K. Suppose WLOG that the half-ball on the negative side of h_i is contained in K (the case when the half-ball on the positive side is contained in K is identical). Define $\rho > 0$ to be the width such that the probability that a new sample X from $\mathcal{P}$ lands in the slice of the ball $S = \{x \in B(\hat{x}_0, r) : h_i(x) \in [-\rho, 0]\}$ is equal to $\tau - \gamma$. Note that, since the half-ball on the negative side of h_i is a subset of K and $\tau - \gamma = \frac{3}{10}\alpha p^{1/2}(r) < p^{1/2}(r)$, such a value of ρ always exists. Since $\tau - \gamma > \gamma$, the uniform convergence argument guarantees that there will be at least one sample point in the slice, say $\hat{x} \in S$. Since $\hat{x}$ is within distance $r = R/2$ of the point x_0, the ball $B(\hat{x}, r)$ is contained in $B(x_0, R)$. Therefore, the ball of radius r centered at $\hat{x}$ only contains points that either belong to class ℓ or no class, since only the linear separator h_i passes through this ball. By construction, the half-ball $B^{1/2}(\hat{x}, r, w_i)$ (where w_i is the coefficient vector defining $h_i(x) = w_i^\top x - b_i$) with face parallel to h_i intersects the set K in a slice of width at most ρ and therefore has probability mass at at most $\tau - \gamma$. It follows that the direction $\hat{w}$ that minimizes the number of samples in the half-ball $B^{1/2}(\hat{x}, r, \hat{w})$ will result in the half-ball containing at most a τ fraction of the sample set, and therefore the pair $(\hat{x}, \hat{w})$ will be included in H, and this will be an α-approximation to h_i. □

Naturally, if a half-ball $B^{1/2}(\hat{x}, r, \hat{w})$ is an α-approximation to the linear function h, we expect that the decision surface of $\hat{h}(x) = \hat{w}^\top(x - \hat{x})$ is similar to the decision surface of h. In turn, this suggests that either $\hat{h}(x)$ or $-\hat{h}(x)$ should take similar function values to $h(x)$ (since the coefficient vectors are normalized). We first give a simple probability lemma that bounds the fraction of a ball contained between two parallel hyperplanes, one passing through the ball's center. The proof of Lemma 2.5 is in Section A.3 of the appendix.

Lemma 2.5. *Let $r > 0$ be any radius and X be a random sample drawn uniformly from the ball of radius r centered at the origin. For any width $0 \leq \rho \leq r/\sqrt{2}$, the probability that the first coordinate of X lands in $[0, \rho]$ can be bounded as follows:*

$$\sqrt{\frac{d}{2^d \pi}}\frac{\rho}{r} \leq \Pr_{X \sim B(r,0)}(X_1 \in [0, \rho]) \leq \sqrt{\frac{d+1}{2\pi}}\frac{\rho}{r}.$$

Using Lemma 2.5, we show the following:

Lemma 2.6. *Let the half-ball $B^{1/2}(\hat{x}, r, \hat{w})$ be an α-approximation to the linear function $h(x) = w^\top x - b$ with $\|w\| = 1$, $\hat{x} \in \mathcal{X}$, and $\alpha < \frac{1}{2}$. Let D be the diameter of $\mathcal{X}$. If $h(\hat{x}) < 0$ then for all $x \in \mathcal{X}$ we have*

$$|h(x) - \hat{h}(x)| \leq \left(2D + \sqrt{\frac{2^d \pi}{d}}\frac{r}{2}\right)\sqrt{\alpha},$$

where $\hat{h}(x) = \hat{w}^\top(x - \hat{x})$. Otherwise, if $h(\hat{x}) > 0$ then the same upper bound holds for $|h(x) + \hat{h}(x)|$.

Proof. Suppose that $h(\hat{x}) < 0$ and let X be a uniformly random sample from the half-ball $B^{1/2} = B^{1/2}(\hat{x}, r, \hat{w})$. By assumption, we know that $\Pr(h(X) < 0) \leq \alpha$.

First we show that $\|w - \hat{w}\|$ is small. Since $\alpha < 1/2$ we have that $w^\top \hat{w} > 0$. To see this, notice that we must have $h(\hat{x} + r\hat{w}) \geq 0$, since otherwise at least half of the half-ball would be on the negative side of h. Define $g(x) = w^\top(x - \hat{x})$ to be the linear function whose decision surface runs

parallel to that of h but passes through the point $\hat{x}$. Since $h(x) = g(x) + h(\hat{x}) \leq g(x)$, we have that $\alpha > \Pr(h(X) < 0) \geq \Pr(g(X) < 0)$. Moreover, since the decision surface of g passes through the center of the half-ball $B^{1/2}$ and the uniform distribution on the half-ball is radially symmetric about the point $\hat{x}$, we have that $\Pr(g(X) < 0) = \frac{\theta(w,\hat{w})}{\pi}$. It follows that $\theta(w, \hat{w}) \leq \pi\alpha$. Using this fact, we can bound $\|w - \hat{w}\|$ as follows:

$$\|w - \hat{w}\|^2 = \|w\|^2 + \|\hat{w}\|^2 - 2w^\top \hat{w} = 2(1 - w^\top \hat{w}).$$

Since $w^\top \hat{w} = \cos(\theta(w, \hat{w}))$ and on the interval $[0, \pi/2]$, the $\cos(\theta)$ function is decreasing and lower bounded by $1 - \frac{2}{\pi}\theta$, we have that $2(1 - w^\top \hat{w}) \leq 4\alpha$. Taking the square root gives that $\|w - \hat{w}\| \leq 2\sqrt{\alpha}$.

Next we show that $|h(\hat{x})|$ (the distance from $\hat{x}$ to the decision surface of h) is not too large. The half-ball $B^{1/2}(\hat{x}, r, w)$, whose direction w matches the coefficient vector of h is one half-ball centered at $\hat{x}$ of radius r minimizing the fraction of its volume contained on the same side of h as $\hat{x}$. This is because every point in the ball $B(\hat{x}, r)$ not on the same side as $\hat{x}$ is contained in $B^{1/2}(\hat{x}, r, w)$. Let Y be uniformly sampled from $B^{1/2}(\hat{x}, r, w)$. By construction of the half-ball Y is sampled from, we have that $\Pr(h(X) < 0) \geq \Pr(h(Y) < 0)$, which gives

$$\alpha \geq \Pr_{X \sim B^{1/2}(\hat{x}, r, \hat{w})}\big(h(X) < 0\big) \geq \Pr_{Y \sim B^{1/2}(\hat{x}, r, w)}\big(h(Y) < 0\big) \geq \sqrt{\frac{d}{2^d \pi}} \frac{2|h(\hat{x})|}{r},$$

which implies that

$$|h(\hat{x})| \leq \sqrt{\frac{2^d \pi}{d}} \frac{r\alpha}{2}.$$

Finally, let x' be any point on the decision surface of h, so that $h(x) = w^\top(x - x')$. Combining the above calculations we have

$$\begin{aligned}
|h(x) - \hat{h}(x)| &= |w^\top(x - x') - \hat{w}^\top(x - \hat{x})| \\
&= |w^\top(x - \hat{x}) + w^\top(\hat{x} - x') - \hat{w}^\top(x - \hat{x})| \\
&= |(w - \hat{w})(x - \hat{x}) + w^\top(\hat{x} - x')| \\
&\leq \|w - \hat{w}\|\|x - \hat{x}\| + |h(\hat{x})| \\
&\leq 2\sqrt{\alpha} D + \sqrt{\frac{2^d \pi}{d}} \frac{r\alpha}{2} \\
&\leq (2D + \sqrt{\frac{2^d \pi}{d}} \frac{r}{2})\sqrt{\alpha},
\end{aligned}$$

as required. The proof of the case when $h(x) > 0$ follows by applying the above arguments to the function $-h$. □

Recall that for any hyperplane $h(x) = w^\top x - b$ with $\|w\|_2 = 1$, the distance from point x to the decision surface of h is $|h(x)|$. The above lemma implies that if $B^{1/2}(\hat{x}, r, \hat{w})$ is an α-approximation to h, then either $\hat{h}$ or $-\hat{h}$ will have the same sign as h for all points in $\mathcal{X}$ except those in a margin of width $O(\sqrt{\alpha})$ around h. Under the uniform distribution on K, the probability mass of the margins surrounding the true hyperplanes isn't large, which results in low error for the classification rule.

Theorem 2.5. *Suppose Assumptions 2.3 and 2.4 hold. For any desired error $\epsilon > 0$, with probability at least $1 - \delta$, running Algorithm 4 with parameters $r \leq R/2$ and $\tau = \alpha p^{1/2}(r)/2$ for a known constant α on on a sample of size $n = \tilde{O}(dm^2 c_{\mathrm{ub}}^2 R^d/(c_{\mathrm{lb}}^2 \epsilon^4))$ will have error at most ϵ.*

Proof. By Lemma 2.4, for the parameter settings τ and r and the given sample size, with probability at least $1-\delta$ every half-ball included in the set H will be an α-approximation to some true hyperplane h_i, and every true hyperplane h_i is α-approximated by at least one half-ball in H. Assume that this high probability event occurs.

Let $H = \{(\hat{x}_1, \hat{w}_1), \ldots, (\hat{x}_M, \hat{w}_M)\}$ be the set of of half-balls produced by the algorithm and define $\hat{h}_i(x) = \hat{w}_i^\top (x - \hat{x}_i)$ for $i = 1, \ldots, M$ to be the corresponding linear functions. Algorithm 4 uses these hyperplanes to partition the space $\mathcal{X}$ into a collection of polygonal regions and assigns a unique class label to each cell in the partition. Notice that negating any of the $\hat{h}_i$ functions does not change the partitioning of the space. Therefore, negating any subset of the $\hat{h}_i$ will not change the permutation-invariant error of the resulting classifier.

Let $I_1, \ldots, I_m$ be a partition of the set of indices $\{1, \ldots, M\}$ such that for all $j \in I_i$, we have that $B^{1/2}(\hat{x}_j, \hat{w}_j, r)$ is an α-approximation to h_i. By Lemma 2.6, we know that for at least one $g \in \{\hat{h}_j, -\hat{h}_j\}$, we have that

$$|h_i(x) - g(x)| \leq \left(2D + \sqrt{\frac{2^d \pi}{d}} \frac{r}{2}\right) \sqrt{\alpha}$$

Since negating any of the functions $\hat{h}_j$ does not change the error of the resulting classifier, assume WLOG that the above holds for $g = \hat{h}_j$.

This implies that whenever $|h_i(x)| > c\sqrt{\alpha}$, where $c = 2D + \sqrt{\frac{2^d \pi}{d}} \frac{R}{4}$, then for every $j \in I_i$, the sign of $\hat{h}_j(x)$ is the same as the sign of $h_i(x)$. It follows that for points that are not within a margin of $c\sqrt{\alpha}$ of any of the true hyperplanes, every $\hat{h}_j$ function with $j \in I_i$ will have the same sign as h_i for all $i = 1, \ldots, m$. It follows that the classifier can only error on points that are within a $c\sqrt{\alpha}$ margin of one of the true hyperplanes.

Using Lemma 2.5 we can bound the probability that a sample X drawn uniformly from K lands in the $c\sqrt{\alpha}$-margin of hyperplane h_i as follows:

$$\Pr(X \text{ in } c\sqrt{\alpha}\text{-margin of } h_i) \leq 2\sqrt{\frac{d+1}{2\pi}} \frac{c\sqrt{\alpha}}{D} D^d v_d c_{\mathrm{ub}},$$

where D is the diameter of X. We can make this upper bound equal to ϵ/m by setting

$$\alpha = \frac{\pi}{2(d+1)} \left(\frac{\epsilon D}{mcD^d v_d c_{\mathrm{lb}}}\right)^2 = \Omega\left(\frac{\epsilon^2}{m^2 2^d R^2 D^{2d} v_d^2 c_{\mathrm{lb}}^2}\right)$$

Applying the union bound to the m hyperplanes $h_1, \ldots, h_m$ shows that the error of $\hat{f}$ is at most ϵ. □

Note that if the scale parameter R is unknown, the conclusions of Theorem 2.5 continue to hold when the parameter r and the unlabeled sample complexity n are set using a conservatively small estimate $\hat{R}$ satisfying $\hat{R} \leq R$.

Theorem A.3 in the appendix extends the above result to the agnostic setting considered in Section 2.7.

2.7 Extensions to the Agnostic Setting

The majority of our algorithms have two phases: first, we extract a partitioning of the unlabeled data into groups that are likely label-homogeneous, and second, we query the label of the largest groups. We can extend our results for these algorithms to the agnostic setting by querying multiple labels from each group and using the majority label.

Specifically, suppose that the data is generated according to a distribution P over $\mathcal{X} \times [L]$ and there exists a labeling function f^* such that $\Pr_{(x,y)\sim P}(f^*(x) \neq y) \leq \eta$ and our assumptions hold when the unlabeled data is drawn from the marginal $P_{\mathcal{X}}$ but the labels are assigned by f^*. That is, the true distribution over class labels disagrees with a function f^* satisfying our assumptions with probability at most η. In this setting, the first phase of our algorithms, which deals with only unlabeled data, behaves exactly as in the realizable setting. The only difference is that we will need to query multiple labels from each group of data to ensure that the majority label is the label predicted by f^*. Suppose that the training data is $(x_1, y_1), \ldots, (x_n, y_n)$ drawn from P (where the labels y_i are initially unobserved). For $n = \tilde{O}(1/\eta^2)$, we are guaranteed that on at most $2\eta n$ of the training points we have that $y_i \neq f^*(x_i)$. Moreover, if we only need to guess the label of large groups of samples, say those containing at least $8\eta n$ points, then we are guaranteed that within each group at least $1/4$ of the sample points will have labels that agree with f^*. Therefore, after querying $O(\log(1/\delta))$ labeled examples from each group, the majority label will agree with f^*. If we use these labels in the second phase of the algorithm, we would be guaranteed that the error of our algorithm would be at most ϵ had the labels been produced by f^*, and therefore the error under the distribution P is at most $\eta + \epsilon$. The appendix contains agnostic versions of Theorems 2.1, 2.4, and 2.5.

Similarly, modifying Algorithm 2 to require that the each cluster in the pruning have a majority label that accounts for at least $3/4$ of the cluster's data can be used to extend the corresponding results to the agnostic setting.

2.8 Conclusion and Discussion

In this work we showed how to exploit the implicit geometric assumptions made by output code techniques under the well studied cases of one-vs-all and well separated codewords, and for a novel boundary features condition that captures the intuition that every binary learning task should be significant. We provide label-efficient learning algorithms for both the consistent and agnostic learning settings with guarantees when the data density has thick level sets or upper and lower bounds. In all cases, our algorithms show that the implicit assumptions of output code learning can be used to learn from very limited labeled data.

In this work we focused on linear output codes, which have been in several practical works. For example Palatucci et al. [116] use linear output codes for neural decoding of thoughts from fMRI data, Berger [28] used them successfully for text classification, and Crammer and Singer [48] show that they perform well on MNIST and several UCI datasets. Many other works use non-linear output codes, and it is a very interesting research direction to extend our work to such cases.

The unlabeled sample complexity of our algorithms is exponential in the dimension because our algorithms require the samples to cover high-density regions. It is common for semi-supervised algorithms to require exponentially more unlabeled data than labeled, e.g. [129, 37]. Our results also show that the unlabeled sample complexity only scales exponentially with the intrinsic dimension, which may be significantly lower than the ambient dimension for real-world problems. An interest-

ing direction for future work is to determine further conditions under which the unlabeled sample complexity can be drastically reduced.

Chapter 3

Online and Private Algorithm Configuration

In this chapter we consider data driven algorithm configuration, where the goal is to use machine learning tools to find the parameters for an algorithm that give the best performance on a specific application domain. For example, given a parameterized family of clustering algorithms, we might want to learn the parameters that lead to the most semantically meaningful clusterings of text documents. To formulate this as a learning problem, we might collect example text document clustering instances, each with a hand-crafted clustering. With this training data in hand, our goal is to find the parameters that result in clusterings with maximal agreement with the hand-crafted clusterings. We focus primarily on two settings: the online setting, where a sequence of problem instances (e.g., clustering datasets) arrives one at a time, and we must choose parameters for each instance. Our goal is to design algorithms for which the cumulative performance of the parameters chosen by the algorithm is nearly as high as the best fixed parameters in hindsight. We also consider the private setting, where each problem instance encodes sensitive information about an individual, and our goal is to find high-performing parameters without divulging that sensitive information. For example, if each clustering instance we encountered consisted of the search queries made by a single user of a search engine, we may want to obtain high quality algorithm parameters while at the same time ensuring that we protect each user's data.

In both the online and private settings, tuning the parameters of combinatorial algorithms often reduces to optimization of piecewise Lipschitz functions. These are particularly challenging optimization problems, since the discontinuities of piecewise Lipschitz functions allows them to be incredibly sensitive near the optimal parameters. Section 3.1 introduces a novel dispersion condition that roughly measures how concentrated the discontinuities of a sequence of piecewise Lipschitz funtions are. We provide matching upper and lower bounds for both private and online optimization that depend on the level of dispersion in the underlying sequence. In Section 3.2, we design very efficient algorithms for online algorithm configuration by taking advantage of rich additional structure present for many combinatorial algorithm configuration problems: running the algorithm on an instance for a given parameter setting can sometimes reveal an entire range of parameter values with similar performance. Exploiting this additional information leads to algorithms with better guarantees than those explored in Section 3.1.

3.1 Private and Online Algorithm Configuration from Dispersion

3.1.1 Introduction

Data-driven algorithm design, that is, choosing the best algorithm for a specific application, is a critical problem in modern data science and algorithm design. Rather than use off-the-shelf algorithms with only worst-case guarantees, a practitioner will often optimize over a family of parametrized algorithms, tuning the algorithm's parameters based on typical problems from his domain. Ideally, the resulting algorithm will have high performance on future problems, but these procedures have historically come with no guarantees. In a seminal work, Gupta and Roughgarden [77] study algorithm selection in a distributional learning setting. Modeling an application domain as a distribution over typical problems, they show that a bound on the intrinsic complexity of the algorithm family prescribes the number of samples sufficient to ensure that any algorithm's empirical and expected performance are close.

We advance the foundations of algorithm selection in several important directions: online and private algorithm selection. In the online setting, problem instances arrive one-by-one, perhaps adversarially. The goal is to select parameters for each instance in order to minimize *regret*, which is the difference between the cumulative performance of those parameters and the optimal parameters in hindsight. We also study private algorithm selection, where the goal is to find high-performing parameters over a set of problems without revealing sensitive information contained therein. Preserving privacy is crucial when problems depend on individuals' medical or purchase data, for example.

We analyze several important, infinite families of parameterized algorithms. These include greedy techniques for canonical subset selection problems such as the knapsack and maximum weight independent set problems. We also study SDP-rounding schemes for problems that can be formulated as integer quadratic programs, such as max-cut, max-2sat, and correlation clustering. In these cases, our goal is to optimize, online or privately, the utility function that measures an algorithm's performance as a function of its parameters, such as the value of the items added to the knapsack by a parameterized knapsack algorithm. The key challenge is the volatility of this function: a small tweak to the algorithm's parameters can cause a cascade of changes in the algorithm's behavior. For example, greedy algorithms typically build a solution by iteratively adding items that maximize a scoring rule. Prior work has proposed parameterizing these scoring rules and tuning the parameter to obtain the best performance for a given application [77]. Slightly adjusting the parameter can cause the algorithm to select items in a completely different order, potentially causing a sharp change in the quality of the selected items.

Despite this challenge, we show that in many cases, these utility functions are well-behaved in several respects and thus can be optimized online and privately. Specifically, these functions are piecewise Lipschitz and moreover, they satisfy a condition we call *dispersion*. Roughly speaking, a collection of piecewise Lipschitz functions is *dispersed* if no small region of space contains discontinuities for many of the functions. We provide general techniques for online and private optimization of the sum or average of dispersed piecewise Lipschitz functions. Taking advantage of dispersion in online learning, we improve over the best-known regret bounds for a variety problems, prove regret bounds for problems not previously studied, and provide matching regret lower bounds. In the privacy setting, we show how to optimize performance while preserving privacy for several important problems, giving matching upper and lower bounds on performance loss due to privacy.

Though our main motivation is algorithm selection, we expect dispersion is even more widely applicable, opening up an exciting research direction. For this reason, we present our main results

more generally for optimizing piecewise Lipschitz functions. We also uncover dispersion in domains beyond algorithm selection, namely, auction design and pricing, so we prove online and privacy guarantees for these problems as well. Finally, we answer several open questions: Cohen-Addad and Kanade [46] asked how to optimize piecewise Lipschitz functions and Gupta and Roughgarden [77] asked which algorithm selection problems can be solved with no regret algorithms. As a bonus, we also show that dispersion implies generalization guarantees in the distributional setting. In this setting, the configuration procedure is given an iid sample of problem instances drawn from an unknown distribution d, and the goal is to find the algorithm parameters with highest expected utility. By bounding the empirical Rademacher complexity, we show that the sample and expected utility for all algorithms in our class are close, implying that the optimal algorithm on the sample is approximately optimal in expectation.

Our contributions

In order to present our contributions, we briefly outline the notation we will use. Let $\mathcal{A}$ be an infinite set of algorithms parameterized by a set $\mathcal{C} \subseteq \mathbb{R}^d$. For example, $\mathcal{A}$ might be the set of knapsack greedy algorithms that add items to the knapsack in decreasing order of $v(i)/s(i)^\rho$, where $v(i)$ and $s(i)$ are the value and size of item i and ρ is a parameter. Next, let Π be a set of problem instances for $\mathcal{A}$, such as knapsack problem instances, and let $u : \Pi \times \mathcal{C} \to [0, H]$ be a utility function where $u(x, \boldsymbol{\rho})$ measures the performance of the algorithm with parameters $\boldsymbol{\rho}$ on problem instance $x \in \Pi$. For example, $u(x, \rho)$ could be the value of the items chosen by the knapsack algorithm with parameter ρ on input x.

We now summarize our main contributions. Since our results apply beyond application-specific algorithm selection, we describe them for the more general problem of optimizing piecewise Lipschitz functions.

Dispersion Let $u_1, \dots, u_T$ be a set of functions mapping a set $\mathcal{C} \subseteq \mathbb{R}^d$ to $[0, H]$. For example, in the application-specific algorithm selection setting, given a collection of problem instances $x_1, \dots, x_T \in \Pi$ and a utility function $u : \Pi \times \mathcal{C} \to [0, H]$, each function $u_i(\cdot)$ might equal the function $u(x_i, \cdot)$, measuring an algorithm's performance on a fixed problem instance as a function of its parameters. Dispersion is a constraint on the functions $u_1, \dots, u_T$. We assume that for each function u_i, we can partition $\mathcal{C}$ into sets $\mathcal{C}_1, \dots, \mathcal{C}_K$ such that u_i is L-Lipschitz on each piece, but u_i may have discontinuities at the boundaries between pieces. In our applications, each set $\mathcal{C}_i$ is connected, but our general results hold for arbitrary sets. Informally, the functions $u_1, \dots, u_T$ are (w, k)-dispersed if every Euclidean ball of radius w contains discontinuities for at most k of those functions (see Section 3.1.2 for a formal definition). This guarantees that although each function u_i may have discontinuities, they do not concentrate in a small region of space. Dispersion is sufficient to prove strong learning generalization guarantees, online learning regret bounds, and private optimization bounds when optimizing the empirical utility $\frac{1}{T}\sum_{i=1}^{T} u_i$. In our applications, $w = T^{\alpha-1}$ and $k = \tilde{O}(T^\alpha)$ with high probability for any $1/2 \leq \alpha \leq 1$, ignoring problem-specific multiplicands.

Online learning We prove that dispersion implies strong regret bounds in online learning, a fundamental area of machine learning [38]. In this setting, a sequence of functions $u_1, \dots, u_T$ arrive one-by-one. At time t, the learning algorithm chooses a parameter vector $\boldsymbol{\rho}_t$ and then either observes the function u_t in the full information setting or the scalar $u_t(\boldsymbol{\rho}_t)$ in the bandit setting. The goal is

to minimize expected regret: $\mathbb{E}[\max_{\boldsymbol{\rho}\in\mathcal{C}} \sum u_t(\boldsymbol{\rho}) - u_t(\boldsymbol{\rho}_t)]$. Under full information, we show that the exponentially-weighted forecaster [38] has regret bounded by $\tilde{O}(H(\sqrt{Td}+k)+TLw)$. When $w = 1/\sqrt{T}$ and $k = \tilde{O}(\sqrt{T})$, this results in $\tilde{O}(\sqrt{T}(H\sqrt{d}+L))$ regret. We also prove a matching lower bound. This algorithm also preserves (ϵ,δ)-differential privacy with regret bounded by $\tilde{O}(H(\sqrt{T}d/\epsilon + k + \delta) + TLw)$. Finally, under bandit feedback, we show that a discretization-based algorithm achieves regret at most $\tilde{O}(H(\sqrt{dT(3R/w)^d}+k)+TLw)$. When $w = T^{-1/(d+2)}$ and $k = \tilde{O}(T^{(d+1)/(d+2)})$, this gives a bound of $\tilde{O}(T^{(d+1)/(d+2)}(H\sqrt{d(3R)^d}+L))$, matching the dependence on T of a lower bound by Kleinberg et al. [91] for (globally) Lipschitz functions.

Online algorithm selection is generally not possible: Gupta and Roughgarden [77] give an algorithm selection problem for which no online algorithm can achieve sub-linear regret. Therefore, additional structure is necessary to prove guarantees, which we characterize using dispersion.

Private batch optimization We demonstrate that it is possible to optimize over a set of dispersed functions while preserving *differential privacy* [64]. In this setting, the goal is to find the parameter $\boldsymbol{\rho}$ that maximizes average utility on a set $\mathcal{S} = \{u_1, \ldots, u_T\}$ of functions $u_i : \mathcal{C} \to \mathbb{R}$ without divulging much information about any single function u_i. Providing privacy at the granularity of functions is suitable when each function encodes sensitive information about one or a small group of individuals and each individual's information is used to define only a small number of functions. For example, in the case of auction design and pricing problems, each function u_i is defined by a set of buyers' bids or valuations for a set of items. If a single buyer's information is only encoded by a single function, then we preserve her privacy by not revealing sensitive information about any one function u_i. This will be the case, for example, if the buyers do not repeatedly return to buy the same items day after day. This is a common assumption in online auction design and pricing [32, 33, 35, 40, 90, 125, 60] because it means the buyers will not be strategic, aiming to trick the algorithm into setting lower prices in the future.

Differential privacy requires that an algorithm is randomized and its output distribution is insensitive to changing a single point in the input data. Formally, two multi-sets $\mathcal{S}$ and $\mathcal{S}'$ of T functions are *neighboring*, denoted $\mathcal{S} \sim \mathcal{S}'$, if $|\mathcal{S}\Delta\mathcal{S}'| \leq 1$. A randomized algorithm $\mathcal{A}$ is (ϵ,δ)-*differentially private* if, for any neighboring multi-sets $\mathcal{S} \sim \mathcal{S}'$ and set $\mathcal{O}$ of outcomes, $\Pr(\mathcal{A}(\mathcal{S}) \in \mathcal{O}) \leq e^\epsilon \Pr(\mathcal{A}(\mathcal{S}') \in \mathcal{O}) + \delta$. In our setting, the algorithm's input is a set $\mathcal{S}$ of T functions, and the output is a point $\boldsymbol{\rho} \in \mathcal{C}$ that approximately maximizes the average of those functions. We show that the exponential mechanism [104] outputs $\hat{\boldsymbol{\rho}} \in \mathcal{C}$ such that with high probability $\frac{1}{T}\sum_{i=1}^T u_i(\hat{\boldsymbol{\rho}}) \geq \max_{\boldsymbol{\rho}\in\mathcal{C}} \frac{1}{T}\sum_{i=1}^T u_i(\boldsymbol{\rho}) - \tilde{O}(\frac{H}{T}(\frac{d}{\epsilon}+k)+Lw)$ while preserving $(\epsilon,0)$-differential privacy. We also give a matching lower bound. Our private algorithms always preserve privacy, even when dispersion does not hold.

Computational efficiency In our settings, the functions have additional structure that enables us to design efficient implementations of our algorithms: for one-dimensional problems, there is a closed-form expression for the integral of the piecewise Lipschitz functions on each piece and for multi-dimensional problems, the functions are piecewise concave. We leverage tools from high-dimensional geometry [26, 101] to efficiently implement the integration and sampling steps required by our algorithms. Our algorithms have running time linear in the number of pieces of the utility function and polynomial in all other parameters.

Dispersion in algorithm selection problems

Algorithm selection. We study algorithm selection for integer quadratic programs (IQPs) of the form $\max_{\boldsymbol{z}\in\{\pm 1\}^n} \boldsymbol{z}^\top A\boldsymbol{z}$, where $A \in \mathbb{R}^{n\times n}$ for some n. Many classic NP-hard problems can be formulated as IQPs, including max-cut [72], max-2SAT [72], and correlation clustering [43]. Many IQP approximation algorithms are semidefinite programming (SDP) rounding schemes; they solve the SDP relaxation of the IQP and round the resulting vectors to binary values. We study two families of SDP rounding techniques: s-linear rounding [67] and outward rotation [152], which include the Goemans-Williamson algorithm [72] as a special case. Due to these algorithms' inherent randomization, finding an optimal rounding function over T problem instances with n variables amounts to optimizing the sum of $(1/T^{1-\alpha}, \tilde{O}(nT^\alpha))$-dispersed functions for $1/2 \leq \alpha < 1$. This holds even for adversarial (non-stochastic) instances, implying strong online learning guarantees.

We also study greedy algorithm selection for two canonical subset selection problems: the knapsack and maximum weight independent set (MWIS) problems. Greedy algorithms are typically defined by a scoring rule determining the order the algorithm adds elements to the solution set. For example, Gupta and Roughgarden [77] introduce a parameterized knapsack algorithm that adds items in decreasing order of $v(i)/s(i)^\rho$, where $v(i)$ and $s(i)$ are the value and size of item i. Under mild conditions — roughly, that the items' values are drawn from distributions with bounded density functions and that each item's size is independent from its value — we show that the utility functions induced by T knapsack instances with n items are $(1/T^{1-\alpha}, \tilde{O}(nT^\alpha))$-dispersed for any $1/2 \leq \alpha < 1$.

Pricing problems and auction design Market designers use machine learning to design auctions and set prices [149, 81]. In the online setting, at each time step there is a set of goods for sale and a set of consumers who place bids for those goods. The goal is to set auction parameters, such as reserve prices, that are nearly as good as the best fixed parameters in hindsight. Here, "best" may be defined in terms of revenue or social welfare, for example. In the offline setting, the algorithm receives a set of bidder valuations sampled from an unknown distribution and aims to find parameters that are nearly optimal in expectation (e.g., [66, 47, 83, 105, 109, 124, 56, 74, 35, 110, 17, 24]). In the paper that this chapter is based on [22], we analyze multi-item, multi-bidder second price auctions with reserves, as well as pricing problems, where the algorithm sets prices and buyers decide what to buy based on their utility functions. These classic mechanisms have been studied for decades in both economics and computer science. We note that data-driven mechanism design problems are effectively algorithm design problems with incentive constraints: the input to a mechanism is the buyers' bids or valuations, and the output is an allocation of the goods and a description of the payments required of the buyers. For ease of exposition, we discuss algorithm and mechanism design separately.

Related work

Gupta and Roughgarden [77] and Balcan et al. [18] study algorithm selection in the distributional learning setting, where there is a distribution d over problem instances. A learning algorithm receives a set $\mathcal{S}$ of samples from d. Those two works provide *uniform convergence guarantees*, which bound the difference between the average performance over $\mathcal{S}$ of any algorithm in a class $\mathcal{A}$ and its expected performance on d. It is known that regret bounds imply generalization guarantees for various online-to-batch conversion algorithms [39], but in this work, we also show that dispersion can be used to explicitly provide uniform convergence guarantees via Rademacher complexity. Beyond this connection, our work is a significant departure from these works since we give guarantees for

private algorithm selection and we give no regret algorithms, whereas Gupta and Roughgarden [77] only study online MWIS algorithm selection, proving their algorithm has small constant per-round regret.

Private empirical risk minimization (ERM) The goal of private ERM is to find the best machine learning model parameters based on private data. Techniques include objective and output perturbation [45], stochastic gradient descent, and the exponential mechanism [26]. These works focus on minimizing data-dependent convex functions, so parameters near the optimum also have high utility, which is not the case in our settings.

Private algorithm configuration Kusner et al. [95] develop private Bayesian optimization techniques for tuning algorithm parameters. Their methods implicitly assume that the utility function is differentiable. Meanwhile, the class of functions we consider have discontinuities between pieces, and it is not enough to privately optimize on each piece, since the boundaries themselves are data-dependent.

Online optimization Prior work on online algorithm selection focuses on significantly more restricted settings. Cohen-Addad and Kanade [46] study single-dimensional piecewise constant functions under a "smoothed adversary," where the adversary chooses a distribution per boundary from which that boundary is drawn. Thus, the boundaries are independent. Moreover, each distribution must have bounded density. Gupta and Roughgarden [77] study online MWIS greedy algorithm selection under a smoothed adversary, where the adversary chooses a distribution per vertex from which its weight is drawn. Thus, the vertex weights are independent and again, each distribution must have bounded density. In contrast, we allow for more correlations among the elements of each problem instance. Our analysis also applies to the substantially more general setting of optimizing piecewise Lipschitz functions. We show several new applications of our techniques in algorithm selection for SDP rounding schemes, price setting, and auction design, none of which were covered by prior work. Furthermore, we provide differential privacy results and generalization guarantees.

Neither Cohen-Addad and Kanade [46] nor Gupta and Roughgarden [77] develop a general theory of dispersion, but we can map their analysis into our setting. In essence, Cohen-Addad and Kanade [46], who provide the tighter analysis, show that if the functions the algorithm sees map from $[0, 1]$ to $[0, 1]$ and are $(w, 1)$-dispersed, then the regret of their algorithm is bounded by $O(\sqrt{T \ln(1/w)})$. Under a smoothed adversary, the functions are $(w, 1)$-dispersed for an appropriate choice of w. In this work, we show that using the more general notion of (w, k)-dispersion is essential to proving tight learning bounds for more powerful adversaries. We provide a sequence of piecewise constant functions $u_1, \ldots, u_T$ mapping $[0, 1]$ to $[0, 1]$ that are $(1/8, \sqrt{T} + 1)$-dispersed, which means that our regret bound is $O(\sqrt{T \log(1/w)} + k) = O(\sqrt{T})$. However, these functions are not $(w, 1)$-disperse for any $w \geq 2^{-T}$, so the regret bound by Cohen-Addad and Kanade [46] is trivial, since $\sqrt{T \log(1/w)}$ with $w = 2^{-T}$ equals T. Similarly, Weed et al. [143] and Feng et al. [68] use a notion similar to $(w, 1)$-dispersion to prove learning guarantees for the specific problem of learning to bid, as do Rakhlin et al. [119] for learning threshold functions under a smoothed adversary.

Our online bandit results are related to those of Kleinberg [89] for the "continuum-armed bandit" problem. They consider bandit problems where the set of arms is the interval $[0, 1]$ and each payout function is uniformly locally Lipschitz. We relax this requirement, allowing each payout function to be Lipschitz with a number of discontinuities. In exchange, we require that the overall sequence of

payout functions is fairly nice, in the sense that their discontinuities do not tightly concentrate. The follow-up work on Multi-armed Bandits in Metric Spaces [91] considers the stochastic bandit problem where the space of arms is an arbitrary metric space and the mean payoff function is Lipschitz. They introduce the zooming algorithm, which has better regret bounds than the discretization approach of Kleinberg [89] when either the max-min covering dimension or the (payout-dependent) zooming dimension are smaller than the covering dimension. In contrast, we consider optimization over $\mathbb{R}^d$ under the ℓ_2 metric, where this algorithm does not give improved regret in the worst case.

Auction design and pricing Several works [32, 33, 35, 40, 90, 125] present stylized online learning algorithms for revenue maximization under specific auction classes. In contrast, our online algorithms are highly general and apply to many optimization problems beyond auction design. Dudík et al. [60] also provide online algorithms for auction design. They discretize each set of mechanisms they consider and prove their algorithms have low regret over the discretized set. When the bidders have simple valuations (unit-demand and single-parameter) minimizing regret over the discretized set amounts to minimizing regret over the entire mechanism class. In contrast, we study bidders with fully general valuations, as well as additive and unit-demand valuations.

A long line of work has studied *generalization guarantees* for auction design and pricing problems (e.g., [66, 47, 83, 105, 109, 124, 56, 74, 35, 110, 73, 17, 24]). These works study the distributional setting where there is an unknown distribution over buyers' values and the goal is to use samples from this distribution to design a mechanism with high expected revenue. Generalization guarantees bound the difference between a mechanism's empirical revenue over the set of samples and expected revenue over the distribution. For example, several of these works [105, 109, 110, 17, 24, 106, 135] use learning theoretic tools such as pseudo-dimension and Rademacher complexity to derive these generalization guarantees. In contrast, we study online and private mechanism design, which requires a distinct set of analysis tools beyond those used in the distributional setting.

Bubeck et al. [35] study auction design in both the online and distributional settings when there is a single item for sale. They take advantage of structure exhibited in this well-studied single-item setting, such as the precise form of the optimal single-item auction [112]. Meanwhile, our algorithms and guarantees apply to the more general problem of optimizing piecewise Lipschitz functions.

3.1.2 Dispersion Condition

In this section we formally define (w, k)-dispersion using the same notation as in Section 3.1.1. Recall that Π is a set of problem instances, $\mathcal{C} \subset \mathbb{R}^d$ is a parameter space, and u is an abstract utility function. Throughout this chapter, we use the ℓ_2 distance and let $B(\boldsymbol{\rho}, r) = \{\boldsymbol{\rho}' \in \mathbb{R}^d : \|\boldsymbol{\rho} - \boldsymbol{\rho}'\|_2 \leq r\}$ denote a ball of radius r centered at $\boldsymbol{\rho}$.

Definition 3.1. Let $u_1, \ldots, u_T : \mathcal{C} \to [0, H]$ be a collection of functions where u_i is piecewise Lipschitz over a partition $\mathcal{P}_i$ of $\mathcal{C}$. We say that $\mathcal{P}_i$ splits a set A if A intersects with at least two sets in $\mathcal{P}_i$ (see Figure 3.1). The collection of functions is (w, k)-*dispersed* if every ball of radius w is split by at most k of the partitions $\mathcal{P}_1, \ldots, \mathcal{P}_T$. More generally, the functions are (w, k)-*dispersed at a maximizer* if there exists a point $\boldsymbol{\rho}^* \in \operatorname{argmax}_{\boldsymbol{\rho} \in \mathcal{C}} \sum_{i=1}^{T} u_i(\boldsymbol{\rho})$ such that the ball $B(\boldsymbol{\rho}^*, w)$ is split by at most k of the partitions $\mathcal{P}_1, \ldots, \mathcal{P}_T$.

Given $\mathcal{S} = \{x_1, \ldots, x_T\} \subseteq \Pi$ and a utility function $u : \Pi \times \mathcal{C} \to [0, H]$, we equivalently say that u is (w, k)-*dispersed for $\mathcal{S}$ (at a maximizer)* if $\{u(x_1, \cdot), \ldots, u(x_T, \cdot)\}$ is (w, k)-dispersed (at a maximizer).

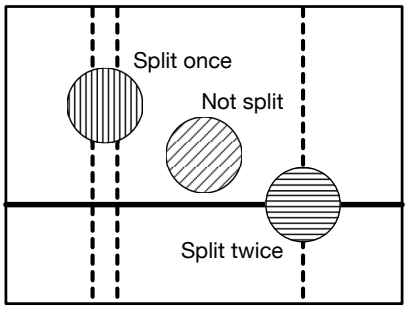


Figure 3.1: The dashed and solid lines correspond to two partitionings of the rectangle. Each of the displayed balls is either not split, split by one partition, or split by both.

We often show that the discontinuities of a piecewise Lipschitz function $u : \mathbb{R} \to \mathbb{R}$ are random variables with *κ-bounded distributions*. A density function $f : \mathbb{R} \to \mathbb{R}$ corresponds to a κ-bounded distribution if $\max\{f(x)\} \leq \kappa$.[1] To prove dispersion we will use the following probabilistic lemma, showing that samples from κ-bounded distributions do not tightly concentrate.

Lemma 3.1. *Let $\mathcal{B} = \{\beta_1, \ldots, \beta_r\} \subset \mathbb{R}$ be a collection of samples where each β_i is drawn from a κ-bounded distribution with density function p_i. For any $\zeta \geq 0$, the following statements hold with probability at least $1 - \zeta$:*

1. *If the β_i are independent, then every interval of width w contains at most $k = O(rw\kappa + \sqrt{r\log(1/\zeta)})$ samples. In particular, for any $\alpha \geq 1/2$ we can take $w = 1/(\kappa r^{1-\alpha})$ and $k = O(r^{\alpha}\sqrt{\log(1/\zeta)})$.*

2. *If the samples can be partitioned into P buckets $\mathcal{B}_1, \ldots, \mathcal{B}_P$ such that each $\mathcal{B}_i$ contains independent samples and $|\mathcal{B}_i| \leq M$, then every interval of width w contains at most $k = O(PMw\kappa + \sqrt{M\log(P/\zeta)}$. In particular, for any $\alpha \geq 1/2$ we can take $w = 1/(\kappa M^{1-\alpha})$ and $k = O(PM^{\alpha}\sqrt{\log(P/\zeta)})$.*

Proof sketch. If the β_i are independent, the expected number of samples in any interval of width w is at most $r\kappa w$. Since the VC-dimension of intervals is 2, it follows that with probability at least $1 - \zeta$, no interval contains more than $r\kappa w + O(\sqrt{r\log(1/\zeta)})$ samples.

The second claim follows by applying this counting argument to each of the buckets $\mathcal{B}_i$ with failure probability $\zeta' = \zeta/P$ and taking the union bound over all buckets. With probability at least $1 - \zeta$, every interval of width w contains at most $M\kappa w + O(\sqrt{M\log(P/\zeta)})$ samples from each bucket, and at most $k = PM\kappa w + O(P\sqrt{M\log(P/\zeta)})$ samples in total from all P buckets. □

Lemma 3.1 allows us to provide dispersion guarantees for "smoothed adversaries" in online learning. Under this type of adversary, the discontinuity locations for each function u_i are random variables, due to the smoothness of the adversary. In our algorithm selection applications, the randomness of discontinuities may be a byproduct of the randomness in the algorithm's inputs. For example, in the case of knapsack algorithm configuration, the item values and sizes may be drawn from distributions chosen by the adversary. This induces randomness in the discontinuity locations of the algorithm's cost function. We can thus apply Lemma 3.1 to guarantee dispersion.

[1]For example, for all $\mu \in \mathbb{R}$, $\mathcal{N}(\mu, \sigma)$ is $\frac{1}{2\pi\sigma}$-bounded.

We also use Lemma 3.1 to guarantee dispersion even when the adversary is not smoothed. Surprisingly, we show that dispersion holds for IQP algorithm configuration without *any* assumptions on the input instances. In this case, we exploit the fact that the algorithms are themselves randomized. This randomness implies that the discontinuities of the algorithm's cost function are random variables, and thus Lemma 3.1 implies dispersion.

3.1.3 Online Optimization

In this setting, a sequence of functions $u_1, \ldots, u_T$ arrive one-by-one. At time t, the learning algorithm chooses a vector $\boldsymbol{\rho}_t$ and then either observes the function $u_t(\cdot)$ in the full information setting or the value $u_t(\boldsymbol{\rho}_t)$ in the bandit setting. The goal is to minimize expected regret: $\mathbb{E}[\max_{\boldsymbol{\rho}\in\mathcal{C}} \sum_{t=1}^{T}(u_t(\boldsymbol{\rho}) - u_t(\boldsymbol{\rho}_t))]$. In our applications, the functions $u_1, \ldots, u_T$ are random, either due to internal randomization in the algorithms we are configuring or from assumptions on the adversary[2]. We show that the functions are (w, k)-dispersed with probability $1 - \zeta$ over the choice of $u_1, \ldots, u_T$. The following regret bounds hold in expectation with an additional term of $HT\zeta$ bounding the effect of the rare event where the functions are not dispersed.

Full information. The *exponentially-weighted forecaster* algorithm samples the vectors $\boldsymbol{\rho}_t$ from the distribution $p_t(\boldsymbol{\rho}) \propto \exp(\lambda \sum_{s=1}^{t-1} u_s(\boldsymbol{\rho}))$. We prove the following regret bound. The full proof is in Appendix B.1.4.

Theorem 3.1. *Let $u_1, \ldots, u_T : \mathcal{C} \to [0, H]$ be any sequence of piecewise L-Lipschitz functions that are (w, k)-dispersed at the maximizer $\boldsymbol{\rho}^*$. Suppose $\mathcal{C} \subset \mathbb{R}^d$ is contained in a ball of radius R and $B(\boldsymbol{\rho}^*, w) \subset \mathcal{C}$. The exponentially weighted forecaster with $\lambda = \sqrt{d \ln(R/w)/T}/H$ has expected regret bounded by*

$$O\left(H\left(\sqrt{Td\log\frac{R}{w}} + k\right) + TLw\right).$$

For all rounds $t \in [T]$, suppose $\sum_{s=1}^{t} u_s$ is piecewise Lipschitz over at most K pieces. When $d = 1$ and $\exp(\sum_{s=1}^{t} u_s)$ can be integrated in constant time on each of its pieces, the running time is $O(TK)$. When $d > 1$ and $\sum_{s=1}^{t} u_s$ is piecewise concave over convex pieces, we provide an efficient approximate implementation. For approximation parameters $\eta = \zeta = 1/\sqrt{T}$ and $\lambda = \sqrt{d \ln(R/w)/T}/H$, this algorithm has the same regret bound as the exact algorithm and runs in time $\tilde{O}(T(K \cdot \mathrm{poly}(d, 1/\eta) + \mathrm{poly}(d, L, 1/\eta))$.

Proof sketch. Let U_t be the function $\sum_{i=1}^{t-1} u_i(\cdot)$ and let $W_t = \int_{\mathcal{C}} \exp(\lambda U_t(\boldsymbol{\rho}))\, d\boldsymbol{\rho}$. We use (w, k)-dispersion to lower bound W_{T+1}/W_1 in terms of the optimal parameter's total payout. Combining this with a standard upper bound on W_{T+1}/W_1 in terms of the learner's expected payout gives the regret bound. To lower bound W_{T+1}/W_1, let $\boldsymbol{\rho}^*$ be the optimal parameter and let $\mathrm{OPT} = U_{T+1}(\boldsymbol{\rho}^*)$. Also, let $\mathcal{B}^*$ be the ball of radius w around $\boldsymbol{\rho}^*$. From (w, k)-dispersion, we know that for all $\boldsymbol{\rho} \in \mathcal{B}^*$,

[2] As we describe in Section 3.1.1, prior research [77, 46] also makes assumptions on the adversary. For example, Cohen-Addad and Kanade [46] focus on adversaries that choose distributions with bounded densities from which the discontinuities of u_t are drawn. In Lemma B.12 of Appendix B.1.4, we show that their smoothness assumption implies dispersion with high probability.

$U_{T+1}(\boldsymbol{\rho}) \geq \mathrm{OPT} - Hk - LTw$. Therefore,

$$\begin{aligned} W_{T+1} &= \int_{\mathcal{C}} \exp(\lambda U_{T+1}(\boldsymbol{\rho}))\, d\boldsymbol{\rho} \geq \int_{\mathcal{B}^*} \exp(\lambda U_{T+1}(\boldsymbol{\rho}))\, d\boldsymbol{\rho} \\ &\geq \int_{\mathcal{B}^*} \exp(\lambda(\mathrm{OPT} - Hk - LTw)) d\boldsymbol{\rho} \\ &\geq \mathrm{Vol}(B(\boldsymbol{\rho}^*, w)) \exp(\lambda(\mathrm{OPT} - Hk - LTw)). \end{aligned}$$

Moreover, $W_1 = \int_{\mathcal{C}} \exp(\lambda U_1(\boldsymbol{\rho}))\, d\boldsymbol{\rho} \leq \mathrm{Vol}(B(\mathbf{0}, R))$. Therefore,

$$\frac{W_{T+1}}{W_1} \geq \frac{\mathrm{Vol}(B(\boldsymbol{\rho}^*, w))}{\mathrm{Vol}(B(\mathbf{0}, R))} \exp(\lambda(\mathrm{OPT} - Hk - LTw)).$$

The volume ratio is equal to $(w/R)^d$, since the volume of a ball of radius r in $\mathbb{R}^d$ is proportional to r^d. Therefore, $W_{T+1}/W_1 \geq (w/R)^d \exp(\lambda(\mathrm{OPT} - Hk - LTw))$. Combining the upper and lower bounds on $\frac{W_{T+1}}{W_1}$ gives the result.

Our efficient algorithm (Algorithm 17 of Appendix B.1.4) approximately samples from p_t. Let $\mathcal{C}_1, \ldots, \mathcal{C}_K$ be the partition of $\mathcal{C}$ over which $\sum u_t(\cdot)$ is piecewise concave. Our algorithm picks $\mathcal{C}_I$ with probability approximately proportional to $\int_{\mathcal{C}_I} p_t$ [101] and outputs a sample from the conditional distribution of p_t on $\mathcal{C}_I$ [26]. Crucially, we prove that the algorithm's output distribution is close to p_t, so every event concerning the outcome of the approximate algorithm occurs with about the same probability as it does under p_t. □

The requirement that $B(\boldsymbol{\rho}^*, w) \subset \mathcal{C}$ is for convenience. In Lemma B.11 of Appendix B.1.4 we show how to transform the problem to satisfy this. Setting $\lambda = \sqrt{d/T}/H$, which does not require knowledge of w, has regret $O(H(\sqrt{Td}\log(R/w) + k) + TLw)$. Under alternative settings of λ, we show that our algorithms are (ϵ, δ)-differentially private with regret bounds of $\tilde{O}(H\sqrt{T}/\epsilon + Hk + LTw)$ in the single-dimensional setting and $\tilde{O}(H\sqrt{T}d/\epsilon + H(k+\delta) + LTw)$ in the d-dimensional setting (see Theorems B.3 and B.4 in Appendix B.1.4).

Next, we prove a matching lower bound. We warm up with a proof for the single-dimensional case and generalize it to the multi-dimensional case in Appendix B.1.4.

Theorem 3.2. *Suppose $T \geq d$. For any algorithm, there are piecewise constant functions $u_1, \ldots, u_T$ mapping $[0,1]^d$ to $[0,1]$ such that if $D = \{(w,k) : \{u_1, \ldots, u_T\}$ is (w,k)-dispersed at the maximizer$\}$, then*

$$\max_{\boldsymbol{\rho} \in [0,1]^d} \mathbb{E}\left[\sum_{t=1}^{T} u_t(\boldsymbol{\rho}) - u_t(\boldsymbol{\rho}_t)\right] = \Omega\left(\inf_{(w,k)\in D}\left\{\sqrt{Td\log\frac{1}{w}} + k\right\}\right),$$

where the expectation is over the random choices $\boldsymbol{\rho}_1, \ldots, \boldsymbol{\rho}_T$ of the adversary.

Proof sketch. For each dimension, the adversary plays a sequence of axis-aligned halfspaces with thresholds that divide the set of optimal parameters in two. The adversary plays each halfspace $\Theta(\frac{T}{d})$ times, randomly switching which side of the halfspace has a positive label, thus forcing regret of at least $\frac{\sqrt{Td}}{64}$. We prove that the resulting set of optimal parameters is contained in a hypercube of side length $\frac{1}{2}$. The adversary then plays $\sqrt{T} + d$ copies of the indicator function of a ball of radius 2^{-T} at the center of this cube. This ensures the functions are not $(w, 0)$-dispersed at the maximizer for any $w \geq 2^{-T}$, and thus prior regret analyses [46] give a trivial bound of T. In order to prove the

theorem, we need to show that $\frac{\sqrt{Td}}{64} = \Omega\left(\inf_{(w,k)\in D}\left\{\sqrt{Td\log\frac{1}{w}} + k\right\}\right)$. Therefore, we need to show that the set of functions played by the adversary is (w,k)-dispersed at the maximizer $\boldsymbol{\rho}^*$ for $w = \Theta(1)$ and $k = O\left(\sqrt{Td}\right)$. The reason this is true is that the only functions with discontinuities in the ball $\left\{\boldsymbol{\rho} : ||\boldsymbol{\rho}^* - \boldsymbol{\rho}|| \leq \frac{1}{8}\right\}$ are the final $\sqrt{T} + d$ functions played by the adversary. Thus, the theorem statement holds. □

Bandit feedback. We now study online optimization under bandit feedback.

Theorem 3.3. *Let $u_1, \ldots, u_T : \mathcal{C} \to [0, H]$ be any sequence of piecewise L-Lipschitz functions that are (w,k)-dispersed at the maximizer $\boldsymbol{\rho}^*$. Moreover, suppose that $\mathcal{C} \subset \mathbb{R}^d$ is contained in a ball of radius R and that $B(\boldsymbol{\rho}^*, w) \subset \mathcal{C}$. There is a bandit-feedback online optimization algorithm with regret*

$$O\left(H\sqrt{Td\left(\frac{3R}{w}\right)^d \log\frac{R}{w}} + TLw + Hk\right).$$

The per-round running time is $O((3R/w)^d)$.

Proof. Let $\boldsymbol{\rho}_1, \ldots, \boldsymbol{\rho}_M$ be a w-net for $\mathcal{C}$. The main insight is that (w,k)-dispersion implies that the difference in utility between the best point in hindsight from the net and the best point in hindsight from $\mathcal{C}$ is at most $Hk + TLw$. Therefore, we only need to compete with the best point in the net. We use the Exp3 algorithm [4] to choose parameters $\hat{\boldsymbol{\rho}}_1, \ldots, \hat{\boldsymbol{\rho}}_T$ by playing the bandit with M arms, where on round t arm i has payout $u_t(\boldsymbol{\rho}_i)$. The expected regret of Exp3 is $\tilde{O}(H\sqrt{TM\log M})$ relative to our net. In Lemma B.13 of Appendix B.1.4, we show $M \leq (3R/w)^d$, so the overall regret is $\tilde{O}(H\sqrt{Td(3R/w)^d\log(R/w)} + TLw + Hk)$ with respect to $\mathcal{C}$. □

If $w = T^{\frac{d+1}{d+2}-1} = \frac{1}{T^{1/(d+2)}}$ and $k = \tilde{O}\left(T^{\frac{d+1}{d+2}}\right)$, Theorem 3.3 gives the optimal exponent on T. Specifically, the regret is $\tilde{O}\left(T^{(d+1)/(d+2)}\left(H\sqrt{d(3R)^d} + L\right)\right)$, and no algorithm can have regret $O\left(T^\gamma\right)$ for $\gamma < (d+1)/(d+2)$ for the special case of (globally) Lispchitz functions [91].

3.1.4 Differentially Private Optimization

We show that the exponential mechanism, which is $(\epsilon, 0)$-differentially private, has high utility when optimizing the mean of dispersed functions. In this setting, the algorithm is given a collection of functions $u_1, \ldots, u_T : \mathcal{C} \to [0, H]$, each of which depends on some sensitive information. In cases where each function u_i encodes sensitive information about one or a small group of individuals and each individual is present in a small number of functions, we can give meaningful privacy guarantees by providing differential privacy for each function in the collection. We say that two sets of T functions are neighboring if they differ on at most one function. Recall that the exponential mechanism outputs a sample from the distribution with density proportional to $f^\epsilon_{\exp}(\boldsymbol{\rho}) = \exp\left(\frac{\epsilon}{2\Delta T}\sum_{i=1}^T u_i(\boldsymbol{\rho})\right)$, where Δ is the sensitivity of the average utility. Since the functions u_i are bounded, the sensitivity of $\frac{1}{T}\sum_{i=1}^T u_i$ satisfies $\Delta \leq H/T$. The following theorem states our utility guarantee. The full proof is in Appendix B.1.5.

Theorem 3.4. *Let $u_1, \dots, u_T : \mathcal{C} \to [0, H]$ be piecewise L-Lipschitz and (w, k)-dispersed at the maximizer $\boldsymbol{\rho}^*$, and suppose that $\mathcal{C} \subset \mathbb{R}^d$ is convex, contained in a ball of radius R, and $B(\boldsymbol{\rho}^*, w) \subset \mathcal{C}$. For any $\epsilon > 0$, with probability at least $1 - \zeta$, the output $\hat{\boldsymbol{\rho}}$ of the exponential mechanism satisfies*

$$\frac{1}{T}\sum_{i=1}^{T} u_i\left(\hat{\boldsymbol{\rho}}\right) \geq \frac{1}{T}\sum_{i=1}^{T} u_i\left(\boldsymbol{\rho}^*\right) - O\left(\frac{H}{T\epsilon}\left(d\log\frac{R}{w} + \log\frac{1}{\zeta}\right) + Lw + \frac{Hk}{T}\right).$$

When $d = 1$, this algorithm is efficient, provided $f_{\exp}^{\epsilon}$ can be efficiently integrated on each piece of $\sum_i u_i$. For $d > 1$ we also provide an efficient approximate sampling algorithm when $\sum_i u_i$ is piecewise concave defined on K convex pieces. This algorithm preserves (ϵ, δ)-differential privacy for $\epsilon > 0$, $\delta > 0$ with the same utility guarantee (with $\zeta = \delta$). The running time of this algorithm is $\tilde{O}(K \cdot \mathrm{poly}(d, 1/\epsilon) + \mathrm{poly}(d, L, 1/\epsilon))$.

Proof sketch. The exponential mechanism can fail to output a good parameter if there are drastically more bad parameters than good. The key insight is that due to dispersion, the set of good parameters is not too small. In particular, we know that every $\boldsymbol{\rho} \in B(\boldsymbol{\rho}^*, w)$ has $\frac{1}{T}\sum_i u_i(\boldsymbol{\rho}) \geq \frac{1}{T}\sum_i u_i(\boldsymbol{\rho}^*) - \frac{Hk}{T} - Lw$ because at most k of the functions u_i for have discontinuities in $B(\boldsymbol{\rho}^*, w)$ and the rest are L-Lipschitz.

In a bit more detail, for a constant c fixed later on, the probability that a sample from $\mu_{\exp}$ lands in $E = \{\boldsymbol{\rho} : \frac{1}{T}\sum_i u_i(\boldsymbol{\rho}) \leq c\}$ is F/Z, where $F = \int_E f_{\exp}$ and $Z = \int_{\mathcal{C}} f_{\exp}$. We know that $F \leq \exp\left(\frac{T\epsilon c}{2H}\right)\mathrm{Vol}(E) \leq \exp\left(\frac{T\epsilon c}{2H}\right)\mathrm{Vol}\big(B(0,R)\big)$, where the second inequality follows from the fact that a ball of radius R contains the entire space $\mathcal{C}$. To lower bound Z, we use the fact that at most k of the functions $u_1, \dots, u_T$ have discontinuities in the ball $B(\boldsymbol{\rho}^*, w)$ and the rest of the functions are L-Lipschitz. It follows that for any $\boldsymbol{\rho} \in B(\boldsymbol{\rho}^*, w)$, we have $\frac{1}{T}\sum_i u_i(\boldsymbol{\rho}) \geq \frac{1}{T}\sum_i u_i(\boldsymbol{\rho}^*) - \frac{Hk}{|\mathcal{S}|} - Lw$. This is because each of the k functions with boundaries can affect the average utility by at most $H/|T|$ and otherwise $\frac{1}{T}\sum_i u_i(\cdot)$ is L-Lipschitz. Since $B(\boldsymbol{\rho}^*, w) \subset \mathcal{C}$, this gives $Z \geq \exp\big(\frac{T\epsilon}{2H}(\frac{1}{T}\sum_i u_i(\boldsymbol{\rho}^*)) - \frac{Hk}{T} - Lw)\big)\mathrm{Vol}\big(B(\boldsymbol{\rho}^*, w)\big)$.

Putting the bounds together, we have that $F/Z \leq \exp\big(\frac{T\epsilon}{2H}(c - \frac{1}{T}\sum_i u_i(\boldsymbol{\rho}^*) + \frac{Hk}{T} + Lw\big) \cdot \frac{\mathrm{Vol}(B(0,R))}{\mathrm{Vol}(B(\boldsymbol{\rho}^*, w))}$. The volume ratio is equal to $(R/w)^d$, since the volume of a ball of radius r in $\mathbb{R}^d$ is proportional to r^d. Setting this bound to ζ and solving for c gives the result.

Our efficient implementation (Algorithm 15 in Appendix B.1.5) relies on the same tools as our approximate implementation of the exponentially weighted forecaster. The main step is proving the distribution of $\hat{\boldsymbol{\rho}}$ is close to the distribution with density $f_{\exp}$. □

In Appendix B.1.8, we also give a discretization-based computationally inefficient algorithm in d dimensions that satisfies $(\epsilon, 0)$-differential privacy.

We can tune the value of w to make the dependence on L logarithmic: if $T \geq \frac{2Hd}{w\epsilon L}$, then with probability $1 - \zeta$, $\frac{1}{T}\sum_i u_i(\hat{\boldsymbol{\rho}}) \geq \frac{1}{T}\sum_i u_i(\boldsymbol{\rho}^*) - O\left(\frac{Hd}{T\epsilon}\log\frac{L\epsilon RT}{Hd} + \frac{Hk}{T} + \frac{H}{T\epsilon}\log\frac{1}{\zeta}\right)$ (Corollary B.5 in Appendix B.1.5).

Finally, we provide a matching lower bound. See Appendix B.1.5 for the full proof. When $d = 1$, we can instantiate these lower bounds using MWIS instances.

Theorem 3.5. *For every dimension $d \geq 1$, privacy parameter $\epsilon > 0$, failure probability $\zeta > 0$, $T \geq \frac{d}{\epsilon}(\frac{\ln 2}{2} - \ln\frac{1}{\zeta})$) and ϵ-differentially private optimization algorithm $\mathcal{A}$ that takes as input a collection of T piecewise constant functions mapping $B(0,1) \subset \mathbb{R}^d$ to $[0,1]$ and outputs an approximate*

maximizer, there exists a multiset $\mathcal{S}$ of such functions so that with probability at least $1-\zeta$, the output $\hat{\boldsymbol{\rho}}$ of $\mathcal{A}(\mathcal{S})$ satisfies

$$\frac{1}{T}\sum_{u\in\mathcal{S}} u(\hat{\boldsymbol{\rho}}) \le \max_{\boldsymbol{\rho}\in B(0,1)} \frac{1}{T}\sum_{u\in\mathcal{S}} u(\boldsymbol{\rho}) - \Omega\left(\inf_{(w,k)} \frac{d}{T\epsilon}\left(\ln\frac{1}{w} - \ln\frac{1}{\zeta}\right) + \frac{k}{T}\right),$$

where the infimum is taken over all (w,k)-dispersion at the maximizer parameters satisfied by $\mathcal{S}$.

Proof sketch. We construct $M = 2^d$ multi-sets of functions $\mathcal{S}_1,\dots,\mathcal{S}_M$, each with T piecewise constant functions. For every pair $\mathcal{S}_i$ and $\mathcal{S}_j$, $|\mathcal{S}_i \Delta \mathcal{S}_j|$ is small but the set $I_{\mathcal{S}_i}$ of parameters maximizing $\sum_{u\in\mathcal{S}_i} u(\boldsymbol{\rho})$ is disjoint from $I_{\mathcal{S}_j}$. Therefore, for every pair $\mathcal{S}_i$ and $\mathcal{S}_j$, the distributions $\mathcal{A}(\mathcal{S}_i)$ and $\mathcal{A}(\mathcal{S}_j)$ are similar, and since $I_{\mathcal{S}_1},\dots,I_{\mathcal{S}_t}$ are disjoint, this means that for some $\mathcal{S}_i$, with high probability, the output of $\mathcal{A}(\mathcal{S}_i) \notin I_{\mathcal{S}_i}$. The key challenge is constructing the sets $\mathcal{S}_i$ so that the suboptimality of any point not in $I_{\mathcal{S}_i}$ is $\frac{d}{T\epsilon}\log\frac{R}{w} + \frac{k}{T}$, where w and k are dispersion parameters for $\mathcal{S}_i$. We construct $\mathcal{S}_i$ so that this suboptimality is $\Theta(\frac{d}{T\epsilon})$, which gives the desired result if $w = \Theta(R)$ and $k = \Theta(\frac{d}{\epsilon})$. To achieve these conditions, we carefully fill each $\mathcal{S}_i$ with indicator functions of balls centered packed in the unit ball $B(0,1)$. □

3.1.5 Dispersion in application-specific algorithm selection

We now analyze dispersion for a range of algorithm configuration problems. In the private setting, the algorithm receives samples $\mathcal{S} \sim d^T$, where d is an arbitrary distribution over problem instances Π. The goal is to privately find a value $\hat{\boldsymbol{\rho}}$ that nearly maximizes $\sum_{x\in\mathcal{S}} u(x,\boldsymbol{\rho})$. In our applications, prior work [110, 77, 18] shows that $\hat{\boldsymbol{\rho}}$ nearly maximizes $\mathbb{E}_{x\sim d}[u(x,\boldsymbol{\rho})]$. In the online setting, the goal is to find a value $\boldsymbol{\rho}$ that is nearly optimal in hindsight over a stream $x_1,\dots,x_T$ of instances, or equivalently, over a stream $u_1 = u(x_1,\cdot),\dots,u_T = u(x_T,\cdot)$ of functions. Each x_t is drawn from a distribution $d^{(t)}$, which may be adversarial. Thus in both settings, $\{x_1,\dots,x_T\} \sim d^{(1)} \times \cdots \times d^{(T)}$, but in the private setting, $d^{(1)} = \cdots = d^{(T)}$.

Greedy algorithms. We study greedy algorithm configuration for two important problems: the maximum weight independent set (MWIS) and knapsack problems. In MWIS, there is a graph and a weight $w(v) \in \mathbb{R}_{\ge 0}$ for each vertex v. The goal is to find a set of non-adjacent vertices with maximum weight. The classic greedy algorithm repeatedly adds a vertex v which maximizes $w(v)/(1+\deg(v))$ to the independent set and deletes v and its neighbors from the graph. Gupta and Roughgarden [77] propose the greedy heuristic $w(v)/(1+\deg(v))^\rho$ where $\rho \in \mathcal{C} = [0,B]$ for some $B \in \mathbb{R}$. When $\rho = 1$, the approximation ratio is $1/D$, where D is the graph's maximum degree [127]. We represent a graph as a tuple $(\boldsymbol{w},\boldsymbol{e}) \in \mathbb{R}^n \times \{0,1\}^{\binom{n}{2}}$, ordering the vertices $v_1,\dots,v_n$ in a fixed but arbitrary way. The function $u(\boldsymbol{w},\boldsymbol{e},\cdot)$ maps a parameter ρ to the weight of the vertices in the set returned by the algorithm parameterized by ρ.

Theorem 3.6. *Suppose all vertex weights are in $(0,1]$ and for each $d^{(i)}$, every pair of vertex weights has a κ-bounded joint distribution. For any $\boldsymbol{w}$ and $\boldsymbol{e}$, $u(\boldsymbol{w},\boldsymbol{e},\cdot)$ is piecewise 0-Lipschitz and for any $\alpha \ge 1/2$, with probability $1-\zeta$ over $\mathcal{S} \sim \bigtimes_{i=1}^T d^{(i)}$, u is*

$$\left(\frac{1}{T^{1-\alpha}\kappa\ln n}, O\left(n^4 T^\alpha \sqrt{\ln\frac{n}{\zeta}}\right)\right)\text{-dispersed}$$

with respect to $\mathcal{S}$.

Proof sketch. The utility $u\left(\boldsymbol{w}^{(t)}, \boldsymbol{e}^{(t)}, \rho\right)$ has a discontinuity when the ordering of two vertices under the greedy score swaps. Thus, the discontinuities have the form

$$\frac{\ln\left(w_i^{(t)}\right) - \ln\left(w_j^{(t)}\right)}{\ln\left(d_1\right) - \ln\left(d_2\right)}$$

for all $t \in [T]$ and $i, j, d_1, d_2 \in [n]$, where $w_j^{(t)}$ is the weight of the j^{th} vertex of $\left(\boldsymbol{w}^{(t)}, \boldsymbol{e}^{(t)}\right)$ [77]. We show that when pairs of vertex weights have κ-bounded joint distributions, then the discontinuities each have $(\kappa \ln n)$-bounded distributions. Let $\mathcal{B}_{i,j,d_1,d_2}$ be the set of discontinuities contributed by vertices i and j with degrees d_1 and d_2 across all instances in $\mathcal{S}$. The buckets $\mathcal{B}_{i,j,d_1,d_2}$ partition the discontinuities into n^4 sets of independent random variables. Therefore, applying Lemma 3.1 with $P = n^4$ and $M = T$ proves the claim. □

In Appendix B.1.6, we prove Theorem 3.6 and demonstrate that it implies strong optimization guarantees. The analysis for the knapsack problem is similar (see Appendix B.1.6).

Integer quadratic programming (IQP) algorithms. We now apply our dispersion analysis to two popular IQP approximation algorithms: s-linear [67] and outward rotation rounding algorithms [152]. The goal is to maximize a function $\sum_{i,j\in[n]} a_{ij} z_i z_j$ over $\boldsymbol{z} \in \{\pm 1\}^n$, where the matrix $A = (a_{ij})$ has non-negative diagonal entries. Both algorithms are generalizations of the Goemans-Williamson (GW) max-cut algorithm [72]. They first solve the SDP relaxation $\sum_{i,j\in[n]} a_{ij}\langle \boldsymbol{u}_i, \boldsymbol{u}_j\rangle$ subject to the constraint that $\|\boldsymbol{u}_i\| = 1$ for $i \in [n]$ and then round the vectors $\boldsymbol{u}_i$ to $\{\pm 1\}$. Under s-linear rounding, the algorithm samples a standard Gaussian $\boldsymbol{Z} \sim \mathcal{N}_n$ and sets $z_i = 1$ with probability $1/2 + \phi_s\left(\langle \boldsymbol{u}_i, \boldsymbol{Z}\rangle\right)/2$ and -1 otherwise, where $\phi_s(y) = -\mathbb{1}_{y<-s} + \frac{y}{s}\cdot\mathbb{1}_{-s\le y\le s} + \mathbb{1}_{y>s}$ and s is a parameter. The outward rotation algorithm first maps each $\boldsymbol{u}_i$ to $\boldsymbol{u}_i' \in \mathbb{R}^{2n}$ by $\boldsymbol{u}_i' = [\cos(\gamma)\,\boldsymbol{u}_i\,;\,\sin(\gamma)\,\boldsymbol{e}_i]$ and sets $z_i = \operatorname{sign}\left(\langle \boldsymbol{u}_i', \boldsymbol{Z}\rangle\right)$, where $\boldsymbol{e}_i$ is the i^{th} standard basis vector, $\boldsymbol{Z} \in \mathbb{R}^{2n}$ is a standard Gaussian, and $\gamma \in [0, \pi/2]$ is a parameter. Feige and Langberg [67] and Zwick [152] prove that these rounding functions provide a better worst-case approximation ratio on graphs with "light" max-cuts, where the max-cut does not constitute a large fraction of the edges.

Our utility u maps the algorithm parameter (either s or γ) to the objective value obtained. We exploit the randomness of these algorithms to guarantee dispersion. To facilitate this analysis, we imagine that the Gaussians $\boldsymbol{Z}$ are sampled ahead of time and included as part of the problem instance. For s-linear rounding, we write the utility as $u_{\text{slin}}(A, \boldsymbol{Z}, s) = \sum_{i=1}^n a_i^2 + \sum_{i\neq j} a_{ij}\phi_s(v_i)\phi_s(v_j)$, where $v_i = \langle \boldsymbol{u_i}, \boldsymbol{Z}\rangle$. For outward rotations, $u_{\text{owr}}(A, \boldsymbol{Z}, \gamma) = \sum_{i,j} a_{ij}\operatorname{sign}(v_i')\operatorname{sign}(v_j')$, where $v_i' = \langle \boldsymbol{u}_i', \boldsymbol{Z}\rangle$.

First, we prove a dispersion guarantee for u_{owr}. The full proof is in Appendix B.1.6, where we also demonstrate the theorem's implications for our optimization settings (Theorems B.14, B.15, B.16, and B.17).

Theorem 3.7. *For any matrix A and vector $\boldsymbol{Z}$, $u_{\text{owr}}(A, \boldsymbol{Z}, \cdot)$ is piecewise 0-Lipschitz. With probability $1 - \zeta$ over $\boldsymbol{Z}^{(1)}, \ldots, \boldsymbol{Z}^{(T)} \sim \mathcal{N}_{2n}$, for any $A^{(1)}, \ldots, A^{(T)} \in \mathbb{R}^{n\times n}$ and any $\alpha \ge 1/2$, u_{owr} is*

$$\left(T^{\alpha-1}, O\left(nT^{\alpha}\sqrt{\log\frac{n}{\zeta}}\right)\right)\text{-dispersed}$$

with respect to $\mathcal{S} = \left\{\left(A^{(t)}, \boldsymbol{Z}^{(t)}\right)\right\}_{t=1}^T$.

Proof sketch. The discontinuities of $u_{\text{owr}}(A, \boldsymbol{Z}, \gamma)$ occur whenever $\langle \boldsymbol{u}_i', \boldsymbol{Z} \rangle$ shifts from positive to negative for some $i \in [n]$. Between discontinuities, the function is constant. By definition of $\boldsymbol{u}_i'$, this happens when $\gamma = \tan^{-1}(-\langle \boldsymbol{u}_i, \boldsymbol{Z}[1, \ldots, n] \rangle / Z[n+i])$, which comes from a $1/\pi$-bounded distribution. The next challenge is that the discontinuities are not independent: the n discontinuities from instance t depend on the same vector $\boldsymbol{Z}^{(t)}$. To overcome this, we let $\mathcal{B}_i$ denote the set of discontinuities contributed by vector $\boldsymbol{u}_i$ across all instances. The buckets $\mathcal{B}_i$ partition the set of discontinuities into $P = n$ sets, each containing at most T discontinuities. We then apply Lemma 3.1 with P and $M = T$ to prove the claim. □

Next, we prove the following guarantee for u_{slin}. The full proof is in Appendix B.1.6, where we also demonstrate the theorem's implications for our optimization settings (Theorems B.18, B.19, and B.20).

Theorem 3.8. *With probability $1-\zeta$ over $\boldsymbol{Z}^{(1)}, \ldots, \boldsymbol{Z}^{(T)} \sim \mathcal{N}_n$, for any matrices $A^{(1)}, \ldots, A^{(T)}$ and any $\alpha \geq 1/2$, the functions $u_{\text{slin}}\left(\boldsymbol{Z}^{(1)}, A^{(1)}, \cdot\right), \ldots, u_{\text{slin}}\left(\boldsymbol{Z}^{(T)}, A^{(T)}, \cdot\right)$ are piecewise L-Lipschitz with $L = \tilde{O}\left(MT^3n^5/\zeta^3\right)$, where $M = \max_{i,j \in [n], t \in [T]} |a_{ij}^{(t)}|$, and u_{slin} is*

$$\left(T^{\alpha-1}, O\left(nT^{\alpha}\sqrt{\log \frac{n}{\zeta}}\right)\right)\text{-dispersed}$$

with respect to $\mathcal{S} = \left\{\left(A^{(t)}, \boldsymbol{Z}^{(t)}\right)\right\}_{t=1}^{T}$.

Proof sketch. We show that over the randomness of $\boldsymbol{Z}^{(1)}, \ldots, \boldsymbol{Z}^{(T)}$, u_{slin} is (w, k)-dispersed. By definition of ϕ_s, the discontinuities of $u_{\text{slin}}\left(A^{(t)}, \boldsymbol{Z}^{(t)}, \cdot\right)$ have the form $s = |\langle \boldsymbol{u}_i^{(t)}, \boldsymbol{Z}^{(t)} \rangle|$, where $\boldsymbol{u}_i^{(t)}$ is the i^{th} vector in the solution to SDP-relaxation of $A^{(t)}$. These random variables have density bounded by $\sqrt{2/\pi}$. Let $\mathcal{B}_i$ be the set of discontinuities contributed by $\boldsymbol{u}_i^{(1)}, \ldots, \boldsymbol{u}_i^{(T)}$. The points within each $\mathcal{B}_i$ are independent. We apply Lemma 3.1 with $P = n$ and $M = T$ and arrive at our dispersion guarantee.

Proving that the piecewise portions of u_{slin} are Lipschitz is complicated by the fact that they are quadratic in $1/s$, so the slope may go to $\pm\infty$ as s goes to 0. However, if s is smaller than the smallest boundary s_0, $u_{\text{slin}}\left(\boldsymbol{Z}^{(t)}, A^{(t)}, \cdot\right)$ is constant because ϕ_s deterministically maps the variables to -1 or 1, as in the GW algorithm. We prove that s_0 is not too small using anti-concentration bounds. The Lipschitz constant is then roughly bounded by n^2/s_0^3, since we take the derivative of the sum of n^2 inverse quadratic functions. □

3.1.6 Generalization guarantees for distributional learning

It is known that regret bounds imply generalization guarantees for various online-to-batch conversion algorithms [39], but we also show that dispersion can be used to explicitly provide *uniform convergence guarantees,* which bound the difference between any function's average value on a set of samples drawn from a distribution and its expected value. Our primary tool is *empirical Rademacher complexity* [93, 25], which is defined as follows. Let $\mathcal{F} = \{f_{\boldsymbol{\rho}} : \Pi \to [0,1] : \boldsymbol{\rho} \in \mathcal{C}\}$, where $\mathcal{C} \subset \mathbb{R}^d$ is a parameter space and let $\mathcal{S} = \{x_1, \ldots, x_T\} \subseteq \Pi$. (We use this notation for the sake of generality beyond algorithm selection, but mapping to the notation from Section 3.1.1, $f_{\boldsymbol{\rho}}(x) = u(x, \boldsymbol{\rho})$.) The empirical Rademacher complexity of $\mathcal{F}$ with respect to $\mathcal{S}$ is defined as $\hat{R}(\mathcal{F}, \mathcal{S}) = \mathbb{E}_{\boldsymbol{\sigma}}\left[\sup_{f \in \mathcal{F}} \frac{1}{T}\sum_{i=1}^{T} \sigma_i f(x_i)\right]$, where $\sigma_i \sim U(\{-1, 1\})$. Classic results from learning theory [93, 25] guarantee that for any distribution d over Π, with probability $1-\zeta$ over $\mathcal{S} =$

$\{x_1, \ldots, x_T\} \sim d^T$, for all $f_{\boldsymbol{\rho}} \in \mathcal{F}$, $\left|\frac{1}{T}\sum_{i=1}^{T} f_{\boldsymbol{\rho}}(x_i) - \mathbb{E}_{x\sim d}[f_{\boldsymbol{\rho}}(x)]\right| = O(\hat{R}(\mathcal{F}, \mathcal{S}) + \sqrt{\log(1/\zeta)/T})$. Our bounds depend on the the dispersion parameters of functions belonging to the *dual* class $\mathcal{G}$. That is, let $\mathcal{G} = \{u_x : \mathcal{C} \to \mathbb{R} : x \in \Pi\}$ be the set of functions $u_x(\boldsymbol{\rho}) = f_{\boldsymbol{\rho}}(x)$ where x is fixed and $\boldsymbol{\rho}$ varies. We bound $\hat{R}(\mathcal{F}, \mathcal{S})$ in terms of the dispersion parameters satisfied by $u_{x_1}, \ldots, u_{x_T} \in \mathcal{G}$. Moreover, even if these functions are not well dispersed, we can always upper bound $\hat{R}(\mathcal{F}, \mathcal{S})$ in terms of the pseudo-dimension of $\mathcal{F}$, denoted by $\mathrm{Pdim}(\mathcal{F})$ (we review the definition in Appendix B.1.7). The full proof of Theorem 3.9 is in Appendix B.1.7.

Theorem 3.9. *Let $\mathcal{F} = \{f_{\boldsymbol{\rho}} : \Pi \to [0,1] : \boldsymbol{\rho} \in \mathcal{C}\}$ be parameterized by $\mathcal{C} \subset \mathbb{R}^d$, where $\mathcal{C}$ lies in a ball of radius R. For any set $\mathcal{S} = \{x_1, \ldots, x_T\}$, suppose the functions $u_{x_i}(\boldsymbol{\rho}) = f_{\boldsymbol{\rho}}(x_i)$ for $i \in [T]$ are piecewise L-Lipschitz and (w, k)-dispersed. Then*

$$\hat{R}(\mathcal{F}, \mathcal{S}) \leq O\left(\min\left\{\sqrt{\frac{d}{T}\log\frac{R}{w}} + Lw + \frac{k}{T}, \sqrt{\frac{\mathrm{Pdim}(\mathcal{F})}{T}}\right\}\right).$$

Proof sketch. The key idea is that when the functions $u_{x_1}, \ldots, u_{x_T}$ are (w, k)-dispersed, any pair of parameters $\boldsymbol{\rho}$ and $\boldsymbol{\rho'}$ with $\|\boldsymbol{\rho} - \boldsymbol{\rho'}\|_2 \leq w$ satisfy $|f_{\boldsymbol{\rho}}(x_i) - f_{\boldsymbol{\rho'}}(x_i)| = |u_{x_i}(\boldsymbol{\rho}) - u_{x_i}(\boldsymbol{\rho'})| \leq Lw$ for all but at most k of the elements in $\mathcal{S}$. Therefore, we can approximate the functions in $\mathcal{F}$ on the set $\mathcal{S}$ with a finite subset $\hat{\mathcal{F}}_w = \{f_{\hat{\boldsymbol{\rho}}} : \hat{\boldsymbol{\rho}} \in \hat{\mathcal{C}}_w\}$, where $\hat{\mathcal{C}}_w$ is a w-net for $\mathcal{C}$. Since $\hat{\mathcal{F}}_w$ is finite, its empirical Rademacher complexity is $O((\log|\hat{\mathcal{F}}_w|/T)^{1/2})$. We then argue that the empirical Rademacher complexity of $\mathcal{F}$ is not much larger, since all functions in $\mathcal{F}$ are approximated by some function in $\hat{\mathcal{F}}_w$. □

3.1.7 Conclusion

We study online and private optimization for application-specific algorithm selection. We introduce a general condition, dispersion, that allows us to provide strong guarantees for both of these settings. As we demonstrate, many problems in algorithm and auction design reduce to optimizing dispersed functions. In this way, we connect learning theory, differential privacy, online learning, bandits, high dimensional sampling, computational economics, and algorithm design. Our main motivation is algorithm selection, but we expect that dispersion is even more widely applicable, opening up an exciting research direction.

3.2 Semi-bandit Optimization in the Dispersed Setting

3.2.1 Introduction

Overview. In this work, we consider the problem of online optimization of piecewise Lipschitz functions with semi-bandit feedback. This is an important family of non-convex optimization problems that arises in algorithm selection problems for combinatorial settings, where the goal is to decide in a data-driven way what algorithm to use from a large family of algorithms for a given problem domain. For example, we may want to decide which clustering algorithm to use from a large family of clustering procedures in order to obtain the highest quality clusterings. In the online version of the algorithm selection problem, on each round the learner chooses an algorithm from the family and receives a new instance of the problem. The problem is characterized by a loss function that measures the performance of each algorithm in the family for the given instance. The goal is to select

algorithms so that the cumulative performance of the learner is nearly as good as the best algorithm in hindsight for that sequence of problems. The major challenge in these settings is that it is potentially computationally expensive for the learner to characterize the loss function for each round, since each run of the chosen algorithm reveals the value of the loss function at just the single point corresponding to the algorithm selected. Moreover, for combinatorial problems, small differences between two algorithms can lead to a cascade of changes in their behavior and dramatically change their performance. However, in many cases the loss function can be shown to be piecewise Lipschitz, so we can phrase the problem as online optimization of piecewise Lipschitz functions.

Prior work on piecewise Lipschitz optimization was limited to two more extreme feedback regimes: Either the learner carries out a computationally expensive process to obtain full-information feedback (i.e., it observes the loss of every algorithm on each instance), or accepts suboptimal regret bounds to work in the bandit feedback setting (i.e., it only observes the loss of only one algorithm for each instance). This creates a tradeoff between computational efficiency and good regret bounds. However, many of these algorithm selection problems exhibit rich additional structure that is ignored by these two approaches. In particular, evaluating the loss function for a single algorithm can sometimes reveal the loss for a range of similar algorithms, essentially for free; in the context of the loss function, we show that an entire Lipshitz region can often be learned at once, which we call the semi-bandit feedback setting. Our new results in the semi-bandit feedback regime achieve the best of both worlds: we can efficiently obtain the necessary feedback while also having regret bounds that are nearly as good as under full-information.

We apply our results to two online algorithm selection problems. Our results for optimizing over a family of greedy knapsack algorithms improve over the procedures of Balcan et al. [22], Gupta and Roughgarden [77], and Cohen-Addad and Kanade [46] by simultaneously being more efficient and having tighter regret bounds. We also provide the first online algorithm selection procedures for a rich family of linkage based clustering algorithms introduced by Balcan et al. [18] in the batch statistical setting that interpolates between single and complete linkage, which are algorithms that are widely used in practice [6, 126, 145] and known to perform optimally in many settings [5, 16, 15, 76].

Problem Setup. We consider the problem of online piecewise Lipschitz optimization. The learning protocol is as follows: on each round t, the learner chooses a parameter ρ_t belonging to a d-dimensional parameter space $\mathcal{C} \subset \mathbb{R}^d$, the adversary chooses a piecewise Lipschitz loss function $\ell_t : \mathcal{C} \to [0, 1]$, and the learner incurs a loss equal to $\ell_t(\rho_t)$. A function $\ell_t : \mathcal{C} \to [0, 1]$ is piecewise L-Lipchitz if we can partition the parameter space $\mathcal{C}$ into regions such that ℓ_t is L-Lipschitz when restricted to each region. Many important algorithm selection problems require optimizing piecewise Lipschitz functions [77, 18, 21, 23, 24]. In these problems, the family of algorithms is parameterized and each parameter $\rho \in \mathcal{C}$ corresponds to one algorithm. We suppose that on each round t there is a partition $A_1^{(t)}, \ldots, A_M^{(t)}$ of the parameter space $\mathcal{C}$, called the feedback system. If the learner's parameter ρ_t belongs to the set $A_i^{(t)}$, then they observe both the set $A_i^{(t)}$ as well as the loss $\ell_t(\rho)$ for every $\rho \in A_i^{(t)}$. We consider the uninformed setting, where the learner does not know the feedback system for round t in advance of selecting a parameter. For simplicity, we consider oblivious adversaries that choose their sequence of loss functions $\ell_1, \ell_2, \ldots$ adversarially, but before the interaction with the learner begins. The learner's goal is to minimize regret, which is the difference between their total accumulated loss and that of the best parameter in hindsight: $\sum_{t=1}^{T} \ell_t(\rho_t) - \min_{\rho \in \mathcal{C}} \sum_{t=1}^{T} \ell_t(\rho)$.

Throughout this chapter, we use the notation $\tilde{O}(\cdot)$ to suppress all logarithmic terms and dependence on parameters other than the time horizon T and the dimension of the parameter space d.

Main Results and Techniques.

Semi-bandit Regret Bounds in the Dispersed Setting. It is known that it is not always possible to achieve sub-linear regret for piecewise Lipschitz loss functions [100, 27, 119]. Balcan et al. [22] provide regret bounds in the full-information and bandit feedback settings under a dispersion condition that roughly measures the number of discontinuous functions in any balls of a given radius, and which is satisfied for a diverse collection of combinatorial algorithm configuration problems. In this chapter, we introduce a simplified version of this condition that captures what is asymptotically important for our regret bounds.

Definition 3.2. The sequence of loss functions $\ell_1, \ell_2, \dots$ is *β-dispersed* for the Lipshitz constant L if for all T and for all $\epsilon \geq T^{-\beta}$, we have that, in expectation, the maximum number of functions among $\ell_1, \dots, \ell_T$ which are not L-Lipshitz in any ϵ-ball of $\mathcal{C}$ is at most $\tilde{O}(\epsilon T)$. That is, for all T and for all $\epsilon \geq T^{-\beta}$, we have

$$\mathbb{E}\big[\max_{\rho \in \mathcal{C}} \big|\{1 \leq i \leq T \mid \ell_i \text{ is not } L\text{-Lipshitz in } B(\rho, \epsilon)\}\big|\big] = \tilde{O}(\epsilon T).$$

In our applications, the sequence of loss functions will be chosen by a smoothed adversary. Informally, the functions chosen by a smoothed adversary are corrupted by a small random perturbations. The expectation in this definition is over this randomness in the sequence of loss functions. (Balcan et al. [22] also show examples where sufficient randomness can arise from the algorithm itself, rather than smoothness constraints on the adversary.) In all of our applications, we prove β-dispersion with $\beta = 1/2$. We provide an algorithm for online piecewise Lipschitz optimization under semi-bandit feedback whose regret is characterized by the β-dispersion parameter of the adversarially chosen functions. In Section 3.2.2, we prove the following result:

Theorem 3.10. *Let $\mathcal{C} \subset \mathbb{R}^d$ be a bounded parameter space and $\ell_1, \ell_2, \cdots : \mathcal{C} \to [0, 1]$ be piecewise Lipschitz functions that are β-dispersed. Running Algorithm 5 under semi-bandit feedback with an appropriate choice of λ has expected regret bounded by*

$$\mathbb{E}\left[\sum_{t=1}^{T} \ell_t(\rho_t) - \ell_t(\rho^*)\right] \leq \tilde{O}\left(\sqrt{dT} + T^{1-\beta}\right).$$

In comparison, the bandit-feedback algorithm of Balcan et al. [22] (the subject of Section 3.1) has expected regret bounded by $\tilde{O}(dT^{\frac{d+1}{d+2}}3^d + T^{1-\beta})$. Even in one-dimensional problems, this leads to a regret of $\tilde{O}(T^{2/3} + T^{1-\beta})$, which is significantly worse than our results. Under different assumptions, the bandit algorithm of Cohen-Addad and Kanade [46] also has $\tilde{O}(T^{2/3})$ regret for the special case of one-dimensional piecewise constant functions.

General Tools for Verifying Dispersion. We also provide general tools for proving that a sequence of piecewise Lipschitz functions satisfies dispersion. When the sequence $\ell_1, \ell_2, \dots$ is random, we can usually directly bound the expected number of loss functions that are not L-Lipschitz on any specific ball of radius ϵ by $\tilde{O}(T\epsilon)$. However, this does not imply that the functions are β-dispersed, since the expected number of non-Lipschitz functions in the *worst* ball of radius ϵ will typically be larger than the expected number in any specific ball. In Section 3.2.3, we consider sequences of loss functions that are independent, have a one-dimensional domain, and a bounded number of discontinuities. If every ball of radius ϵ has at most $\tilde{O}(T\epsilon)$ non-Lipschitz functions among $\ell_1, \dots, \ell_T$ in expectation, then we show that the functions are $\frac{1}{2}$-dispersed.

Practical Online Algorithm Selection. In Section 3.2.4, we apply our semi-bandit optimization results to design practical algorithm selection procedures for parameterized families of algorithms for linkage-based clustering and the knapsack problem. We show how to efficiently obtain semi-bandit feedback from by running a single algorithm in the family together with a small amount of additional bookkeeping. In contrast, selection procedures that require full-information feedback typically must run the many algorithm from the family per round in order to observe the loss associated with every parameter value. Using semi-bandit feedback provides significant computational improvements over full-information optimization. At the same time, our procedures achieve regret bounds that are nearly as good as those given by prior work for the full-information setting. We also prove dispersion guarantees for each problem when the adversary is smoothed.

Explicit Comparison for Knapsack. In Section 3.2.4 we consider selecting the best algorithm from a parameterized family of greedy algorithms for the knapsack problem. The algorithm with parameter ρ assigns the score $\sigma_\rho(i) = v_i/s_i^\rho$ to item i, where the value and size of item i are v_i and s_i, respectively. Then, the algorithm greedily selects items in decreasing order of their score. In each round of the online game, the algorithm chooses a parameter ρ, a new knapsack instance with n items arrives, and our goal is for the total value of items selected by the learner to be close to the total value of the best fixed parameter ρ in hindsight. We compare our results to the best prior full-information and bandit feedback procedures for this problem.

Full-information. Balcan et al. [22] show that the exponentially weighted forecaster, when given access to full-information feedback, achieves a regret bound of $\tilde{O}(n^2\sqrt{T})$. Our tighter analysis improves the bound to $\tilde{O}(\sqrt{T})$. Obtaining full-information feedback involves running the greedy algorithm $O(n^2)$ times per round, each costing $O(n \log n)$ time, leading to a total cost of $O(n^3 \log n)$ per round.

Bandit Feedback. The discretization-based bandit algorithm of Balcan et al. [22] achieves a regret bound of $\tilde{O}(T^{2/3}n^2)$, but only requires a single run of the greedy algorithm per round, costing $O(n \log n)$ time.

Semi-bandit Feedback. Finally, in this chapter we give an algorithm whose regret is bounded by $\tilde{O}(n\sqrt{T})$ using semi-bandit feedback obtainable in time $O(n \log n)$ per round. Note that our algorithm is as efficient as the bandit-feedback algorithm, yet achieves essentially the same regret bound as the full-information algorithm.

Related Work. For online optimization of one-dimensional piecewise constant functions, Cohen-Addad and Kanade [46] provide full-information and bandit online optimization procedures. Balcan et al. [22] consider the more general setting of multi-dimensional piecewise Lipschitz functions. They introduce a dispersion condition that roughly measures how many functions are not Lipschitz in any ball, and provide algorithms with dispersion-dependent full-information and bandit regret bounds. They also verify that dispersion is satisfied for a diverse collection of algorithm selection problems.

Semi-bandit feedback was first considered for online shortest path problems, where on each round the learner selects a path through a graph and observes the length of the edges along that path (but not for other edges) [78, 85]. Alon et al. [2] introduce the Exp3-SET algorithm for semi-bandit feedback for finite-armed bandits. They consider cases where on each round t, there is a feedback graph G_t over the arms of the bandit and playing arm i reveals the loss for arm i and all arms adjacent in the graph G_t.

There is a rich literature on data-driven algorithm selection. Most prior work focuses on the statistical setting, where the learner is given a large i.i.d. sample of problem instances from some distribution, and the goal is to find the algorithm with the best performance in expectation. Gupta and Roughgarden [77] introduced this formal setting and provide sample complexity results for several families of greedy algorithms. Balcan et al. [18] consider semidefinite rounding schemes for integer quadratic programs and linkage based clustering algorithms, Balcan et al. [21] consider learning the best branch and bound algorithms for mixed integer programs, and Balcan et al. [23] consider learning the best initialization procedures for k-means clustering. In addition to these formal results, this statistical setting has been the predominant model for data-driven algorithm configuration in artificial intelligence [120], combinatorial auctions [99], numerical linear algebra [55], vehicle routing [36], and SAT solving [147].

Another related line of work aims to select an algorithm with the fastest average running time for a collection of problems even when some algorithms have very large running times and without making any structural assumptions about the family of algorithms [92, 144].

3.2.2 Semi-bandit Optimization of Piecewise Lispchitz Functions

In this section we provide an algorithm for online piecewise Lispchitz optimization and analyze its regret under dispersion. We consider the following the semi-bandit feedback setting.

Definition 3.3 (Uninformed Semi-bandit Feedback.)**.** An online optimization problem with loss functions $\ell_1, \ell_2, \dots$ has semi-bandit feedback if for each time t, there is partition $A_1^{(t)}, \dots, A_M^{(t)}$ of the parameter space $\mathcal{C}$, called a feedback system, such that when the learner plays point $\rho_t \in A_i^{(t)}$, they observe the set $A_i^{(t)}$ and $\ell_t(\rho)$ for all $\rho \in A_i^{(t)}$. For any $\rho \in \mathcal{C}$, we let $A^{(t)}(\rho)$ denote the feedback set that contains ρ.

We analyze a continuous version of the Exp3-SET algorithm of Alon et al. [2]. This algorithm uses importance weighting to construct unbiased estimates of the complete loss function on each round, which it passes as input to a continuous version of the exponentially weighted forecaster. Pseudocode is given in Algorithm 5. In Appendix B.2.1, we show how to implement this algorithm with $O(\log T)$ per round computational complexity for the case of piecewise constant losses in one dimension.

Algorithm 5 Continuous Exp3-SET

Parameter: Step size $\lambda \in [0, 1]$

1. Let $w_1(\rho) = 1$ for all $\rho \in \mathcal{C}$
2. For $t = 1, \dots, T$
   (a) Let $p_t(\rho) = w_t(\rho)/W_t$, where $W_t = \int_{\mathcal{C}} w_t(\rho)\, d\rho$.
   (b) Sample ρ_t from p_t, play it, and observe feedback set $A^{(t)}(\rho)$ and losses $\ell_t(\rho)$ for all $\rho \in A_t$.
   (c) Let $\hat{\ell}_t(\rho) = \frac{\mathbb{I}\{\rho \in A^{(t)}(\rho_t)\}}{p_t(A^{(t)}(\rho_t))} \ell_t(\rho)$, where $p_t(A^{(t)}(\rho_t)) = \int_{A^{(t)}(\rho_t)} p_t(\rho)\, d\rho$.
   (d) Let $w_{t+1}(\rho) = w_t(\rho) \exp(-\lambda \hat{\ell}_t(\rho))$ for all $\rho \in \mathcal{C}$.

Given the learner's observations on round t, Algorithm 5 uses importance weighting to estimate the complete loss function by $\hat{\ell}_t(\rho) = \frac{\mathbb{I}\{\rho \in A^{(t)}(\rho_t)\}}{p_t(A^{(t)}(\rho_t))} \ell_t(\rho)$. The key property of $\hat{\ell}_t$ is that it is an unbiased estimate of the true loss function conditioned on the history until the beginning of round

t. More formally, let $\mathbb{E}_t[\cdot] = \mathbb{E}[\cdot|\rho_1, \ldots, \rho_{t-1}, \ell_1, \ldots, \ell_t]$ denote the conditional expectation given the learner's choices until round $t-1$ and the first t loss functions. This expectation is only over the randomness of the learner's choice of ρ_t at time t. For clarity, we also use the notation $\mathbb{E}_{<t}[\cdot]$ to denote the expectation of any random variable that is a function of only $\rho_1, \ldots, \rho_{t-1}$ and $\ell_1, \ldots, \ell_t$ so that for any random quantity X, we have $\mathbb{E}[X] = \mathbb{E}_{<t}\big[\mathbb{E}_t[X]\big]$. For any fixed $\rho \in \mathcal{C}$ and any time t, a straight forward calculation shows that $\mathbb{E}_t[\hat{\ell}_t(\rho)] = \ell_t(\rho)$. Intuitively, the importance weight $1/p_t(A^{(t)}(\rho_t))$ increases the estimated losses of points that belong to feedback sets assigned low probabilities by the algorithm to compensate for the fact that they are less likely to be observed.

To simplify presentation, we use the following technical condition.

Definition 3.4 (r_0-interior optimum)**.** Let $\mathcal{C} \subset \mathbb{R}^d$ be a parameter space and consider a sequence of random functions $\ell_1, \ell_2, \cdots : \mathcal{C} \to \mathbb{R}$. We say these functions have an r_0-*interior optimum* if for all times $T \in \mathbb{N}$, with probability one there exists $\rho^* \in \operatorname{argmin}_{\rho \in \mathcal{C}} \sum_{t=1}^{T} \ell_t(\rho)$ such that $B(\rho^*, r_0) \subset \mathcal{C}$.

Note that we can usually modify a sequence of loss functions to obtain an equivalent optimization problem that is guaranteed to have an r_0-interior minimizer. For example, when the parameter space $\mathcal{C}$ is convex (e.g., a cube in $\mathbb{R}^d$, which covers most algorithm configuration applications), we can apply the following transformation: define an enlarged parameter space $\mathcal{C}' = \bigcup_{\rho \in \mathcal{C}} B(\rho, r_0)$ and a modified sequence of loss functions $\ell'_t : \mathcal{C} \to [0, 1]$ given by $\ell'_t(\rho') = \ell_t(\Pi_{\mathcal{C}}(\rho'))$, where $\Pi_{\mathcal{C}}$ denotes the Euclidean projection onto $\mathcal{C}$. Using the fact that projections onto convex sets are contractions, it follows that the sequence $\ell'_1, \ell'_2, \ldots$ is also L-Lispchitz and f-dispersed. Moreover, it has an r_0-interior minimizer and any sequence of parameters $\rho'_1, \rho'_2, \cdots \in \mathcal{C}'$ can be converted into $\rho_1, \rho_2, \ldots$ with $\rho_t = \Pi_{\mathcal{C}}(\rho'_t)$ so that $\ell_t(\rho_t) = \ell'_t(\rho'_t)$ for all t. In particular, an algorithm with low regret playing against $\ell'_1, \ell'_2, \ldots$ can be converted into one that plays against $\ell_1, \ell_2, \ldots$ with an identical regret bound.

We bound the regret of Algorithm 5 under a generalization of both both β-dispersion and the (w, k)-dispersion definition of Balcan et al. [22], leading to more precise bounds and broader applicability.

Definition 3.5. The sequence of loss functions $\ell_1, \ell_2, \ldots$ is f-*dispersed* for the Lipshitz constant L and dispersion function $f : \mathbb{N} \times [0, \infty) \to \mathbb{R}$ if for all T and for all $\epsilon > 0$, we have

$$\mathbb{E}\big[\max_{\rho \in \mathcal{C}}\big(\big|\{1 \le i \le T \mid \ell_i \text{ is not } L\text{-Lipshitz in } B(\rho, \epsilon)\}\big|\big] \le f(T, \epsilon).$$

We can express both (w, k)-dispersion and β-dispersion in terms of f-dispersion. In particular, suppose that the functions $\ell_1, \ell_2, \ldots$ are β-dispersed. Then for any $\epsilon \ge T^{-\beta}$, at most $\tilde{O}(\epsilon T)$ of the functions are not Lipschitz on the worst ball of radius ϵ. In particular, for $\epsilon = T^{-\beta}$, this gives that the number of non-Lipschitz functions is bounded by $T^{1-\beta}$ and this bound also applies to smaller balls, since decreasing the radius can only decrease the number of non-Lipschitz functions. It follows that the functions are f-dispersed for $f(T, \epsilon) = \tilde{O}(\epsilon T + T^{1-\beta})$. Next, suppose $\ell_1, \ell_2, \ldots$ are (w, k)-dispersed. Then they are f-dispersed where $f(T, \epsilon) = k$ for all $\epsilon \le w$ and $f(T, \epsilon) = T$ otherwise.

We bound the regret of Algorithm 5 in terms of the f-dispersion of the losses.

Theorem 3.11. *Let $\mathcal{C} \subset \mathbb{R}^d$ be contained in a ball of radius R and $\ell_1, \ell_2, \cdots : \mathcal{C} \to [0, 1]$ be piecewise L-Lipschitz functions that are f-dispersed with an r_0-interior minimizer. Moreover, suppose the learner gets semi-bandit feedback and, on each round t, the feedback system $A_1^{(t)}, \ldots, A_M^{(t)}$ has M*

feedback sets. For any $r \in (0, r_0]$, running Algorithm 5 with $\lambda = \sqrt{d \log(R/r)/(TM)}$ satisfies the following regret bound:

$$\mathbb{E}\big[\sum_{t=1}^{T} \ell_t(\rho_t) - \ell_t(\rho^*)\big] \le O\big(\sqrt{dTM \log(R/r)} + f(T, r) + TLr\big).$$

Proof sketch. Following the proof for the finite-armed case of Alon et al. [2], we upper and lower bound the quantity $\mathbb{E}[\log(W_{T+1}/W_1)]$, where $W_t = \int_{\mathcal{C}} w_t(\rho)\, d\rho$ is the normalizing constant at round t. The upper bound is in terms of the expected total loss of the algorithm, while the lower bound depends on the expected total loss of the best parameter ρ^* in hindsight. Dispersion plays a crucial role in the lower bound. Combining these inequalities leads to the desired regret bound.

Following a continuous analogue of the arguments used by Alon et al. [2], we arrive at the following upper bound:

$$\mathbb{E}\big[\log \frac{W_{T+1}}{W_1}\big] \le -\lambda\, \mathbb{E}\big[\sum_{t=1}^{T} \ell_t(\rho_t)\big] + \frac{\lambda^2}{2}\, \mathbb{E}\big[\sum_{t=1}^{T} \int_{\mathcal{C}} p_t(\rho) \hat{\ell}_t(\rho)^2\, d\rho\big],$$

where the second term roughly quantifies our dependence on the second moment of the estimated losses $\hat{\ell}_t(\rho)$. We show that for any fixed $\rho \in \mathcal{C}$ we have that $\mathbb{E}_t[\hat{\ell}_t(\rho)^2] \le \frac{1}{p_t(A^{(t)}(\rho))}$. This implies that

$$\mathop{\mathbb{E}}_{t}\big[\int_{\mathcal{C}} p_t(\rho) \hat{\ell}_t(\rho)^2\, d\rho\big] \le \int_{\mathcal{C}} p_t(\rho) \frac{1}{p_t(A^{(t)}(\rho))}\, d\rho = \sum_{i=1}^{M} \int_{A_i^{(t)}} p_t(\rho) \frac{1}{p_t(A_i^{(t)})}\, d\rho = M,$$

where the first equality follows from the fact that the feedback system $A_1^{(t)}, \ldots, A_M^{(t)}$ is a partition of $\mathcal{C}$. Substituting this into our upper bound, we have

$$\mathbb{E}\big[\log \frac{W_{T+1}}{W_1}\big] \le -\lambda\, \mathbb{E}\big[\sum_{t=1}^{T} \ell_t(\rho_t)\big] + \frac{\lambda^2}{2} TM.$$

Next, let $\rho^* \in \operatorname{argmin}_\rho \sum_{t=1}^{T} \ell_t(\rho)$ be a minimizer with $B(\rho^*, r_0) \subset \mathcal{C}$, let r be any radius satisfying $r \le r_0$ and let $\mathcal{B} = B(\rho^*, r)$ be the ball of radius r about ρ^*. We lower bound $\log W_{T+1}$ by integrating w_{T+1} over just the ball $\mathcal{B}$ consisting of parameters close to ρ^*:

$$\log W_{T+1} = \log\big(\int_{\mathcal{C}} \exp\big(-\lambda \sum_{t=1}^{T} \hat{\ell}_t(\rho)\big)\, d\rho\big) \ge \log\big(\int_{\mathcal{B}} \exp\big(-\lambda \sum_{t=1}^{T} \hat{\ell}_t(\rho)\big)\, d\rho\big).$$

Using the fact that $\log$ is concave, applying Jensen's inequality to the right hand side gives $\log W_{T+1} \ge \log(\mathrm{Vol}(\mathcal{B})) - \frac{\lambda}{\mathrm{Vol}(\mathcal{B})} \int_{\mathcal{B}} \sum_{t=1}^{T} \hat{\ell}_t(\rho)\, d\rho$. Let $\mathbb{E}_\ell[\cdot] = \mathbb{E}[\cdot | \ell_1, \ldots, \ell_T]$ denote the expectation conditioned on the sequence of loss functions (i.e., only taking the expectation over the randomness of the algorithm). Then, since $W_1 = \mathrm{Vol}(\mathcal{C})$, we have that $\mathbb{E}_\ell[\frac{W_{T+1}}{W_1}] \ge \log \frac{\mathrm{Vol}(\mathcal{B})}{\mathrm{Vol}(\mathcal{C})} - \frac{\lambda}{\mathrm{Vol}(\mathcal{B})} \int_{\mathcal{B}} \sum_{t=1}^{T} \ell_t(\rho)$. Here, we used the fact that $\hat{\ell}_t$ is an unbiased estimate of ℓ_t. Finally, letting D denote the number of non-Lipschitz functions in $\ell_1, \ldots, \ell_T$ on the ball $\mathcal{B}$, we can upper bound the loss of all $\rho \in \mathcal{B}$ by $\sum_{t=1}^{T} \ell_t(\rho) \le \sum_{t=1}^{T} \ell_t(\rho^*) + TLr + D$. Since the functions $\ell_1, \ell_2, \ldots$ are f-dispersed, we know that the expected number of non-Lipschitz functions in the worst ball of radius r is at most $f(T, r)$.

In particular, this implies that $\mathbb{E}[D] \leq f(T, r)$. Therefore, taking the expectation of our lower bound gives $\mathbb{E}[\frac{W_{T+1}}{W_1}] \geq \log \frac{\text{Vol}(\mathcal{B})}{\text{Vol}(\mathcal{C})} - \lambda(\sum_{t=1}^{T} \ell_t(\rho^*) + TLr + f(T, r))$.

Combining the upper and lower bounds on $\mathbb{E}[\log(W_{T+1}/W_1)]$, using the fact that $\frac{\text{Vol}(\mathcal{B})}{\text{Vol}(\mathcal{C})} \geq (\frac{r}{R})^d$, and rearranging gives

$$\mathbb{E}\big[\sum_{t=1}^{T} \ell_t(\rho_t) - \ell_t(\rho^*)\big] \leq \frac{\lambda}{2} TM + \frac{d}{\lambda} \log \frac{R}{r} + TLr + f(T, r).$$

Substituting the given value of λ completes the proof. □

Our regret bound for β-dispersed losses given in Theorem 3.10 follows immediately from Theorem 3.11. In particular, when a sequence of loss functions is β-dispersed, then it is also f-dispersed for $f(T, \epsilon) = \tilde{O}(T\epsilon + T^{1-\beta})$. Applying Theorem 3.11 with $r = \frac{1}{T^\beta(L+1)}$ bounds the algorithm's expected regret by $\tilde{O}(\sqrt{Td} + T^{1-\beta})$.

Note that our results are also applicable in two closely related settings: maximizing piecewise Lipschitz and dispersed utility functions, and the case when losses are bounded in $[0, H]$ for some known bound H instead of $[0, 1]$. A discussion of the necessary transformations can be found in Appendix B.1.4.

3.2.3 General Tools for Verifying Dispersion

In this section we provide a general tool for demonstrating that a sequence of loss functions $\ell_1, \ell_2, \ldots$ is dispersed. For many sequences of loss functions and any fixed ball $B(\rho, \epsilon)$, we are able to bound the expected number of functions among $\ell_1, \ldots, \ell_T$ that are not L-Lipschitz on that ball. However, this does not directly imply that the functions are dispersed, since in general the expected number of non-Lipschitz functions in the worst ball will be larger than the expected number of non-Lipschitz functions in any specific ball. Our main result in this section shows that for sequences of independent loss functions $\ell_1, \ell_2, \ldots$ that are defined over a one-dimensional parameter space with at most K discontinuities each, to prove dispersion for the sequence, it is sufficient to bound the expected number of non-Lipschitz functions for any fixed ball.

Theorem 3.12. *Let $\ell_1, \ell_2, \cdots : \mathbb{R} \to \mathbb{R}$ be independent piecewise L-Lipschitz functions, each having at most K discontinuities. Let $D(T, \epsilon, \rho) = \big|\{1 \leq t \leq T \,|\, \ell_t \text{ is not } L\text{-Lipschitz on } [\rho - \epsilon, \rho + \epsilon]\}\big|$ be the number of functions in $\ell_1, \ldots, \ell_T$ that are not L-Lipschitz on the ball $[\rho - \epsilon, \rho + \epsilon]$. Then we have*

$$\mathbb{E}[\sup_{\rho \in \mathbb{R}} D(T, \epsilon, \rho)] \leq \sup_{\rho \in \mathbb{R}} \mathbb{E}[D(T, \epsilon, \rho)] + O(\sqrt{T \log(TK)}).$$

Proof sketch. For simplicity, we assume that every function has exactly K discontinuities and let $\alpha^{(t)} \in \mathbb{R}^K$ be the vector whose entries are the discontinuity locations of ℓ_t. The vectors $\alpha^{(1)}, \alpha^{(2)}, \ldots$ are independent.

For any interval $I \subset \mathbb{R}$, define the function $f_I : \mathbb{R}^K \to \{0, 1\}$ by $f_I(\alpha) = 1$ if any component of α belongs to I and $f_I(\alpha) = 0$ otherwise. The sum $\sum_{t=1}^{T} f_I(\alpha^{(t)})$ counts the number of vectors $\alpha^{(1)}, \ldots, \alpha^{(T)}$ that have a component in the interval I or, equivalently, the number of functions $\ell_1, \ldots, \ell_T$ that are not L-Lipschitz on I. The main result will follow from VC-dimension based uniform convergence arguments applied to the class $\mathcal{F} = \{f_I : \mathbb{R}^K \to \{0, 1\} \,|\, I \subset \mathbb{R} \text{ is an interval}\}$.

First, we bound the VC-dimension of $\mathcal{F}$ by $O(\log K)$. The key insight is the following connection between $\mathcal{F}$ and the class of indicator functions for intervals: let $S = \{x^{(1)}, \ldots, x^{(n)}\} \subset \mathbb{R}^K$ be any collection of n vectors in $\mathbb{R}^K$ and let $P = \{x_1^{(1)}, \ldots, x_K^{(1)}, \ldots, x_1^{(n)}, \ldots, x_K^{(n)}\} \subset \mathbb{R}$ denote the set containing the union of their combined nK component values. Now consider any pair of intervals I and I'. If the indicator functions for I and I' agree on all the points in P (i.e., the intervals contain exactly the same subset of P), then we must have that f_I and $f_{I'}$ agree on every vector in S. This is because if I and I' contain exactly the same subset of P, then for each vector $x^{(i)}$, both intervals contain the same subset of its component values and we must have $f_I(x^{(i)}) = f_{I'}(x^{(i)})$. Therefore, that the number of distinct ways that the class $\mathcal{F}$ can label the set of vectors S is at most the number of ways that intervals can label the set of points P. The bound on the VC-dimension of $\mathcal{F}$ follows from the fact that, by Sauer's Lemma, intervals can only label the points in P in $O((Kn)^2)$ distinct ways, which limits the size of the largest set S that is shatterable by $\mathcal{F}$.

Applying VC-dimension based uniform convergence arguments to the class $\mathcal{F}$ and using the fact that $D(T, \epsilon, \rho) = \sum_{t=1}^{T} f_I(\alpha^{(t)})$, where $I = [\rho - \epsilon, \rho + \epsilon]$, it follows that for all time horizons T and all radiuses ϵ, with probability at least $1 - \delta$ we have $\sup_{\rho \in \mathbb{R}} D(T, \epsilon, \rho) \leq \sup_{\rho \in \mathbb{R}} \mathbb{E}[D(T, \epsilon, \rho)] + O(\sqrt{T \log(K/\delta)})$. Converting this high-probability bound into a bound in expectation completes the proof. □

If we can show that for all times T, radiuses $\epsilon > 0$ and any fixed interval I of radius ϵ, the expected number of non-Lipschitz functions on interval I is at most $\tilde{O}(T\epsilon)$, then Theorem 3.12 guarantees that the losses are β-dispersed with $\beta = 1/2$. Similarly, if the expected number of non-Lipschitz functions on the interval I is at most $g(T, \epsilon)$, then the functions are f-dispersed for $f(T, \epsilon) = g(T, \epsilon) + O(\sqrt{T \log(TK)})$.

Balcan et al. [22] also provide general tools for proving (w, k)-dispersion that can be adapted to β and f-dispersion. Our bounds provide an exponential improvement in the dependence on K, the number of discontinuities per function. Under the same assumptions as Theorem 3.12, they are able to show $\mathbb{E}[\sup_{\rho \in \mathbb{R}} D(T, \epsilon, \rho)] \leq \sup_{\rho \in \mathbb{R}} \mathbb{E}[\tilde{D}(T, \epsilon, \rho)] + O(K\sqrt{T \log(TK)})$, where $\tilde{D}(T, \epsilon, \rho)$ is the total number of discontinuities across the functions $\ell_1, \ldots, \ell_T$ that fall in the interval of radius ϵ centered at ρ (i.e., it counts multiple discontinuities from each loss function ℓ_t) (see Lemma B.33 in Appendix B.2.2). The dependence of our additive term on K is exponentially smaller and leads to improved dispersion analysis.

3.2.4 Online Algorithm Selection with Semi-bandit Feedback

In this section we apply our semi-bandit optimization results to online algorithm selection for two rich parameterized families of algorithms. For each family, we show that semi-bandit feedback can be obtained efficiently by running a single algorithm together with a small amount of bookkeeping. We also provide dispersion analysis for these problems under the assumption that the adversary is smoothed. In both cases, we obtain $\tilde{O}(\sqrt{T})$ regret bounds in the semi-bandit feedback setting. Finally, in Appendix B.2.3 we show how to use binary search to obtain semi-bandit feedback for a large class single-parameter algorithm families.

Smoothed adversaries. We consider adversaries that are smoothed in the sense of Spielman and Teng [130], where their decisions are corrupted by small random perturbations. Formally, we say that a parameter chosen by the adversary is κ-smooth if it is a random variable whose density is bounded by κ. After the adversary chooses the density for each smoothed parameter, nature samples each

parameter value independently from their corresponding distributions. Small values of κ correspond to larger random perturbations of the problem parameters, while in the limit as $\kappa \to \infty$, the adversary is able to choose the parameters deterministically. In each application, we will specify which problem parameters are smoothed, together with the bound κ on their density. For simplicity, we assume that all κ-smooth random variables are independent (i.e., the corruption of the adversary's choices is not correlated across variables).

Greedy Algorithms for Knapsack

First, we consider selecting the best algorithm from a parameterized family of a greedy algorithms for the knapsack problem. Recall that an instance of the knapsack problem consists of n items, where item i has a value v_i and a size s_i, and a knapsack capacity C. Our goal is to find the most valuable subset of items whose total size does not exceed C. Gupta and Roughgarden [77] propose using the following parameterized family of greedy knapsack algorithms: for a given parameter $\rho \in [0, R]$, set the score of item i to be $\sigma_\rho(i) = v_i/s_i^\rho$. Then, in decreasing order of score, add each item to the knapsack if there is enough capacity left. This algorithm runs in time $O(n \log n)$. In our analysis, we assume that the adversary's item values are κ-smooth.

First, we show how to perform a small amount of bookkeeping in order to obtain semi-bandit feedback. Pseudocode for the modified algorithm is given in Algorithm 6.

Algorithm 6 Greedy Knapsack Algorithm with Bookkeeping

Input: Parameter $\rho \geq 0$, item values $v_1, \ldots, v_n$, item sizes $s_1, \ldots, s_n$, knapsack capacity $C \geq 0$.

1. Let $\pi : [n] \to [n]$ be the item permutation such that $\sigma_\rho(\pi(1)) \geq \cdots \geq \sigma_\rho(\pi(n))$.
2. Initialize $S \leftarrow \emptyset$.
3. For $i = 1, \ldots, n$: if $s_{\pi(i)} \leq C$ then add $\pi(i)$ to S and set $C \leftarrow C - s_{\pi(i)}$.
4. For $i = 1, \ldots, n-1$: let $c_i \leftarrow \log(v_{\pi(i)}/v_{\pi(i+1)})/\log(s_{\pi(i)}/s_{\pi(i+1)})$.
5. Let $\rho_{\min} \leftarrow \max\{c_i \mid c_i \leq \rho\}$ and $\rho_{\max} \leftarrow \min\{c_i \mid c_i > \rho\}$.
6. Return S and interval $A = (\rho_{\min}, \rho_{\max})$.

Lemma 3.2. *Consider a knapsack instance with capacity C and n items with values $v_1, \ldots, v_n$, and sizes $s_1, \ldots, s_n$. Algorithm 6 runs in time $O(n \log n)$. Moreover, there is feedback system $A_1, \ldots, A_M$ partitioning $\mathcal{C}$ into $M = O(n^2)$ intervals such that set of items output by the algorithm is constant for $\rho \in A_i$. When run with parameter ρ, in addition to the item set S, the algorithm outputs the interval A_i containing ρ.*

Proof. Computing the permutation π takes $O(n \log n)$ time and finding the item set S and interval A only require linear passes through the items, resulting in $O(n \log n)$ total running time.

Gupta and Roughgarden [77] show that for any knapsack instance, the algorithm's output is a piecewise constant function of the parameter ρ with at most $O(n^2)$ discontinuities. In particular, for each pair of items i and j, there is a critical parameter value $c_{ij} = \log(v_i/v_j)/\log(s_i/s_j)$ such that the relative order of items i and j changes at $\rho = c_{ij}$. These critical parameter values partition $\mathcal{C}$ into $M = O(n^2)$ sets $A_1, \ldots, A_M$ such that the item ordering is fixed for all $\rho \in A_i$. Algorithm 6 computes the critical values for the each consecutive pair of items $\pi(i)$ and $\pi(i+1)$ and outputs the largest interval A containing ρ and none of these critical values. For all $\rho' \in A$, we must have $\sigma_{\rho'}(\pi(i)) \geq \sigma_{\rho'}(\pi(i+1))$ for $i = 1, \ldots, n-1$, and therefore the item ordering is constant for $\rho' \in A$.

It follows that that A does not contain c_{ij} for any pair of items i and j. On the other hand, the end points of A are critical values, so A must be equal to one of the M sets A_i. □

Next, we provide a dispersion analysis for selecting the parameter $\rho \in [0, R]$ in order to maximize the value of items selected. We assume that each instance has the same capacity C, item sizes are in $[1, C]$, and the item values are in $[0, 1]$ and κ-smooth. The corresponding loss function is $\ell(\rho) = C - \sum_{i \in S_\rho} v_i$, where S_ρ is the set of items selected by Algorithm 6 when run with parameter ρ. Since each item has $v_i/s_i \leq 1$, the maximum achievable value is C and this loss takes values in $[0, C]$.

Lemma 3.3. *Consider an adversary choosing knapsack instances with a fixed knapsack capacity C where the t^{th} instance has item sizes $s_1^{(t)}, \ldots, s_n^{(t)} \in [1, C]$, and κ-smooth item values $v_1^{(t)}, \ldots, v_n^{(t)} \in [0, 1]$. The loss functions $\ell_1, \ell_2, \ldots$ defined above are piecewise constant and f-dispersed for $f(T, \epsilon) = T\epsilon n^2\kappa^2 \ln(C) + O(\sqrt{T \log(Tn)})$ and β-dispersed for $\beta = 1/2$.*

Proof. Let $c_{ij}^{(t)} = \log(v_i^{(t)}/v_j^{(t)})/\log(s_i^{(t)}/s_j^{(t)})$ be the critical parameter value such that at $\rho = c_{ij}^{(t)}$, items i and j swap their relative order in the t^{th} instance. Balcan et al. [22] show that each critical value $c_{ij}^{(t)}$ is random and has a density function bounded by $\kappa^2 \ln(C)/2$. It follows that for any interval I of radius ϵ, the expected total number of critical values $c_{ij}^{(t)}$ summed over all pairs of items and $t = 1, \ldots, T$ is at most $T\epsilon n^2\kappa^2 \ln(C)$. This is also an upper bound on the expected number of loss functions in $\ell_1, \ldots, \ell_T$ that are not constant on I. Applying Theorem 3.12, it follows that the functions are f-dispersed for $f(T, \epsilon) = T\epsilon n^2\kappa^2 \ln(C) + O(\sqrt{T \log(Tn)}) = \tilde{O}(T\epsilon + \sqrt{T})$, which implies β-dispersion with $\beta = 1/2$.

□

Note that our analysis is tighter than that of Balcan et al. [22] due to the better bound given by Theorem 3.12. Running Algorithm 5 using the semi-bandit feedback returned by Algorithm 6, we obtain the following.

Corollary 3.1. *Under the same conditions as Lemma 3.3, using Algorithm 5 to tune the parameter $\rho \in [0, R]$ of Algorithm 6 under semi-bandit feedback has expected regret bounded by*

$$O(Cn\sqrt{T \log(RTn\kappa \log(C))}).$$

The full-information regret bound obtained by Balcan et al. [22] is $\tilde{O}(Cn^2\sqrt{T})$, which is worse than our semi-bandit bound. Moreover, applying our analysis gives a $\tilde{O}(C\sqrt{T})$ regret bound under full-information.

Interpolating between Single and Complete Linkage Clustering

Next, we consider a rich family of linkage-based clustering algorithms introduced by Balcan et al. [18] that interpolates between the classic single and complete linkage procedures. Clustering instances are described by a matrix $D = (d_{ij}) \in \mathbb{R}^{n \times n}$ giving the pairwise distances between a collection of n data points and the goal is to organize the points into a hierarchy or cluster tree. We provide the first dispersion analysis and online configuration procedures for this class of algorithms. We assume that each distance d_{ij} is κ-smooth.

The algorithm family we consider, called ρ-linkage, is family of agglomerative clustering algorithms with a single parameter $\rho \in [0, 1]$. These algorithms take as input a distance matrix

$D = (d_{ij}) \in \mathbb{R}^{n \times n}$ and the parameter value $\rho \in [0, 1]$ and output a cluster tree, which is a binary tree where each node corresponds to a cluster in the data. The leaves of the tree are the individual data points, while the root node corresponds to the entire dataset. The children of each node subdivide that cluster into two subclusters. The ρ-linkage algorithm starts with each point belonging to its own cluster. Then, it repeatedly merges the closest pair of clusters according the distance defined by $\mathrm{d}_\rho(A, B) = (1 - \rho)\,\mathrm{d}_{\min}(A, B) + \rho\,\mathrm{d}_{\max}(A, B)$, where A and B are clusters (i.e., subsets of $[n]$), $\mathrm{d}_{\min}(A, B) = \min_{a \in A, b \in B} d_{ab}$ and $\mathrm{d}_{\max}(A, B) = \max_{a \in A, b \in B} d_{ab}$. When there is only a single cluster remaining, the algorithm outputs the constructed cluster tree. Running this algorithm with $\rho = 0$ and $\rho = 1$ recovers single and complete linkage, respectively. To simplify notation in the rest of this section, given any clusters C_1, C_2, C_1' and C_2', we let $c(C_1, C_2, C_1', C_2') = \Delta_{\min}/(\Delta_{\min} - \Delta_{\max})$, where $\Delta_{\min} = \mathrm{d}_{\min}(C_1', C_2') - \mathrm{d}_{\min}(C_1, C_2)$ and $\Delta_{\max} = \mathrm{d}_{\max}(C_1', C_2') - \mathrm{d}_{\max}(C_1, C_2)$, denote the unique parameter value such that for $\rho = c(C_1, C_2, C_1', C_2')$ we have $\mathrm{d}_\rho(C_1, C_2) = \mathrm{d}_\rho(C_1', C_2')$.

First, we show that we can obtain semi-bandit feedback when running ρ-linkage by performing a small amount of bookkeeping. This algorithm maintains an interval $(\rho_{\min}, \rho_{\max})$ with the invariant that at any iteration, for all parameters $\rho' \in (\rho_{\min}, \rho_{\max})$, the algorithm would make the same merges that have been made so far. Pseudocode for this procedure is given in Algorithm 7

Algorithm 7 ρ-Linkage with Bookkeeping

Input: Parameter $\rho \in [0, 1]$, symmetric matrix $D \in [0, M]^{n \times n}$.

1. Let $S \leftarrow \{\mathrm{Leaf}(i) \text{ for } i \in [n]\}$.
2. Let $\rho_{\min} \leftarrow 0$ and $\rho_{\max} \leftarrow 1$.
3. While $|S| > 1$:
   (a) Let (C_1, C_2) be the pair of clusters in S minimizing $d_\rho(C_1, C_2)$.
   (b) For each pair of clusters $(C_1', C_2') \neq (C_1, C_2)$ in S
      i. Let $c' \leftarrow c(C_1, C_2, C_1', C_2')$.
      ii. If $c' > \rho$ then set $\rho_{\max} \leftarrow \min(\rho_{\max}, c')$, otherwise set $\rho_{\min} \leftarrow \max(\rho_{\min}, c')$.
   (c) Remove C_1 and C_2 from S and add $\mathrm{Node}(C_1, C_2)$ to S.
4. Return the only element T of S, which is the constructed cluster tree, and the interval $A = [\rho_{\min}, \rho_{\max}]$.

Lemma 3.4. *Consider a clustering instance with distance matrix $D \in \mathbb{R}^{n \times n}$. Algorithm 7 runs in time $O(n^3)$. Moreover, there is a feedback system $A_1, \ldots, A_M$ partitioning $[0, 1]$ into $M = O(n^8)$ intervals such that the cluster tree output by the algorithm is constant for $\rho \in A_i$. When run with parameter ρ, in addition to the cluster tree T, the algorithm outputs the interval A_i containing ρ.*

Proof. The algorithm performs $n - 1$ merges, and for each merge it makes two passes through the $O(n^2)$ clusters to find the closest pair and to update $\rho_{\min}$ and $\rho_{\max}$, giving a total running time of $O(n^3)$.

Balcan et al. [18] prove that there exists a partition $A_1, \ldots, A_M$ of $\mathcal{C}$ into $M = O(n^8)$ intervals such that the algorithm output is constant for $\rho \in A_i$. In particular, for any pair of possible cluster merges (C_1, C_2) and (C_1', C_2') with $\mathrm{d}_{\min}(C_1, C_2) < \mathrm{d}_{\min}(C_1', C_2')$, the algorithm prefers to merge C_1 and C_2 over C_1' and C_2' for all values of the parameter $\rho < c(C_1, C_2, C_1', C_2')$. Moreover, since $c(C_1, C_2, C_1', C_2')$ only depends on 8 points—the closest and farthest pairs of points between C_1 and C_2 and between C_1' and C_2'—and there are only $O(n^8)$ ways to select 8 points, these critical parameter values partition $\mathcal{C}$ into the $M = O(n^8)$. For $\rho \in A_i$, the ordering on all possible merges is fixed, so the algorithm will output the same cluster tree.

Finally, an invariant satisfied at every iteration of the algorithm is that for all values of $\rho' \in (\rho_{\min}, \rho_{\max})$, running the algorithm with parameter ρ' would make the same sequence of merges made up until the current iteration. Moreover, since $\rho_{\max}$ and $\rho_{\min}$ are always equal to one of the critical parameter values above, it follows that the output interval A is equal to some set A_i from the piecewise constant partition. □

Next, we provide a dispersion analysis for selecting the parameter ρ of Algorithm 7 when the clustering instances are chosen by a smoothed adversary. In particular, we suppose that on each round the adversary chooses a distance matrix $D^{(t)}$ where each distance $d_{ij}^{(t)}$ is κ-smooth and takes values in $[0, B]$, for some maximum distance B. The quantity $B/(1/\kappa) = B\kappa$ roughly captures the scale of the perturbations relative to the true distances. Our analysis leads to regret that depends on $B\kappa$ only logarithmically.

Fix any loss function g where $g(D, T)$ measures the cost of cluster tree T for distance matrix D (e.g., the distance from a ground-truth target clustering). We consider the sequence of loss functions defined by $\ell_t(\rho) = g(D^{(t)}, \mathcal{A}(D^{(t)}; \rho))$, where $\mathcal{A}(D; \rho)$ denotes the output of Algorithm 7 run on D with parameter ρ.

Lemma 3.5. *Consider an adversary choosing clustering instances where the t^{th} instance has symmetric distance matrix $D^{(t)} \in [0, B]^{n \times n}$ and for all $i \leq j$, $d_{ij}^{(t)}$ is κ-smooth. The loss functions $\ell_1, \ell_2, \ldots$ defined above are piecewise constant and f-dispersed for $f(T, \epsilon) = 32T\epsilon n^8\kappa^2M^2 + O(\sqrt{T\log(Tn)})$ and β-dispersed for $\beta = 1/2$.*

Proof sketch. In the proof of Lemma 3.4, we showed that for each time t, there are $O(n^8)$ critical parameter values partitioning $\mathcal{C}$ into regions so that the algorithm output is constant on each region. Since the loss ℓ_t only depends on ρ through the algorithm output, ℓ_t is also piecewise constant with at most $O(n^8)$ pieces.

Moreover, we argued that every discontinuity of ℓ_t occurs at a critical parameter value of the form $c = (d_{rr'}^{(t)} - d_{ii'}^{(t)})/(d_{jj'}^{(t)} - d_{ii'}^{(t)} + d_{rr'}^{(t)} - d_{ss'}^{(t)})$ where $i, i', j, j', r, r', s, s'$ are 8 point indices. Similarly to the knapsack example, we show that each critical parameter value is random and has a density function bounded by $16(\kappa B)^2$. From this, it follows that for any interval I of radius ϵ, the expected total number of critical values summing over all instances $t = 1, \ldots, T$ that land in interval I is at most $32T\epsilon(\kappa B)^2$. This also bounds the expected number of functions $\ell_1, \ldots, \ell_T$ that are not constant on I. By Theorem 3.12, the functions are f-dispersed for $f(T, \epsilon) = 32T\epsilon(\kappa B)^2 + \sqrt{T\log(Tn)} = \tilde{O}(T\epsilon + \sqrt{T})$, also implying $\frac{1}{2}$-dispersion.

There are several cases when bounding the density of the critical value c, depending on whether any of the 4 distances correspond to the same entry in the distance matrix D. We give the argument for the case when all 4 distances are distinct entries and therefore independent. The remaining cases are similar and considered in Appendix B.1.6. Let $X = d_{rr'} - d_{ii'}$ and $Y = d_{jj'} - d_{ss'}$ so that $c = X/(X + Y)$. The variables X and Y are independent. Since X and Y are each the sum of κ-smooth random variables, Lemma B.37 implies that they are each have κ-bounded densities. Using the fact that $|X + Y| \leq 2B$, applying Lemma B.39 implies that the ratio $c = X/(X + Y)$ has a $16(\kappa B)^2$ bounded density, as required. □

Running Algorithm 5 using the semi-bandit feedback returned by Algorithm 7, we obtain the following:

Corollary 3.2. *Under the same conditions as Lemma 3.5, using Algorithm 5 to tune the parameter $\rho \in [0, 1]$ of Algorithm 7 under semi-bandit feedback has expected regret bounded by*

$$O(n^4\sqrt{T\log(Tn\kappa B)}).$$

Chapter 4

Data-driven Algorithm Configuration and Metric Learning for Clustering

4.1 Introduction

Clustering is an important part of many modern data analysis pipelines. For example, we might cluster emails based on content as a pre-processing step for spam detection, or we might cluster individuals in a social network in order to suggest new connections. There are a myriad of different clustering algorithms, and it is not always clear what algorithm will give the best performance on a specific clustering task. Similarly, we often have multiple different ways to measure distances between data points, and it is not obvious which distance metric will lead to the best performance. In this work, we study data-driven algorithm selection and metric learning for clustering problems, where the goal is to use data to learn the best algorithm or metric for a specific application such as clustering emails or users of a social network. Each specific application is modeled as a distribution over clustering tasks, we observe an i.i.d. sample of clustering instances drawn from that distribution, and our goal is to choose an approximately optimal algorithm from a parameterized family of algorithms (according to some well-defined loss function). This corresponds to settings where we repeatedly solve clustering instances (e.g., clustering the emails that arrive each day) and we want to use historic instances to learn the best clustering algorithm.

We build on a recent line of work that provides learning-theoretical guarantees for data-driven algorithm configuration [77, 18, 20, 23]. These papers analyze the intrinsic complexity of parameterized algorithm families in order to provide sample complexity guarantees; that is, bounds on the number of sample instances needed in order to select an approximately optimal algorithm. In this work, we have three main contributions: first, we provide sample complexity guarantees for learning the best metric to use for a given clustering application domain. Second, we design computationally efficient procedures for data-driven clustering algorithm selection. These procedures take a sample of clustering instances and output the algorithm from the family with the lowest empirical cost. Finally, we use our efficient algorithm selection procedures to evaluate data-driven clustering algorithm selection in a number of different clustering application domains.

We study parameterized families of agglomerative or linkage-based clustering algorithms. These procedures take as input a clustering instance S and output a hierarchical clustering of S represented as a binary *cluster tree*. Each node in the tree represents a cluster in the data at one level of granularity, with the leaves corresponding to individual data points and the root node corresponding to the entire

dataset. Each internal node represents a cluster obtained by merging its two children. Linkage-based clustering algorithms build a cluster tree from the leaves up, starting with each point belonging to its own cluster and repeatedly merging the "closest" pair of clusters until only one remains.

Our algorithm families vary two aspects of this algorithm template. The first family allows us to vary the metric used to measure the distance between points in the clustering instance. Optimizing over this family can equivalently be viewed as learning the best metric for a given clustering application. The second family allows us to vary the meaning of "closest" by changing how we measure distances between clusters (in terms of the distances between their points). Two classic examples are single linkage, which measures distances in terms of the closest pair of points, and complete linkage, which measures distance in terms of the farthest pair of points. Varying each of these aspects can lead to significantly different clustering algorithms.

For these families of algorithms, we provide computationally efficient empirical risk minimization (ERM) procedures. These procedures are given a sample of clustering instances and must output an algorithm from the family that has the lowest empirical loss. A key challenge of the ERM problem for our algorithm families is that, for each clustering task S, the loss we optimize is a piecewise constant function of the algorithm parameter. This implies that the optimization problem is non-convex and the loss derivative is zero anywhere that it is defined, rendering gradient descent and similar approaches ineffective. Instead, our ERM procedures exploit structure in the piecewise constant partitionings of the parameter space to efficiently find optimal parameters.

Problem Formulation. Let $\mathcal{X}$ be a data domain. Each clustering instance consists of a point set $S = \{x_1, \ldots, x_n\} \subset \mathcal{X}$ and an (unknown) target clustering $\mathcal{Y} = (C_1, \ldots, C_k)$, where the sets $C_1, \ldots, C_k$ partition S into k clusters. Linkage-based clustering algorithms output a hierarchical clusterings, represented by a cluster tree. We measure the agreement of a cluster tree T with the target clustering $\mathcal{Y} = (C_1, \ldots, C_k)$ in terms of the Hamming distance between the best pruning of T into k clusters (i.e., k disjoint subtrees that cover the leaves of T). More formally, we define the loss

$$\ell(T, \mathcal{Y}) = \min_{P_1,\ldots,P_k} \min_{\sigma \in \mathbb{S}_n} \frac{1}{|S|} \sum_{i=1}^{K} |C_i - P_{\sigma_i}|, \tag{4.1}$$

where the first minimum is over all prunings $P_1, \ldots, P_k$ of the cluster tree T and the second minimum is over all permutations of the k cluster indices. This formulation allows us to handle the case where each clustering task has a different number of clusters, and where the desired number might not be known in advance. Our analysis applies to any loss function ℓ measuring the quality of the output cluster tree T, but we focus on the Hamming distance for simplicity. Given a distribution $\mathcal{D}$ over clustering instances (i.e., point sets together with target clusterings), our goal is to find the algorithm A from a family $\mathcal{A}$ with the lowest expected loss for an instance sampled from $\mathcal{D}$. As training data, we assume that we are given an i.i.d. sample of clustering instances annotated with their target clusterings drawn from the application distribution $\mathcal{D}$.

4.2 Learning Clustering Metrics

For many clustering problems, we have multiple different ways to measure distances between points. For example, in a dataset of captioned images, we can measure the distance between two examples in terms of either the images or the captions. Alternatively, we might have both a hand-crafted feature

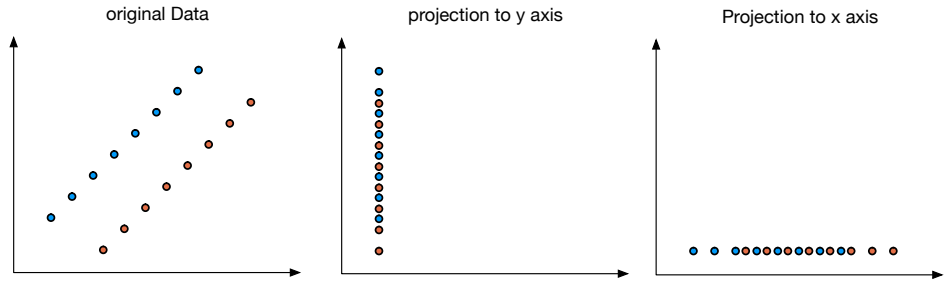


Figure 4.1: A two dimensional dataset with well separated clusters that become interleaved when projecting onto either the x or y axes.

representation incorporating expert knowledge of the clustering domain, as well as a neural network feature embedding. How should we combine these distance metrics in order to obtain high quality clusterings? In this section we introduce a parameterized family of algorithms that can use any linear combination of two given metrics. After introducing the algorithm family, we prove sample complexity guarantees for learning the optimal parameter values from sample clustering instances.

Fix two metrics $\mathrm{d}_0, \mathrm{d}_1 : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ defined on the data universe. We define an algorithm family $\mathcal{A}_{\text{metric}}(\mathrm{d}_0, \mathrm{d}_1)$ parameterized by $\beta \in [0, 1]$. The algorithm with parameter β runs complete linkage using the metric $\mathrm{d}_\beta(x, x') = (1 - \beta)\,\mathrm{d}_0(x, x') + \beta\,\mathrm{d}_1(x, x')$. That is, it builds a cluster tree as follows: starting with each point in a cluster of its own, it repeatedly merges the pair of clusters that are the closest according to the distance function $\mathrm{D}_{\max}(A, B; \beta) = \max_{a \in A, b \in B} \mathrm{d}_\beta(a, b)$ until only one cluster remains. Each merge introduces a new interal node in the cluster tree. Pseudocode is given in Algorithm 8. When the two metrics d_0 and d_1 are clear from context, we let $A_\beta^{\text{metric}}(S)$ denote the cluster tree output by this algorithm when run on clustering instance S.

Algorithm 8 β-linkage Clustering

Input: Metrics d_0 and d_1, parameter $\beta \in [0, 1]$, and clustering instance $S = \{x_1, \ldots, x_n\}$.

1. Let $\mathcal{N} = \{\mathrm{Leaf}(x_1), \ldots, \mathrm{Leaf}(x_n)\}$ be the initial set of nodes (one leaf per point).
2. While $|\mathcal{N}| > 1$
   (a) Let $A, B \in \mathcal{N}$ be the clusters in $\mathcal{N}$ minimizing $\max_{a \in A, b \in B} \mathrm{d}_\beta(a, b)$.
   (b) Remove nodes A and B from $\mathcal{N}$ and add $\mathrm{Node}(A, B)$ to $\mathcal{N}$.
3. Return the cluster tree (the only element of $\mathcal{N}$).

Next, we show that using a mixture of two metrics d_0 and d_1 can lead to lower loss clusterings than either base metric alone. In Figure 4.1 we construct a two-dimensional dataset and suppose that d_0 and d_1 are distance metrics that only use one of the two dimensions. Clustering the data according to either d_0 or d_1 is equivalent to clustering the data after projecting to either the x or y axis, which interleaves points belonging to the two target clusters and leads to high error. On the other hand, the algorithm A_β^{metric} with $\beta = 0.5$ is is equivalent to using the ℓ_1-distance for the original 2 dimensional data, which keeps the two clusters well separated, and leads to low error.

Finally, we give sample-complexity guarantees for learning the best algorithm from this family (or equivalently, the best metric). These results ensure that for a sufficiently large sample of clustering instances, with high probability, the empirical loss of every parameter $\beta \in [0, 1]$ is close to its expected cost on the application specific distribution. This implies that an empirically optimal parameter is also approximately optimal in expectation. Our sample complexity results are a conseqeuence

of the following key structural property of the algorithm family.

Theorem 4.1. *For any distance metrics* d_0 *and* d_1 *and any clustering instance* $S = \{x_1, \ldots, x_n\}$*, there exists a partition of* $[0, 1]$ *into* $M = O(n^4)$ *intervals* $I_1, \ldots, I_M$ *such that for any* $i \in [M]$ *and any* $\beta, \beta' \in I_i$*, we have* $A_\beta^{\textit{metric}}(S) = A_{\beta'}^{\textit{metric}}(S)$*. That is, the algorithm outputs the same cluster tree for all values of* β *in the interval* I_i*.*

Proof. Complete linkage can be implemented in such a way that it only makes comparisons of pairwise distances between points (e.g., is $\mathrm{d}_\beta(x, x')$ bigger or smaller than $\mathrm{d}_\beta(y, y')$?) but never uses the exact values of the pairwise distances. In particular, finding the farthest pair of points between two clusters can be done with only distance comparisons and, similarly, once we have the farthest pair of points between each pair of clusters, determining which of those pairs is closest can again be done using only distance comparisons. It follows that if two metrics d_β and $d_{\beta'}$ give the same answers to all pairwise distance comparisons, then running complete linkage with both metrics will output exactly the same cluster tree.

We will argue that for any fixed clustering instance $S = \{x_1, \ldots, x_n\}$, there is a partitioning of $[0, 1]$ into $M = O(n^4)$ intervals $I_1, \ldots, I_M$ with the following property: For every interval I_i in the partition and any pair of parameters $\beta, \beta' \in I_i$, for all points $x, x', y, y' \in S$, we have $d_\beta(x, x') < d_\beta(y, y')$ if and only if $d_{\beta'}(x, x') < d_{\beta'}(y, y')$. In other words, all comparisons between pairwise distances have the same outcome under the metrics d_β and $d_{\beta'}$. By the above argument, it follows there is one cluster tree that complete linkage will output when run with the metric d_β for any $\beta \in I_i$.

All that remains is to construct the partition described above. Fix any subset of 4 points $x, x', y, y' \in S$. We start by characterizing the subset of β values for which $d_\beta(x, x') < d_\beta(y, y')$. Expanding the definition of d_β, the following inequality describes this subset.

$$(1 - \beta) \cdot d_0(x, x') + \beta d_1(x, x') < (1 - \beta) \cdot d_0(y, y') + \beta d_1(y, y'). \tag{4.2}$$

Since (4.2) is linear in the parameter β, it follows that either $d_\beta(x, x') < d_\beta(y, y')$ for all or none of the values $\beta \in [0, 1]$, or there exists a unique critical parameter value at

$$c = \frac{d_0(y, y') - d_0(x, x')}{d_1(x, x') - d_0(x, x') + d_0(y, y') - d_1(y, y')},$$

such that the outcome of the comparison between $d_\beta(x, x')$ and $d_\beta(y, y')$ only changes at $\beta = c$. Since there are only $O(n^4)$ subsets of 4 points from S, it follows that there are at most $O(n^4)$ of these critical parameter values. The collection of critical parameter values partitions $[0, 1]$ into $O(n^4)$ intervals, such that on each interval the outcome of every comparison of pairwise distances is fixed, as required. □

Following similar arguments as Balcan et al. [18], this leads to the following uniform convergence guarantee, which we prove in Appendix C.

Theorem 4.2. *Consider the family* $\mathcal{A}_{\textit{metric}}(\mathrm{d}_0, \mathrm{d}_1)$ *and let* $(S_1, \mathcal{Y}_1), \ldots, (S_N, \mathcal{Y}_N)$ *be an i.i.d. sample of clustering instances with target clusterings of size* $N = O\big(\frac{1}{\epsilon^2}(\log n + \log \frac{1}{\delta})\big)$*, where* n *is a bound on the number of points per instance. With probability at least* $1 - \delta$*, the following holds for all* $\beta \in [0, 1]$

$$\left| \frac{1}{N} \sum_{i=1}^{N} \ell(A_\beta^{\textit{metric}}(S_i), \mathcal{Y}_i) - \mathop{\mathbb{E}}_{(S,\mathcal{Y})\sim\mathcal{D}}[\ell(A_\beta^{\textit{metric}}(S), \mathcal{Y})] \right| \leq \epsilon.$$

4.3 Learning Merge Functions

We also provide efficient algorithm configuration procedures for a family of linkage-based clustering algorithms introduced by Balcan et al. [20]. We briefly introduce the algorithm family and state their sample complexity results for this family. Linkage based clustering algorithms construct cluster trees by repeatedly merging the "closest" pair of clusters. However, there are many different ways to measure the distance between a pair of clusters in terms of the distance between their points. To avoid confusion with learning the best metric, we refer to a distance function defined on clusters as a *merge function*. For example, single linkage measures distances in terms of the closest pair of points; that is, it uses the merge function $\mathrm{D}_{\text{min}}(A, B) = \min_{a \in A, b \in B} \mathrm{d}(a, b)$, where d is a fixed metric defined on the data universe $\mathcal{X}$. Alternatively, complete linkage uses the farthest pair of points: $\mathrm{D}_{\text{max}}(A, B) = \max_{a \in A, b \in B} \mathrm{d}(a, b)$. Another popular linkage based algorithm is average linkage, which uses the merge function $\mathrm{D}_{\text{avg}}(A, B) = \frac{1}{|A||B|} \sum_{a \in A, b \in B} \mathrm{d}(a, b)$. For any pair of merge functions D_0 and D_1, we define an algorithm family $\mathcal{A}_{\text{merge}}(\mathrm{D}_0, \mathrm{D}_1)$ parameterized by $\alpha \in [0, 1]$. The algorithm with parameter α uses the merge function $\mathrm{D}_\alpha(A, B) = (1 - \alpha)\,\mathrm{D}_0(A, B) + \alpha\,\mathrm{D}_1(A, B)$. When the merge functions D_0 and D_1 are clear from context, we let $A_\alpha^{\text{merge}}(S)$ denote the cluster tree output by the algorithm with parameter α run on clustering instance S. Pseudocode is given in Algorithm 9.

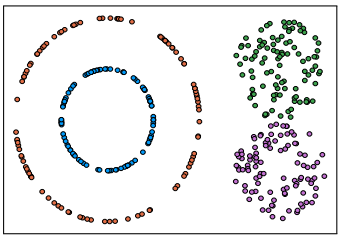
Algorithm 9 α-linkage Clustering

Input: Merge functions D_0 and D_1, parameter $\alpha \in [0, 1]$, and clustering instance $S = \{x_1, \ldots, x_n\}$.

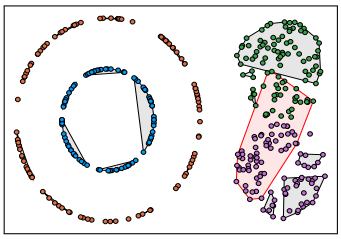
1. Let $\mathcal{N} = \{\mathrm{Leaf}(x_1), \ldots, \mathrm{Leaf}(x_n)\}$ be the initial set of nodes (one leaf per point).
2. While $|\mathcal{N}| > 1$
   (a) Let $A, B \in \mathcal{N}$ be the clusters in $\mathcal{N}$ minimizing $\mathrm{D}_\alpha(A, B) = (1-\alpha) \cdot \mathrm{D}_0(A, B) + \alpha \cdot \mathrm{D}_1(A, B)$.
   (b) Remove nodes A and B from $\mathcal{N}$ and add $\mathrm{Node}(A, B)$ to $\mathcal{N}$.
3. Return the cluster tree (the only element of $\mathcal{N}$).

To motivate this algorithm family, we show that mixing two merge functions can lead to significantly lower loss clusterings than either of the two original merge functions. Figure 4.2 shows a two-dimensional dataset where both single and complete linkage fail to find high-quality clusterings, yet a combination of them has zero error. Intuitively, the two ring-shaped clusters are easy for single linkage to recover, but difficult for complete linkage, while the two disk-shaped clusters are easy for complete linkage, but hard for single linkage. Figures 4.2b and 4.2d show the clusters obtained by single and complete linkage, respectively, after some number of merges have been performed. In both cases, the algorithms have already merged many points that belong to different target clusters, and therefore no pruning of the cluster tree will have low Hamming distance to the target clustering. Single linkage merges large portions of the two disk-shaped clusters together because there are very near points that belong to different clusters. On the other hand, complete linkage struggles to keep the two ring-shaped clusters separated; once the diameter of a cluster of points belonging to the inner or outer ring becomes larger than the gap between the two rings, complete linkage will prefer to merge that cluster with points from the opposite ring instead of points belonging to the same ring. However, running the algorithm with $\alpha \in [0.07, 0.16]$ results in zero error. These values of α put most of their weight on single linkage, and succeed at recovering the ring-shaped clusters. The small weight placed on complete linkage penalizes merges that produce large-diameter clusters, and this discourages merging clusters across the two disk clusters before merging each disk cluster itself, since this produces clusters of larger diameter.

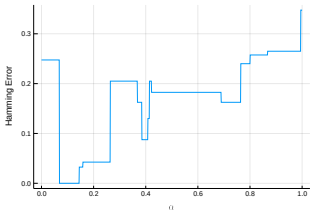
Balcan et al. [18] provide sample complexity guarantees for a number of linkage-based clustering



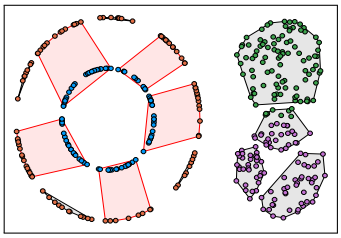
(a) A clustering instance with 4 target clusters.



(b) Clusters for single linkage after 370 merges.



(c) The error of α-linkage as a function of α.



(d) Clusters for complete linkage after 389 merges.

Figure 4.2: An example clustering instance where both single and complete linkage produce high-cost clusterings, but a mixture of the two merge functions leads to zero cost.

algorithms. They explicitly consider the family $\mathcal{A}_{\text{merge}}(\mathrm{D}_{\min}, \mathrm{D}_{\max})$ that interpolates between single and complete linkage. Their key results show that for any fixed clustering instance S with n points, the parameter space $[0, 1]$ can be partitioned into $O(n^8)$ intervals such that for each interval I, the cluster tree produced by A_α^{merge} for $\alpha \in I$ is fixed. Using this, they prove the following sample complexity guarantee:

Theorem 4.3 ([18]). *Consider the family $\mathcal{A}_{merge}(\mathrm{D}_{min}, \mathrm{D}_{max})$ and let $(S_1, \mathcal{Y}_1), \ldots, (S_N, \mathcal{Y}_N)$ be an i.i.d. sample of clustering instances with target clusterings of size $N = O\big(\frac{1}{\epsilon^2}(\log n + \log \frac{1}{\delta})\big)$, where n is a bound on the number of points per instance. With probability at least $1 - \delta$, the following holds for all $\alpha \in [0, 1]$*

$$\left| \frac{1}{N} \sum_{i=1}^{N} \ell(A_\alpha^{merge}(S_i), \mathcal{Y}_i) - \mathop{\mathbb{E}}_{(S,\mathcal{Y})\sim\mathcal{D}} [\ell(A_\alpha^{merge}(S), \mathcal{Y})] \right| \leq \epsilon.$$

4.4 Efficient Algorithm Selection

In this section we design efficient procedures for finding the algorithm from the families introduced in Section 4.2 and Section 4.3 that has the lowest average loss on a sample of labeled clustering instances $(S_1, \mathcal{Y}_1), \ldots, (S_N, \mathcal{Y}_N)$ where $\mathcal{Y}_i = (C_1^{(i)}, \ldots, C_{k_i}^{(i)})$ is the target clustering for instance S_i. Recall that the loss function $\ell(T, \mathcal{Y})$ defined in (4.1) computes the Hamming distance between the target clustering $\mathcal{Y}$ and the best pruning of the cluster tree T. Formally, our goal is to solve the following optimization problems:

$$\operatorname*{argmin}_{\alpha\in[0,1]} \frac{1}{N} \sum_{i=1}^{N} \ell(A_\alpha^{\text{merge}}(S_i), \mathcal{Y}_i) \qquad \text{and} \qquad \operatorname*{argmin}_{\beta\in[0,1]} \frac{1}{N} \sum_{i=1}^{N} \ell(A_\beta^{\text{metric}}(S_i), \mathcal{Y}_i).$$

The key challenge of these optimization problems is that, for a fixed clustering instance S and any of our algorithm families, we can partition the parameter space $[0, 1]$ into finitely many intervals such that for each interval I, the cluster tree output by the algorithm is the same for every parameter in I. It follows that the loss function is a piecewise constant function of the algorithm parameter. Therefore, both optimization problems are non-convex the loss derivative is zero wherever it is defined, rendering gradient descent and similar algorithms ineffective.

We solve the optimization problems by explicitly computing the piecewise constant loss function for each instance S. That is, we find a collection of discontinuity locations $0 = c_0 < \cdots < c_M = 1$ and values $v_1, \ldots, v_M \in \mathbb{R}$ so that for each $i \in [M]$, running the algorithm on instance S with a parameter in $[c_{i-1}, c_i)$ has loss equal to v_i. Given this representation of the loss functions for all N instances, finding the parameter with minimal loss can be done in $O(\sum_{i=1}^{N} M_i \log N)$ time, where M_i is the number of discontinuities for S_i. The bulk of the computational cost is incurred by computing the piecewise constant loss functions, which we focus on for the rest of the section.

Our efficient optimization procedures exploit a more powerful structural property of the algorithm families we study: for a clustering instance S, not only is the output cluster tree a piecewise constant function of the algorithm parameter, but for any length t, the sequence of first t merges performed by the algorithm is a piecewise constant function of the parameter. For length $t = 0$, the partition is a single region containing all parameters in $[0, 1]$, since every algorithm trivially starts with the empty sequence of merges. For each length $t > 0$, the piecewise constant partition for the first t merges is a refinement of the partition for $t - 1$ merges. We can represent this sequence of partitions using a *partition tree*, which is a tree where each node is labeled by an interval, for all depths t the nodes at depth t partition $[0, 1]$, and edges represent subset relationships. The tree described above represents all possible execution paths for the algorithm family when run on the instance S as we vary the algorithm parameter. In particular, each path from the root node to a leaf corresponds to one possible sequence of merges. We therefore call this tree the *execution tree* of the algorithm family when run on an instance S. Figure 4.3 shows an example execution tree for the family $\mathcal{A}_{\text{merge}}(\mathrm{D}_{\text{min}}, \mathrm{D}_{\text{max}})$ on a small dataset with 4 points. To find the piecewise constant loss function for a clustering instance S, we perform a depth-first traversal of the leaves of the execution tree and compute the loss for the cluster tree produced at each leaf.

4.4.1 Optimizing the Merge Function

In this section we show that for any merge functions D_0 and D_1 anda ny clustering instance S, the execution tree of $\mathcal{A}_{\text{merge}}(\mathrm{D}_0, \mathrm{D}_1)$ when run on S is well defined. Moreover, we give an efficient algorithm for finding the children of any node in the tree. Finally, we show how to compute the piecewise constant loss function for $\mathcal{A}_{\text{merge}}(\mathrm{D}_0, \mathrm{D}_1)$ on the instance S by performing a depth-first traversal of the execution tree.

Lemma 4.1. *For any merge functions D_0 and D_1 and any clustering instance S, the execution tree for $\mathcal{A}_{merge}(\mathrm{D}_0, \mathrm{D}_1)$ when run on S is well defined. That is, there exists a partition tree such that for any node v at depth t, the same sequence of first t merges is performed by A_α^{merge} for all α in the interval for node v.*

Proof. The proof is by induction on the depth t. The base case is for depth $t = 0$, in which case we can use a single node whose interval is $[0, 1]$. Since all algorithms in the family start with an empty-sequence of merges, this satisfies the execution tree property.

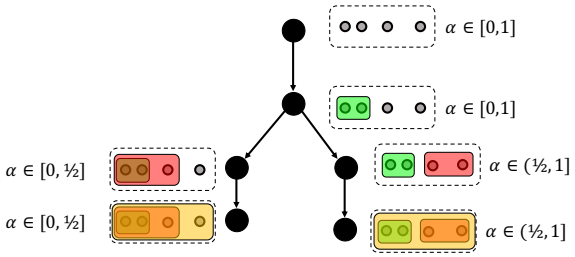


Figure 4.3: An example of the execution tree of $\mathcal{A}_{\text{merge}}(\mathrm{D}_{\text{min}}, \mathrm{D}_{\text{max}})$ for a clustering instance with 4 points. We also show the clustering of the points produced by the sequence of merges associated with each node in the tree. Each colored rectangle represents a cluster (and for clarity we also show the two children of each cluster). The leaves show the two possible cluster trees that can be output by any algorithm from the family $\mathcal{A}_{\text{merge}}(\mathrm{D}_{\text{min}}, \mathrm{D}_{\text{max}})$ for this instance.

Now suppose that there is a tree of depth t with the execution tree property. If $t = |S| - 1$ then we are finished, since the algorithms in $\mathcal{A}_{\text{merge}}(\mathrm{D}_0, \mathrm{D}_1)$ make exactly $|S| - 1$ merges. Otherwise, consider any leaf node v of the depth t tree with parameter interval I_v. It is sufficient to show that we can partition I_v into subintervals such that for α in each subinterval the next merge performed is constant. By the inductive hypothesis, we know that the first t merges made by A_α^{merge} are the same for all $\alpha \in I_v$. After performing these merges, the algorithm will have arrived at some set of clusters $C_1, \ldots, C_m$ with $m = |S| - t$. For each pair of clusters C_i and C_j, the distance $\mathrm{D}_\alpha(C_i, C_j) = (1-\alpha)\,\mathrm{D}_0(C_i, C_j) + \alpha\,\mathrm{D}_1(C_i, C_j)$ is a linear function of the parameter α. Therefore, for any clusters C_i, C_j, C_k, and C_l, the algorithm will prefer to merge C_i and C_j over C_j and C_k for a (possibly empty) sub-interval of I_v, corresponding to the values of $\alpha \in I_v$ where $\mathrm{D}_\alpha(C_i, C_j) < \mathrm{D}_\alpha(C_k, C_l)$. For any fixed pair of clusters C_i and C_j, taking the intersection of these intervals over all other pairs C_j and C_k guarantees that clusters C_i and C_j will be merged exactly for parameter values in some subinterval of I_v. For each merge with a non-empty parameter interval, we can introduce a child node of v labeled by that parameter interval. These children partition I_v into intervals where the next merge is constant, as required. □

Figure 4.3 shows the execution tree for the family $\mathcal{A}_{\text{merge}}(\mathrm{D}_{\text{min}}, \mathrm{D}_{\text{max}})$ on a clustering instance with $n = 4$ points. For this clustering instance, only the second merge depends on the algorithm parameter α, resulting in two possible output cluster trees.

Next, we provide an efficient algorithm for finding the children of a node in the execution tree. Given the node's parameter interval I and the set of clusters $C_1, \ldots, C_m$ resulting from that node's merge sequence, we use a sweep-line algorithm to determine all possible next merges and the corresponding parameter intervals. First, we calculate the merge for $\alpha = \alpha_{\text{lo}}$ by enumerating all pairs of clusters. Suppose clusters C_i and C_j are the optimal merge for $\alpha = \alpha_{\text{lo}}$. We then determine the largest value α' for which we will still merge these clusters by solving the linear equation $\mathrm{D}_\alpha(C_i, C_j) = \mathrm{D}_\alpha(C_k, C_l)$ for all other pairs of clusters C_k and C_l, keeping track of the minimal solution larger than α. Denote the minimal solution larger than α by $c \in I$. We are guaranteed that $A_{\alpha'}^{\text{merge}}$ will merge clusters C_i and C_j for all $\alpha' \in [\alpha, c)$. We repeat this procedure starting from $\alpha = c$ to determine the next merge and corresponding interval, and so on, until $\alpha \geq \alpha_{\text{hi}}$. Pseudocode is given in Algorithm 10. Our next result bounds the running time of this procedure.

Lemma 4.2. *Let $C_1, \ldots, C_m$ be a collection of clusters, D_0 and D_1 be any pair of merge functions,*

and $[\alpha_{lo}, \alpha_{hi})$ be a subset of the parameter space. If there are M distinct cluster pairs C_i, C_j that minimize $\mathrm{D}_\alpha(C_i, C_j)$ for values of $\alpha \in [\alpha_{lo}, \alpha_{hi})$, then the running time of Algorithm 10 is $O(Mm^2K)$, where K is the cost of evaluating the merge functions D_0 and D_1.

Proof. The loop in step 4 of Algorithm 10 runs once for each possible merge, giving a total of M iterations. Each iteration finds the closest pair of clusters according to D_α using $O(m^2)$ evaluations of the merge functions D_0 and D_1. Calculating the critical parameter value c involves solving $O(m^2)$ linear equations whose coefficients are determined by four evaluations of D_0 and D_1. It follows that the cost of each iteration is $O(m^2K)$, where K is the cost of evaluating D_0 and D_1, and the overall running time is $O(Mm^2K)$. □

Algorithm 10 Find all merges for $\mathcal{A}_{\text{merge}}\mathrm{D}_0, \mathrm{D}_1$

Input: Set of clusters $C_1, \ldots, C_m$, merge functions $\mathrm{D}_0, \mathrm{D}_1$, parameter interval $[\alpha_{\text{lo}}, \alpha_{\text{hi}})$.

1. Let $\mathcal{M} = \emptyset$ be the initially empty set of possible merges.
2. Let $\mathcal{I} = \emptyset$ be the initially empty set of parameter intervals.
3. Let $\alpha = \alpha_{\text{lo}}$.
4. While $\alpha < \alpha_{\text{hi}}$:
   (a) Let C_i, C_j be the pair of clusters minimizing $(1-\alpha) \cdot \mathrm{D}_0(C_i, C_j) + \alpha \cdot \mathrm{D}_1(C_i, C_j)$.
   (b) For each $k, l \in [m]$, let $c_{kl} = \Delta_0/(\Delta_0 - \Delta_1)$, where $\Delta_p = \mathrm{D}_{\text{p}}(C_i, C_j) - \mathrm{D}_{\text{p}}(C_k, C_l)$ for $p \in \{0, 1\}$.
   (c) Let $c = \min\big(\{c_{kl} \mid c_{kl} > \alpha\} \cup \{\beta_{\text{hi}}\}\big)$.
   (d) Add merge (C_i, C_j) to $\mathcal{M}$ and $[\alpha, c)$ to $\mathcal{I}$.
   (e) Set $\alpha = c$.
5. Return $\mathcal{M}$ and $\mathcal{I}$.

With this, our algorithm for computing the piecewise constant loss function for an instance S performs a depth-first traversal of the leaves of the execution tree for $\mathcal{A}_{\text{merge}}(\mathrm{D}_0, \mathrm{D}_1)$, using Algorithm 10 to determine the children of each node. When we reach a leaf in the depth-first traversal, we have both the corresponding parameter interval $I \subset [0, 1]$, as well as the cluster tree T such that $A_\alpha^{\text{merge}}(S) = T$ for all $\alpha \in I$. We then evaluate the loss $\ell(T, \mathcal{Y})$ to get one piece of the piecewise constant loss function. Detailed pseudocode for this approach is given in Algorithm 11. The following Lemma characterizes the overall running time of the algorithm.

Theorem 4.4. *Let $S = \{x_1, \ldots, x_n\}$ be a clustering instance and D_0 and D_1 be any two merge functions. Suppose that the execution tree of $\mathcal{A}_{\text{merge}}(\mathrm{D}_0, \mathrm{D}_1)$ on S has E edges. Then the total running time of Algorithm 11 is $O(En^2K)$, where K is the cost of evaluating the merge functions D_0 and D_1.*

Proof. Fix any node v in the execution tree with m clusters $C_1, \ldots, C_m$ and M outgoing edges (i.e., M possible merges from the state represented by v). We run Algorithm 10 to determine the children of v, which by Lemma 4.2 costs $O(Mn^2K)$, since $m \leq n$. Summing over all non-leaves of the execution tree, the total cost is $O(En^2K)$. In addition to computing the children of a given node, we need to construct the children nodes, but this takes constant time per child. □

We can also express the running time of Algorithm 11 in terms of the number of discontinuities of the function $\alpha \mapsto A_\alpha^{\text{merge}}(S)$. There is one leaf of the execution tree for each constant interval of this function, and the path from the root of the execution tree to that leaf is of length $n-1$. Therefore,

Algorithm 11 Depth-first Enumeration of α-linkage Execution Tree

Input: Point set $x_1, \ldots, x_n$, cluster distance functionds d_1 and d_2.

1. Let r be the root node of the execution tree with $r.\mathcal{N} = \{(x_1), \ldots, (x_n)\}$ and $r.I = [0, 1]$.
2. Let s be a stack of execution tree nodes, initially containing the root r.
3. Let $\mathcal{T} = \emptyset$ be the initially empty set of possible cluster trees.
4. Let $\mathcal{I} = \emptyset$ be the initially empty set of intervals.
5. While the stack s is not empty:
   (a) Pop execution tree node e off stack s.
   (b) If $e.\mathcal{N}$ has a single cluster, add $e.\mathcal{N}$ to $\mathcal{T}$ and $e.I$ to $\mathcal{I}$.
   (c) Otherwise, for each merge (C_i, C_j) and interval I_c returned by Algorithm 10 run on $e.\mathcal{N}$ and $e.I$:
      i. Let c be a new node with state given by $e.\mathcal{N}$ after merging C_i and C_j and $c.I = I_c$.
      ii. Push c onto the stack s.
6. Return $\mathcal{T}$ and $\mathcal{I}$.

the cost associated with that path is at most $O(Kn^3)$ and enumerating the execution tree to obtain the piecewise constant loss function for a given instance S spends $O(Kn^3)$ time for each constant interval of $\alpha \mapsto A_\alpha^{\text{merge}}(S)$. In contrast, the combinatorial approach of Balcan et al. [18] requires that we run α-linkage once for every interval in their partition of $[0, 1]$, which always contains $O(n^8)$ intervals (i.e., it is a refinement of the piecewise constant partition). Since each run of α-Linkage costs $O(Kn^2 \log n)$ time, this leads to a running time of $O(Kn^{10} \log n)$. The key advantage of our approach stems from the fact that the number of discontinuities of the function $\alpha \mapsto A_\alpha^{\text{merge}}(S)$ is often several orders of magnitude smaller than $O(n^8)$.

It is crucial to use a depth-first instead of breadth-first traversal. The depth of the execution tree of $\mathcal{A}_{\text{merge}}(\mathrm{D}_0, \mathrm{D}_1)$ on a clustering instance $S = \{x_1, \ldots, x_n\}$ is $n - 1$, while the width of the tree can be very large. In a depth-first traversal, we only need to store one path of the tree in memory at a time, and we can save intervals of the piecewise constant loss function to disk as they are encountered. On the other hand, breadth-first traversal requires that we keep an entire layer of the execution tree in memory, which can be very large.

4.4.2 Optimizing the Metric

Next, we show that a similar algorithmic approach can be used for finding the piecewise constant loss function for the family $\mathcal{A}_{\text{metric}}(\mathrm{d}_0, \mathrm{d}_1)$ on a clustering instance S. We prove that the execution tree is well defined, and provide an efficient algorithm for finding the children of each node in the execution tree, allowing us to use a depth-first traversal to find the piecewise constant loss function for any clustering instance S.

Lemma 4.3. *For any metrics d_0 and d_1 and any clustering instance S, the execution tree for the family $\mathcal{A}_{\text{metric}}(\mathrm{d}_0, \mathrm{d}_1)$ when run on S is well defined. That is, there exists a partition tree such that for any node v at depth t, the same sequence of first t merges is performed by A_β^{metric} for all β in the interval for node v.*

Proof. The proof is by induction on the depth t. The base case is for depth $t = 0$, in which case we can use a single node whose interval is $[0, 1]$. Since all algorithms in the family start with an empty-sequence of merges, this satisfies the execution tree property.

Now suppose that there is a tree of depth t with the execution tree property. If $t = |S| - 1$ then we are finished, since the algorithms in $\mathcal{A}_{\text{metric}}(\mathrm{d}_0, \mathrm{d}_1)$ make exactly $|S| - 1$ merges. Otherwise, consider any leaf node v of the depth t tree with parameter interval I_v. It is sufficient to show that we can partition I_v into subintervals such that for β in each subinterval the next merge performed is constant. By the inductive hypothesis, we know that the first t merges made by A_β^{metric} are the same for all $\beta \in I_v$. After performing these merges, the algorithm will have arrived at some set of clusters $C_1, \ldots, C_m$ with $m = |S| - t$. Recall that algorithms in the family $\mathcal{A}_{\text{metric}}(\mathrm{d}_0, \mathrm{d}_1)$ run complete linkage using the metric d_β. Complete linkage can be implemented in such a way that it only makes comparisons between pairwise point distances (i.e., is $\mathrm{d}_\beta(x, x')$ larger or smaller than $\mathrm{d}_\beta(y, y')$?). To see this, for any pair of clusters, we can find the farthest pair of points between them using only distance comparisons. And, once we have the farthest pair of points between all pairs of clusters, we can find the pair of clusters to merge by again making only pairwise comparisons. It follows that if two parameters β and β' have the same outcome for all pairwise distance comparisons, then the next merge to be performed must be the same. We use this observation to partition the interval I_v into subintervals where the next merge is constant. For any pair of points $x, x' \in S$, the distance $\mathrm{d}_\beta(x, x') = (1 - \beta)\,\mathrm{d}_0(x, x') + \beta\,\mathrm{d}_1(x, x')$ is a linear function of the parameter β. Therefore, for any points $x, x', y, y' \in S$, there is at most one critical parameter value where the relative order of $\mathrm{d}_\beta(x, x')$ and $\mathrm{d}_\beta(y, y')$ changes. Between these $O(|S|^4)$ critical parameter values, the ordering on all pairwise merges is constant, and the next merge performed by the algorithm will also be constant. Therefore, there must exist a partitioning of I_v into at most $O(|S|^4)$ sub-intervals such that the next merge is constant on each interval. We let the children of v correspond to the coarsest such partition. □

Next, we provide an efficient procedure for determining the children of a node v in the execution tree of $\mathcal{A}_{\text{metric}}(\mathrm{d}_0, \mathrm{d}_1)$. Given the node's parameter interval $I = [\beta_{\text{lo}}, \beta_{\text{hi}})$ and the set of clusters $C_1, \ldots, C_m$ resulting from that node's sequence of merges, we again use a sweep-line procedure to find the possible next merges and the corresponding parameter intervals. First, we determine the pair of clusters that will be merged by A_β^{metric} for $\beta = \beta_{\text{lo}}$ by enumerating all pairs of clusters. Suppose the winning pair is C_i and C_j and let $x \in C_i$ and $x' \in C_j$ be the farthest pair of points between the two clusters. Next, we find the largest value of β' for which we will still merge the clusters C_i and C_j. To do this, we enumerate all other pairs of clusters C_k and C_l and all pairs of points $y \in C_k$ and $y' \in C_l$, and solve the linear equation $\mathrm{d}_{\beta'}(x, x') = \mathrm{d}_\beta(y, y')$, keeping track of the minimal solution larger than β. Denote the minimal solution larger than β by c. We are guaranteed that for all $\beta' \in [\beta, c)$, the pair of clusters merged will be C_i and C_j. Then we repeat the process with $\beta = c$ to find the next merge and corresponding interval, and so on, until $\beta \geq \beta_{\text{hi}}$. Pseudocode for this procedure is given in Algorithm 12. The following Lemma bounds the running time:

Lemma 4.4. *Let $C_1, \ldots, C_m$ be a collection of clusters, d_0 and d_1 be any pair of metrics, and $[\beta_{lo}, \beta_{hi})$ be a subset of the parameter space. If there are M distinct cluster pairs C_i, C_j that complete linkage would merge when using the metric d_β for $\beta \in [\beta_{lo}, \beta_{hi})$, then the running time of Algorithm 12 is $O(Mn^2)$.*

Proof. The loop in step 4 of Algorithm 12 runs once for each possible merge, giving a total of M iterations. Each iteration finds the merge performed by complete linkage using the d_β metric, which takes $O(n^2)$ time, and then solves $O(n^2)$ linear equations to determine the largest value of β' such that the same merge is performed. It follows that the cost of each iteration is $O(n^2)$, leading to an overall running time of $O(Mn^2)$. Note, we assume that the pairwise distances $\mathrm{d}_\beta(x, x')$ can be evaluated in

constant time. This can always be achieved by precomputing two $n \times n$ distance matrices for the base metrics d_0 and d_1, respectively. □

Algorithm 12 Find all merges for $\mathcal{A}_{\text{metric}}\mathrm{d}_0, \mathrm{d}_1$

Input: Set of clusters $C_1, \ldots, C_m$, metrics $\mathrm{d}_0, \mathrm{d}_1$, parameter interval $[\beta_{\text{lo}}, \beta_{\text{hi}})$.

1. Let $\mathcal{M} = \emptyset$ be the initially empty set of possible merges.
2. Let $\mathcal{I} = \emptyset$ be the initially empty set of parameter intervals.
3. Let $\beta = \beta_{\text{lo}}$.
4. While $\beta < \beta_{\text{hi}}$:
   (a) Let C_i, C_j be the pair of clusters minimizing $\max_{a \in A, b \in B} \mathrm{d}_\beta(a, b)$.
   (b) Let $x \in C_i$ and $x' \in C_j$ be the farthest points between C_i and C_j.
   (c) For all pairs of points y and y' belonging to different clusters, let $c_{yy'} = \Delta_0/(\Delta_0 - \Delta_1)$ where $\Delta_p = \mathrm{d}_{\mathrm{p}}(y, y') - \mathrm{d}_{\mathrm{p}}(x, x')$ for $p \in \{0, 1\}$.
   (d) Let $c = \min\big(\{c_{yy'} \mid c_{yy'} > \alpha\} \cup \{\alpha_{\text{hi}}\}\big)$.
   (e) Add merge (C_i, C_j) to $\mathcal{M}$ and $[\beta, c)$ to $\mathcal{I}$.
   (f) Set $\beta = c$.
5. Return $\mathcal{M}$ and $\mathcal{I}$.

Our algorithm for computing the piecewise constant loss function for an instance S performs a depth-first traversal of the leaves of the execution tree for $\mathcal{A}_{\text{metric}}(\mathrm{d}_0, \mathrm{d}_1)$, using Algorithm 12 to determine the children of each node. When we reach a leaf in the depth-first traversal, we have both the corresponding parameter interval $I \subset [0, 1]$, as well as the cluster tree T such that $A_\beta^{\text{metric}}(S) = T$ for all $\beta \in I$. We then evaluate the loss $\ell(T, \mathcal{Y})$ to get one piece of the piecewise constant loss function. Detailed pseudocode for this approach is given in Algorithm 13. The following Lemma characterizes the overall running time of the algorithm.

Theorem 4.5. *Let $S = \{x_1, \ldots, x_n\}$ be a clustering instance and d_0 and d_1 be any two merge functions. Suppose that the execution tree of $\mathcal{A}_{\text{metric}}(\mathrm{d}_0, \mathrm{d}_1)$ on S has E edges. Then the total running time of Algorithm 13 is $O(En^2)$.*

Proof. Fix any node v in the execution tree with m clusters $C_1, \ldots, C_m$ and M outgoing edges (i.e., M possible merges from the state represented by v). We run Algorithm 12 to determine the children of v, which by Lemma 4.4 costs $O(Mn^2)$. Summing over all non-leaves of the execution tree, the total cost is $O(En^2)$. □

Again, we can express the running time of Algorithm 13 in terms of the number of discontinuities of the function $\beta \mapsto A_\beta^{\text{metric}}(S)$. There is one leaf of the execution tree for each constant interval of this function, and the path from the root of the execution tree to that leaf is of length $n - 1$. Therefore, the cost associated with that path is at most $O(n^3)$ and enumerating the execution tree to obtain the piecewise constant loss function for a given instance S spends $O(n^3)$ time for each constant interval of $\beta \mapsto A_\beta^{\text{metric}}(S)$.

4.5 Experiments

In this section we cary out experiments for both algorithm families we study on clustering applications derived from classification datasets. Our experiments demonstrate that the best algorithm for different

Algorithm 13 Depth-first Enumeration of β-linkage Execution Tree

Input: Point set $x_1, \ldots, x_n$, cluster distance functionds d_1 and d_2.

1. Let r be the root node of the execution tree with $r.\mathcal{N} = \{(x_1), \ldots, (x_n)\}$ and $r.I = [0, 1]$.
2. Let s be a stack of execution tree nodes, initially containing the root r.
3. Let $\mathcal{T} = \emptyset$ be the initially empty set of possible cluster trees.
4. Let $\mathcal{I} = \emptyset$ be the initially empty set of intervals.
5. While the stack s is not empty:
   (a) Pop execution tree node e off stack s.
   (b) If $e.\mathcal{N}$ has a single cluster, add $e.\mathcal{N}$ to $\mathcal{T}$ and $e.I$ to $\mathcal{I}$.
   (c) Otherwise, for each merge (C_i, C_j) and interval I_c returned by Algorithm 12 run on $e.\mathcal{N}$ and $e.I$:
      i. Let c be a new node with state given by $e.\mathcal{N}$ after merging C_i and C_j and $c.I = I_c$.
      ii. Push c onto the stack s.
6. Return $\mathcal{T}$ and $\mathcal{I}$.

application varies greatly (i.e., there is no single algorithm from the families that works well across all the distributions we consider). Second, we also observe that in many cases it is possible to achieve non-trivial improvements in clustering quality by using a mixture of two base algorithms or metrics.

Experimental setup. In each of our experiments we define a distribution $\mathcal{D}$ over clustering tasks (i.e., clustering instances S together with target clusterings $\mathcal{Y}$). For each clustering instance, the loss of the cluster tree output by a clustering algorithm is measured in terms of the loss defined in (4.1), which computes the Hamming distance between the target clustering and the best pruning of the cluster tree. The losses take values in $[0, 1]$ and correspond to the fraction of the dataset assigned to clusters that disagree with the target clustering. To evaluate algorithm performance, we draw N sample clustering tasks from the given distribution and use the algorithms developed in Section 4.4 to evaluate the average empirical loss every algorithm in one algorithm family. In each experiment, we specify the pair of merge functions D_0 and D_1 or the pair of metrics d_0 and d_1 that we are interpolating between.

Clustering instance distributions. We perform experiments on the following application-specific distributions:

MNIST Subsets. Our first distribution over clustering tasks corresponds to clustering subsets of the MNIST dataset [98], which contains 80,000 hand-written examples of the digits 0 through 9. We generate a random clustering instance from the MNIST data as follows: first, we select $k = 5$ digits from $\{0, \ldots, 9\}$ at random, then we randomly select 200 examples belonging to each of the selected digits, giving a total of $n = 1000$ images. The target clustering for this instance is given by the ground-truth digit labels. We measure distances between any pair of digits in terms of the the Euclidean distance between their images represented as vectors of pixel intensities.

CIFAR-10 Subsets. We also consider a distribution over clustering tasks that corresponds to clustering subsets of the CIFAR-10 dataset [94]. This dataset contains 6000 images of each of the following classes: airplate, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. Each example is a 32×32 color image with 3 color channels. We pre-process the data to obtain neural-network

feature representations for each example. We include 50 randomly rotated and cropped versions of each example and obtain feature representations from layer 'in4d' of a pre-trained Google inception network[1]. This gives a 144-dimensional feature representation for each of the 2500000 examples (50 randomly rotated copies of the 6000 examples for each of the 10 classes). Like in the MNIST Subsets distribution, we generate clustering tasks from CIFAR-10 as follows: first, select $k = 5$ classes at random, then choose 50 examples belonging to each of the selected classes, giving a total of $n = 250$ images. The target clsutering for this instance is given by the ground-truth class labels. We measure distance between any pair of images as the distance between their feature embeddings.

Omniglot Subsets. Next, we consider a distribution over clustering tasks corresponding to clustering subsets of the Omniglot dataset [96]. The Omniglot dataset consists of written characters from 50 different alphabets with a total of 1623 different characters. The dataset includes 20 examples of each character, leading to a total of 32,460 examples. We generate a random clustering instance from the Omniglot data as follows: first, we choose one of the alphabets at random. Next, we choose k uniformly in $\{5, \dots, 10\}$ and choose k random characters from that alphabet. The clustering instance consists of all $20k$ examples for the selected characters, and the target clustering is given by the ground-truth character labels.

We use two different distance metrics on the Omniglot dataset. First, we use the cosine distance between neural network feature embeddings. The neural network was trained to perform digit classification on MNIST. Second, each example has both an image of the written character, as well as the stroke trajectory for the written character (i.e., a time series of (x, y) coordinates of the tip of the pen when the character was written). We also use the following distance defined in terms of the strokes: Given two pen stroke trajectories $s = (x_t, y_t)_{t=1}^{T}$ and $s' = (x'_t, y'_t)_{t=1}^{T}$, we define the distance between them by

$$d(s, s') = \frac{1}{T + T'} \left(\sum_{t=1}^{T} d\big((x_t, y_t), s'\big) + \sum_{t=1}^{T'} d\big((x'_t, y'_t), s\big) \right),$$

where $d\big((x_t, y_t), s'\big)$ denotes the Euclidean distance from the point (x_t, y_t) to the closest point in s'. This is the average distance from any point from either trajectory to the nearest point on the other trajectory.

Synthetic Rings and Dicsc. We consider a two dimensional synthetic distribution similar to the rings and disks example given in Figure 4.2. Each clustering instance has 4 clusters, where two are ring-shaped and two are disk-shaped. To generate each instance we sample 100 points uniformly at random from each ring or disc. The two rings have radiuses 0.4 and 0.8, respectively, and are both centered at the origin. The two disks have radius 0.4 and are centered at $(1.5, 0.4)$ and $(1.5, -0.4)$, respectively. For this data, we measure distances between points in terms of the Euclidean distance between them.

Results

Learning merge functions for MNIST. First we evaluate the family $\mathcal{A}_{\text{merge}}(\mathrm{D}_{\text{min}}, \mathrm{D}_{\text{max}})$ that interpolates between single and complete linkage, and the family $\mathcal{A}_{\text{merge}}(\mathrm{D}_{\text{avg}}, \mathrm{D}_{\text{max}})$ that interpolate between average and complete linkage on the MNIST data. Figure 4.4 shows the loss for both families averaged over $N = 1000$ samples from the MNIST clustering task distribution. The best standard

[1]The exact feature extraction code used for the CIFAR-10 data can be found at `https://github.com/dmlc/mxnet-notebooks/blob/master/python/moved-from-mxnet/cifar10-recipe.ipynb`

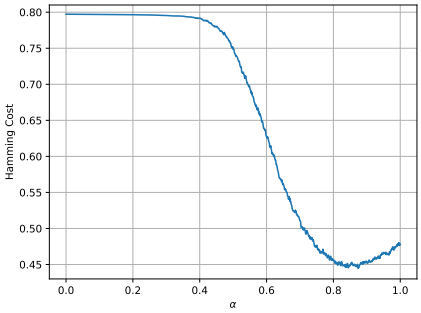
algorithm for this distribution is complete linkage, which achieves an average loss of 0.478. When interpolating between single and complete linkage, the optimal parameter value is given by $\alpha = 0.874$ which achieves an average loss of 0.4392, which is an improvement of 0.037. When interpolating between average and complete linkage, the optimal parameter is $\alpha = 0.599$, which has average loss of 0.4431. In both cases, the optimal intermediate algorithm gives a non-trivial improvement over the best classic algorithm, and the optimal mixing parameter is different for the two considered algorithm families.

Learning merge functions for CIFAR-10. Figure 4.5 shows the results of learning the best merge function from the families $\mathcal{A}_{\text{merge}}(\text{D}_{\text{min}}, \text{D}_{\text{max}})$ and $\mathcal{A}_{\text{merge}}(\text{D}_{\text{avg}}, \text{D}_{\text{max}})$ on the CIFAR-10 Subsets distribution averaged over $N = 1000$ sampled instances. When interpolating between single and complete linkage, the optimal parameter is $\alpha = 0.98$ which achieves average loss 0.543. When interpolating between average and complete linkage, the optimal parameter is $\alpha = 0.954$ and achieves average loss 0.544. The average loss of complete linkage ($\alpha = 1.0$) is 0.548. For this distribution, complete linkage performs very well. One explanation is that the neural network feature embedding might be grouping the points belonging to each class into separated ball-shape groups, which is the best-case for complete linkage.

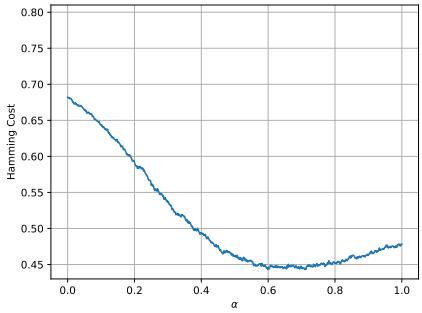
Learning the merge function for Rings and Discs. Figure 4.6 shows the results of learning the best merge function from the families $\mathcal{A}_{\text{merge}}(\text{D}_{\text{min}}, \text{D}_{\text{max}})$ and $\mathcal{A}_{\text{merge}}(\text{D}_{\text{avg}}, \text{D}_{\text{max}})$ When interpolating between single and complete linkage, the optimal parameter is $\alpha = 0.179$, which achieves average loss of 0.042. In contrast, single linkage ($\alpha = 0$) has average loss 0.236 and complete linkage ($\alpha = 1$) has average loss 0.269. Optimizing over the family results in clusterings that correctly classify almost an additional 20% of the data. When interpolating between average and complete linkage, however, we find that there is significantly less variation in the cost. The explanation is that both average and complete linkage succeed at clustering the disk-shaped clusters, and both fail at clustering the ring-shaped clusters.

Learning the merge function for Omniglot Subsets. Figure 4.7 shows the results of learning the best merge function from the families $\mathcal{A}_{\text{merge}}(\text{D}_{\text{min}}, \text{D}_{\text{max}})$ and $\mathcal{A}_{\text{merge}}(\text{D}_{\text{avg}}, \text{D}_{\text{max}})$. For this experiment, we use the distance based on cosine distance between the neural network feature embedding for each image. When interpolating between single and complete linkage, the optimal parameter is $\alpha = 0.179$, which achieves average loss of 0.042. In contrast, single linkage ($\alpha = 0$) has average loss 0.236 and complete linkage ($\alpha = 1$) has average loss 0.269. Optimizing over the family results in clusterings that correctly classify almost an additional 20% of the data. When interpolating between average and complete linkage, however, we find that there is significantly less variation in the cost. The explanation is that both average and complete linkage succeed at clustering the disk-shaped clusters, and both fail at clustering the ring-shaped clusters.

Learning the metric for Omniglot. Next we consider learning the best metric for the Omniglot clustering distribution. In this case, we interpolate between the two distance metrics described above: the cosine distance between neural network embeddings of each example image, and a hand-designed stroke distance. Figure 4.8 shows the empirical loss for each parameter β averaged over $N = 4000$ samples from the Omniglot distribution. The best "base" metric is the neural network feature embedding, which results in an average loss of 0.421. The optimal parameter value occurs at $\beta = 0.514$ with an average loss of 0.330, giving an improvement of 0.091. In other words, running complete linkage with the mixed metric correctly clusters almost an extra 10% of the data correctly.

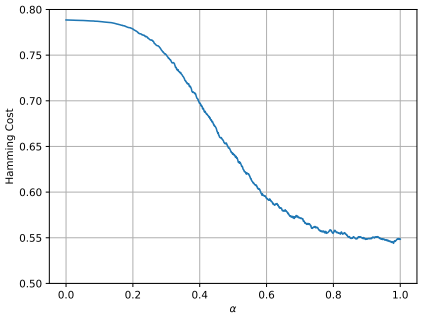


(a) Empirical loss for $\mathcal{A}_{\text{merge}}(\mathrm{D}_{\text{min}}, \mathrm{D}_{\text{max}})$.

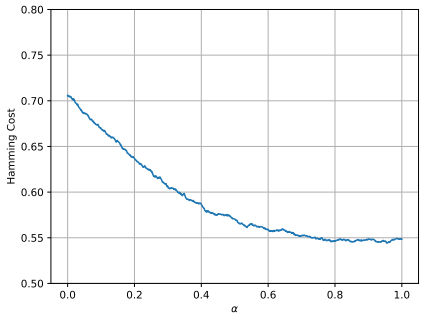


(b) Empirical loss for $\mathcal{A}_{\text{merge}}(\mathrm{D}_{\text{avg}}, \mathrm{D}_{\text{max}})$.

Figure 4.4: Empirical loss for interpolating between single and complete linkage as well as average and complete linkage on 1000 randomly sampled tasks from the MNIST subsets distribution.

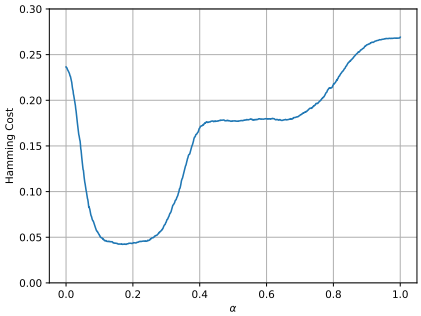


(a) Empirical loss for $\mathcal{A}_{\text{merge}}(\mathrm{D}_{\text{min}}, \mathrm{D}_{\text{max}})$.

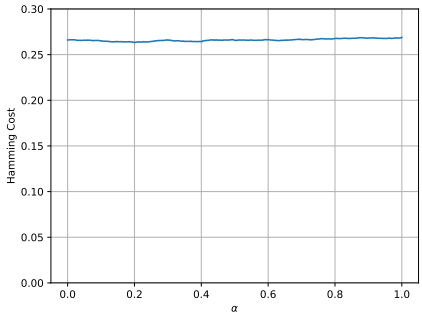


(b) Empirical loss for $\mathcal{A}_{\text{merge}}(\mathrm{D}_{\text{avg}}, \mathrm{D}_{\text{max}})$.

Figure 4.5: Empirical loss for interpolating between single and complete linkage as well as average and complete linkage on 1000 randomly sampled tasks from the CIFAR-10 Subsets distribution.

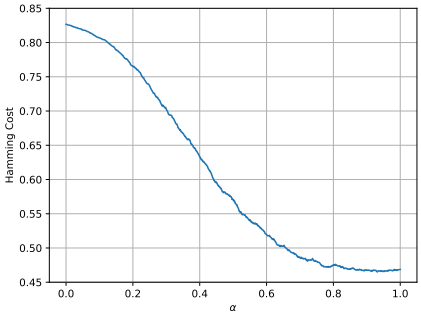


(a) Empirical loss for $\mathcal{A}_{\text{merge}}(\mathrm{D}_{\text{min}}, \mathrm{D}_{\text{max}})$.

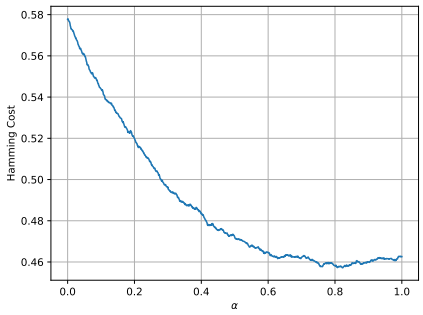


(b) Empirical loss for $\mathcal{A}_{\text{merge}}(\mathrm{D}_{\text{avg}}, \mathrm{D}_{\text{max}})$.

Figure 4.6: Empirical loss for interpolating between single and complete linkage as well as average and complete linkage on 1000 randomly sampled tasks from the rings and disks distribution.



(a) Empirical loss for $\mathcal{A}_{\text{merge}}(\text{D}_{\text{min}}, \text{D}_{\text{max}})$.



(b) Empirical loss for $\mathcal{A}_{\text{merge}}(\text{D}_{\text{avg}}, \text{D}_{\text{max}})$.

Figure 4.7: Empirical loss for interpolating between single and complete linkage as well as average and complete linkage on 1000 randomly sampled tasks from the Omniglot Subsets distribution.

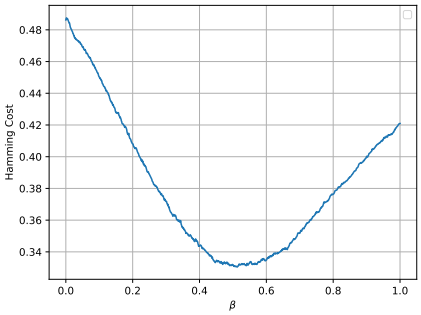


Figure 4.8: Empirical loss for interpolating between the neural network feature embedding metric and stroke metric for Omniglot data. $\beta = 0$ corresponds to the stroke distance, while $\beta = 1$ corresponds to the neural network embedding.

Chapter 5

A New Approach to Individual Fairness: Envy-free Classification

5.1 Introduction

The study of fairness in machine learning is driven by an abundance of examples where learning algorithms were perceived as discriminating against protected groups [134, 53]. Addressing this problem requires a conceptual — perhaps even philosophical — understanding of what fairness means in this context. In other words, the million dollar question is (arguably[1]) this: What are the formal constraints that fairness imposes on learning algorithms?

In this chapter, we propose a new measure of algorithmic fairness. It draws on an extensive body of work on rigorous approaches to fairness, which — modulo one possible exception (see Section 5.1.2) — has not been tapped by machine learning researchers: the literature on *fair division* [34, 111]. The most prominent notion is that of *envy-freeness* [69, 142], which, in the context of the allocation of goods, requires that the utility of each individual for his allocation be at least as high as his utility for the allocation of any other individual; this is the gold standard of fairness for problems such as cake cutting [121, 118] and rent division [133, 71]. In the classification setting, envy-freeness would simply mean that the utility of each individual for his distribution over outcomes is at least as high as his utility for the distribution over outcomes assigned to any other individual.

It is important to say upfront that envy-freeness is *not* suitable for several widely-studied problems where there are only two possible outcomes, one of which is 'good' and the other 'bad'; examples include predicting whether an individual would default on a loan, and whether an offender would recidivate. In these degenerate cases, envy-freeness would require that the classifier assign each and every individual the exact same probability of obtaining the 'good' outcome, which, clearly, is not a reasonable constraint.

By contrast, we are interested in situations where there is a diverse set of possible outcomes, and individuals have diverse preferences for those outcomes. For example, consider a system responsible for displaying credit card advertisements to individuals. There are many credit cards with different eligibility requirements, annual rates, and reward programs. An individual's utility for seeing a card's advertisement will depend on his eligibility, his benefit from the rewards programs, and potentially other factors. It may well be the case that an envy-free advertisement assignment shows Bob adver-

[1] Recent work takes a somewhat different view [87].

tisements for a card with worse annual rates than those shown to Alice; this outcome is not unfair if Bob is genuinely more interested in the card offered to him. Such rich utility functions are also evident in the context of job advertisements [53]: people generally want higher paying jobs, but would presumably have higher utility for seeing advertisements for jobs that better fit their qualifications and interests.

A second appealing property of envy-freeness is that its fairness guarantee binds at the level of individuals. Fairness notions can be coarsely characterized as being either individual notions, or group notions, depending on whether they provide guarantees to specific individuals, or only on average to a protected subgroup. The majority of work on fairness in machine learning focuses on group fairness [102, 62, 151, 80, 84, 150].

There is, however, one well-known example of individual fairness: the influential fair classification model of Dwork et al. [62]. The model involves a set of individuals and a set of outcomes. The centerpiece of the model is a *similarity metric* on the space of individuals; it is specific to the classification task at hand, and ideally captures the ethical ground truth about relevant attributes. For example, a man and a woman who are similar in every other way should be considered similar for the purpose of credit card offerings, but perhaps not for lingerie advertisements. Assuming such a metric is available, fairness can be naturally formalized as a Lipschitz constraint, which requires that individuals who are close according to the similarity metric be mapped to distributions over outcomes that are close according to some standard metric (such as total variation).

As attractive as this model is, it has one clear weakness from a practical viewpoint: the availability of a similarity metric. Dwork et al. [62] are well aware of this issue; they write that justifying this assumption is "one of the most challenging aspects" of their approach. They add that "in reality the metric used will most likely only be society's current best approximation to the truth." But, despite recent progress on automating ethical decisions in certain domains [115, 70], the task-specific nature of the similarity metric makes even a credible approximation thereof seem unrealistic. In particular, if one wanted to learn a similarity metric, it is unclear what type of examples a relevant dataset would consist of.

In place of a metric, envy-freeness requires access to individuals' utility functions, but — by contrast — we do not view this assumption as a barrier to implementation. Indeed, there are a variety of techniques for learning utility functions [41, 114, 13]. Moreover, in our running example of advertising, one can use standard measures like expected click-through rate (CTR) as a good proxy for utility.

It is worth noting that the classification setting is different from classic fair division problems in that the "goods" (outcomes) are non-excludable. In fact, one envy-free solution simply assigns each individual to his favorite outcome. But this solution may be severely suboptimal according to another (standard) component of our setting, the *loss function*, which, in the examples above, might represent the expected revenue from showing an ad to an individual. Typically the loss function is not perfectly aligned with individual utilities, and, therefore, it may be possible to achieve smaller loss than the naïve solution without violating the envy-freeness constraint.

In summary, we view envy-freeness as a compelling, well-established, and, importantly, practicable notion of individual fairness for classification tasks with a diverse set of outcomes when individuals have heterogeneous preferences. Our goal is to understand its learning-theoretic properties.

5.1.1 Our Results

The challenge is that the space of individuals is potentially huge, yet we seek to provide universal envy-freeness guarantees. To this end, we are given a sample consisting of individuals drawn from an unknown distribution. We are interested in learning algorithms that minimize loss, subject to satisfying the envy-freeness constraint, *on the sample*. Our primary technical question is that of generalizability, that is, *given a classifier that is envy free on a sample, is it approximately envy free on the underlying distribution?* Surprisingly, Dwork et al. [62] do not study generalizability in their model, and we are aware of only one subsequent paper that takes a learning-theoretic viewpoint on individual fairness and gives theoretical guarantees (see Section 5.1.2).

In Section 5.3, we do not constrain the classifier. Therefore, we need some strategy to extend a classifier that is defined on a sample; assigning an individual the same outcome as his *nearest neighbor* in the sample is a popular choice. However, we show that *any* strategy for extending a classifier from a sample, on which it is envy free, to the entire set of individuals is unlikely to be approximately envy free on the distribution, unless the sample is exponentially large.

For this reason, in Section 5.4, we focus on structured families of classifiers. On a high level, our goal is to relate the combinatorial richness of the family to generalization guarantees. One obstacle is that standard notions of dimension do not extend to the analysis of randomized classifiers, whose range is *distributions* over outcomes (equivalently, real vectors). We circumvent this obstacle by considering mixtures of *deterministic* classifiers that belong to a family of bounded Natarajan dimension (an extension of the well-known VC dimension to multi-class classification). Our main theoretical result asserts that, under this assumption, envy-freeness on a sample does generalize to the underlying distribution, even if the sample is relatively small (its size grows almost linearly in the Natarajan dimension).

In the full version of the paper [19], we also design and implement an algorithm that learns (almost) envy-free mixtures of linear one-vs-all classifiers. These empirical results suggest that in practice, solving the empirical risk minimization problem subject to envy-freeness constraints is computationally tractible.

5.1.2 Related Work

Conceptually, our work is most closely related to work by Zafar et al. [150]. They are interested in group notions of fairness, and advocate preference-based notions instead of parity-based notions. In particular, they assume that each group has a utility function for *classifiers*, and define the *preferred treatment* property, which requires that the utility of each group for its own classifier be at least its utility for the classifier assigned to any other group. Their model and results focus on the case of binary classification where there is a desirable outcome and an undesirable outcome, so the utility of a group for a classifier is simply the fraction of its members that are mapped to the desirable outcome. Although, at first glance, this notion seems similar to envy-freeness, it is actually fundamentally different.[2] Our paper is also completely different from that of Zafar et al. in terms of technical results; theirs are purely empirical in nature, and focus on the increase in accuracy obtained when parity-based notions of fairness are replaced with preference-based ones.

[2]On a philosophical level, the fair division literature deals exclusively with individual notions of fairness. In fact, even in group-based extensions of envy-freeness [103] the allocation is shared by groups, but individuals must not be envious. We subscribe to the view that group-oriented notions (such as statistical parity) are objectionable, because the outcome can be patently unfair to individuals.

Concurrent work by Rothblum and Yona [123] provides generalization guarantees for the metric notion of individual fairness introduced by Dwork et al. [62], or, more precisely, for an approximate version thereof. There are two main differences compared to our work: first, we propose envy-freeness as an alternative notion of fairness that circumvents the need for a similarity metric. Second, they focus on randomized *binary* classification, which amounts to learning a real-valued function, and so are able to make use of standard Rademacher complexity results to show generalization. By contrast, standard tools do not directly apply in our setting. It is worth noting that several other papers provide generalization guarantees for notions of group fairness, but these are more distantly related to our work [151, 146, 58, 86, 82].

A very recent paper by Kim et al. [88] proposes a fairness notion called preference-informed individual fairness (PIIF) that is a relaxation of both the metric fairness of Dwork et al. [62] and the envy-freeness notion of this work. Informally, their fairness notion requires that a pair of similar individuals should receive similar treatment, but this constraint can be violated when the assigned outcomes only increase each individual's utility. In particular, similar individuals may receive very different treatment, provided they prefer that treatment. One of the main strengths of envy-freeness is that it does not require a suitable distance metric on individuals encoding our social expectations about when individuals should be treated similarly. In contrast, PIIF requires both a suitable similarity metric and knowledge of individual preferences. Like Dwork et al. [62], they focus on the problem of finding PIIF allocations for finite sets of individuals with a known distance metric and utilities and do not give algorithms for optimizing over parameterized allocation functions or generalization guarantees.

5.2 The Model

We assume that there is a space $\mathcal{X}$ of individuals, a finite space $\mathcal{Y}$ of outcomes, and a utility function $u : \mathcal{X} \times \mathcal{Y} \to [0, 1]$ encoding the preferences of each individual for the outcomes in $\mathcal{Y}$. In the advertising example, individuals are users, outcomes are advertisements, and the utility function reflects the benefit an individual derives from being shown a particular advertisement. For any distribution $p \in \Delta(\mathcal{Y})$ (where $\Delta(\mathcal{Y})$ is the set of distributions over $\mathcal{Y}$) we let $u(x, p) = \mathbb{E}_{y\sim p}[u(x, y)]$ denote individual x's expected utility for an outcome sampled from p. We refer to a function $h : \mathcal{X} \to \Delta(\mathcal{Y})$ as a *classifier*, even though it can return a distribution over outcomes.

5.2.1 Envy-Freeness

Roughly speaking, a classifier $h : \mathcal{X} \to \Delta(\mathcal{Y})$ is envy free if no individual prefers the outcome distribution of someone else over his own.

Definition 5.1. A classifier $h : \mathcal{X} \to \Delta(\mathcal{Y})$ is *envy free (EF)* on a set S of individuals if $u(x, h(x)) \geq u(x, h(x'))$ for all $x, x' \in S$. Similarly, h is (α, β)-*EF* with respect to a distribution P on $\mathcal{X}$ if

$$\Pr_{x,x'\sim P}\big(u(x, h(x)) < u(x, h(x')) - \beta\big) \leq \alpha.$$

Finally, h is (α, β)-*pairwise EF* on a set of pairs of individuals $S = \{(x_i, x_i')\}_{i=1}^n$ if

$$\frac{1}{n}\sum_{i=1}^{n} \mathbb{I}\{u(x_i, h(x_i)) < u(x_i, h(x_i')) - \beta\} \leq \alpha.$$

Any classifier that is EF on a sample S of individuals is also (α, β)-pairwise EF on any pairing of the individuals in S, for any $\alpha \geq 0$ and $\beta \geq 0$. The weaker pairwise EF condition is all that is required for our generalization guarantees to hold.

5.2.2 Optimization and Learning

Our formal learning problem can be stated as follows. Given sample access to an unknown distribution P over individuals $\mathcal{X}$ and their utility functions, and a known loss function $\ell : \mathcal{X} \times \mathcal{Y} \to [0, 1]$, find a classifier $h : \mathcal{X} \to \Delta(\mathcal{Y})$ that is (α, β)-EF with respect to P minimizing expected loss $\mathbb{E}_{x \sim P}[\ell(x, h(x))]$, where for $x \in \mathcal{X}$ and $p \in \Delta(\mathcal{Y})$, $\ell(x, p) = \mathbb{E}_{y \sim p}[\ell(x, y)]$.

We follow the empirical risk minimization (ERM) learning approach, i.e., we collect a sample of individuals drawn i.i.d from P and find an EF classifier with low loss on the sample. Formally, given a sample of individuals $S = \{x_1, \ldots, x_n\}$ and their utility functions $u_{x_i}(\cdot) = u(x_i, \cdot)$, we are interested in a classifier $h : S \to \Delta(\mathcal{Y})$ that minimizes $\sum_{i=1}^{n} \ell(x_i, h(x_i))$ among all classifiers that are EF on S.

Recall that we consider randomized classifiers that can assign a distribution over outcomes to each of the individuals. However, one might wonder whether the EF classifier that minimizes loss on a sample happens to always be deterministic. Or, at least, the optimal deterministic classifier on the sample might incur a loss that is very close to that of the optimal randomized classifier. If this were true, we could restrict ourselves to classifiers of the form $h : \mathcal{X} \to \mathcal{Y}$, which would be much easier to analyze. Unfortunately, it turns out that this is not the case. In fact, there could be an arbitrary (multiplicative) gap between the optimal randomized EF classifier and the optimal deterministic EF classifier. The intuition behind this is as follows. A deterministic classifier that has very low loss on the sample, but is not EF, would be completely discarded in the deterministic setting. On the other hand, a randomized classifier could take this loss-minimizing deterministic classifier and mix it with a classifier with high "negative envy", so that the mixture ends up being EF and at the same time has low loss. This is made concrete in Example D.1 in Appendix D.

5.3 Arbitrary Classifiers

An important (and typical) aspect of our learning problem is that the classifier h needs to provide an outcome distribution for every individual, not just those in the sample. For example, if h chooses advertisements for visitors of a website, the classifier should still apply when a new visitor arrives. Moreover, when we use the classifier for new individuals, it must continue to be EF. In this section, we consider two-stage approaches that first choose outcome distributions for the individuals in the sample, and then extend those decisions to the rest of $\mathcal{X}$.

In more detail, we are given a sample $S = \{x_1, \ldots, x_n\}$ of individuals and a classifier $h : S \to \Delta(\mathcal{Y})$ assigning outcome distributions to each individual. Our goal is to extend these assignments to a classifier $\overline{h} : \mathcal{X} \to \Delta(\mathcal{Y})$ that can be applied to new individuals as well. For example, h could be the loss-minimizing EF classifier on the sample S.

For this section, we assume that $\mathcal{X}$ is equipped with a distance metric d. Moreover, we assume in this section that the utility function u is L-Lipschitz on $\mathcal{X}$. That is, for every $y \in \mathcal{Y}$ and for all $x, x' \in \mathcal{X}$, we have $|u(x, y) - u(x', y)| \leq L \cdot d(x, x')$.

Under the foregoing assumptions, one natural way to extend the classifier on the sample to all of $\mathcal{X}$ is to assign new individuals the same outcome distribution as their nearest neighbor in the sample.

Formally, for a set $S \subset \mathcal{X}$ and any individual $x \in \mathcal{X}$, let $\mathrm{NN}_S(x) \in \arg\min_{x' \in S} d(x, x')$ denote the nearest neighbor of x in S with respect to the metric d (breaking ties arbitrarily). The following simple result (whose proof is relegated to Appendix D) establishes that this approach preserves envy-freeness in cases where the sample is exponentially large.

Theorem 5.1. *Let d be a metric on $\mathcal{X}$, P be a distribution on $\mathcal{X}$, and u be an L-Lipschitz utility function. Let S be a set of individuals such that there exists $\hat{\mathcal{X}} \subset \mathcal{X}$ with $P(\hat{\mathcal{X}}) \geq 1 - \alpha$ and $\sup_{x \in \hat{\mathcal{X}}}(d(x, \mathrm{NN}_S(x)) \leq \beta/(2L)$. Then for any classifier $h : S \to \Delta(\mathcal{Y})$ that is EF on S, the extension $\overline{h} : \mathcal{X} \to \Delta(\mathcal{Y})$ given by $\overline{h}(x) = h(\mathrm{NN}_S(x))$ is (α, β)-EF on P.*

The conditions of Theorem 5.1 require that the set of individuals S is a $\beta/(2L)$-net for at least a $(1 - \alpha)$-fraction of the mass of P on $\mathcal{X}$. In several natural situations, an exponentially large sample guarantees that this occurs with high probability. For example, if $\mathcal{X}$ is a subset of $\mathbb{R}^q$, $d(x, x') = \|x - x'\|_2$, and $\mathcal{X}$ has diameter at most D, then for any distribution P on $\mathcal{X}$, if S is an i.i.d. sample of size $O(\frac{1}{\alpha}(\frac{LD\sqrt{q}}{\beta})^q(q \log \frac{LD\sqrt{q}}{\beta} + \log \frac{1}{\delta}))$, it will satisfy the conditions of Theorem 5.1 with probability at least $1 - \delta$. This sampling result is folklore, but, for the sake of completeness, we prove it in Lemma D.1 of Appendix D.

However, the exponential upper bound given by the nearest neighbor strategy is as far as we can go in terms of generalizing envy-freeness from a sample (without further assumptions). In the full version of the paper, we also prove a lower bound showing that *any* algorithm — even randomized — for extending classifiers from the sample to the entire space $\mathcal{X}$ requires an exponentially large sample of individuals to ensure envy-freeness on the distribution P. The proof of Theorem 5.2 can be found in Appendix D.

Theorem 5.2. *There exists a space of individuals $\mathcal{X} \subset \mathbb{R}^q$, and a distribution P over $\mathcal{X}$ such that, for every randomized algorithm $\mathcal{A}$ that extends classifiers on a sample to $\mathcal{X}$, there exists an L-Lipschitz utility function u such that, when a sample of individuals S of size $n = 4^q/2$ is drawn from P without replacement, there exists an EF classifier on S for which, with probability at least $1 - 2\exp(-4^q/100) - \exp(-4^q/200)$ jointly over the randomness of $\mathcal{A}$ and S, its extension by $\mathcal{A}$ is not (α, β)-EF with respect to P for any $\alpha < 1/25$ and $\beta < L/8$.*

We remark that a similar result would hold even if we sampled S with replacement; we sample here without replacement purely for ease of exposition.

5.4 Low-Complexity Families of Classifiers

In this section we show that (despite Theorem 5.2) generalization for envy-freeness is possible using much smaller samples of individuals, as long as we restrict ourselves to classifiers from a family of relatively low complexity.

In more detail, two classic complexity measures are the VC-dimension [141] for binary classifiers, and the Natarajan dimension [113] for multi-class classifiers. However, to the best of our knowledge, there is no suitable dimension directly applicable to functions ranging over distributions, which in our case can be seen as $|\mathcal{Y}|$-dimensional real vectors. One possibility would be to restrict ourselves to deterministic classifiers of the type $h : \mathcal{X} \to \mathcal{Y}$. However, we have seen in Section 5.2 that envy-freeness is a very strong constraint on deterministic classifiers. Instead, we will consider a family $\mathcal{H}$ consisting of randomized mixtures of deterministic classifiers belonging to a family $\mathcal{G} \subset \{g : \mathcal{X} \to \mathcal{Y}\}$ of low Natarajan dimension. This allows us to adapt Natarajan-dimension-based generalization results to our setting while still working with randomized classifiers.

5.4.1 Natarajan Dimension Primer

Before presenting our main result, we briefly summarize the definition and relevant properties of the Natarajan dimension. For more details, we refer the reader to [128].

We say that a family $\mathcal{G}$ *multi-class shatters* a set of points $x_1, \ldots, x_n$ if there exist labels $y_1, \ldots y_n$ and $y'_1, \ldots, y'_n$ such that for every $i \in [n]$ we have $y_i \neq y'_i$, and for any subset $C \subset [n]$ there exists $g \in \mathcal{G}$ such that $g(x_i) = y_i$ if $i \in C$ and $g(x_i) = y'_i$ otherwise. The Natarajan dimension of a family $\mathcal{G}$ is the cardinality of the largest set of points that can be multi-class shattered by $\mathcal{G}$.

For example, suppose we have a feature map $\Psi : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^q$ that maps each individual-outcome pair to a q-dimensional feature vector, and consider the family of functions that can be written as $g(x) = \arg\max_{y \in \mathcal{Y}} w^\top \Psi(x, y)$ for weight vectors $w \in \mathbb{R}^q$. This family has Natarajan dimension at most q.

For a set $S \subset \mathcal{X}$ of points, we let $\mathcal{G}\big|_S$ denote the restriction of $\mathcal{G}$ to S, which is any subset of $\mathcal{G}$ of minimal size such that for every $g \in \mathcal{G}$ there exists $g' \in \mathcal{G}\big|_S$ such that $g(x) = g'(x)$ for all $x \in S$. The size of $\mathcal{G}\big|_S$ is the number of different labelings of the sample S achievable by functions in $\mathcal{G}$. The following Lemma is the analogue of Sauer's lemma for binary classification.

Lemma 5.1 (Natarajan)**.** *For a family $\mathcal{G}$ of Natarajan dimension d and any subset $S \subset \mathcal{X}$, we have $\left|\mathcal{G}\big|_S\right| \leq |S|^d |\mathcal{Y}|^{2d}$.*

Classes of low Natarajan dimension also enjoy the following uniform convergence guarantee.

Lemma 5.2. *Let $\mathcal{G}$ have Natarajan dimension d and fix a loss function $\ell : \mathcal{G} \times \mathcal{X} \to [0, 1]$. For any distribution P over $\mathcal{X}$, if S is an i.i.d. sample drawn from P of size $O(\frac{1}{\epsilon^2}(d \log |\mathcal{Y}| + \log \frac{1}{\delta}))$, then with probability at least $1 - \delta$ we have $\sup_{g \in \mathcal{G}} \left|\mathbb{E}_{x \sim P}[\ell(g, x)] - \frac{1}{n} \sum_{x \in S} \ell(g, x)\right| \leq \epsilon$.*

5.4.2 Main Result

We consider the family of classifiers that can be expressed as a randomized mixture of m deterministic classifiers selected from a family $\mathcal{G} \subset \{g : \mathcal{X} \to \mathcal{Y}\}$. Our generalization guarantees will depend on the complexity of the family $\mathcal{G}$, measured in terms of its Natarajan dimension, and the number m of functions we are mixing. More formally, let $\boldsymbol{g} = (g_1, \ldots, g_m) \in \mathcal{G}^m$ be a vector of m functions in $\mathcal{G}$ and $\eta \in \Delta_m$ be a distribution over $[m]$, where $\Delta_m = \{p \in \mathbb{R}^m : p_i \geq 0, \sum_i p_i = 1\}$ is the m-dimensional probability simplex. Then consider the function $h_{\boldsymbol{g},\eta} : \mathcal{X} \to \Delta(\mathcal{Y})$ with assignment probabilities given by $\Pr(h_{\boldsymbol{g},\eta}(x) = y) = \sum_{i=1}^m \mathbb{I}\{g_i(x) = y\}\eta_i$. Intuitively, for a given individual x, $h_{\boldsymbol{g},\eta}$ chooses one of the g_i randomly with probability η_i, and outputs $g_i(x)$. Let

$$\mathcal{H}(\mathcal{G}, m) = \{h_{\boldsymbol{g},\eta} : \mathcal{X} \to \Delta(\mathcal{Y}) : \boldsymbol{g} \in \mathcal{G}^m, \eta \in \Delta_m\}$$

be the family of classifiers that can be written this way. Our main technical result shows that envy-freeness generalizes for this class.

Theorem 5.3. *Suppose $\mathcal{G}$ is a family of deterministic classifiers of Natarajan dimension d, and let $\mathcal{H} = \mathcal{H}(\mathcal{G}, m)$ for $m \in \mathbb{N}$. For any distribution P over $\mathcal{X}$, $\gamma > 0$, and $\delta > 0$, if $S = \{(x_i, x'_i)\}_{i=1}^n$ is an i.i.d. sample of pairs drawn from P of size*

$$n \geq O\left(\frac{1}{\gamma^2}\left(dm^2 \log \frac{dm|\mathcal{Y}| \log(m|\mathcal{Y}|/\gamma)}{\gamma} + \log \frac{1}{\gamma}\right)\right),$$

then with probability at least $1 - \delta$, every classifier $h \in \mathcal{H}$ that is (α, β)-pairwise-EF on S is also $(\alpha + 7\gamma, \beta + 4\gamma)$-EF on P.

Theorem 5.3 is only effective insofar as families of classifiers of low Natarajan dimension are useful. And, indeed, several prominent families have low Natarajan dimension [49], including one vs. all (which is a special case of the example given in Section 5.4.1), multiclass SVM, tree-based classifiers, and error correcting output codes.

We now turn to the theorem's proof, which consists of two steps. First, we show that envy-freeness generalizes for finite classes. Second, we show that $\mathcal{H}(\mathcal{G}, m)$ can be approximated by a finite subset.

Lemma 5.3. *Let $\mathcal{H} \subset \{h : \mathcal{X} \to \Delta(\mathcal{Y})\}$ be a finite family of classifiers. For any $\gamma > 0$, $\delta > 0$, and $\beta \geq 0$ if $S = \{(x_i, x_i')\}_{i=1}^n$ is an i.i.d. sample of pairs from P of size $n \geq \frac{1}{2\gamma^2} \ln \frac{|\mathcal{H}|}{\delta}$, then with probability at least $1 - \delta$, every $h \in \mathcal{H}$ that is (α, β)-pairwise-EF on S (for any α) is also $(\alpha + \gamma, \beta)$-EF on P.*

Proof. Let $f(x, x', h) = \mathbb{I}\{u(x, h(x)) < u(x, h(x')) - \beta\}$ be the indicator that x is envious of x' by at least β under classifier h. Then $f(x_i, x_i', h)$ is a Bernoulli random variable with success probability $\mathbb{E}_{x,x' \sim P}[f(x, x', h)]$. Applying Hoeffding's inequality to any fixed hypothesis $h \in \mathcal{H}$ guarantees that $\Pr_S(\mathbb{E}_{x,x' \sim P}[f(x, x', h)] \geq \frac{1}{n} \sum_{i=1}^n f(x_i, x_i', h) + \gamma) \leq \exp(-2n\gamma^2)$. Therefore, if h is (α, β)-EF on S, then it is also $(\alpha + \gamma, \beta)$-EF on P with probability at least $1 - \exp(-2n\gamma^2)$. Applying the union bound over all $h \in \mathcal{H}$ and using the lower bound on n completes the proof. □

Next, we show that $\mathcal{H}(\mathcal{G}, m)$ can be covered by a finite subset. Since each classifier in $\mathcal{H}$ is determined by the choice of m functions from $\mathcal{G}$ and mixing weights $\eta \in \Delta_m$, we will construct finite covers of $\mathcal{G}$ and Δ_m. Our covers $\hat{\mathcal{G}}$ and $\hat{\Delta}_m$ will guarantee that for every $g \in \mathcal{G}$, there exists $\hat{g} \in \hat{\mathcal{G}}$ such that $\Pr_{x \sim P}(g(x) \neq \hat{g}(x)) \leq \gamma/m$. Similarly, for any mixing weights $\eta \in \Delta_m$, there exists $\hat{\eta} \in \Delta_m$ such that $\|\eta - \hat{\eta}\|_1 \leq \gamma$. If $h \in \mathcal{H}(\mathcal{G}, m)$ is the mixture of $g_1, \ldots, g_m$ with weights η, we let $\hat{h}$ be the mixture of $\hat{g}_1, \ldots, \hat{g}_m$ with weights $\hat{\eta}$. This approximation has two sources of error: first, for a random individual $x \sim P$, there is probability up to γ that at least one $g_i(x)$ will disagree with $\hat{g}_i(x)$, in which case h and $\hat{h}$ may assign completely different outcome distributions. Second, even in the high-probability event that $g_i(x) = \hat{g}_i(x)$ for all $i \in [m]$, the mixing weights are not identical, resulting in a small perturbation of the outcome distribution assigned to x.

Lemma 5.4. *Let $\mathcal{G}$ be a family of deterministic classifiers with Natarajan dimension d, and let $\mathcal{H} = \mathcal{H}(\mathcal{G}, m)$ for some $m \in \mathbb{N}$. For any $\gamma > 0$, there exists a subset $\hat{\mathcal{H}} \subset \mathcal{H}$ of size $O\big(\frac{(dm|\mathcal{Y}|^2 \log(m|\mathcal{Y}|/\gamma))^{dm}}{\gamma^{(d+1)m}}\big)$ such that for every $h \in \mathcal{H}$ there exists $\hat{h} \in \mathcal{H}$ satisfying:*

1. $\Pr_{x \sim P}(\|h(x) - \hat{h}(x)\|_1 > \gamma) \leq \gamma$.
2. *If S is an i.i.d. sample of individuals of size $O(\frac{m^2}{\gamma^2}(d \log |\mathcal{Y}| + \log \frac{1}{\delta}))$ then w.p. $\geq 1 - \delta$, we have $\|h(x) - \hat{h}(x)\|_1 \leq \gamma$ for all but a 2γ-fraction of $x \in S$.*

Proof. As described above, we begin by constructing finite covers of Δ_m and $\mathcal{G}$. First, let $\hat{\Delta}_m \subset \Delta_m$ be the set of distributions over $[m]$ where each coordinate is a multiple of γ/m. Then we have $|\hat{\Delta}_m| = O((\frac{m}{\gamma})^m)$ and for every $p \in \Delta_m$, there exists $q \in \hat{\Delta}_m$ such that $\|p - q\|_1 \leq \gamma$.

In order to find a small cover of $\mathcal{G}$, we use the fact that it has low Natarajan dimension. This implies that the number of effective functions in $\mathcal{G}$ when restricted to a sample S' grows only polynomially in the size of S'. At the same time, if two functions in $\mathcal{G}$ agree on a large sample, they will also agree with high probability on the distribution.

Formally, let S' be an i.i.d. sample drawn from P of size $O(\frac{m^2}{\gamma^2} d \log |\mathcal{Y}|)$, and let $\hat{\mathcal{G}} = \mathcal{G}\big|_{S'}$ be any minimal subset of $\mathcal{G}$ that realizes all possible labelings of S' by functions in $\mathcal{G}$. We now argue that with

probability 0.99, for every $g \in \mathcal{G}$ there exists $\hat{g} \in \hat{\mathcal{G}}$ such that $\Pr_{x\sim P}(g(x) \neq \hat{g}(x)) \leq \gamma/m$. For any pair of functions $g, g' \in \mathcal{G}$, let $(g, g') : \mathcal{X} \to \mathcal{Y}^2$ be the function given by $(g, g')(x) = (g(x), g'(x))$, and let $\mathcal{G}^2 = \{(g, g') : g, g' \in \mathcal{G}\}$. The Natarajan dimension of $\mathcal{G}^2$ is at most $2d$ (see Lemma D.2 below). Moreover, consider the loss $c : \mathcal{G}^2 \times \mathcal{X} \to \{0, 1\}$ given by $c(g, g', x) = \mathbb{I}\{g(x) \neq g'(x)\}$. Applying Lemma 5.2 with the chosen size of $|S'|$ ensures that with probability at least 0.99 every pair $(g, g') \in \mathcal{G}^2$ satisfies

$$\left| \mathop{\mathbb{E}}_{x\sim P}[c(g, g', x)] - \frac{1}{|S'|} \sum_{x\in S'} c(g, g', x) \right| \leq \frac{\gamma}{m}.$$

By the definition of $\hat{\mathcal{G}}$, for every $g \in \mathcal{G}$, there exists $\hat{g} \in \hat{\mathcal{G}}$ for which $c(g, \hat{g}, x) = 0$ for all $x \in S'$, which implies that $\Pr_{x\sim P}(g(x) \neq \hat{g}(x)) \leq \gamma/m$.

Using Lemma 5.1 to bound the size of $\hat{\mathcal{G}}$, we have that

$$|\hat{\mathcal{G}}| \leq |S'|^d |\mathcal{Y}|^{2d} = O\left(\left(\frac{m^2}{\gamma^2} d|\mathcal{Y}|^2 \log|\mathcal{Y}|\right)^d\right).$$

Since this construction succeeds with non-zero probability, we are guaranteed that such a set $\hat{\mathcal{G}}$ exists. Finally, by an identical uniform convergence argument, it follows that if S is a fresh i.i.d. sample of the size given in Item 2 of the lemma's statement, then, with probability at least $1 - \delta$, every g and $\hat{g}$ will disagree on at most a $2\gamma/m$-fraction of S, since they disagree with probability at most γ/m on P.

Next, let $\hat{\mathcal{H}} = \{h_{\boldsymbol{g},\eta} : \boldsymbol{g} \in \hat{G}^m, \eta \in \hat{\Delta}_m\}$ be the same family as $\mathcal{H}$, except restricted to choosing functions from $\hat{\mathcal{G}}$ and mixing weights from $\hat{\Delta}_m$. Using the size bounds above and the fact that $\binom{N}{m} = O((\frac{N}{m})^m)$, we have that

$$|\hat{\mathcal{H}}| = \binom{|\hat{\mathcal{G}}|}{m} \cdot |\hat{\Delta}_m| = O\left(\frac{(dm^2|\mathcal{Y}|^2 \log(m|\mathcal{Y}|/\gamma))^{dm}}{\gamma^{(2d+1)m}}\right).$$

Suppose that h is the mixture of $g_1, \ldots, g_m \in \mathcal{G}$ with weights $\eta \in \Delta_m$. Let $\hat{g}_i$ be the approximation to g_i for each i, let $\hat{\eta} \in \hat{\Delta}_m$ be such that $\|\eta - \hat{\eta}\|_1 \leq \gamma$, and let $\hat{h}$ be the random mixture of $\hat{g}_1, \ldots, \hat{g}_m$ with weights $\hat{\eta}$. For an individual x drawn from P, we have $g_i(x) \neq \hat{g}_i(x)$ with probability at most γ/m, and therefore they all agree with probability at least $1 - \gamma$. When this event occurs, we have $\|h(x) - \hat{h}(x)\|_1 \leq \|\eta - \hat{\eta}\|_1 \leq \gamma$.

The second part of the claim follows by similar reasoning, using the fact that for the given sample size $|S|$, with probability at least $1-\delta$, every $g \in \mathcal{G}$ disagrees with its approximation $\hat{g} \in \hat{\mathcal{G}}$ on at most a $2\gamma/m$-fraction of S. This means that $\hat{g}_i(x) = g_i(x)$ for all $i \in [m]$ on at least a $(1 - 2\gamma)$-fraction of the individuals x in S. For these individuals, $\|h(x) - \hat{h}(x)\|_1 \leq \|\eta - \hat{\eta}\|_1 \leq \gamma$. □

Combining the generalization guarantee for finite families given in Lemma 5.3 with the finite approximation given in Lemma 5.4, we are able to show that envy-freeness also generalizes for $\mathcal{H}(\mathcal{G}, m)$.

Proof of Theorem 5.3. Let $\hat{\mathcal{H}}$ be the finite approximation to $\mathcal{H}$ constructed in Lemma 5.4. If the sample is of size $|S| = O(\frac{1}{\gamma^2}(dm \log(dm|\mathcal{Y}| \log|\mathcal{Y}|/\gamma) + \log\frac{1}{\delta}))$, we can apply Lemma 5.3 to this finite family, which implies that for any $\beta' \geq 0$, with probability at least $1 - \delta/2$ every $\hat{h} \in \hat{\mathcal{H}}$ that is (α', β')-pairwise-EF on S (for any α') is also $(\alpha' + \gamma, \beta')$-EF on P. We apply this lemma with

$\beta' = \beta + 2\gamma$. Moreover, from Lemma 5.4, we know that if $|S| = O(\frac{m^2}{\gamma^2}(d \log |\mathcal{Y}| + \log \frac{1}{\delta}))$, then with probability at least $1 - \delta/2$, for every $h \in \mathcal{H}$, there exists $\hat{h} \in \hat{\mathcal{H}}$ satisfying $\|h(x) - \hat{h}(x)\|_1 \leq \gamma$ for all but a 2γ-fraction of the individuals in S. This implies that on all but at most a 4γ-fraction of the pairs in S, h and $\hat{h}$ satisfy this inequality for both individuals in the pair. Assume these high probability events occur. Finally, from Item 1 of the lemma we have that $\Pr_{x_1,x_2\sim P}(\max_{i=1,2} \|h(x_i) - \hat{h}(x_i)\|_1 > \gamma) \leq 2\gamma$.

Now let $h \in \mathcal{H}$ be any classifier that is (α, β)-pairwise-EF on S. Since the utilities are in $[0, 1]$ and $\max_{x=x_i,x_i'} \|h(x) - \hat{h}(x)\|_1 \leq \gamma$ for all but a 4γ-fraction of the pairs in S, we know that $\hat{h}$ is $(\alpha+4\gamma, \beta+2\gamma)$-pairwise-EF on S. Applying the envy-freeness generalization guarantee (Lemma 5.3) for $\hat{\mathcal{H}}$, it follows that $\hat{h}$ is also $(\alpha + 5\gamma, \beta + 2\gamma)$-EF on P. Finally, using the fact that

$$\Pr_{x_1,x_2\sim P}\left(\max_{i=1,2} \|h(x_i) - \hat{h}(x_i)\|_1 > \gamma\right) \leq 2\gamma,$$

it follows that h is $(\alpha + 7\gamma, \beta + 4\gamma)$-EF on P. □

It is worth noting that the (exponentially large) approximation $\hat{\mathcal{H}}$ is only used in the generalization analysis; importantly, an ERM algorithm need not construct it. Also note that the number of outcomes $|\mathcal{Y}|$ only appears in the logarithmic terms of the sample complexity bounds, allowing these results to handle very large outcome spaces, provided that we choose a suitable family $\mathcal{G}$ of deterministic classifiers.

Bibliography

[1] E. Allwein, R. Schapire, and Y. Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. In *Journal of Machine Learning Research*, 2000. 2.2

[2] Noga Alon, Nicolò Cesa-Bianchi, Claudio Gentile, Shie Mannor, Yishay Mansour, and Ohad Shamir. Nonstochastic multi-armed bandits with graph-structured feedback. *SIAM J. Comput.*, 46(6):1785–1826, 2017. 3.2.1, 3.2.2, 3.2.2, B.2.1, B.2.5, B.2.5

[3] Martin Anthony and Peter Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, 2009. B.23

[4] Peter Auer, Nicoló Cesa-Bianchi, Yoav Freund, and Robert Shapire. The nonstochastic multi-armed bandit problem. In *SIAM Journal on Computing*, 2003. 3.1.3

[5] Pranjal Awasthi, Avrim Blum, and Or Sheffet. Center-based clustering under perturbation stability. In *Information Processing Letters*, 2012. 3.2.1

[6] Pranjal Awasthi, Maria-Florina Balcan, and Konstantin Voevodski. Local algorithms for interactive clustering. In *ICML*, 2014. 3.2.1

[7] K. Balasubramanian, P. Donmez, and G. Lebanon. Unsupervised supervised learning ii: Margin-based classification without labels. In *AISTATS*, pages 137–145, 2011. 2.2

[8] K. Balasubramanian, P. Donmez, and G. Lebanon. Unsupervised supervised learning ii: Margin-based classification without labels. In *Journal of Machine Learning Research*, volume 12, pages 3119–3145, 2011. 2.2

[9] M-F. Balcan and A. Blum. A discriminative model for semi-supervised learning. In *Journal of the ACM*, 2010. 2.2

[10] M-F. Balcan and R. Urner. Active learning. In *Survey in the Encyclopedia of Algorithms*, 2015. 2.2

[11] M-F. Balcan, A. Blum, and K. Yang. Co-training and expansion: Towards bridging theory and practice. In *NeurIPS*, 2004. 2.2

[12] M-F. Balcan, A. Beygelzimer, and J. Lanford. Agnostic active learing. In *ICML*, 2006. 2.2

[13] M.-F. Balcan, F. Constantin, S. Iwata, and L. Wang. Learning valuation functions. In *25th Proceedings of the Conference on Learning Theory (COLT)*, pages 4.1–4.24, 2012. 5.1

[14] M.-F. Balcan, A. Blum, and Y. Mansour. Exploiting ontology structures and unlabeled data for learning. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, pages 1112–1120, 2013. 2.2

[15] Maria-Florina Balcan and Yingyu Liang. Clustering under perturbation resilience. In *SIAM Journal on Computing*, 2016. 3.2.1

[16] Maria-Florina Balcan, Nika Haghtalab, and Colin White. k-center clustering under perturbation resilience. In *Proceedings of the Annual International Colloquium on Automata, Languages, and Programming (ICALP)*, 2016. 3.2.1

[17] Maria-Florina Balcan, Tuomas Sandholm, and Ellen Vitercik. Sample complexity of automated mechanism design. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2016. 3.1.1, 3.1.1

[18] Maria-Florina Balcan, Vaishnavh Nagarajan, Ellen Vitercik, and Colin White. Learning-theoretic foundations of algorithm configuration for combinatorial partitioning problems. *Proceedings of the Conference on Learning Theory (COLT)*, 2017. 3.1.1, 3.1.5, 3.2.1, 3.2.1, 3.2.1, 3.2.4, 3.2.4, 4.1, 4.2, 4.3, 4.3, 4.4.1, B.1.6, B.22, B.1.6, B.1.6, B.30, B.2.3

[19] Maria-Florina Balcan, Travis Dick, Ritesh Noothigattu, and Ariel D. Procaccia. Envy-free classification. *CoRR*, abs/1809.08700, 2018. URL `http://arxiv.org/abs/1809.08700`. 5.1.1

[20] Maria-Florina Balcan, Travis Dick, Tuomas Sandholm, and Ellen Vitercik. Learning to branch. *ICML*, 2018. 4.1, 4.3

[21] Maria-Florina Balcan, Travis Dick, Tuomas Sandholm, and Ellen Vitercik. Learning to branch. In *ICML*, 2018. 3.2.1, 3.2.1

[22] Maria-Florina Balcan, Travis Dick, and Ellen Vitercik. Dispersion for data-driven algorithm design, online learning, and private optimization. In *FOCS*, 2018. 3.1.1, 3.2.1, 3.2.1, 3.2.1, 3.2.1, 3.2.1, 3.2.1, 3.2.2, 3.2.3, 3.2.4, 3.2.4, B.2.2, B.35, B.36

[23] Maria-Florina Balcan, Travis Dick, and Colin White. Data-driven clustering via parameterized lloyd's families. In *NeurIPS*, 2018. 3.2.1, 3.2.1, 4.1

[24] Maria-Florina Balcan, Tuomas Sandholm, and Ellen Vitercik. A general theory of sample complexity for multi-item profit maximization. *Proceedings of the ACM Conference on Economics and Computation (EC)*, 2018. 3.1.1, 3.1.1, 3.2.1

[25] Peter L Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3(Nov):463–482, 2002. 3.1.6, B.1.1

[26] Raef Bassily, Adam Smith, and Abhradeep Thakurta. Differentially private empirical risk minimization: Efficient algorithms and tight error bounds. In *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS)*, 2014. 3.1.1, 3.1.1, 3.1.3, B.1.3, B.1.3, B.1

[27] Shai Ben-David, David Pal, and Shai Shalev-Shwartz. Agnostic online learning. In *COLT*, 2009. 3.2.1

[28] A. Berger. Error-correcting output coding for text classification. In *IJCAI Workshop on machine learning for information filtering*, 1999. 2.8

[29] A. Beygelzimer, J. Langford, and P. Ravikumar. Error-correcting tournaments. *ALT*, 2009. 2.1, 2.2

[30] Francesca Biagini and Massimo Campanino. *Elements of Probability and Statistics: An Introduction to Probability with de Finetti's Approach and to Bayesian Statistics*, volume 98. Springer, 2016. B.1.6

[31] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *COLT*, 1998. 2.2

[32] Avrim Blum and Jason D. Hartline. Near-optimal online auctions. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1156–1163. Society for Industrial and Applied Mathematics, 2005. 3.1.1, 3.1.1

[33] Avrim Blum, Vijay Kumar, Atri Rudra, and Felix Wu. Online learning in online auctions. *Theoretical Computer Science*, 324(2-3):137–146, 2004. 3.1.1, 3.1.1

[34] S. J. Brams and A. D. Taylor. *Fair Division: From Cake-Cutting to Dispute Resolution*. Cambridge University Press, 1996. 5.1

[35] Sébastien Bubeck, Nikhil R Devanur, Zhiyi Huang, and Rad Niazadeh. Online auctions and multi-scale online learning. *Proceedings of the ACM Conference on Economics and Computation (EC)*, 2017. 3.1.1, 3.1.1, 3.1.1

[36] Yves Caseau, François Laburthe, and Glenn Silverstein. A meta-heuristic factory for vehicle routing problems. In *International Conference on Principles and Practice of Constraint Programming*, pages 144–158. Springer, 1999. 3.2.1

[37] V. Castelli and T. Cover. On the exponential value of labeled samples. In *Pattern Recognition Letters*, 1995. 2.8

[38] Nicolo Cesa-Bianchi and Gábor Lugosi. *Prediction, learning, and games*. Cambridge university press, 2006. 3.1.1

[39] Nicoló Cesa-Bianchi, Alex Conconi, and Claudio Gentile. On the generalization ability of on-line learning algorithms. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 359–366, 2002. 3.1.1, 3.1.6

[40] Nicolo Cesa-Bianchi, Claudio Gentile, and Yishay Mansour. Regret minimization for reserve prices in second-price auctions. *IEEE Transactions on Information Theory*, 61(1):549–564, 2015. 3.1.1, 3.1.1

[41] U. Chajewska, D. Koller, and D. Ormoneit. Learning an agent's utility function by observing behavior. In *18th Proceedings of the International Conference on Machine Learning (ICML)*, pages 35–42, 2001. 5.1

[42] O. Chapelle, B. Schlkopf, and A. Zien. *Semi-Supervised Learning*. The MIT Press, 1st edition, 2010. ISBN 0262514125, 9780262514125. 2.2

[43] Moses Charikar and Anthony Wirth. Maximizing quadratic programs: extending Grothendieck's inequality. In *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS)*, 2004. 3.1.1, B.1.6

[44] K. Chaudhuri and S Dasgupta. Rates of convergence for the cluster tree. In *Advances in Neural Information Processing 23 (NeurIPS)*, pages 343–351, 2010. 2.5, 2.5

[45] Kamalika Chaudhuri, Claire Monteleoni, and Anand D Sarwate. Differentially private empirical risk minimization. *Journal of Machine Learning Research*, 12(Mar):1069–1109, 2011. 3.1.1

[46] Vincent Cohen-Addad and Varun Kanade. Online Optimization of Smoothed Piecewise Constant Functions. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2017. 3.1.1, 3.1.1, 2, 3.1.3, 3.2.1, 3.2.1, 3.2.1, B.1.4, B.1, B.2.1, B.2.1, B.2.1

[47] Richard Cole and Tim Roughgarden. The sample complexity of revenue maximization. In *Proceedings of the Annual Symposium on Theory of Computing (STOC)*, 2014. 3.1.1, 3.1.1

[48] K. Crammer and Y. Singer. Improved output coding for classification using continuous relaxation. In *NeurIPS*, 2000. 2.8

[49] A. Daniely, S. Sabato, and S. Shalev-Shwartz. Multiclass learning approaches: A theoretical comparison with implications. In *25th Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 485–493, 2012. 5.4.2

[50] A. Daniely, M. Schapira, and G. Shahaf. Multiclass learning approaches: A theoretical comparison with implications. In *NeurIPS*, 2012. 2.2

[51] S. Dasgupta. Two faces of active learning. In *Theoretical Computer Science*, 2011. 2.2

[52] S. Dasgupta and K. Sinha. Randomized partition trees for exact nearest neighbor search. In *COLT*, 2013. 2.4

[53] A. Datta, M. C. Tschantz, and A. Datta. Automated experiments on ad privacy settings: A tale of opacity, choice, and discrimination. In *15th*, pages 92–112, 2015. 5.1

[54] Anindya De. Lower bounds in differential privacy. In *Proceedings of the Theory of Cryptography Conference (TCC)*, pages 321–338, 2012. B.1.5

[55] Jim Demmel, Jack Dongarra, Victor Eijkhout, Erika Fuentes, Antoine Petitet, Rich Vuduc, R Clint Whaley, and Katherine Yelick. Self-adapting linear algebra algorithms and software. *Proceedings of the IEEE*, 93(2):293–312, 2005. 3.2.1

[56] Nikhil R Devanur, Zhiyi Huang, and Christos-Alexandros Psomas. The sample complexity of auctions with side information. In *Proceedings of the Annual Symposium on Theory of Computing (STOC)*, 2016. 3.1.1, 3.1.1

[57] T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, pages 263–286, 1995. 2.1, 2.1, 2.2

[58] M. Donini, L. Oneto, S. Ben-David, J. Shawe-Taylor, and M. Pontil. Empirical Risk Minimization under Fairness Constraints. arXiv:1802.08626, 2018. 5.1.2

[59] P. Donmez, G. Lebanon, and K. Balasubramanian. Unsupervised supervised learning i: Estimating classification and regression errors without labels. In *Journal of Machine Learning Research*, volume 11, pages 1323–1351, 2010. 2.2

[60] Miroslav Dudík, Nika Haghtalab, Haipeng Luo, Robert E Schapire, Vasilis Syrgkanis, and Jennifer Wortman Vaughan. Oracle-efficient learning and auction design. *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS)*, 2017. 3.1.1, 3.1.1

[61] R. M. Dudley. The sizes of compact subsets of Hilbert space and continuity of Gaussian processes. *Journal of Functional Analysis*, 1(3):290 – 330, 1967. B.1.1, B.1.7

[62] C. Dwork, M. Hardt, T. Pitassi, O. Reingold, and R. S. Zemel. Fairness through awareness. In *3rd Proceedings of the ACM Conference on Innovations in Theoretical Computer Science (ITCS)*, pages 214–226, 2012. 5.1, 5.1.1, 5.1.2

[63] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(34):211–407, 2014. B.1.5

[64] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Proceedings of the Theory of Cryptography Conference (TCC)*, pages 265–284. Springer, 2006. 3.1.1

[65] Cynthia Dwork, Guy N Rothblum, and Salil Vadhan. Boosting and differential privacy. In *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS)*, 2010. B.15

[66] Edith Elkind. Designing and learning optimal finite support auctions. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2007. 3.1.1, 3.1.1

[67] Uriel Feige and Michael Langberg. The RPR^2 rounding technique for semidefinite programs. *Journal of Algorithms*, 60(1):1–23, 2006. 3.1.1, 3.1.5

[68] Zhe Feng, Chara Podimata, and Vasilis Syrgkanis. Learning to bid without knowing your value. *Proceedings of the ACM Conference on Economics and Computation (EC)*, 2018. 3.1.1

[69] D. Foley. Resource allocation and the public sector. *Yale Economics Essays*, 7:45–98, 1967. 5.1

[70] R. Freedman, J. Schaich Borg, W. Sinnott-Armstrong, J. P. Dickerson, and V. Conitzer. Adapting a kidney exchange algorithm to align with human values. In *32nd Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1636–1645, 2018. 5.1

[71] Y. Gal, M. Mash, A. D. Procaccia, and Y. Zick. Which is the fairest (rent division) of them all? *Journal of the ACM*, 64(6): article 39, 2017. 5.1

[72] Michel X Goemans and David P Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 42(6):1115–1145, 1995. 3.1.1, 3.1.5

[73] Kira Goldner and Anna R Karlin. A prior-independent revenue-maximizing auction for multiple additive bidders. In *Proceedings of the Conference on Web and Internet Economics (WINE)*, 2016. 3.1.1

[74] Yannai A Gonczarowski and Noam Nisan. Efficient empirical revenue maximization in single-parameter auction environments. In *Proceedings of the Annual Symposium on Theory of Computing (STOC)*, pages 856–868, 2017. 3.1.1, 3.1.1

[75] Robert D Gordon. Values of Mills' ratio of area to bounding ordinate and of the normal probability integral for large values of the argument. *The Annals of Mathematical Statistics*, 12(3):364–366, 1941. B.27

[76] Anna Grosswendt and Heiko Roeglin. Improved analysis of complete linkage clustering. In *European Symposium of Algorithms*, 2015. 3.2.1

[77] Rishi Gupta and Tim Roughgarden. A PAC approach to application-specific algorithm selection. *SIAM Journal on Computing*, 46(3):992–1017, 2017. 3.1.1, 3.1.1, 3.1.1, 3.1.1, 3.1.1, 2, 3.1.5, 3.1.5, 3.2.1, 3.2.1, 3.2.1, 3.2.4, 3.2.4, 4.1, B.18, B.1.6, B.19, B.1.6, B.20, B.21

[78] András György, Tamás Linder, and György Ottucsák. The shortest path problem under partial monitoring. In *COLT*, 2006. 3.2.1

[79] S. Hanneke. Theory of active learning. *Foundations and Trends in Machine Learning*, 7(2–3), 2014. 2.2

[80] M. Hardt, E. Price, and N. Srebro. Equality of opportunity in supervised learning. In *30th Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 3315–3323, 2016. 5.1

[81] Xinran He, Junfeng Pan, Ou Jin, Tianbing Xu, Bo Liu, Tao Xu, Yanxin Shi, Antoine Atallah, Ralf Herbrich, Stuart Bowers, et al. Practical lessons from predicting clicks on ads at Facebook. In *Proceedings of the International Workshop on Data Mining for Online Advertising*, 2014. 3.1.1

[82] Ú. Hébert-Johnson, M. P. Kim, O. Reingold, and G. N. Rothblum. Calibration for the (computationally-identifiable) masses. In *35th Proceedings of the International Conference on Machine Learning (ICML)*, 2018. Forthcoming. 5.1.2

[83] Zhiyi Huang, Yishay Mansour, and Tim Roughgarden. Making the most of your samples. In *Proceedings of the ACM Conference on Economics and Computation (EC)*, 2015. 3.1.1, 3.1.1

[84] M. Joseph, M. Kearns, J. Morgenstern, and A. Roth. Fairness in learning: Classic and contextual bandits. In *30th Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 325–333, 2016. 5.1

[85] Satyen Kale, Lev Reyzin, and Robert E. Shapire. Non-stochastic bandit slate problems. In *NeurIPS*, 2010. 3.2.1

[86] M. Kearns, S. Neel, A. Roth, and S. Wu. Computing parametric ranking models via rank-breaking. In *35th Proceedings of the International Conference on Machine Learning (ICML)*, 2018. 5.1.2

[87] N. Kilbertus, M. Rojas-Carulla, G. Parascandolo, M. Hardt, D. Janzing, and B. Schölkopf. Avoiding discrimination through causal reasoning. In *31st Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 656–666, 2017. 1

[88] Michael P. Kim, Aleksandra Korolova, Guy N. Rothblum, and Gal Yona. Preference-informed fairness, 2019. 5.1.2

[89] Robert Kleinberg. Nearly tight bounds for the continuum-armed bandit problem. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2004. 3.1.1

[90] Robert Kleinberg and Tom Leighton. The value of knowing a demand curve: Bounds on regret for online posted-price auctions. In *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS)*, 2003. 3.1.1, 3.1.1

[91] Robert Kleinberg, Aleksandrs Slivkins, and Eli Upfal. Multi-armed bandits in metric spaces. In *Proceedings of the Annual Symposium on Theory of Computing (STOC)*, 2008. 3.1.1, 3.1.1, 3.1.3

[92] Robert Kleinberg, Kevin Leyton-Brown, and Brendan Lucier. Efficiency through procrastination: Approximately optimal algorithm configuration with runtime guarantees. In *IJCAI*, 2017. 3.2.1

[93] Vladimir Koltchinskii. Rademacher penalties and structural risk minimization. *IEEE Transactions on Information Theory*, 47(5):1902–1914, 2001. 3.1.6, B.1.1

[94] Alex Krizhevsky. Learning multiple layers of features from tiny images. In *Technical Report*, 2009. 4.5

[95] Matt Kusner, Jacob Gardner, Roman Garnett, and Kilian Weinberger. Differentially private Bayesian optimization. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 918–927, 2015. 3.1.1

[96] Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015. doi: 10.1126/science.aab3050. 4.5

[97] J. Langford and A. Beygelzimer. Sensitive error correcting output codes. *COLT*, 2005. 2.1, 2.2

[98] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, 1998. 4.5

[99] Kevin Leyton-Brown, Eugene Nudelman, and Yoav Shoham. Empirical hardness models: Methodology and a case study on combinatorial auctions. *Journal of the ACM (JACM)*, 56(4): 22, 2009. 3.2.1

[100] N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. In *Machine Learning*, 1998. 3.2.1

[101] László Lovász and Santosh Vempala. Fast algorithms for logconcave functions: Sampling, rounding, integration, and optimization. In *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS)*, 2006. 3.1.1, 3.1.3, B.1.3, B.1

[102] B. T. Luong, S. Ruggieri, and F. Turini. k-NN as an implementation of situation testing for discrimination discovery and prevention. In *17th Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 502–510, 2011. 5.1

[103] P. Manurangsi and W. Suksompong. Asymptotic existence of fair divisions for groups. *Mathematical Social Sciences*, 89:100–108, 2017. 2

[104] Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 94–103, 2007. 3.1.1

[105] Andres Munoz Medina and Mehryar Mohri. Learning theory and algorithms for revenue optimization in second price auctions with reserve. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2014. 3.1.1, 3.1.1

[106] Andrés Muñoz Medina and Sergei Vassilvitskii. Revenue optimization with approximate bid predictions. *Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2017. 3.1.1

[107] T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling. Never-ending learning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI-15)*, 2015. 2.1

[108] M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of Machine Learning*. MIT press, 2012. 2.1, 2.2

[109] Jamie Morgenstern and Tim Roughgarden. On the pseudo-dimension of nearly optimal auctions. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2015. 3.1.1, 3.1.1

[110] Jamie Morgenstern and Tim Roughgarden. Learning simple auctions. In *Proceedings of the Conference on Learning Theory (COLT)*, 2016. 3.1.1, 3.1.1, 3.1.5

[111] H. Moulin. *Fair Division and Collective Welfare*. MIT Press, 2003. 5.1

[112] Roger Myerson. Optimal auction design. *Mathematics of Operation Research*, 6:58–73, 1981. 3.1.1

[113] B. K. Natarajan. On learning sets and functions. *Machine Learning*, 4(1):67–97, 1989. 5.4

[114] T. D. Nielsen and F. V. Jensen. Learning a decision maker's utility function from (possibly) inconsistent behavior. *Artificial Intelligence*, 160(1–2):53–78, 2004. 5.1

[115] R. Noothigattu, S. S. Gaikwad, E. Awad, S. Dsouza, I. Rahwan, P. Ravikumar, and A. D. Procaccia. A voting-based system for ethical decision making. In *32nd Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1587–1594, 2018. 5.1

[116] M. Palatucci, D. Pomerleau, G. Hinton, and T. Mitchell. Zero-shot learning with semantic output codes. In *NeurIPS*, 2009. 2.2, 2.8

[117] David Pollard. *Convergence of Stochastic Processes*. Springer, 1984. B.1.7, C

[118] A. D. Procaccia. Cake cutting: Not just child's play. *Communications of the ACM*, 56(7): 78–87, 2013. 5.1

[119] Alexander Rakhlin, Karthik Sridharan, and Ambuj Tewari. Online learning: Stochastic, constrained, and smoothed adversaries. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2011. 3.1.1, 3.2.1

[120] John R Rice. The algorithm selection problem. In *Advances in computers*, volume 15, pages 65–118. Elsevier, 1976. 3.2.1

[121] J. M. Robertson and W. A. Webb. *Cake Cutting Algorithms: Be Fair If You Can*. A. K. Peters, 1998. 5.1

[122] Vijay K Rohatgi and AK Md Ehsanes Saleh. *An introduction to probability and statistics*. John Wiley & Sons, 2015. B.1.2

[123] G. N. Rothblum and G. Yona. Probably approximately metric-fair learning. arXiv:1803.03242, 2018. 5.1.2

[124] Tim Roughgarden and Okke Schrijvers. Ironing in the dark. In *Proceedings of the ACM Conference on Economics and Computation (EC)*, 2016. 3.1.1, 3.1.1

[125] Tim Roughgarden and Joshua R Wang. Minimizing regret with multiple reserves. In *Proceedings of the ACM Conference on Economics and Computation (EC)*, pages 601–616. ACM, 2016. 3.1.1, 3.1.1

[126] Mehreen Saeed, Onaiza Maqbool, Haroon Atique Babri, Syed Zahoor Hassan, and S. Mansoor Sarwar. Software clustering techniques and the use of combined algorithm. In *European Conference on Software Maintenance and Reengineering*, 2003. 3.2.1

[127] Shuichi Sakai, Mitsunori Togasaki, and Koichi Yamazaki. A note on greedy algorithms for the maximum weighted independent set problem. *Discrete Applied Mathematics*, 126(2):313–322, 2003. 3.1.5

[128] S. Shalev-Shwartz and S. Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge University Press, 2014. 5.4.1, B.1.1

[129] A. Singh, X. Zhu, and R. Nowak. Unlabeled data: Now it helps, now it doesn't. In *NeurIPS*, 2008. 2.8

[130] Daniel A Spielman and Shang-Hua Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *Journal of the ACM (JACM)*, 51(3):385–463, 2004. 3.2.4

[131] J. Steinhardt and P. Liang. Unsupervised risk estimation using only conditional independence structure. In *NeurIPS*, 2016. 2.2

[132] I. Steinwart. Fully adaptive density based clustering. In *Annals of Statistics*, volume 43, pages 2132–2167, 2015. 2.4

[133] F. E. Su. Rental harmony: Sperner's lemma in fair division. *American Mathematical Monthly*, 106(10):930–942, 1999. 5.1

[134] L. Sweeney. Discrimination in online ad delivery. *Communications of the ACM*, 56(5):44–54, 2013. 5.1

[135] Vasilis Syrgkanis. A sample complexity measure with applications to learning optimal auctions. *Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2017. 3.1.1

[136] S. Thrun. *Explanation-Based Neural Network Learning: A Lifelong Learning Approach*. Kluwer Academic Publishers, Boston, MA, 1996. 2.1

[137] S. Thrun and T. Mitchell. Learning one more thing. In *Proc. 14th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1217–1225, 1995. 2.1

[138] Sebastian Thrun and Tom M. Mitchell. Lifelong robot learning. *Robotics and Autonomous Systems*, 15(1-2):25–46, 1995. 2.1

[139] Henk Tijms. *Understanding probability*. Cambridge University Press, 2012. B.1.2, B.1.6

[140] Alexandre Tsybakov. *Introduction to Nonparametric Estimation*. Springer-Verlag New York, 2009. B.1.4

[141] V. Vapnik and A. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264–280, 1971. 2.4, 2.5, 2.6, 5.4

[142] H. Varian. Equity, envy and efficiency. *Journal of Economic Theory*, 9:63–91, 1974. 5.1

[143] Jonathan Weed, Vianney Perchet, and Philippe Rigollet. Online learning in repeated auctions. In *Proceedings of the Conference on Learning Theory (COLT)*, pages 1562–1583, 2016. 3.1.1, B.14, B.1.4, B.1.4

[144] Gellért Weisz, András György, and Csaba Szepesvári. Leapsandbounds: A method for approximately optimal algorithm configuration. In *ICML*, 2018. 3.2.1

[145] James R. White, Saket Navlakha, Niranjan Nagarajan, Mohammad-Reza Ghodsi, Carl Kingsford, and Mihai Pop. Alignment and clustering of phylogenetic markers—implications for microbial diversity studies. In *BCM Bioinformatics*, 2010. 3.2.1

[146] B. Woodworth, S. Gunasekar, M. I. Ohannessian, and N. Srebro. Learning non-discriminatory predictors. In *30th Proceedings of the Conference on Learning Theory (COLT)*, pages 1920–1953, 2017. 5.1.2

[147] Lin Xu, Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Satzilla: portfolio-based algorithm selection for sat. *Journal of artificial intelligence research*, 32:565–606, 2008. 3.2.1

[148] A. C. Yao. Probabilistic computations: Towards a unified measure of complexity. In *17th Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 222–227, 1977. D

[149] Hector Yee and Bar Ifrach. Aerosolve: Machine learning for humans. *Open Source*, 2015. URL `http://nerds.airbnb.com/aerosolve/`. 3.1.1

[150] M. B. Zafar, I. Valera, M. Gomez-Rodriguez, K. P. Gummadi, and A. Weller. From parity to preference-based notions of fairness in classification. In *31st Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 228–238, 2017. 5.1, 5.1.2

[151] R. Zemel, Y. Wu, K. Swersky, T. Pitassi, and C. Dwork. Learning fair representations. In *30th Proceedings of the International Conference on Machine Learning (ICML)*, pages 325–333, 2013. 5.1, 5.1.2

[152] Uri Zwick. Outward rotations: a tool for rounding solutions of semidefinite programming relaxations, with applications to max cut and other problems. In *Proceedings of the Annual Symposium on Theory of Computing (STOC)*, 1999. 3.1.1, 3.1.5

Appendix A

Appendix for Chapter 2

A.1 Appendix for Error Correcting Output Codes

First, we show that the line segment $[x, y]$ crosses the decision surface of the linear separator h_k if and only if $h(x)$ and $h(y)$ differ on the k^{th} entry.

Lemma A.1. *Let $i \neq j$ be any pair of classes whose codewords disagree on the k^{th} bit. Then for any points $x \in K_i$ and $y \in K_j$, the line segment $[x, y]$ intersects with the line $h_k = 0$.*

Proof. Without loss of generality, suppose that $C_{ik} = 1$ and $C_{jk} = -1$. Then, from the definition of K_i and K_j, we have that $h_k(x) > 0$ and $h_k(y) < 0$. The function $f(t) = h_k((1-t)x + ty)$ is continuous and satisfies $f(0) = h_k(x) > 0$ and $f(1) = h_k(y) < 0$. It follows that there must be some $t_0 \in (0, 1)$ such that $f(t_0) = 0$. But this implies that the point $z = (1 - t_0)x + t_0 y \in [x, y]$ satisfies $h_k(z) = 0$ and it follows that $h_k = 0$ intersects with $[x, y]$ at the point z. □

Next, we show that when the consistent linear output code makes at most β errors when predicting the code word of a new example and the Hamming distance of the code words is at least $2\beta + d + 1$, then there must be a minimum gap $g > 0$ between any pair of points belonging to different classes.

Lemma 2.1. *Under Assumption 2.1, there exists $g > 0$ such that if points x and x' belong to different classes, then $\|x - x'\| > g$.*

Proof. For sets A and B, let $d(A, B) = \min_{a \in A, b \in B} \|a - b\|$ denote the distance between them and recall that for each $i = 1, \ldots, L$, we defined $K_i = \{x \in \mathcal{X} : d_{\text{Ham}}(h(x), C_i) \leq \beta\}$ to be the set of points that belong to class i.

Fix any pair of classes i and j and suppose for contradiction that $d(K_i, K_j) = 0$. This implies that there are two code words $c, c' \in \{\pm 1\}^m$ such that $d_{\text{Ham}}(c, C_i) \leq \beta$, $d_{\text{Ham}}(c', C_j) \leq \beta$, and the distance between $A = \{x \in \mathcal{X} : h(x) = c\}$ and $B = \{x \in \mathcal{X} : h(x) = c'\}$ is 0. First, we construct a point x that belongs to $\overline{A} \cap \overline{B}$, where $\overline{A}$ and $\overline{B}$ denote the closure of A and B, respectively. Since $d(A, B) = 0$, there exists a sequence of points $x_1, x_2, \ldots \in A$ such that $d(x_n, B) \to 0$ as $n \to \infty$. But, since A is bounded, so is the sequence (x_n), and therefore by the Bolzano-Weierstrass theorem, (x_n) has a convergent subsequence. Without loss of generality, suppose that (x_n) itself converges to the point x. Then x is a limit point of A and therefore belongs to the closure of A. On the other hand, since the function $z \mapsto d(z, B)$ is continuous, it follows that $d(x, B) = \lim_{n \to \infty} d(x_n, B) = 0$ and therefore x is also in the closure of B.

Now let k be any index such that the code words c and c' differ on the k^{th} entry. Next, we show that $h_k(x) = 0$. For each integer $n > 0$, let $C_n = B(x, 1/n)$ be the ball of radius $1/n$ centered at x. Since x belongs to the closure of A and C_n is a neighborhood of x, we can find some point, say x_n that belongs to the intersection $A \cap C_n$. Similarly, we can find a point y_n belonging to $B \cap C_n$. Since the line segment $[x_n, y_n]$ passes from A to B, Lemma A.1 guarantees that there is a point $z_n \in [x_n, y_n] \subset C_n$ such that $h_k(z_n) = 0$. But, by construction, the sequence z_n is converging to x and, since linear functions are continuous, it follows that $h_k(x) = \lim_{n\to\infty} h_k(z_n) = 0$.

But this leads to a contradiction: since the codewords c and c' must disagree on at least $d + 1$ entries, at least $d+1$ of the linear separators $h_1, \ldots, h_m$ intersect at the point x, which contradicts our assumption that at most d lines intersect at any point $x \in \mathcal{X}$. Therefore, we must have $d(K_i, K_j) > 0$. Since there are finitely many classes, taking $g = \min_{i,j} d(K_i, K_j)$ completes the proof. □

Next, we prove a similar result to Theorem 2.1 that holds in the agnostic setting of Section 2.7.

Theorem A.1. *Assume Assumption 2.1,* $\operatorname{err}(f^*) \le \eta$*, and* p *has* C*-thick level sets. For* $0 < \epsilon \le \eta$*, suppose* $\{p \ge \epsilon/(2\operatorname{Vol}(K))\}$ *has* N *connected components, each with probability at least* 7η*. With probability at least* $1 - \delta$*, running Algorithm 1 with parameter* $r_c < g$ *on an unlabeled sample of size* $n = \tilde{O}(\frac{1}{\epsilon^2}((4C)^{2d}d^{d+1}/r_c^2 d + N))$ *and querying* $t = O(\ln N/\delta)$ *labels per cluster will have error at most* $\eta + \epsilon$ *after querying at most* Nt *labels.*

Proof. Define $\lambda = \epsilon/(2\operatorname{Vol}(K))$ and let $A_1, \ldots, A_N$ be the connected components of $\{p \ge \lambda\}$. Since Assumption 2.1 holds, Lemma 2.1 guarantees that there is a distance $g > 0$ such that whenever $f^*(x) \ne f^*(x')$, we must have $\|x - x'\| \ge g$. This implies that for any $\lambda' > 0$, f^* must be constant on the connected components of $\{p \ge \lambda'\}$, since otherwise we could construct a pair of points closer than g with $f^*(x) \ne f^*(x')$. In particular, we know that f^* is constant on each of the A_i sets.

Since the clustering produced by Algorithm 1 does not see the labeled examples, an identical covering argument to the one in the proof of Theorem 2.1 shows that for $n = O((4C)^{2d}d^{d+1}/(\epsilon^2 r_c^{2d}))$ with probability at least $1 - \delta$, for each set A_i there is a unique cluster, say $\hat{A}_i$, such that $\hat{A}_i$ contains $S \cap A_i$, the closest cluster to every point in A_i is $\hat{A}_i$. Assume this high probability event occurs.

Similarly to the proof of Theorem 2.1, for $n = O(\frac{N}{\epsilon^2}\ln\frac{1}{\delta})$, we have that with probability at least $1 - \delta$, for any subset of indices $I \subset [N]$, we have that

$$\left||S \cap A_I|/n - P_{\mathcal{X}}(A_I)\right| \le \epsilon,$$

where $A_I = \bigcup_{i\in I} A_i$. Assume this high probability event occurs.

Now let $y_1, \ldots, y_n$ be the (unobserved) labels corresponding to the unlabeled sample $x_1, \ldots, x_n$. Since $\Pr_{(x,y)\sim P}(f^*(x) \ne y) \le \eta$, if $n = O(\frac{1}{\eta^2}\ln\frac{1}{\delta})$, then with probability at least $1 - \delta$, we have that $f^*(x_i) \ne y_i$ for at most $2\eta n$ of the sample points.

Now, for any connected component A_i, let $\hat{A}_i$ be the cluster containing $A_i \cap S$. Since we have uniform convergence for all unions of the A_i sets, and $P_{\mathcal{X}}(A_i) \ge 7\eta$, we know that the set A_i contains at least $6\eta n$ sample points. Therefore, even if every point whose label y_i disagrees with f^* belongs to $\hat{A}_i$, we know that at most a $2\eta n/(6\eta n) = 1/3$ fraction of the points belonging to the cluster $\hat{A}_i$ will have labels other than $f^*(A_i)$. If we query the label of $t = 32\ln\frac{2N}{\delta} = O(\ln\frac{N}{\delta})$ points belonging to cluster $\hat{A}_i$, then with probability at least $1 - \delta/N$ the majority label will agree with f^* on A_i. Applying the union bound over the connected components $A_1, \ldots, A_N$ gives the same guarantee for all connected components with probability at least $1 - \delta$.

Let $\hat{f}$ be the classifier output by Algorithm 1 and $Q \subset [N]$ be the indices of the A_i sets for which the algorithm queried the label of the corresponding cluster $\hat{A}_i$. The above arguments show that with

probability at least $1 - 4\delta$, we have that $\hat{f}(x) = f^*(x)$ for any $x \in \cup_{i \in Q} A_i$ and, as in Theorem 2.1, we know that $P_{\mathcal{X}}(\bigcup_{i \notin Q} A_i) \leq \epsilon/2$. This gives the following bound on the error of $\hat{f}$: Let $(x, y) \sim P$, then

$$\begin{aligned}
\Pr(\hat{f}(x) \neq y) &= \Pr(\hat{f}(x) \neq y, x \in \{p < \lambda\}) \\
&\quad + \Pr(\hat{f}(x) \neq y, x \in \bigcup_{i \in Q} A_i) \\
&\quad + \Pr(\hat{f}(x) \neq y, x \in \bigcup_{i \notin Q} A_i) \\
&\leq \Pr(x \in \{p < \lambda\}) \\
&\quad + \Pr(f^*(x) \neq y) \\
&\quad + \Pr(x \in \bigcup_{i \notin Q} A_i).
\end{aligned}$$

By our choice of λ, the first term is at most $\epsilon/2$, by assumption the second term is at most η, and the last term is at most $\epsilon/2$, giving the final error bound of $\eta + \epsilon$. □

A.2 Appendix For One-vs-all on the Unit Ball

The following result is similar to Theorem 2.4 and shows that Algorithm 3 continues to work in the agnostic setting of Section 2.7.

Theorem A.2. *Suppose the data is drawn from distribution P over $\mathcal{X} \times [L]$ and that there exists a labeling function f^* such that $\Pr_{(x,y)\sim P}(f^*(x) \neq y) \leq \eta$ and Assumptions 2.2 and 2.3 hold when labels are assigned by f^*. Assume that $\Pr_{x \sim P_{\mathcal{X}}}(f^*(x) = i) \geq 19\eta$ for all classes i. For any excess error ϵ, There exists an r_c satisfying $r_c = \Omega(\epsilon c_{\text{lb}}/(c_{\text{ub}}^2 b_{\min}))$ such that with probability at least $1 - \delta$, running Algorithm 3 with parameter r_c on an unlabeled sample of size $n = \tilde{O}((c_{\text{ub}}^4 d/(\epsilon^2 c_{\text{lb}}^2 b_{\min}^2))^d)$ and querying $t = O(\ln \frac{N}{\delta})$ labels from each cluster will output a classifier with error at most $\eta + \epsilon$ and query at most tL labels.*

Proof. For small enough ϵ, we know that at least half of the probability mass of the points assigned to class i will belong to the ϵ-level set of $\{q_{\text{ub}}^{(i)} \geq \epsilon\}$ (in the notation of Theorem 2.4). Therefore, the probability mass of each of the sets $A_1, \ldots, A_L$ in the proof of Theorem 2.4 is at least 9η. It follows that if we see an unlabeled set of size $n = \tilde{O}(\frac{1}{\eta^2})$, then with probability at least $1 - \delta$ every A_i set will contain at least $8\eta n$ points. Since these points belong to A_i, we know that they will be active, included in the graph G, and connected to the cluster that contains samples belonging to A_i. Moreover, under the same high probability event, we know that there are at most $2\eta n$ points whose labels disagree with f^*. Therefore, the cluster that contains samples from A_i must have at least $8\eta n$ points, at most $2\eta n$ of which can have labels that disagree with f^*, so the label assigned by f^* will account for at least a $3/4$ fraction of the points belonging to the cluster containing A_i. It follows that if we query $O(\log(L/\delta))$ labels from each A_i set then with probability at least $1 - \delta$, we will output a classification rule that agrees with f^* except with probability ϵ. It follows that the error with respect to P at most $\eta + \epsilon$. □

A.3 Appendix for Boundary Features Condition

We begin by proving the probability bounds for slices of a d-dimensional ball under the uniform distribution.

Lemma 2.5. *Let $r > 0$ be any radius and X be a random sample drawn uniformly from the ball of radius r centered at the origin. For any width $0 \le \rho \le r/\sqrt{2}$, the probability that the first coordinate of X lands in $[0, \rho]$ can be bounded as follows:*

$$\sqrt{\frac{d}{2^d \pi}} \frac{\rho}{r} \le \Pr_{X \sim B(r,0)}(X_1 \in [0, \rho]) \le \sqrt{\frac{d+1}{2\pi}} \frac{\rho}{r}.$$

Proof. Let B be the ball of radius r centered at the origin and $S = \{x \in B : x_1 \in [0, \rho]\}$ be the slice of B for which the first coordinate is in the interval $[0, \rho]$. The probability that a uniformly random sample from B lands in the subset S is given by $\mathrm{Vol}(S)/\mathrm{Vol}(B)$, where Vol denotes the (Lebesgue) volume of a set.

We bound the volume of the set S by writing the volume as a double integral over the first coordinate x_1 and the remaining $d-1$ coordinates x_R.

$$\mathrm{Vol}(S) = \int_0^\rho \int_{\mathbb{R}^{d-1}} \mathbb{I}\{\|x_R\|_2^2 \le r^2 - x_1^2\} \, dx_R \, dx_1$$

Noticing that the inner integral is actually the volume of a $d-1$ dimensional ball of radius $\sqrt{r^2 - x_1^2}$, and using the fact that for any d, the volume of a d-dimensional ball of radius r is $r^d v_d$, where v_d is the volume of the d-dimensional unit ball, we have

$$\mathrm{Vol}(S) = v_{d-1} \int_0^\rho (r^2 - x_1^2)^{(d-1)/2} \, dx_1.$$

Upper bounding the integrand by r^{d-1} gives that $\mathrm{Vol}(S) \le v_{d-1} \rho r^{d-1}$. Lower bounding the integrand by $(r^2 - \rho^2)^{(d-1)/2}$ and using the fact that $\rho \le \frac{r}{\sqrt{2}}$ we have that $\mathrm{Vol}(S) \ge v_{d-1} \frac{1}{\sqrt{2^{d-1}}} \rho r^{d-1}$. Dividing both inequalities by the volume of B, which is $r^d v_d$, and using the fact that for all d we have $\frac{v_{d-1}}{v_d} \in [\sqrt{\frac{d}{2\pi}}, \sqrt{\frac{d+1}{2\pi}}]$ gives

$$\sqrt{\frac{d}{2^d \pi}} \frac{\rho}{r} \le \Pr_{X \sim B}(X \in S) \le \sqrt{\frac{d+1}{2\pi}} \frac{\rho}{r},$$

as required. □

The following is an extension of Theorem 2.5 to the agnostic setting described in Section 2.7.

Theorem A.3. *Suppose the data is drawn from distribution P over $\mathcal{X} \times [L]$ and that there exists a labeling function f^* such that $\Pr_{(x,y)\sim P}(f^*(x) \ne y) \le \eta$ and Assumptions 2.3 and 2.4 hold when labels are assigned by f^*. Moreover, assume that $\Pr_{x \sim P_{\mathcal{X}}}(f^*(x) = i) \ge 10\eta$ for all classes i. For any excess error $0 < \epsilon \le \eta$, with probability at least $1 - \delta$, running Algorithm 4 with parameters $r \le R/2$ and $\tau = \alpha p^{1/2}(r)/2$ for a known constant α on on a sample of size $n = \tilde{O}(dm^2 c_{\mathrm{ub}}^2 R^d / (c_{\mathrm{lb}}^2 \epsilon^4))$ and querying $t = O(\ln(N/\delta))$ labels from the L largest clusters will have error at most $\eta + \epsilon$.*

Proof. In the proof of Theorem 2.5 we argued that with the set of hyperplanes produced by Algorithm 4 will be good approximations to the true hyperplanes. We additionally showed that the set of hyperplanes approximating one of the linear separators h_i defining the output code will agree with high probability with h_i except in a small margin and we bounded the probability mass of these margins around each h_i by ϵ. It follows that for each class i, the probability mass of the set of points in that class not contained in these margins is at least $10\eta - \epsilon \geq 9\eta$, and it follows that if our unlabeled sample is of size at least $\tilde{O}(\frac{1}{\eta^2})$ that with probability at least $1-\delta$, we will see at least $8\eta n$ points from each class which are not contained in the small margins. Under the same high probability event, we know that at most $2\eta n$ of the labels we query can disagree with f^*, which implies that the majority label within the L largest cells will be the label predicted by f^* on these cells. It follows that if we query the labels of $O(\ln N/\delta)$ labels from each class, then with probability at least $1-\delta$ the resulting classifier will predict labels that disagree with f^* with probability at most ϵ. It follows that the error of the classifier with respect to the distribution P is at most $\eta + \epsilon$. □

Appendix B

Appendix for Chapter 3

B.1 Appendix for Section 3.1

B.1.1 Generic lemmas for dispersion

In this appendix we provide several general tools for demonstrating that a collection of functions will be (w, k)-dispersed. The dispersion analyses for each of our applications leverages the general tools presented here. We first recall the definition of dispersion.

Definition 3.1. Let $u_1, \ldots, u_T : \mathcal{C} \to [0, H]$ be a collection of functions where u_i is piecewise Lipschitz over a partition $\mathcal{P}_i$ of $\mathcal{C}$. We say that $\mathcal{P}_i$ splits a set A if A intersects with at least two sets in $\mathcal{P}_i$ (see Figure 3.1). The collection of functions is (w, k)-*dispersed* if every ball of radius w is split by at most k of the partitions $\mathcal{P}_1, \ldots, \mathcal{P}_T$. More generally, the functions are (w, k)-*dispersed at a maximizer* if there exists a point $\boldsymbol{\rho}^* \in \operatorname{argmax}_{\boldsymbol{\rho} \in \mathcal{C}} \sum_{i=1}^{T} u_i(\boldsymbol{\rho})$ such that the ball $B(\boldsymbol{\rho}^*, w)$ is split by at most k of the partitions $\mathcal{P}_1, \ldots, \mathcal{P}_T$.

We begin by proving the dispersion lemma from Section 3.1.2.

Lemma 3.1. *Let $\mathcal{B} = \{\beta_1, \ldots, \beta_r\} \subset \mathbb{R}$ be a collection of samples where each β_i is drawn from a κ-bounded distribution with density function p_i. For any $\zeta \geq 0$, the following statements hold with probability at least $1 - \zeta$:*

1. *If the β_i are independent, then every interval of width w contains at most $k = O(rw\kappa + \sqrt{r \log(1/\zeta)})$ samples. In particular, for any $\alpha \geq 1/2$ we can take $w = 1/(\kappa r^{1-\alpha})$ and $k = O(r^{\alpha}\sqrt{\log(1/\zeta)})$.*

2. *If the samples can be partitioned into P buckets $\mathcal{B}_1, \ldots, \mathcal{B}_P$ such that each $\mathcal{B}_i$ contains independent samples and $|\mathcal{B}_i| \leq M$, then every interval of width w contains at most $k = O(PMw\kappa + \sqrt{M \log(P/\zeta)}$. In particular, for any $\alpha \geq 1/2$ we can take $w = 1/(\kappa M^{1-\alpha})$ and $k = O(PM^{\alpha}\sqrt{\log(P/\zeta)})$.*

Proof. We begin by proving part 1 of the statement. The expected number of samples that land in any interval I of width w is at most $w\kappa r$, since for each $i \in [r]$, the probability β_i lands in I is at most $w\kappa$. If the distributions $p_1, \ldots, p_r$ were identical, then the β_i would be i.i.d. samples and we could apply standard uniform convergence results leveraging the fact that the VC-dimension of intervals is 2. It is folklore that these uniform convergence results also apply for independent but not identically

distributed random variables. We provide a proof of this fact in Lemma B.1 for completeness. By Lemma B.1, we know that with probability at least $1-\zeta$ over the draw of the set $\mathcal{B}$,

$$\sup_{a,b\in\mathbb{R},a<b}\left(\sum_{i=1}^{r}\mathbf{1}_{\beta_i\in(a,b)}-\mathop{\mathbb{E}}_{\mathcal{B}'}\left[\sum_{i=1}^{r}\mathbf{1}_{\beta_i'\in(a,b)}\right]\right)\leq O\left(\sqrt{r\log\frac{1}{\zeta}}\right),$$

where $\mathcal{B}'=\{\beta_1',\ldots,\beta_r'\}$ is another sample drawn from $p_1,\ldots,p_r$. This implies that with probability at least $1-\zeta$, every interval I of width w satisfies $|\mathcal{B}\cap I|\leq w\kappa r+O(\sqrt{r\log(1/\zeta)})$. For any $\alpha\geq 1/2$, setting $w=r^{\alpha-1}/\kappa$ gives $|\mathcal{B}\cap I|=O(r^{\alpha}\sqrt{\log 1/\zeta})$ for all intervals of width w with probability at least $1-\zeta$.

Next we prove part 2. Applying the argument from part 1 to each bucket $\mathcal{B}_i$, we know that with probability at least $1-\zeta/P$, any interval of width w contains at most $w\kappa M+O(\sqrt{M\log(P/\zeta)})$ samples belonging to $\mathcal{B}_i$. Taking the union bound over the P buckets, it follows that with probability at least $1-\zeta$, every interval of width w contains at most $P(w\kappa M+O(\sqrt{M\log(1/\zeta)}))$ samples in total from all P buckets. For any $\alpha\geq 1/2$, setting $w=M^{\alpha-1}/\kappa$ guarantees that the number of samples in any interval of width w is at most $O(PM^{\alpha}\sqrt{\log(P/\zeta)})$. □

Corollary B.1. *Let $\mathcal{B}=\{\beta_1,\ldots,\beta_r\}$ be a collection of samples where $\beta_i\sim\mathrm{Uniform}([a_i,a_i+W])$ and $a_1,\ldots,a_r,W$ are arbitrary parameters. For any $\zeta>0$ and $\alpha\geq 1/2$, with probability at least $1-\zeta$, every interval of width $w=\frac{W}{r^{1-\alpha}}$ contains at most $O\left(r^{\alpha}\sqrt{\log\frac{1}{\zeta}}\right)$ points.*

Proof. The density function for a uniform random variable on an interval of width W is $1/W$. Therefore, the corollary follows from part 1 of Lemma 3.1. □

Finally, for completeness, we include the following folklore lemma which allows us to use uniform convergence for non-identical random variables, whereas typical uniform convergence bounds are written in terms of identical random variables. It follows by modifying the well-known proof for uniform convergence using Rademacher complexity [25, 93, 128].

Lemma B.1. *Let $\mathcal{B}=\{\beta_1,\ldots,\beta_r\}\subset\mathbb{R}$ be a set of random variables where $\beta_i\sim p_i$. For any $\zeta>0$, with probability at least $1-\zeta$ over the draw of the set $\mathcal{B}$,*

$$\sup_{a,b\in\mathbb{R},a<b}\left(\sum_{i=1}^{r}\boldsymbol{1}_{\beta_i\in(a,b)}-\mathop{\mathbb{E}}_{\mathcal{B}'}\left[\sum_{i=1}^{r}\boldsymbol{1}_{\beta_i'\in(a,b)}\right]\right)\leq O\left(\sqrt{r\ln\frac{1}{\zeta}}\right),$$

where $\mathcal{B}'=\{\beta_1',\ldots,\beta_r'\}$ is another sample drawn from $p_1,\ldots,p_r$.

Proof. Let $\boldsymbol{\sigma}$ be a vector of Rademacher random variables. Since the VC-dimension of intervals is 2, we know from work by Dudley [61] that

$$\mathop{\mathbb{E}}_{\boldsymbol{\sigma}}\left[\sup_{a,b\in\mathbb{R},a<b}\sum_{i=1}^{r}\sigma_i\mathbf{1}_{\beta_i\in(a,b)}\right]\leq O\left(\sqrt{r}\right). \tag{B.1}$$

Also, we have that

$$\begin{aligned}\sup_{a,b\in\mathbb{R},a<b}\left(\sum_{i=1}^{r}\mathbf{1}_{\beta_i\in(a,b)}-\mathop{\mathbb{E}}_{\mathcal{B}'}\left[\sum_{i=1}^{r}\mathbf{1}_{\beta_i'\in(a,b)}\right]\right)&=\sup_{a,b\in\mathbb{R},a<b}\mathop{\mathbb{E}}_{\mathcal{B}'}\left[\sum_{i=1}^{r}\mathbf{1}_{\beta_i\in(a,b)}-\sum_{i=1}^{r}\mathbf{1}_{\beta_i'\in(a,b)}\right]\\&\leq\mathop{\mathbb{E}}_{\mathcal{B}'}\left[\sup_{a,b\in\mathbb{R},a<b}\left(\sum_{i=1}^{r}\mathbf{1}_{\beta_i\in(a,b)}-\mathbf{1}_{\beta_i'\in(a,b)}\right)\right].\end{aligned}$$

Taking the expectation over the draw of $\mathcal{B}$, we have that

$$\mathop{\mathbb{E}}_{\mathcal{B}}\left[\sup_{a,b\in\mathbb{R},a<b}\left(\sum_{i=1}^{r}\mathbf{1}_{\beta_i\in(a,b)}-\mathop{\mathbb{E}}_{\mathcal{B}'}\left[\sum_{i=1}^{r}\mathbf{1}_{\beta_i'\in(a,b)}\right]\right)\right]\leq\mathop{\mathbb{E}}_{\mathcal{B},\mathcal{B}'}\left[\sup_{a,b\in\mathbb{R},a<b}\left(\sum_{i=1}^{r}\mathbf{1}_{\beta_i\in(a,b)}-\mathbf{1}_{\beta_i'\in(a,b)}\right)\right].$$

For each i, β_i and β_i' are independent and identically distributed. Therefore, we can switch them without replacing the expectation, as follows.

$$\mathop{\mathbb{E}}_{\mathcal{B},\mathcal{B}'}\left[\sup_{a,b\in\mathbb{R},a<b}\left(\sum_{i=1}^{r}\mathbf{1}_{\beta_i\in(a,b)}-\mathbf{1}_{\beta_i'\in(a,b)}\right)\right]=\mathop{\mathbb{E}}_{\mathcal{B},\mathcal{B}'}\left[\sup_{a,b\in\mathbb{R},a<b}\left(\sum_{i=1}^{r}\mathbf{1}_{\beta_i'\in(a,b)}-\mathbf{1}_{\beta_i\in(a,b)}\right)\right].$$

Letting σ_i be a Rademacher random variable, we have that

$$\mathop{\mathbb{E}}_{\mathcal{B},\mathcal{B}'}\left[\sup_{a,b\in\mathbb{R},a<b}\left(\sum_{i=1}^{r}\mathbf{1}_{\beta_i\in(a,b)}-\mathbf{1}_{\beta_i'\in(a,b)}\right)\right]=\mathop{\mathbb{E}}_{\boldsymbol{\sigma},\mathcal{B},\mathcal{B}'}\left[\sup_{a,b\in\mathbb{R},a<b}\left(\sum_{i=1}^{r}\sigma_i\left(\mathbf{1}_{\beta_i\in(a,b)}-\mathbf{1}_{\beta_i'\in(a,b)}\right)\right)\right].$$

Since

$$\sup_{a,b\in\mathbb{R},a<b}\left(\sum_{i=1}^{r}\sigma_i\left(\mathbf{1}_{\beta_i\in(a,b)}-\mathbf{1}_{\beta_i'\in(a,b)}\right)\right)\leq\sup_{a,b\in\mathbb{R},a<b}\sum_{i=1}^{r}\sigma_i\mathbf{1}_{\beta_i\in(a,b)}+\sup_{a,b\in\mathbb{R},a<b}\sum_{i=1}^{r}-\sigma_i\mathbf{1}_{\beta_i'\in(a,b)},$$

we have that

$$\begin{aligned}
&\mathop{\mathbb{E}}_{\boldsymbol{\sigma},\mathcal{B},\mathcal{B}'}\left[\sup_{a,b\in\mathbb{R},a<b}\left(\sum_{i=1}^{r}\sigma_i\left(\mathbf{1}_{\beta_i\in(a,b)}-\mathbf{1}_{\beta_i'\in(a,b)}\right)\right)\right]\\
\leq&\mathop{\mathbb{E}}_{\boldsymbol{\sigma},\mathcal{B}}\left[\sup_{a,b\in\mathbb{R},a<b}\sum_{i=1}^{r}\sigma_i\mathbf{1}_{\beta_i\in(a,b)}\right]+\mathop{\mathbb{E}}_{\boldsymbol{\sigma},\mathcal{B}'}\left[\sup_{a,b\in\mathbb{R},a<b}\sum_{i=1}^{r}\sigma_i\mathbf{1}_{\beta_i'\in(a,b)}\right]\\
=&2\mathop{\mathbb{E}}_{\boldsymbol{\sigma},\mathcal{B}}\left[\sup_{a,b\in\mathbb{R},a<b}\sum_{i=1}^{r}\sigma_i\mathbf{1}_{\beta_i\in(a,b)}\right].
\end{aligned}$$

All in all, this means that

$$\sup_{a,b\in\mathbb{R},a<b}\left(\sum_{i=1}^{r}\mathbf{1}_{\beta_i\in(a,b)}-\mathop{\mathbb{E}}_{\mathcal{B}'}\left[\sum_{i=1}^{r}\mathbf{1}_{\beta_i'\in(a,b)}\right]\right)\leq 2\mathop{\mathbb{E}}_{\boldsymbol{\sigma},\mathcal{B}}\left[\sup_{a,b\in\mathbb{R},a<b}\sum_{i=1}^{r}\sigma_i\mathbf{1}_{\beta_i\in(a,b)}\right]. \tag{B.2}$$

We now apply McDiarmid's Inequality to

$$\mathop{\mathbb{E}}_{\boldsymbol{\sigma}\sim\{-1,1\}^r}\left[\sup_{a,b\in\mathbb{R},a<b}\sum_{i=1}^{r}\sigma_i\mathbf{1}_{\beta_i\in(a,b)}\right]. \tag{B.3}$$

Notice that if we switch β_j with an arbitrary β_j', Equation (B.3) will change by at most 1. Therefore, with probability at least $1-\zeta$ over the draw of $\mathcal{B}$,

$$\left|\mathop{\mathbb{E}}_{\boldsymbol{\sigma}}\left[\sup_{a,b\in\mathbb{R},a<b}\sum_{i=1}^{r}\sigma_i\mathbf{1}_{\beta_i\in(a,b)}\right]-\mathop{\mathbb{E}}_{\boldsymbol{\sigma},\mathcal{B}}\left[\sup_{a,b\in\mathbb{R},a<b}\sum_{i=1}^{r}\sigma_i\mathbf{1}_{\beta_i\in(a,b)}\right]\right|\leq\sqrt{\frac{r}{2}\ln\frac{2}{\zeta}}. \tag{B.4}$$

Combining Equations (B.1), (B.2), and (B.4), we have that with probability at least $1-\zeta$,

$$\sup_{a,b\in\mathbb{R},a<b}\left(\sum_{i=1}^{r}\mathbf{1}_{\beta_i\in(a,b)}-\mathop{\mathbb{E}}_{\mathcal{B}'}\left[\sum_{i=1}^{r}\mathbf{1}_{\beta_i'\in(a,b)}\right]\right)\leq O\left(\sqrt{r\ln\frac{1}{\zeta}}\right).$$

□

B.1.2 Properties of κ-bounded distributions

In order to prove dispersion for many of our applications, we start by assuming there is some randomness present in the relevant problem parameters and show that this implies that the resulting utility functions are (w, k)-dispersed with meaningful parameters. The key step of these arguments is to show that the discontinuity locations resulting from the randomness in the problem parameters have κ-bounded density functions. The following lemmas are helpful for reasoning about how transformations of a κ-bounded random variable affect the density upper bound.

Lemma B.2. *Suppose X and Y are independent, real-valued random variables drawn from κ-bounded distributions. Let $Z = |X - Y|$. Then Z is drawn from a 2κ-bounded distribution.*

Proof. Let f_X and f_Y be the density functions of X and Y. The cumulative density function for Z is

$$\begin{aligned} F_Z(z) &= \Pr[Z \le z] = \Pr[Y - X \le z \text{ and } X - Y \le z] = \Pr[Y - z \le X \le z + Y] \\ &= \int_{-\infty}^{\infty} \int_{y-z}^{y+z} f_{X,Y}(x, y)\, dx dy = \int_{-\infty}^{\infty} \int_{y-z}^{y+z} f_X(x) f_Y(y)\, dx dy \\ &= \int_{-\infty}^{\infty} (F_X(y + z) - F_X(y - z)) f_Y(y)\, dy. \end{aligned}$$

Therefore, applying the fundamental theorem of calculus, the density function of Z can be bounded as follows:

$$\begin{aligned} f_Z(z) &= \frac{d}{dz} F_Z(z) = \int_{-\infty}^{\infty} \frac{d}{dz} (F_X(y + z) - F_X(y - z)) f_Y(y)\, dy \\ &= \int_{-\infty}^{\infty} (f_X(y + z) + f_X(y - z)) f_Y(y)\, dy \le 2\kappa \int_{-\infty}^{\infty} f_Y(y)\, dy = 2\kappa. \end{aligned}$$

□

Next, we show that even when X and Y are dependent random variables with a κ-bounded joint distribution, $X - Y$ has a $W\kappa$-bounded distribution, as long as the support set of X and Y are of width at most W.

Lemma B.3. *Suppose X and Y are real-valued random variables taking values in $[a, a + W]$ and $[b, b + W]$ for some $a, b, W \in \mathbb{R}$ and suppose that their joint distribution is κ-bounded. Let $Z = X - Y$. Then Z is drawn from a $W\kappa$-bounded distribution.*

Proof. The cumulative density function for Z is

$$\begin{aligned} F_Z(z) &= \Pr[Z \le z] = \Pr[X - Y \le z] = \Pr[X \le z + Y] \\ &= \int_{b}^{b+W} \int_{a}^{y+z} f_{X,Y}(x, y)\, dx dy. \end{aligned}$$

The density function for Z is

$$\begin{aligned} f_Z(z) &= \frac{d}{dz} F_Z(z) \\ &= \frac{d}{dz} \int_b^{b+W} \int_a^{y+z} f_{X,Y}(x,y)\, dxdy \\ &= \int_b^{b+W} \frac{d}{dz} \int_a^{y+z} f_{X,Y}(x,y)\, dxdy \\ &= \int_b^{b+W} \left(\frac{d}{dz} \int_a^{y} f_{X,Y}(x,y)\, dx + \frac{d}{dz} \int_0^{z} f_{X,Y}(y+t,y)\, dt \right) dy \\ &= \int_b^{b+W} (0 + f_{X,Y}(y+z,y))\, dy \leq W\kappa, \end{aligned}$$

as claimed. □

Finally, we prove that if X and Y have support in $(0,1]$ and a κ-bounded joint distribution, then $\ln(X)$ and $\ln(Y)$ have a κ-bounded joint distribution as well. We will use this fact to show that $\ln(X) - \ln(Y)$ is $\kappa/2$-bounded. These results are primarily useful for the maximum weight independent set and knapsack algorithm selection dispersion analyses.

Lemma B.4. *Suppose X and Y are random variables taking values in $(0,1]$ and suppose that their joint distribution is κ-bounded. Let $A = \ln X$ and $B = \ln Y$. Then A and B have a κ-bounded joint distribution.*

Proof. We will perform a change of variables using the function $g(x,y) = (\ln x, \ln y)$. Let $g^{-1}(a,b) = h(a,b) = (e^a, e^b)$. Then $f_{A,B}(a,b) = f_{X,Y}(a,b)|J_h(a,b)| \leq \kappa e^a e^b \leq \kappa$, where J_h is the Jacobian matrix of h. □

Lemma B.5. *Suppose X and Y are random variables taking values in $(0,1]$ and suppose that their joint distribution is κ-bounded. Then the distribution of $\ln(X) - \ln(Y)$ is $\kappa/2$ bounded.*

Proof. Let $Z = \ln(X) - \ln(Y)$. We will perform change of variables using the function $g(x,y) = (x, \ln(x) - \ln(y))$. Let $g^{-1}(x,z) = h(x,z) = (x, xe^{-z})$. Then

$$J_h(x,z) = \det \begin{pmatrix} 1 & e^{-z} \\ 0 & -xe^{-z} \end{pmatrix} = -xe^{-z}.$$

Therefore, $f_{X,Z}(x,z) = xe^{-z} f_{X,Y}(x, xe^{-z})$. This means that $f_Z(z) = \int_0^1 xe^{-z} f_{X,Y}(x, xe^{-z})\, dx \leq \frac{\kappa}{2e^z}$, so when $z \geq 0$, $f_Z(z) \leq \kappa/2$.

Next, we will perform change of variables using the function $g(x,y) = (\ln(x) - \ln(y), y)$. Let $g^{-1}(z,y) = h(z,y) = (ye^z, y)$. Then

$$J_h(z,y) = \det \begin{pmatrix} ye^z & 0 \\ e^z & 1 \end{pmatrix} = ye^z.$$

Therefore, $f_{Z,Y}(z,y) = ye^z f_{X,Y}(ye^z, y)$. This means that $f_Z(z) = \int_0^1 ye^z f_{X,Y}(ye^z, y)\, dy \leq \frac{\kappa e^z}{2}$, so when $z \leq 0$, $f_Z(z) \leq \kappa/2$.

Combining these two bounds, we see that $f_Z(z) \leq \kappa/2$. □

Lemma B.6. *Suppose X and Y are two independent continuous random variables. Suppose that Y has a κ-bounded density function and $-W \leq X \leq W$ with probability 1. Then Y/X has a κW-bounded density function.*

Proof. Let $Z = \frac{Y}{X}$ and let f_Z be the probability density function of Z. We want to show that for all $z \in \mathbb{R}$, $f_Z(z) \leq \kappa W$.

It is well-known (e.g., [122]) that because X and Y are independent,

$$f_Z(z) = \int_{-\infty}^{\infty} |x| f_X(x) f_Y(zx)\, dx.$$

Since Y has a κ-bounded density function and $-W \leq X \leq W$ with probability 1, this means that

$$\begin{aligned} f_Z(z) &= \int_{-\infty}^{\infty} |x| f_X(x) f_Y(zx)\, dx \leq \kappa \int_{-\infty}^{\infty} |x| f_X(x)\, dx = \kappa \int_{-W}^{W} |x| f_X(x)\, dx \leq \kappa W \int_{-W}^{W} f_X(x)\, dx \\ &= \kappa W. \end{aligned}$$

The first inequality follows because Y has a κ-bounded density function, the second equality follows because $-W \leq X \leq W$ with probability 1, and the final equality follows because f_X is a density function. □

Lemma B.7. *Suppose X is a random variable with κ-bounded distribution and suppose $c \neq 0$ is a constant. Then $\frac{X}{c}$ has a $|c|\kappa$-bounded distribution.*

Proof. Let f_X be the density function of the variable X. It is well-known [139] that if the function $v(x)$ is strictly increasing or strictly decreasing, then the probability density of the random variable $Y = v(X)$ is given by $f_X(a(y))\,|a'(y)|$, where $a(y)$ is the inverse function of $v(x)$. In our setting $v(x) = \frac{x}{c}$, so $a(x) = cx$. Therefore, the probability density of $Y = \frac{X}{c}$ is $|c| f_X(cx)$. Since X has a κ-bounded distribution, $\max |c| f_X(cx) \leq |c|\kappa$. □

B.1.3 Efficient sampling

Both our differential privacy and online algorithms critically rely on our ability to sample from a particular type of distribution. Specifically, let g be a piecewise Lipschitz function mapping vectors in the set $\mathcal{C} \subseteq \mathbb{R}^d$ to $\mathbb{R}$. These applications require us to sample from a distribution μ with density proportional to $e^{g(\boldsymbol{\rho})}$. We use the notation $f_\mu(\boldsymbol{\rho}) = e^{g(\boldsymbol{\rho})} / \int_C e^{g(\boldsymbol{\rho'})}\, d\boldsymbol{\rho'}$ to denote the density function of μ. In this section we provide efficient algorithms for approximately sampling from μ. Our utility guarantees, privacy guarantees, and regret bounds in the following sections include bounds that hold under approximate sampling procedures.

Efficient implementation for 1-dimensional piecewise Lipschitz functions

We begin with an efficient and exact algorithm for sampling from μ in 1-dimensional problems. Our algorithms for higher dimensional sampling have the same basic structure. First, our algorithm requires that the parameter space $\mathcal{C}$ is an interval on the real line. Second, it requires that f_μ is piecewise defined with efficiently computable integrals on each piece of the domain. More formally, suppose there are intervals $\{[a_i, b_i)\}_{i=1}^K$ partitioning $\mathcal{C}$ such that the indefinite integral F_i of f_μ restricted to $[a_i, b_i)$ is efficient to compute. We propose a two-stage sampling algorithm. First, it randomly chooses

one of the intervals $[a_i, b_i)$ with probability proportional to $\int_{a_i}^{b_i} f_\mu(\rho)\,d\rho = F_i(b_i) - F_i(a_i)$. Then, it outputs a sample from the conditional distribution on that interval. By breaking the problem into two stages, we take advantage of the fact that f_μ has a simple form on each of its components. We thus circumvent the fact that f_μ may be a complicated function globally. We provide the pseudocode in Algorithm 14.

Algorithm 14 One-dimensional sampling algorithm

Require: Function g, intervals $\{[a_i, b_i)\}_{i=1}^K$ partitioning $\mathcal{C}$

1: Define $h(\rho) = \exp\big(g(\rho)\big)$ and let H_i be the indefinite integral of h on $[a_i, b_i)$.
2: Let $Z_i = H_i(b_i) - H_i(a_i)$ and define $P_i(\rho) = \frac{1}{Z_i}\big(H_i(\rho) - H_i(a_i)\big)$.
3: Choose random interval index $I = i$ with probability $Z_i / \sum_j Z_j$.
4: Let U be uniformly distributed in $[0, 1]$ and set $\hat{\rho} = P_I^{-1}(U)$.

Ensure: $\hat{\rho}$

The following lemma shows that Algorithm 14 exactly outputs a sample from $f_\mu(\rho) \propto e^{g(\rho)}$.

Lemma B.8. *Algorithm 14 outputs samples from the distribution μ with density $f_\mu(\rho) \propto e^{g(\rho)}$.*

Proof. Let μ be the target distribution. The density function for μ is given by $f_\mu(\rho) = h(\rho)/Z$, where $h(\rho) = e^{g(\rho)}$ and $Z = \int_{\mathcal{C}} g(\rho)\,d\rho = \sum_{i=1}^K Z_i$. Let $\hat{\rho}$ be the output of Algorithm 14. We need to show that $\Pr(\hat{\rho} \le \tau) = \int_{a_1}^{\tau} f_\mu(\rho)\,d\rho$ for all $\tau \in \mathcal{C}$.

Fix any $\tau \in \mathcal{C}$ and let T be the largest index i such that $b_i \le \tau$. Then we have

$$\Pr(\hat{\rho} \le \tau) = \sum_{i=1}^{T} \Pr(\hat{\rho} \in [a_i, b_i)) + \Pr(\hat{\rho} \in [a_{T+1}, \tau)) = \frac{1}{Z}\sum_{i=1}^{T} Z_i + \frac{1}{Z}(H_{T+1}(\tau) - H_{T+1}(a_{T+1}))$$

$$= \frac{1}{Z}\sum_{i=1}^{T}\int_{a_i}^{b_i} h(\rho)\,d\rho + \frac{1}{Z}\int_{a_{T+1}}^{\tau} f(\rho)\,d\rho = \frac{1}{Z}\int_{a_1}^{\tau} h(\rho)\,d\rho = \int_{a_1}^{\tau} f_\mu(\rho)\,d\rho,$$

as required. □

Efficient approximate sampling in multiple dimensions

In this section, we turn to the multi-dimensional setting. We present an efficient algorithm for approximately sampling from μ with density $f_\mu(\boldsymbol{\rho}) \propto e^{g(\boldsymbol{\rho})}$. It applies to the case where the input function g is piecewise concave and each piece of the domain is a convex set. As in the single dimensional case, the algorithm first chooses one piece of the domain with probability proportional to the integral of f_μ on that piece, and then it outputs a sample from the conditional distribution on that piece. See Algorithm 15 for the pseudo-code. Our algorithm uses techniques from high dimensional convex geometry. These tools allow us to approximately integrate and sample efficiently. Bassily et al. [26] used similar techniques for differentially private convex optimization. Their algorithm also approximately samples from the exponential mechanism's output distribution. We generalize these techniques to apply to cases when the function g is only piecewise concave.

We will frequently measure the distance between two probability measures in terms of the relative (multiplicative) distance D_∞. This is defined as $D_\infty(\chi, \sigma) = \sup_{\boldsymbol{\rho}} \big|\log \frac{d\chi}{d\sigma}(\boldsymbol{\rho})\big|$, where $\frac{d\chi}{d\sigma}$ denotes the Radon-Nikodym derivative. The following lemma characterizes the D_∞ metric in terms of the probability mass of sets:

Lemma B.9. *For any probability measures χ and σ, we have that $D_\infty(\chi, \sigma) \le \beta$ if and only if for every set S we have $e^{-\beta}\sigma(S) \le \chi(S) \le e^{\beta}\sigma(S)$.*

Proof. First, suppose that $D_\infty(\chi, \sigma) \le \beta$. Then for every $\boldsymbol{\rho}$, we have that $-\beta \le \log \frac{d\chi}{d\sigma}(\boldsymbol{\rho}) \le \beta$. Exponentiating both sides gives $e^{-\beta} \le \frac{d\chi}{d\sigma}(\boldsymbol{\rho}) \le e^{\beta}$. Now fix any set A. We have:

$$\chi(A) = \int_A \frac{d\chi}{d\sigma}(\boldsymbol{\rho})\, d\sigma(\boldsymbol{\rho}) \le e^{\beta} \int_A 1\, d\sigma(s) = e^{\beta}\sigma(A).$$

Similarly, $\chi(A) \ge e^{-\beta}\sigma(A)$.

Now suppose that $e^{-\beta}\sigma(A) \le \chi(A) \le e^{\beta}\sigma(A)$ for all sets A and let $\boldsymbol{\rho}$ be any point. Let $B_i = B(x, 1/i)$ be a sequence of decreasing balls converging to $\boldsymbol{\rho}$. The Lebesgue differentiation theorem gives that

$$\frac{d\chi}{d\sigma}(\boldsymbol{\rho}) = \lim_{i\to 0} \frac{1}{\sigma(B_i)} \int_{B_i} \frac{d\chi}{d\sigma}(\boldsymbol{y})\, d\sigma(\boldsymbol{y}) = \lim_{i\to 0} \frac{\chi(B_i)}{\sigma(B_i)}.$$

Since $e^{-\beta} \le \frac{\chi(B_i)}{\sigma(B_i)} \le e^{\beta}$ for all i, it follows that $-\beta \le \log \frac{d\chi}{d\sigma}(\boldsymbol{\rho}) \le \beta$, as required. □

Our algorithm depends on two subroutines from high-dimensional convex computational geometry. These subroutines use rapidly mixing random walks to approximately integrate and sample from μ. These procedures are efficient when the function we would like to integrate or sample is logconcave. which holds in our setting, since f_μ is piecewise logconcave when g is piecewise concave. Formally, we assume that we have access to two procedures, $\mathcal{A}_{\text{integrate}}$ and $\mathcal{A}_{\text{sample}}$, with the following guarantees. Let $h : \mathbb{R}^d \to \mathbb{R}_{\ge 0}$ be any logconcave function, we assume

1. For any accuracy parameter $\alpha > 0$ and failure probability $\zeta > 0$, running $\mathcal{A}_{\text{integrate}}(h, \alpha, \zeta)$ outputs a number $\hat{Z}$ such that with probability at least $1 - \zeta$ we have $e^{-\alpha} \int h \le \hat{Z} \le e^{\alpha} \int h$.
2. For any accuracy parameter $\beta > 0$ and failure probability $\zeta > 0$, running $\mathcal{A}_{\text{sample}}(h, \beta, \zeta)$ outputs a sample $\hat{X}$ drawn from a distribution $\hat{\mu}_h$ such that with probability at least $1-\zeta$, $D_\infty(\hat{\mu}_h, \mu_h) \le \beta$. Here, μ_h is the distribution with density proportional to h.

For example, the integration algorithm of Lovász and Vempala [101] satisfies our assumptions on $\mathcal{A}_{\text{integrate}}$ and runs in time $\text{poly}(d, \frac{1}{\alpha}, \log \frac{1}{\zeta}, \log \frac{R}{r})$, where the domain of h is contained in a ball of radius R, and the level set of h of probability mass $1/8$ contains a ball of radius r. Similarly, Algorithm 6 of Bassily et al. [26] satisfies our assumptions on $\mathcal{A}_{\text{sample}}$ with probability 1 and runs in time $\text{poly}(d, L, \frac{1}{\beta}, \log \frac{R}{r})$. When we refer to Algorithm 15 in the rest of the chapter, we use these integration and sampling procedures.

Algorithm 15 Multi-dimensional sampling algorithm for piecewise concave functions

Require: Piecewise concave function g, partition $\mathcal{C}_1, \ldots, \mathcal{C}_K$ on which g is concave, approximation parameter η, confidence parameter ζ.

1: Define $\alpha = \beta = \eta/3$.
2: Let $h(\boldsymbol{\rho}) = \exp(g(\boldsymbol{\rho}))$ and $h_i(\boldsymbol{\rho}) = \mathbb{I}\{\boldsymbol{\rho} \in \mathcal{C}_i\}h(\boldsymbol{\rho})$ be h restricted to $\mathcal{C}_i$.
3: For each $i \in [K]$, let $\hat{Z}_i = \mathcal{A}_{\text{integrate}}(h_i, \alpha, \zeta/(2K))$.
4: Choose random partition index $I = i$ with probability $\hat{Z}_i / \sum_j \hat{Z}_j$.
5: Let $\hat{\boldsymbol{\rho}}$ be the sample output by $\mathcal{A}_{\text{sample}}(h_I, \beta, \zeta/2)$.

Ensure: $\boldsymbol{\rho}$

The main result in this section is that with high probability the output distribution of Algorithm 15 is close to μ.

Lemma B.10. *With probability at least $1-\zeta$ all the approximate integration and sampling operations performed by Algorithm 15 succeed. Let $\hat{\mu}$ be the output distribution of Algorithm 15 conditioned on all integration and sampling operations succeeding and let μ be the distribution with density $f_\mu(\boldsymbol{\rho}) \propto e^{g(\boldsymbol{\rho})}$. Then we have $D_\infty(\hat{\mu}, \mu) \leq \eta$.*

Proof. First, with probability at least $1-\zeta$ every call to the subprocedures $\mathcal{A}_{\mathrm{integrate}}$ and $\mathcal{A}_{\mathrm{sample}}$ succeeds. Assume this high probability event occurs for the remainder of the proof.

Let $\mathcal{C}_1, \ldots, \mathcal{C}_K$, f_μ, and $h_1, \ldots, h_K$ be as defined in Algorithm 15. Let $E \subset \mathcal{C}$ be any set of outcomes and let $\hat{\mu}_i$ denote the output distribution of $\mathcal{A}_{\mathrm{sample}}(h_i, \beta, \delta'/(2K))$. We have

$$\hat{\mu}(E) = \Pr(\hat{\boldsymbol{\rho}} \in E) = \sum_{i=1}^{K} \Pr(\hat{\boldsymbol{\rho}} \in E | \hat{\boldsymbol{\rho}} \in \mathcal{C}_i) \Pr(\hat{\boldsymbol{\rho}} \in \mathcal{C}_i) = \sum_{i=1}^{K} \hat{\mu}_i(E) \cdot \frac{\hat{Z}_i}{\sum_j \hat{Z}_j}.$$

Using the guarantees on $\mathcal{A}_{\mathrm{integrate}}$ and $\mathcal{A}_{\mathrm{sample}}$ and Lemma B.9, it follows that

$$\hat{\mu}(E) \leq \sum_{i=1}^{K} e^{\beta} \mu_i(E) e^{2\alpha} \frac{Z_i}{\sum_j Z_j} = e^{\eta} \mu(E),$$

where $Z_i = \int_{\mathcal{C}_i} f_\mu$ and μ_i is the distribution with density proportional to $\boldsymbol{\rho} \mapsto \mathbb{I}\{\boldsymbol{\rho} \in \mathcal{C}_i\} \cdot h(\boldsymbol{\rho})$. Similarly, we have that $\hat{\mu}(E) \geq e^{-\eta} \mu(E)$. By Lemma B.9 it follows that $D_\infty(\hat{\mu}, \mu) \leq \eta$. □

B.1.4 Proofs for online learning (Section 3.1.3)

In our regret bounds and utility guarantees for differentially private optimization, we assume that the ball of radius w centered at an optimal point $\boldsymbol{\rho}^*$ is contained in the parameter space $\mathcal{C}$. Lemma B.11 shows that when $\mathcal{C}$ is convex, we can transform the problem so that this condition is satisfied, at the cost of doubling the radius of $\mathcal{C}$.

Lemma B.11. *Let $\mathcal{C} \subset \mathbb{R}^d$ be a convex parameter space contained in a ball of radius R and let $u_1, \ldots, u_T : \mathcal{C} \to [0, H]$ be any piecewise L-Lipschitz and (w, k)-dispersed utility functions. There exists an enlarged parameter space $\mathcal{C}' \supset \mathcal{C}$ contained in a ball of radius $2R$ and extended utility functions $q_1, \ldots, q_T : \mathcal{C}' \to [0, H]$ such that:*

1. *Any maximizer of $\sum_t q_t$ can be transformed into a maximizer for $\sum_t u_t$ by projecting onto $\mathcal{C}$.*
2. *The functions $q_1, \ldots, q_t$ are piecewise L-Lipschitz and (w, k)-dispersed.*
3. *There exists an optimizer $\boldsymbol{\rho}^* \in \operatorname{argmax}_{\boldsymbol{\rho} \in \mathcal{C}'} \sum_t q_t(\boldsymbol{\rho})$ such that $B(\boldsymbol{\rho}^*, R) \subset \mathcal{C}'$.*

Proof. For any $\boldsymbol{\rho} \in \mathbb{R}^d$, let $\mathcal{C}(\boldsymbol{\rho}) = \operatorname{argmin}_{\boldsymbol{\rho}' \in \mathcal{C}} \|\boldsymbol{\rho} - \boldsymbol{\rho}'\|_2$ denote the Euclidean projection of $\boldsymbol{\rho}$ onto $\mathcal{C}$. Define $\mathcal{C}' = \{\boldsymbol{\rho} \in \mathbb{R}^d : \|\boldsymbol{\rho} - \mathcal{C}(\boldsymbol{\rho})\|_2 \leq R\}$ to be the set of points within distance R of $\mathcal{C}$, and let $q_t : \mathcal{C}' \to [0, H]$ be given by $q_t(\boldsymbol{\rho}) = u_t(\mathcal{C}(\boldsymbol{\rho}))$ for $t \in [T]$. Since $\mathcal{C}$ is contained in a ball of radius R and every point in $\mathcal{C}'$ is within distance R of $\mathcal{C}$, it follows that $\mathcal{C}'$ is contained in a ball of radius $2R$.

Part 1. Let $\boldsymbol{\rho}^* \in \operatorname{argmax}_{\boldsymbol{\rho} \in \mathcal{C}'} \sum_{t=1}^{T} q_t(\boldsymbol{\rho})$ be any maximizer of $\sum_t q_t$. We need to show that $\mathcal{C}(\boldsymbol{\rho}^*)$ is a maximizer of $\sum_t u_t$. First, since for any $\boldsymbol{\rho} \in \mathcal{C}'$ we have $q_t(\boldsymbol{\rho}) = u_t(\mathcal{C}(\boldsymbol{\rho}))$, it follows that

$\max_{\boldsymbol{\rho}\in\mathcal{C}'} \sum_{t=1}^{T} q_t(\boldsymbol{\rho}) = \max_{\boldsymbol{\rho}\in\mathcal{C}} \sum_{t=1}^{T} u_t(\boldsymbol{\rho})$ (i.e., the maximum value attained by $\sum_t q_t$ over $\mathcal{C}'$ is equal to the maximum value attained by $\sum_t u_t$ over $\mathcal{C}$). Since $\boldsymbol{\rho}^*$ is a maximizer of $\sum_t q_t$, we have $\max_{\boldsymbol{\rho}\in\mathcal{C}} \sum_{t=1}^{T} u_t(\boldsymbol{\rho}) = \sum_{t=1}^{T} q_t(\boldsymbol{\rho}^*) = \sum_{t=1}^{T} u_t(\mathcal{C}(\boldsymbol{\rho}^*))$ and it follows that $\mathcal{C}(\boldsymbol{\rho}^*)$ is a maximizer for $\sum_t u_t$.

Part 2. Next, we show that each function q_t is piecewise L-Lipschitz. Let $\mathcal{C}_1, \ldots, \mathcal{C}_N$ be the partition of $\mathcal{C}$ such that u_t is L-Lipschitz on each piece, and define $\mathcal{C}'_1, \ldots, \mathcal{C}'_N$ by $\mathcal{C}'_i = \{\boldsymbol{\rho} \in \mathcal{C}' : \mathcal{C}(\boldsymbol{\rho}) \in \mathcal{C}_i\}$ for each $i \in [N]$. We will show that q_t is L-Lipschitz on each set $\mathcal{C}'_i$. To see this, we use the fact that projections onto convex sets are contractions (i.e., $\|\boldsymbol{\rho} - \boldsymbol{\rho}'\|_2 \geq \|\mathcal{C}(\boldsymbol{\rho}) - \mathcal{C}(\boldsymbol{\rho}')\|_2$). From this it follows that for any $\boldsymbol{\rho}, \boldsymbol{\rho}' \in \mathcal{C}'_i$ we have

$$|q_t(\boldsymbol{\rho}) - q_t(\boldsymbol{\rho}')| = |u_t(\mathcal{C}(\boldsymbol{\rho})) - u_t(\mathcal{C}(\boldsymbol{\rho}'))| \leq L \cdot \|\mathcal{C}(\boldsymbol{\rho}) - \mathcal{C}(\boldsymbol{\rho}')\|_2 \leq L \cdot \|\boldsymbol{\rho} - \boldsymbol{\rho}'\|_2,$$

where the first inequality follows from the fact that $\mathcal{C}(\boldsymbol{\rho})$ and $\mathcal{C}(\boldsymbol{\rho}')$ belong to $\mathcal{C}_i$ and u_t is L-Lipschitz on $\mathcal{C}_i$.

Next, we show that $q_1, \ldots, q_T$ are (w, k)-dispersed. Fix any function index t, let $B = B(\boldsymbol{\rho}, w)$ be any ball of radius w and suppose that B is split by the partition $\mathcal{C}'_1, \ldots, \mathcal{C}'_N$ of $\mathcal{C}'$ defined above for which q_t is piecewise Lipschitz. This implies that we can find two points $\boldsymbol{\rho}_1$ and $\boldsymbol{\rho}_2$ in B such that (after possibly renaming the partitions) we have $\boldsymbol{\rho}_1 \in \mathcal{C}'_1$ and $\boldsymbol{\rho}_2 \in \mathcal{C}'_2$. By definition of the sets $\mathcal{C}'_i$, it follows that $\mathcal{C}(\boldsymbol{\rho}_1) \in \mathcal{C}_1$ and $\mathcal{C}(\boldsymbol{\rho}_2) \in \mathcal{C}_2$. Moreover, since projections onto convex sets are contractions, we have that $\mathcal{C}(\boldsymbol{\rho}_1)$ and $\mathcal{C}(\boldsymbol{\rho}_2)$ are both contained in $B(\mathcal{C}(\boldsymbol{\rho}), w)$. Therefore, the ball $B(\mathcal{C}(\boldsymbol{\rho}), w)$ is split by the partition $\mathcal{C}_1, \ldots, \mathcal{C}_T$ of $\mathcal{C}$ on which u_t is piecewise L-Lipschitz. It follows that if no ball of radius w is split by more than k of the piecewise Lipschitz partitions for the functions $u_1, \ldots, u_T$, then the same is true for $q_1, \ldots, q_T$.

Part 3. Finally, let $\boldsymbol{\rho}^* \in \operatorname{argmax}_{\boldsymbol{\rho}\in\mathcal{C}} \sum_t u_t(\boldsymbol{\rho})$. This point is also a maximizer for $\sum_t q_t$, and is contained in the R-interior of $\mathcal{C}'$. □

We now turn to proving our main result for online piecewise Lipschitz optimization in the full information setting.

Algorithm 16 Online learning algorithm for single-dimensional piecewise functions

Require: $\lambda \in (0, 1/H]$

1: Set $u_0(\cdot) = 0$ to be the constant 0 function over $\mathcal{C}$.
2: **for** $t = 1, 2, \ldots, T$ **do**
3: Obtain a point ρ_t using Algorithm 14 with $g = \lambda \sum_{s=0}^{t-1} u_s$. (The point ρ_t is sampled with probability proportional to $e^{g(\rho_t)}$.)
4: Observe the the function $u_t(\cdot)$ and receive payoff $u_t(\rho_t)$.

Theorem 3.1. *Let $u_1, \ldots, u_T : \mathcal{C} \to [0, H]$ be any sequence of piecewise L-Lipschitz functions that are (w, k)-dispersed at the maximizer $\boldsymbol{\rho}^*$. Suppose $\mathcal{C} \subset \mathbb{R}^d$ is contained in a ball of radius R and $B(\boldsymbol{\rho}^*, w) \subset \mathcal{C}$. The exponentially weighted forecaster with $\lambda = \sqrt{d \ln(R/w)/T}/H$ has expected regret bounded by*

$$O\left(H\left(\sqrt{Td\log\frac{R}{w}} + k\right) + TLw\right).$$

Algorithm 17 Online learning algorithm for multi-dimensional piecewise concave functions

Require: $\lambda \in (0, 1/H]$, $\eta, \zeta \in (0, 1)$.

1: Set $u_0(\cdot) = 0$ to be the constant 0 function over $\mathcal{C}$.
2: **for** $t = 1, 2, \ldots, T$ **do**
3: Obtain a vector $\boldsymbol{\rho}_t$ using Algorithm 15 with $g = \lambda \sum_{s=0}^{t-1} u_s$, approximation parameter $\eta/4$, and confidence parameter ζ/T. (The vector $\boldsymbol{\rho}_t$ is sampled with probability that is approximately proportional to $e^{g(\boldsymbol{\rho}_t)}$.)
4: Observe the function $u_t(\cdot)$ and receive payoff $u_t(\boldsymbol{\rho}_t)$.

For all rounds $t \in [T]$, suppose $\sum_{s=1}^{t} u_s$ is piecewise Lipschitz over at most K pieces. When $d = 1$ and $\exp(\sum_{s=1}^{t} u_s)$ can be integrated in constant time on each of its pieces, the running time is $O(TK)$. When $d > 1$ and $\sum_{s=1}^{t} u_s$ is piecewise concave over convex pieces, we provide an efficient approximate implementation. For approximation parameters $\eta = \zeta = 1/\sqrt{T}$ and $\lambda = \sqrt{d \ln(R/w)/T}/H$, this algorithm has the same regret bound as the exact algorithm and runs in time $O(T(K \cdot \mathrm{poly}(d, 1/\eta) + \mathrm{poly}(d, L, 1/\eta))$.

Proof. Define $u_0(\boldsymbol{\rho}) = 0$ and $U_t(\boldsymbol{\rho}) = \sum_{s=0}^{t-1} u_s(\boldsymbol{\rho})$ for each $t \in [T]$. Let $W_t = \int_{\mathcal{C}} \exp(\lambda U_t(\boldsymbol{\rho}))\, d\boldsymbol{\rho}$ be the normalizing constant at round t and let $P_t = \mathbb{E}_{\boldsymbol{\rho} \sim p_t}[u_t(\boldsymbol{\rho})]$ denote the expected payoff achieved by the algorithm in round t, where the expectation is only with respect to sampling $\boldsymbol{\rho}_t$ from p_t. Also, let $P(\mathcal{A}) = \sum_{t=1}^{T} P_t$ be the expected payoff of the algorithm (with respect to its random choices). We begin by upper bounding W_{t+1}/W_t by $\exp\left(\left(e^{\lambda} - 1\right) P_t\right)$.

$$
\begin{aligned}
\frac{W_{t+1}}{W_t} &= \frac{\int_{\mathcal{C}} \exp(\lambda U_{t+1}(\boldsymbol{\rho}))\, d\boldsymbol{\rho}}{\int_{\mathcal{C}} \exp(\lambda U_t(\boldsymbol{\rho}))\, d\boldsymbol{\rho}} \\
&= \frac{\int_{\mathcal{C}} \exp(\lambda U_t(\boldsymbol{\rho})) \cdot \exp(\lambda u_t(\boldsymbol{\rho}))\, d\boldsymbol{\rho}}{\int_{\mathcal{C}} \exp(\lambda U_t(\boldsymbol{\rho}))\, d\boldsymbol{\rho}} && (U_{t+1} = U_t + u_t) \\
&= \int_{\mathcal{C}} p_t(\boldsymbol{\rho}) \exp(\lambda u_t(\boldsymbol{\rho}))\, d\boldsymbol{\rho} && (\text{By definition of } p_t) \\
&\le \int_{\mathcal{C}} p_t(\boldsymbol{\rho}) \left(1 + (e^{H\lambda} - 1)\frac{u_t(\boldsymbol{\rho})}{H}\right) d\boldsymbol{\rho} && (\text{For } z \in [0,1], e^{\lambda z} \le 1 + (e^{\lambda} - 1)z) \\
&\le 1 + (e^{H\lambda} - 1)\frac{P_t}{H} \le \exp\left((e^{H\lambda} - 1)\frac{P_t}{H}\right) && (1 + z \le e^{z}).
\end{aligned}
$$

Therefore,

$$
\frac{W_{T+1}}{W_1} \le \exp\left(\frac{e^{H\lambda} - 1}{H} \sum_{i=1}^{T} P_t\right) = \exp\left(\frac{P(\mathcal{A})\left(e^{H\lambda} - 1\right)}{H}\right). \tag{B.5}
$$

We now lower bound W_{T+1}/W_1. To do this, let $\boldsymbol{\rho}^*$ be the optimal parameter and let $\mathrm{OPT} = U_{T+1}(\boldsymbol{\rho}^*)$. Also, let $\mathcal{B}^*$ be the ball of radius w around $\boldsymbol{\rho}^*$. From (w, k)-dispersion, we know that for

all $\boldsymbol{\rho} \in \mathcal{B}^*$, $U_{T+1}(\boldsymbol{\rho}) \geq \mathrm{OPT} - Hk - LTw$. Therefore,

$$\begin{aligned} W_{T+1} &= \int_{\mathcal{C}} \exp(\lambda U_{T+1}(\boldsymbol{\rho}))\, d\boldsymbol{\rho} \\ &\geq \int_{\mathcal{B}^*} \exp(\lambda U_{T+1}(\boldsymbol{\rho}))\, d\boldsymbol{\rho} \\ &\geq \int_{\mathcal{B}^*} \exp(\lambda(\mathrm{OPT} - Hk - LTw)) d\boldsymbol{\rho} \\ &\geq \mathrm{Vol}(B(\boldsymbol{\rho}^*, w)) \exp(\lambda(\mathrm{OPT} - Hk - LTw)). \end{aligned}$$

Moreover, $W_1 = \int_{\mathcal{C}} \exp(\lambda U_1(\boldsymbol{\rho}))\, d\boldsymbol{\rho} \leq \mathrm{Vol}(B(\mathbf{0}, R))$. Therefore,

$$\frac{W_{T+1}}{W_1} \geq \frac{\mathrm{Vol}(B(\boldsymbol{\rho}^*, w))}{\mathrm{Vol}(B(\mathbf{0}, R))} \exp(\lambda(\mathrm{OPT} - Hk - LTw)).$$

The volume ratio is equal to $(w/R)^d$, since the volume of a ball of radius r in $\mathbb{R}^d$ is proportional to r^d. Therefore,

$$\frac{W_{T+1}}{W_1} \geq \left(\frac{w}{R}\right)^d \exp(\lambda(\mathrm{OPT} - Hk - LTw)). \tag{B.6}$$

Combining Equations B.5 and B.6, taking the log, and rearranging terms, we have that

$$\mathrm{OPT} \leq \frac{P(\mathcal{A})(e^{H\lambda} - 1)}{H\lambda} + \frac{d\ln(R/w)}{\lambda} + Hk + LTw.$$

We subtract $P(\mathcal{A})$ on either side have that

$$\mathrm{OPT} - P(\mathcal{A}) \leq \frac{P(\mathcal{A})(e^{H\lambda} - 1 - H\lambda)}{H\lambda} + \frac{d\ln(R/w)}{\lambda} + Hk + LTw.$$

We use the fact that for $z \in [0, 1]$, $e^z \leq 1 + z + (e-2)z^2$ and the that $P(\mathcal{A}) \leq HT$ to conclude that

$$\mathrm{OPT} - P(\mathcal{A}) \leq H^2 T\lambda + \frac{d\ln(R/w)}{\lambda} + Hk + LTw.$$

The analysis of the efficient multi-dimensional algorithm that uses approximate sampling is given in Theorem B.1. □

Next, we argue that the dependence on the Lipschitz constant can be made logarithmic by tuning the parameter w exploiting the fact that any functions that are (w, k)-dispersed are also (w', k)-dispersed for $w' \leq w$.

Corollary B.2. *Let $u_1, \ldots, u_T$ be the functions observed by Algorithm 16 and suppose they satisfy the conditions of Theorem 3.1. Suppose $T \geq 1/(Lw)$. Setting $\lambda = 1/(H\sqrt{T})$, the regret of Algorithm 16 is bounded by $H\sqrt{T}\,(1 + d\ln(RNL)) + Hk + 1$.*

Proof. This bound follows from applying Theorem 3.1 using the (w', k)-disperse critical boundaries condition with $w' = 1/(LT)$. The lower bound on requirement on T ensures that $w' \leq w$. □

Lemma B.12 shows that when the sequence of functions u_1, …, u_T are chosen by a smoothed adversary in the sense of Cohen-Addad and Kanade [46] then the set of functions is (w, k)-dispersed with non-trivial parameters.

Lemma B.12. *Let $u_1, \dots, u_T$ be a sequence of functions chosen by a κ-smoothed adversary. That is, each function u_t has at most τ discontinuities, each drawn independently from a potentially different κ-bounded density. For any $\alpha \geq 1/2$, with probability at least $1 - \zeta$ the functions $u_1, \dots, u_T$ are (w, k)-dispersed with $w = \frac{1}{\kappa(T\tau)^{1-\alpha}}$ and $k = O((T\tau)^{\alpha}\sqrt{\log 1/\zeta})$.*

Proof. There are a total of $T\tau$ discontinuities from the T functions, each independently drawn from a κ-bounded density. Applying the first part of Lemma 3.1 guarantees that with high probability, every interval of width w contains at most $O(T\tau\kappa w + \sqrt{T\tau \log(1/\zeta)})$ discontinuities. Setting $w = \frac{1}{\kappa(T\tau)^{1-\alpha}}$ completes the proof. □

Bandit Online Optimization

Our algorithm for online learning under bandit feedback requires that we construct a w-net for the parameter space $\mathcal{C}$. The following Lemma shows that there exists a w-net for any set contained in a ball of radius R of size $(3R/w)^d$. This is a standard result, but we include the proof for completeness.

Lemma B.13. *Let $\mathcal{C} \subset \mathbb{R}^d$ be contained in a ball of radius R. Then there exists a subset $\hat{\mathcal{C}}_w \subset \mathcal{C}$ such that $|\hat{\mathcal{C}}_w| \leq (3R/w)^d$ and for every $\rho \in \mathcal{C}$ there exists $\hat{\rho} \in \hat{\mathcal{C}}_w$ such that $\|\rho - \hat{\rho}\|_2 \leq w$.*

Proof. Consider the following greedy procedure for constructing $\hat{\mathcal{C}}_w$: while there exists any point in $\mathcal{C}$ further than distance w from $\hat{\mathcal{C}}_w$, pick any such point and it to the $\hat{\mathcal{C}}_w$. Suppose this greedy procedure has added points $rho_1, \dots, \rho_n$ to the covering so far. We will argue that the algorithm must terminate with $n \leq (3R/w)^d$.

By construction, we know that the distance between any ρ_i and ρ_j is at least w, which implies that the balls $B(\rho_1, w/2)$, …, $B(\rho_n, w/2)$ are all disjoint. Moreover, since their centers are contained $\mathcal{C}$ which is contained in a ball of radius R, we are guaranteed that the balls of radius $w/2$ centered on ρ_1, …, ρ_n are also contained in a ball of radius $R + w/2$. Therefore, we have $\mathrm{Vol}(\bigcup_{i=1}^n B(\rho_i, w/2)) \leq \mathrm{Vol}(B(0, R + w/2))$. Since the balls $B(\rho_i, w/2)$ are disjoint, we have $\mathrm{Vol}(\bigcup_{i=1}^n B(\rho_i, w/2)) = \sum_{i=1}^n \mathrm{Vol}(B(\rho_i, w/2)) = n(w/2)^d v_d$, where v_d is the volume of the unit ball in d dimensions. Similarly, $\mathrm{Vol}(B(0, R + w/2)) = (R + w/2)^d v_d$. Therefore, we have $n \leq \left(\frac{2(R+w/2)}{w}\right)^d \leq \left(\frac{3R}{w}\right)^d$, where the last inequality follows from the fact that $w < R$. □

Approximate sampling for online learning

Theorem B.1. *Let $u_1, \dots, u_T : \mathcal{C} \to [0, H]$ be the sequence of functions observed by Algorithm 17. Suppose that each u_t is piecewise L-Lipschitz and concave on convex pieces. Moreover, suppose that $u_1, \dots, u_T$ are (w, k)-disperse, $\mathcal{C} \subset \mathbb{R}^d$ is convex and contained in a ball of radius R, and that for some $\boldsymbol{\rho}^* \in \mathrm{argmax}_{\boldsymbol{\rho} \in \mathcal{C}} \sum_{t=1}^T u_t(\boldsymbol{\rho})$ we have $B(\boldsymbol{\rho}^*, w) \subset \mathcal{C}$. Then for any $\eta, \zeta \in (0, 1)$, the expected regret of Algorithm 17 with $\lambda = \sqrt{d \ln(R/w)/T}/H$ is bounded by*

$$O(H(\sqrt{Td \ln(R/w)} + k) + TLw + \eta HT + \zeta HT).$$

Moreover, suppose there are K intervals partitioning $\mathcal{C}$ so that $\sum_{t=1}^T u_t$ is piecewise L-Lipschitz on each region. Also, suppose that we use the integration algorithm of Lovász and Vempala [101] and the sampling algorithm of Bassily et al. [26] to implement Algorithm 15. The running time of Algorithm 17 is

$$T\left(K \cdot \mathrm{poly}\left(d, \frac{1}{\eta}, \log\frac{TK}{\zeta}, \log\frac{R}{r}\right) + \mathrm{poly}\left(d, L, \frac{1}{\eta}, \log\frac{R}{r}\right)\right).$$

Proof. On each round we use Algorithm 15 to approximately sample a point from the distribution proportional to $g_t(\boldsymbol{\rho}) = \exp(\lambda \sum_{t=1}^{T} u_t(\boldsymbol{\rho}))$. Each invocation of Algorithm 15 has failure probability $\zeta' = \zeta/T$, which implies that with probability at least $1 - \zeta$ the sampler succeeds on every round. Assume this high probability event holds for the remainder of the proof. In this case, Lemma B.10 guarantees that if $\hat{\mu}_t$ is the output distribution of Algorithm 15 oun round t and μ_t is the distribution with density proportional to g_t, then we have $D_\infty(\hat{\mu}_t, \mu_t) \leq \eta$.

Next, we show that the expected utility per round of the approximate sampler is at most a $(1 - \eta)$ factor smaller than the expected utility per round of the exact sampler. Let $\hat{\boldsymbol{\rho}}_t \sim \hat{\mu}_t$ and $\boldsymbol{\rho}_t \sim \mu_t$ be samples drawn from the approximate and exact samplers at round t, respectively. Then we have

$$\mathbb{E}[u_t(\hat{\boldsymbol{\rho}}_t)] = \int_0^\infty \Pr(u_t(\hat{\boldsymbol{\rho}}_t) \geq \tau)\, d\tau \geq e^{-\eta} \int_0^\infty \Pr(u_t(\boldsymbol{\rho}_t) \geq \tau)\, d\tau = e^{-\eta} \cdot \mathbb{E}[u_t(\boldsymbol{\rho}_t)] \geq (1-\eta) \cdot \mathbb{E}[u_t(\boldsymbol{\rho}_t)].$$

where the first inequality follows from Lemma B.9 (i.e., since $D_\infty(\hat{\mu}_t, \mu_t) \leq \eta$, we know that the probability mass of any event under $\hat{\mu}_t$ is at least $e^{-\eta}$ of its mass under μ_t). Using this, we can bound the excess regret suffered by the approximate sampler compared to the exact sampling algorithm:

$$\mathbb{E}\left[\sum_{t=1}^{T} u_t(\boldsymbol{\rho}_t) - u_t(\hat{\boldsymbol{\rho}}_t)\right] \leq \mathbb{E}\left[\sum_{t=1}^{T} u_t(\boldsymbol{\rho}_t)\right] - (1-\eta) \cdot \mathbb{E}\left[\sum_{t=1}^{T} u_t(\boldsymbol{\rho}_t)\right] = \eta \cdot \mathbb{E}\left[\sum_{t=1}^{T} u_t(\boldsymbol{\rho}_t)\right] \leq \eta HT.$$

Combining this with the regret bound for the exact sampling algorithm gives a regret bound of

$$H^2 T\lambda + \frac{d \ln(R/W)}{\lambda} + Hk + TLw + \eta HT + \zeta HT.$$

where the ζHT term comes from the ζ-probability event that at least one invocation of the approximate sampler fails, in which case the maximum possible regret is HT. Setting $\eta = \zeta = 1/\sqrt{T}$ and λ as in Theorem 3.1 gives a regret bound of

$$O(H(\sqrt{Td\log(R/w)} + k) + TLw).$$

□

Lower bound for single-dimensional parameter spaces

We will use the following adversarial construction to prove our lower bound.

Lemma B.14 (Weed et al. [143])**.** *Define the two functions $u^{(0)} : [0,1] \to [0,1]$ and $u^{(1)} : [0,1] \to [0,1]$ such that*

$$u^{(0)}(\rho) = \begin{cases} \frac{1}{2} & \text{if } \rho < \frac{1}{2} \\ 0 & \text{if } \rho \geq \frac{1}{2} \end{cases} \text{ and } u^{(1)}(\rho) = \begin{cases} \frac{1}{2} & \text{if } \rho < \frac{1}{2} \\ 1 & \text{if } \rho \geq \frac{1}{2}. \end{cases}$$

There exists a pair of adversaries U and L defining two distributions μ_U and μ_L over $\left\{u^{(0)}, u^{(1)}\right\}$ such that for any learning algorithm,

$$\max_{A \in \{U,L\}} \max_{\rho \in [0,1]} \mathbb{E}\left[\sum_{t=1}^{T} u_t(\rho) - \sum_{t=1}^{T} u_t(\rho_t)\right] \geq \frac{1}{32}\sqrt{T},$$

where the expectation is over $u_1, \ldots, u_T \sim \mu_A$ and the random choices $\rho_1, \ldots, \rho_T$ of the algorithm. Moreover, under adversary U, any parameter $\rho \geq \frac{1}{2}$ is optimal and under adversary L, any parameter $\rho < \frac{1}{2}$ is optimal.

Specifically, the adversary U defined by Weed et al. [143] selects the function $u^{(0)}$ with probability $\frac{1}{2} - \frac{1}{8\sqrt{T}}$ and $u^{(1)}$ with probability $\frac{1}{2} + \frac{1}{8\sqrt{T}}$. Meanwhile, the adversary L selects the function $u^{(0)}$ with probability $\frac{1}{2} + \frac{1}{8\sqrt{T}}$ and $u^{(1)}$ with probability $\frac{1}{2} - \frac{1}{8\sqrt{T}}$. The theorem's proof follows from standard information theoretic techniques for lower bounds (e.g., Tsybakov [140]).

Weed et al. [143] study the specific problem of learning to bid in an online setting. A single item is sold at each round. The learner is a potential buyer, and he does not know his value for the item at any given round. The seller sells each item in a second-price auction. The other buyers' values may be adversarially selected. If the buyer wins the item, he learns his value, but if he does not win the item, he learns nothing about his value at that round. Thus, the buyer must learn to bid without knowing his value. Weed et al. [143] prove that the buyer's optimization problem amounts to the online optimization of threshold functions with a specific structure. They do not develop a general theory of dispersion, but we can map their analysis into our setting. In essence, they prove that if these threshold functions are $(w, 0)$-dispersed at the maximizer, then the adversary's regret is bounded by $O\left(\sqrt{T \log \frac{1}{w}}\right)$. They use Lemma B.14 to prove a matching lower bound.

Theorem B.2. *For any learning algorithm and $T \geq 3$, there is a sequence $u_1, \ldots, u_T$ of piecewise constant functions mapping $[0, 1]$ to $[0, 1]$ such that if*

$$D = \{(w, k) : \{u_1, \ldots, u_T\} \text{ is } (w, k)\text{-dispersed at the maximizer}\},$$

then

$$\max_{\rho \in [0,1]} \mathbb{E}\left[\sum_{t=1}^{T} u_t(\rho) - u_t\left(\rho_t\right)\right] = \Omega\left(\inf_{(w,k)\in D}\left\{\sqrt{T \log \frac{1}{w}} + k\right\}\right).$$

Proof. We begin with an outline of the proof. For the first $T - \sqrt{T}$ rounds, our adversary behaves exactly like the worse of the two adversaries defined in Lemma B.14, playing threshold functions at each round. Each threshold function has a discontinuity at $\rho = \frac{1}{2}$. Since these functions are piecewise constant, either $\frac{1}{4}$ or $\frac{3}{4}$ maximizes the sum $\sum_{t=1}^{T-\sqrt{T}} u_t$. Denoting this maximizer as ρ^*, our adversary then plays $\sqrt{T}$ copies of the indicator function corresponding to the interval $\left[\rho^* - 2^{-T}, \rho^* + 2^{-T}\right]$. At the end of all T rounds, ρ^* maximizes the sum $\sum_{t=1}^{T} u_t$. We prove that the expected regret incurred by this adversary is at least $\frac{\sqrt{T}}{64}$, which follows from Lemma B.14. In order to prove the theorem, we need to show that $\frac{\sqrt{T}}{64} = \Omega\left(\inf_{(w,k)\in D}\left\{\sqrt{T \log \frac{1}{w}} + k\right\}\right)$. Therefore, we need to show that the set of functions played by the adversary is (w, k)-dispersed at the maximizer ρ^* for $w = \Theta(1)$ and $k = O\left(\sqrt{T}\right)$. The reason this is true is that the only functions with discontinuities in the interval $\left[\rho^* - \frac{1}{8}, \rho^* + \frac{1}{8}\right]$ are the final $\sqrt{T}$ functions played by the adversary. Thus, the theorem statement holds.

Regret lower bound. Fix the learning algorithm. We begin be demonstrating the existence of a sequence of functions inducing a regret lower bound of $\Omega\left(\sqrt{T}\right)$.

Claim B.1. *Let $T' = \left\lfloor T - \sqrt{T} \right\rfloor$. There is a sequence $u_1, \ldots, u_{T'}$ of piecewise constant functions mapping $[0, 1]$ to $[0, 1]$ such that:*

1. *The expected regret is lower bounded as follows: $\max_{\rho \in [0,1]} \mathbb{E}\left[\sum_{t=1}^{T'} u_t(\rho) - u_t\left(\rho_t\right)\right] \geq \frac{\sqrt{T}}{64}$, where the expectation is over the random choices $\rho_1, \ldots, \rho_{T'}$ of the learner.*

2. *Each function u_t is a threshold function with a discontinuity at $\frac{1}{2}$.*

3. *Either $\left[0, \frac{1}{2}\right] = \operatorname{argmax}_{\rho \in [0,1]} \sum_{t=1}^{T'} u_t(\rho)$ or $\left(\frac{1}{2}, 1\right] = \operatorname{argmax}_{\rho \in [0,1]} \sum_{t=1}^{T'} u_t(\rho)$.*

Proof of Claim B.1. By Lemma B.14, there exists a randomized adversary such that

$$\max_{\rho \in [0,1]} \mathbb{E}\left[\sum_{t=1}^{T'} u_t(\rho) - u_t\left(\rho_t\right)\right] \geq \frac{1}{32}\sqrt{T'} = \frac{1}{32}\sqrt{\left\lfloor T - \sqrt{T}\right\rfloor} \geq \frac{1}{32}\sqrt{\frac{T-\sqrt{T}}{2}} \geq \frac{\sqrt{T}}{64},$$

where the expectation is over the random sequence $u_1, \ldots, u_{T'}$ of functions chosen by the adversary and the random choices $\rho_1, \ldots, \rho_{T'}$ of the learner. Since this inequality holds in expectation over the adversary's choices, there must be a sequence $u_1, \ldots, u_{T'}$ of functions such that

$$\max_{\rho \in [0,1]} \mathbb{E}\left[\sum_{t=1}^{T'} u_t(\rho) - u_t\left(\rho_t\right)\right] \geq \frac{\sqrt{T}}{64},$$

where the expectation is only over the random choices $\rho_1, \ldots, \rho_{T'}$ of the learner. Therefore, the first part of the claim holds. By Lemma B.14, we know that each function is piecewise constant with a discontinuity at $\frac{1}{2}$, so the second part of the claim holds. Finally, Lemma B.14 guarantees that either every parameter in $[0, 1/2]$ is optimal, or every parameter in $(1/2, 1]$ is optimal, so the third part of the claim holds. □

Construction of the final $\sqrt{T}$ functions. From the previous claim, we know that either $\left[0, \frac{1}{2}\right] = \operatorname{argmax}_{\rho \in [0,1]} \sum_{t=1}^{T'} u_t(\rho)$ or $\left(\frac{1}{2}, 1\right] = \operatorname{argmax}_{\rho \in [0,1]} \sum_{t=1}^{T'} u_t(\rho)$. We define the parameter $\rho^* \in \left\{\frac{1}{4}, \frac{3}{4}\right\}$ such that $\rho^* = \frac{1}{4}$ in the former case, and $\rho^* = \frac{3}{4}$ in the latter case. Under this definition, ρ^* maximizes the sum $\sum_{t=1}^{T'} u_t$. We now define the functions $u_{T'+1}, \ldots, u_T$ to all be equal to the function $\rho \mapsto \mathbf{1}_{\{\rho \in [\rho^* - 2^{-T}, \rho^* + 2^{-T}]\}}$. Under this definition, the parameter ρ^* remains a maximizer of the sum $\sum_{t=1}^{T} u_t$.

In our final regret bound, we will use the following property of the functions $u_{T'+1}, \ldots, u_T$.

Claim B.2. *For any parameters $\rho_{T'+1}, \ldots, \rho_T$, $\sum_{t=T'+1}^{T} u_t\left(\rho^*\right) - u_t\left(\rho_t\right) \geq 0$.*

Proof of Claim B.2. By definition, $\sum_{t=T'+1}^{T} u_t\left(\rho^*\right) = T - T' + 1$. Since the range of each function u_t is contained in $[0, 1]$, for any parameters $\rho_{T'+1}, \ldots, \rho_T$, $\sum_{t=T'+1}^{T} u_t\left(\rho_t\right) \leq T - T' + 1$. Therefore, the claim holds. □

Dispersion parameters. We now prove that the only functions with discontinuities in the interval $\left[\rho^* - \frac{1}{8}, \rho^* + \frac{1}{8}\right]$ are the functions $u_{T'+1}, \ldots, u_T$. Since $T \geq 3$, if $\rho^* = \frac{1}{4}$, then $\left[\rho^* - 2^{-T}, \rho^* + 2^{-T}\right] \subseteq \left[\rho^* - \frac{1}{8}, \rho^* + \frac{1}{8}\right] \subset \left[0, \frac{1}{2}\right]$ and $\rho^* = \frac{3}{4}$, then $\left[\rho^* - 2^{-T}, \rho^* + 2^{-T}\right] \subseteq \left[\rho^* - \frac{1}{8}, \rho^* + \frac{1}{8}\right] \subset \left(\frac{1}{2}, 1\right]$. Since the discontinuities of the functions $u_1, \ldots, u_{T'}$ only fall at $\frac{1}{2}$, this means that the interval $\left[\rho^* - \frac{1}{8}, \rho^* + \frac{1}{8}\right]$ only contains the discontinuities of the functions $u_{T'+1}, \ldots, u_T$. Since $T - T' = T - \left\lfloor T - \sqrt{T}\right\rfloor \leq T - \left(T - \sqrt{T} - 1\right) = \sqrt{T} + 1$, the set $\{u_1, \ldots, u_T\}$ is $\left(\frac{1}{8}, \sqrt{T} + 1\right)$-dispersed

at the maximizer ρ^*. Therefore,

$$\begin{aligned}
&\inf_{(w,k)\in D}\left\{\sqrt{T\log\frac{1}{w}}+k\right\}\\
&\leq \sqrt{T\log 8}+\sqrt{T}+1\\
&\leq 4\sqrt{T}+0\\
&\leq 256\max_{\rho\in[0,1]}\mathbb{E}\left[\sum_{t=1}^{T'}u_t(\rho)-u_t\left(\rho_t\right)\right]+\mathbb{E}\left[\sum_{t=T'+1}^{T}u_t\left(\rho^*\right)-u_t\left(\rho_t\right)\right] && \text{(Claims B.1 and B.2)}\\
&= 256\,\mathbb{E}\left[\sum_{t=1}^{T'}u_t\left(\rho^*\right)-u_t\left(\rho_t\right)\right]+\mathbb{E}\left[\sum_{t=T'+1}^{T}u_t\left(\rho^*\right)-u_t\left(\rho_t\right)\right] && \left(\rho^*\in\operatorname*{argmax}_{\rho\in[0,1]}\sum_{t=1}^{T'}u_t(\rho)\right)\\
&\leq 256\,\mathbb{E}\left[\sum_{t=1}^{T}u_t\left(\rho^*\right)-u_t\left(\rho_t\right)\right]\\
&\leq 256\max_{\rho\in[0,1]}\mathbb{E}\left[\sum_{t=1}^{T}u_t\left(\rho\right)-u_t\left(\rho_t\right)\right].
\end{aligned}$$

Therefore,

$$\max_{\rho\in[0,1]}\mathbb{E}\left[\sum_{t=1}^{T}u_t(\rho)-u_t\left(\rho_t\right)\right]=\Omega\left(\inf_{(w,k)\in D}\left\{\sqrt{T\log\frac{1}{w}}+k\right\}\right),$$

as claimed. □

Remark B.1. As we describe in Section 3.1.1, Cohen-Addad and Kanade [46] show that if the functions their full-information, online optimization algorithm sees are piecewise constant, map from $[0,1]$ to $[0,1]$, and are $(w,0)$-dispersed at the maximizer, then their algorithm's regret is bounded by $O\left(\sqrt{T\ln(1/w)}\right)$. The worst-case, piecewise constant functions $u_1,\ldots,u_T$ from Theorem B.2 map from $[0,1]$ to $[0,1]$ and are $\left(\frac{1}{8},\sqrt{T}+1\right)$-dispersed at the maximizer, which means that our regret upper bound (Theorem 3.1) is $O\left(\sqrt{T\log(1/w)}+k\right)=O\left(\sqrt{T}\right)$. However, these functions are not $(w,0)$-dispersed at the maximizer for any $w\geq 2^{-T}$, so the regret bound by Cohen-Addad and Kanade [46] is trivial, since $\sqrt{T\log(1/w)}$ with $w=2^{-T}$ equals T.

Lower bound for multi-dimensional parameter spaces

We begin with the following corollary of Lemma B.14 by Weed et al. [143] which simply generalizes the adversarial functions from single-dimensional thresholds to multi-dimensional thresholds (i.e., axis-aligned hyperplanes).

Corollary B.3 (Corollary of Lemma B.14)**.** *For any $i\in[d]$, define the two functions $u^{(0)}:[0,1]^d\to[0,1]$ and $u^{(1)}:[0,1]^d\to[0,1]$ such that*

$$u^{(0)}(\boldsymbol{\rho})=\begin{cases}\frac{1}{2} & \text{if } \rho[i]<\frac{1}{2}\\ 0 & \text{if } \rho[i]\geq\frac{1}{2}\end{cases} \text{ and } u^{(1)}(\boldsymbol{\rho})=\begin{cases}\frac{1}{2} & \text{if } \rho[i]<\frac{1}{2}\\ 1 & \text{if } \rho[i]\geq\frac{1}{2}.\end{cases}$$

There exists a pair of adversaries U and L defining two distributions μ_U and μ_L over $\left\{u^{(0)}, u^{(1)}\right\}$ such that for any learning algorithm,

$$\max_{A\in\{U,L\}} \max_{\boldsymbol{\rho}\in[0,1]^d} \mathbb{E}\left[\sum_{t=1}^{T} u_t(\boldsymbol{\rho}) - \sum_{t=1}^{T} u_t(\boldsymbol{\rho}_t)\right] \geq \frac{1}{32}\sqrt{T},$$

where the expectation is over $u_1, \ldots, u_T \sim \mu_A$ and the random choices $\boldsymbol{\rho}_1, \ldots, \boldsymbol{\rho}_T$ of the algorithm. Moreover, under adversary U, any parameter vector $\boldsymbol{\rho}$ such that $\rho[i] > \frac{1}{2}$ is optimal and under adversary L, any parameter vector $\boldsymbol{\rho}$ such that $\rho[i] \leq \frac{1}{2}$ is optimal.

Theorem 3.2. *Suppose $T \geq d$. For any algorithm, there are piecewise constant functions $u_1, \ldots, u_T$ mapping $[0,1]^d$ to $[0,1]$ such that if $D = \{(w,k) : \{u_1, \ldots, u_T\}$ is (w,k)-dispersed at the maximizer$\}$, then*

$$\max_{\boldsymbol{\rho}\in[0,1]^d} \mathbb{E}\left[\sum_{t=1}^{T} u_t\left(\boldsymbol{\rho}\right) - u_t\left(\boldsymbol{\rho}_t\right)\right] = \Omega\left(\inf_{(w,k)\in D}\left\{\sqrt{Td\log\frac{1}{w}} + k\right\}\right),$$

where the expectation is over the random choices $\boldsymbol{\rho}_1, \ldots, \boldsymbol{\rho}_T$ of the adversary.

Proof. The proof of this theorem is a straightforward generalization of Theorem B.2. We begin with an outline of the proof. For each dimension $i \in [d]$, the adversary plays $\left\lfloor \frac{T-\sqrt{T}}{d} \right\rfloor$ thresholds aligned with the i^{th} axis, behaving exactly like the worse of the two adversaries defined in Corollary B.3. Each threshold function has a discontinuity along the hyperplane $\left\{\boldsymbol{\rho} \in [0,1]^d : \rho[i] = \frac{1}{2}\right\}$. Since these functions are piecewise constant, either $\left\{\boldsymbol{\rho} \in [0,1]^d : \rho[i] \leq \frac{1}{2}\right\}$ is the set of points maximizing the sum of these $\left\lfloor \frac{T-\sqrt{T}}{d} \right\rfloor$ thresholds or $\left\{\boldsymbol{\rho} \in [0,1]^d : \rho[i] > \frac{1}{2}\right\}$. Denoting this maximizing set as $\mathcal{P}_i^*$, let $\mathcal{P}^* = \bigcap_{i=1}^{d} \mathcal{P}_i^*$ be the set of points maximizing all $\left\lfloor \frac{T-\sqrt{T}}{d} \right\rfloor$ thresholds over all d dimensions. By definition of the sets $\mathcal{P}_i^*$, this set is a hypercube with side-length $\frac{1}{2}$. Let $\boldsymbol{\rho}^*$ be the center of the hypercube $\mathcal{P}^*$. Our adversary then plays $T - d\left\lfloor \frac{T-\sqrt{T}}{d} \right\rfloor \leq \sqrt{T} + d$ copies of the indicator function corresponding to the ball $\left\{\boldsymbol{\rho} : ||\boldsymbol{\rho}^* - \boldsymbol{\rho}|| \leq 2^{-T}\right\}$. At the end of all T rounds, $\boldsymbol{\rho}^*$ maximizes the sum $\sum_{t=1}^{T} u_t$. We prove that the expected regret incurred by this adversary is at least $\frac{\sqrt{Td}}{64}$, which follows from Corollary B.3. In order to prove the theorem, we need to show that $\frac{\sqrt{Td}}{64} = \Omega\left(\inf_{(w,k)\in D}\left\{\sqrt{Td\log\frac{1}{w}} + k\right\}\right)$. Therefore, we need to show that the set of functions played by the adversary is (w,k)-dispersed at the maximizer $\boldsymbol{\rho}^*$ for $w = \Theta(1)$ and $k = O\left(\sqrt{Td}\right)$. The reason this is true is that the only functions with discontinuities in the ball $\left\{\boldsymbol{\rho} : ||\boldsymbol{\rho}^* - \boldsymbol{\rho}|| \leq \frac{1}{8}\right\}$ are the final $\sqrt{T} + d$ functions played by the adversary. Thus, the theorem statement holds.

Regret lower bound. Fix the learning algorithm. We begin be demonstrating the existence of a sequence of functions inducing a regret lower bound of $\Omega\left(\sqrt{Td}\right)$.

Claim B.3. *Let $T' = \left\lfloor \frac{T-\sqrt{T}}{d} \right\rfloor$. There is a sequence $u_1, \ldots, u_{T'd}$ of piecewise constant functions mapping $[0,1]^d$ to $[0,1]$ such that:*

1. *The expected regret is lower bounded as follows: $\max_{\boldsymbol{\rho}\in[0,1]} \mathbb{E}\left[\sum_{t=1}^{T'd} u_t(\boldsymbol{\rho}) - u_t\left(\boldsymbol{\rho}_t\right)\right] \geq \frac{\sqrt{Td}}{64}$, where the expectation is over the random choices $\boldsymbol{\rho}_1, \ldots, \boldsymbol{\rho}_{T'd}$ of the learner.*

2. *The set of points maximizing $\sum_{t=1}^{T'd} u_t$ is a hypercube of side length $\frac{1}{2}$.*

Proof of Claim B.3. Corollary B.3 with $i = 1$ tells us there exists a randomized adversary such that

$$\max_{\boldsymbol{\rho}\in[0,1]^d} \mathbb{E}\left[\sum_{t=1}^{T'} u_t^{(1)}(\boldsymbol{\rho}) - u_t^{(1)}\left(\boldsymbol{\rho}_t\right)\right] \geq \frac{1}{32}\sqrt{T'},$$

where the expectation is over the random sequence $u_1^{(1)}, \ldots, u_{T'}^{(1)}$ of functions chosen by the adversary and the random choices $\boldsymbol{\rho}_1, \ldots, \boldsymbol{\rho}_{T'}$ of the learner. Next, for each $i \in \{2, \ldots, d\}$, we apply Corollary B.3 to get T' random functions $u_1^{(i)}, \ldots, u_{T'}^{(i)}$ such that

$$\max_{\boldsymbol{\rho}\in[0,1]^d} \mathbb{E}\left[\sum_{t=1}^{T'} u_t^{(i)}(\boldsymbol{\rho}) - u_t^{(i)}\left(\boldsymbol{\rho}_{(i-1)T'+t}\right)\right] \geq \frac{1}{32}\sqrt{T'}, \tag{B.7}$$

where the expectation is over the random sequence $u_1^{(i)}, \ldots, u_{T'}^{(i)}$ of functions chosen by the adversary and the random choices $\boldsymbol{\rho}_{(i-1)T'+1}, \ldots, \boldsymbol{\rho}_{iT'}$ of the learner. Since for each $i \in [d]$, Equation (B.7) holds in expectation over the adversary's choices, there must be a sequence $u_1^{(i)}, \ldots, u_{T'}^{(i)}$ of functions such that

$$\max_{\boldsymbol{\rho}\in[0,1]^d} \mathbb{E}\left[\sum_{t=1}^{T'} u_t^{(i)}(\boldsymbol{\rho}) - u_t^{(i)}\left(\boldsymbol{\rho}_{(i-1)T'+t}\right)\right] \geq \frac{1}{32}\sqrt{T'},$$

where the expectation is only over the random choices $\boldsymbol{\rho}_{(i-1)T'+1}, \ldots, \boldsymbol{\rho}_{iT'}$ of the learner.

From Corollary B.3, we know that either

$$\left\{\boldsymbol{\rho} \in [0,1]^d : \rho[i] \leq \frac{1}{2}\right\} = \operatorname*{argmax}_{\boldsymbol{\rho}\in[0,1]^d}\left\{\mathbb{E}\left[\sum_{t=1}^{T'} u_t^{(i)}(\boldsymbol{\rho}) - u_t^{(i)}\left(\boldsymbol{\rho}_{(i-1)T'+t}\right)\right]\right\}$$

or

$$\left\{\boldsymbol{\rho} \in [0,1]^d : \rho[i] > \frac{1}{2}\right\} = \operatorname*{argmax}_{\boldsymbol{\rho}\in[0,1]^d}\left\{\mathbb{E}\left[\sum_{t=1}^{T'} u_t^{(i)}(\boldsymbol{\rho}) - u_t^{(i)}\left(\boldsymbol{\rho}_{(i-1)T'+t}\right)\right]\right\}.$$

Call this set of maximizing points $\mathcal{P}_i^*$. Note that the intersection $\mathcal{P}^* = \bigcap_{i=1}^d \mathcal{P}_i^*$ of these d sets is a hypercube with side length $\frac{1}{2}$. Therefore, for any $\boldsymbol{\rho} \in \mathcal{P}^*$,

$$\mathbb{E}\left[\sum_{i=1}^{d}\sum_{t=1}^{T'} u_t^{(i)}(\boldsymbol{\rho}) - u_t^{(i)}\left(\boldsymbol{\rho}_{(i-1)T'+t}\right)\right] \geq \frac{d}{32}\sqrt{T'} = \frac{d}{32}\sqrt{\left\lfloor\frac{T-\sqrt{T}}{d}\right\rfloor} \geq \frac{d}{32}\sqrt{\frac{T}{4d}} = \frac{\sqrt{Td}}{64}.$$

For ease of notation, we relabel the functions $u_1^{(1)}, \ldots, u_{T'}^{(1)}, \ldots, u_1^{(d)}, \ldots, u_{T'}^{(d)}$ as $u_1, \ldots, u_{T'd}$. □

Construction of the final $T - T'd$ functions. Let $\boldsymbol{\rho}^*$ be the center of the hypercube $\mathcal{P}^*$. We now define the functions $u_{T'd+1}, \ldots, u_T$ to all be equal to the function $\boldsymbol{\rho} \mapsto \mathbf{1}_{\{||\boldsymbol{\rho}-\boldsymbol{\rho}^*||\leq 2^{-T}\}}$. Under this definition, the parameter $\boldsymbol{\rho}^*$ remains a maximizer of the sum $\sum_{t=1}^T u_t$.

In our final regret bound, we will use the following property of the functions $u_{T'd+1}, \ldots, u_T$.

Claim B.4. *For any parameters $\boldsymbol{\rho}_{T'd+1}, \ldots, \boldsymbol{\rho}_T$, $\sum_{t=T'd+1}^T u_t\left(\boldsymbol{\rho}^*\right) - u_t\left(\boldsymbol{\rho}_t\right) \geq 0$.*

Proof of Claim B.4. By definition, $\sum_{t=T'd+1}^{T} u_t(\boldsymbol{\rho}^*) = T - T'd + 1$. Since the range of each function u_t is contained in $[0,1]$, for any parameters $\boldsymbol{\rho}_{T'd+1}, \ldots, \boldsymbol{\rho}_T$, $\sum_{t=T'd+1}^{T} u_t(\boldsymbol{\rho}_t) \leq T - T'd + 1$. Therefore, the claim holds. □

Dispersion parameters. We now prove that the only functions with discontinuities in the ball $\left\{\boldsymbol{\rho} : ||\boldsymbol{\rho}^* - \boldsymbol{\rho}|| \leq \frac{1}{8}\right\}$ are the functions $u_{T'd+1}, \ldots, u_T$. Since $\mathcal{P}^*$ is a hypercube with side length $\frac{1}{2}$ and $\boldsymbol{\rho}^*$ is the center of that hypercube, $\left\{\boldsymbol{\rho} : ||\boldsymbol{\rho}^* - \boldsymbol{\rho}|| \leq \frac{1}{8}\right\} \subset \mathcal{P}^*$. Therefore, the ball $\left\{\boldsymbol{\rho} : ||\boldsymbol{\rho}^* - \boldsymbol{\rho}|| \leq \frac{1}{8}\right\}$ only contains the discontinuities of the functions $u_{T'd+1}, \ldots, u_T$. Since $T - T'd = T - d\left\lfloor \frac{T-\sqrt{T}}{d} \right\rfloor \leq T - d\left(\frac{T-\sqrt{T}}{d} - 1\right) = \sqrt{T} + d$, the set $\{u_1, \ldots, u_T\}$ is $\left(\frac{1}{8}, \sqrt{T} + d\right)$-dispersed at the maximizer $\boldsymbol{\rho}^*$. Therefore,

$$
\begin{aligned}
&\inf_{(w,k)\in D} \left\{ \sqrt{Td \log \frac{1}{w}} + k \right\} \\
&\leq \sqrt{T \log 8} + \sqrt{T} + d \\
&\leq 4\sqrt{Td} + 0 \\
&\leq 256 \max_{\boldsymbol{\rho}\in[0,1]^d} \mathbb{E}\left[\sum_{t=1}^{T'd} u_t(\boldsymbol{\rho}) - u_t(\boldsymbol{\rho}_t) \right] + \mathbb{E}\left[\sum_{t=T'd+1}^{T} u_t(\boldsymbol{\rho}^*) - u_t(\boldsymbol{\rho}_t) \right] && \text{(Claims B.3 and B.4)} \\
&= 256\, \mathbb{E}\left[\sum_{t=1}^{T'd} u_t(\boldsymbol{\rho}^*) - u_t(\boldsymbol{\rho}_t) \right] + \mathbb{E}\left[\sum_{t=T'd+1}^{T} u_t(\boldsymbol{\rho}^*) - u_t(\boldsymbol{\rho}_t) \right] && \left(\boldsymbol{\rho}^* \in \operatorname*{argmax}_{\boldsymbol{\rho}\in[0,1]^d} \sum_{t=1}^{T'd} u_t(\boldsymbol{\rho})\right) \\
&\leq 256\, \mathbb{E}\left[\sum_{t=1}^{T} u_t(\boldsymbol{\rho}^*) - u_t(\boldsymbol{\rho}_t) \right] \\
&\leq 256 \max_{\boldsymbol{\rho}\in[0,1]^d} \mathbb{E}\left[\sum_{t=1}^{T} u_t(\boldsymbol{\rho}) - u_t(\boldsymbol{\rho}_t) \right].
\end{aligned}
$$

Therefore,

$$
\max_{\boldsymbol{\rho}\in[0,1]^d} \mathbb{E}\left[\sum_{t=1}^{T} u_t(\boldsymbol{\rho}) - u_t(\boldsymbol{\rho}_t) \right] = \Omega\left(\inf_{(w,k)\in D} \left\{ \sqrt{Td \log \frac{1}{w}} + k \right\} \right),
$$

as claimed. □

Differentially Private Online Learning

Lemma B.15 (Dwork et al. [65])**.** *Given target privacy parameters $\epsilon \in (0,1)$ and $\delta > 0$, to ensure $(\epsilon, \tau\delta' + \delta)$ cumulative privacy loss over τ mechanisms, it suffices that each mechanism is (ϵ', δ')-differentially private, where*

$$
\epsilon' = \frac{\epsilon}{2\sqrt{2\tau \ln(1/\delta)}}.
$$

Theorem B.3. *Let $u_1, \ldots, u_T$ be the sequence of functions observed by Algorithm 16 and suppose they satisfy the conditions of Theorem 3.1. Let $\epsilon \in (0,1)$ and $\delta > 0$ be privacy parameters. If $\lambda = \frac{\epsilon}{4H\sqrt{2T\ln(1/\delta)}}$, then Algorithm 16 is (ϵ, δ)-differentially private. Its regret is bounded by*

$$H\sqrt{T}\left(\frac{\epsilon}{4\sqrt{2\ln(1/\delta)}} + \frac{4\ln(R/w)\sqrt{2\ln(1/\delta)}}{\epsilon}\right) + Hk + LTw.$$

Moreover, suppose there are K intervals partitioning $\mathcal{C}$ so that $\sum_{t=1}^{T} = u_t$ is piecewise L-Lipschitz on each interval. Then the running time of Algorithm 16 is $T \cdot poly(K)$.

Proof. For all $t \in [T]$, the sensitivity of the function $\sum_{i=0}^{t-1} u_i$ is bounded by H. Therefore, at each time step t, Algorithm 16 samples from the exponential mechanism with privacy parameters $\epsilon' = \frac{\epsilon}{2\sqrt{2T\ln(1/\delta)}}$ and $\delta = 0$. The privacy guarantee therefore follows from Lemma B.15. The regret bound follows from Theorem 3.1. The running time follows from the running time of Algorithm 14. □

Corollary B.4. *Let $u_1, \ldots, u_T$ be the sequence of functions observed by Algorithm 16 and suppose they satisfy the conditions of Theorem 3.1. Let $\epsilon \in (0,1)$ and $\delta > 0$ be privacy parameters. Suppose $T \geq 1/(Lw)$. If $\lambda = \frac{\epsilon}{4H\sqrt{2T\ln(1/\delta)}}$, then Algorithm 16 is (ϵ, δ)-differentially private. Its regret is bounded by*

$$H\sqrt{T}\left(\frac{\epsilon}{4\sqrt{2\ln(1/\delta)}} + \frac{4\ln(RLT)\sqrt{2\ln(1/\delta)}}{\epsilon}\right) + Hk + 1.$$

Proof. This bound follows from applying Theorem B.3 using the (w', k)-disperse critical boundaries condition with $w' = 1/(LT)$. The lower bound on requirement on T ensures that $w' \leq w$. □

For multi-dimensional parameter spaces, we prove a similar theorem with respect to Algorithm 17.

Theorem B.4. *Let $u_1, \ldots, u_T$ be the sequence of functions observed by Algorithm 17 and suppose they satisfy the conditions of Theorem 3.1. Moreover, suppose $\sum_{t=1}^{T} u_t$ is piecewise concave on convex pieces. Let $\epsilon \in (0,1)$ and $\delta > 0$ be privacy parameters. Also, let $\epsilon' = \epsilon / \left(2\sqrt{2T\ln(2/\delta)}\right)$, $\lambda = \epsilon'/(6H)$, $\eta = \epsilon'/3$, and $\zeta = \delta / \left(2T\left(1 + e^{\epsilon'}\right)\right)$. Algorithm 17 with input λ, η, and ζ is (ϵ, δ)-differentially private. Moreover, its regret is bounded by*

$$\frac{H\epsilon}{12}\sqrt{\frac{T}{2\ln(1/\delta)}} + \frac{12Hd\ln(R/w)\sqrt{2T\ln(1/\delta)}}{\epsilon} + H\left(k + 2 + \frac{\delta}{2}\right) + LTw.$$

Moreover, suppose there are K intervals partitioning $\mathcal{C}$ so that $U(\mathcal{S}, \cdot)$ is piecewise L-Lipschitz on each interval. Then the running time of Algorithm 17 is $TK \cdot poly\left(d, H, T, K, \frac{1}{\epsilon}, \log\frac{1}{\delta}, \log\frac{R}{r}\right)$.

Proof. For all $t \in [T]$, the sensitivity of the function $\sum_{i=0}^{t-1} u_i$ is bounded by H. By Lemma B.16, at each time step t, Algorithm 17 samples from a distribution that is $(\epsilon', \delta/(2T))$-differentially private. By Lemma B.15, this means that Algorithm 17 is $(\epsilon, 2\delta)$-differentially private. The regret and running time bounds follow from Theorem B.1. □

B.1.5 Proofs for differential privacy (Section 3.1.4)

Theorem 3.4. *Let $u_1, \dots, u_T : \mathcal{C} \to [0, H]$ be piecewise L-Lipschitz and (w, k)-dispersed at the maximizer $\boldsymbol{\rho}^*$, and suppose that $\mathcal{C} \subset \mathbb{R}^d$ is convex, contained in a ball of radius R, and $B(\boldsymbol{\rho}^*, w) \subset \mathcal{C}$. For any $\epsilon > 0$, with probability at least $1 - \zeta$, the output $\hat{\boldsymbol{\rho}}$ of the exponential mechanism satisfies*

$$\frac{1}{T}\sum_{i=1}^{T} u_i\left(\hat{\boldsymbol{\rho}}\right) \geq \frac{1}{T}\sum_{i=1}^{T} u_i\left(\boldsymbol{\rho}^*\right) - O\left(\frac{H}{T\epsilon}\left(d\log\frac{R}{w} + \log\frac{1}{\zeta}\right) + Lw + \frac{Hk}{T}\right).$$

When $d = 1$, this algorithm is efficient, provided $f_{\exp}^{\epsilon}$ can be efficiently integrated on each piece of $\sum_i u_i$. For $d > 1$ we also provide an efficient approximate sampling algorithm when $\sum_i u_i$ is piecewise concave defined on K convex pieces. This algorithm preserves (ϵ, δ)-differential privacy for $\epsilon > 0$, $\delta > 0$ with the same utility guarantee (with $\zeta = \delta$). The running time of this algorithm is $\tilde{O}(K \cdot \mathrm{poly}(d, 1/\epsilon) + \mathrm{poly}(d, L, 1/\epsilon))$.

Proof. The proof follows the same outline as the utility guarantee for the exponential mechanism given by Dwork and Roth [63] when the set of outcomes is finite. The main additional challenge is lower bounding the normalizing constant for $f_{\exp}$, which is the key place where we use dispersion.

Let $f_{\exp}(\boldsymbol{\rho}) = \exp\left(\frac{\epsilon T}{2H} \cdot \frac{1}{T}\sum_{t=1}^{T} u_t(\boldsymbol{\rho})\right)$ be the unnormalized density sampled by the exponential mechanism. For a utility threshold c that we will set later, let $E = \{\boldsymbol{\rho} \in \mathcal{C} : \frac{1}{T}\sum_{t=1}^{T} u_t(\boldsymbol{\rho}) \leq c\}$ be the set of output points with average utility at most c. We can write the probability that a sample drawn from $f_{\exp}$ lands in E as F/Z, where $F = \int_E f_{\exp}$ and $Z = \int_{\mathcal{C}} f_{\exp}$. We bound F and Z independently.

First, we have

$$F = \int_E f_{\exp}(\boldsymbol{\rho})\, d\boldsymbol{\rho} \leq \int_E \exp\left(\frac{\epsilon T c}{2H}\right) d\boldsymbol{\rho} = \exp\left(\frac{\epsilon T c}{2H}\right) \cdot \mathrm{Vol}(E) \leq \exp\left(\frac{\epsilon T c}{2H}\right) \cdot \mathrm{Vol}(\mathcal{C}).$$

To lower bound Z, we use the fact that at most k of the functions $u_1, \dots, u_T$ have discontinuities in the ball $B(\boldsymbol{\rho}^*, w)$ and the rest are L-Lipschitz. This implies that every $\boldsymbol{\rho} \in B(\boldsymbol{\rho}^*, w)$ satisfies $\frac{1}{T}\sum_{t=1}^{T} u_t(\rho) \geq OPT - Lw - Hk/T$, where $OPT = \frac{1}{T}\sum_{t=1}^{T} u_t(\boldsymbol{\rho}^*)$. Therefore, we have

$$Z = \int_{\mathcal{C}} f_{\exp}(\boldsymbol{\rho})\, d\boldsymbol{\rho} \geq \int_{B(\boldsymbol{\rho}^*, w)} f_{\exp}(\boldsymbol{\rho})\, d\boldsymbol{\rho} \geq \exp\left(\frac{\epsilon T}{2H}(OPT - Lw - Hk/T)\right) \cdot \mathrm{Vol}(B(\boldsymbol{\rho}^*, w)).$$

Combining these bounds gives

$$\frac{F}{Z} \leq \exp\left(\frac{\epsilon T}{2H}(c - OPT + Lw + Hk/T)\right) \frac{\mathrm{Vol}(\mathcal{C})}{\mathrm{Vol}(B(\boldsymbol{\rho}^*, w))} \leq \exp\left(\frac{\epsilon T}{2H}(c - OPT + Lw + Hk/T)\right)\left(\frac{R}{w}\right)^d,$$

where the second inequality follows from the fact that $\mathcal{C}$ is contained in a ball of radius R, and the volume of a ball of radius r is proportional to r^d. Choosing c so that this bound on the probability of outputting a point with average utility at most c is at most ζ completes the proof.

Our efficient sampling algorithm is given in Algorithm 15. Given target privacy parameters $\epsilon > 0$ and $\delta > 0$, we use Algorithm 15 to approximately sample from the unnormalized density $g(\boldsymbol{\rho}) = \frac{\epsilon' T}{2H}\frac{1}{T}\sum_{t=1}^{T} u_t(\boldsymbol{\rho})$ with parameters $\epsilon' = \eta = \epsilon/3$ and $\zeta = \delta/(1 + e^{\epsilon})$. In Lemma B.16 we show that for these parameter settings, the algorithm preserves (ϵ, δ)-differential privacy and still has high utility. □

Next, as in the full-information online learning setting, we show that the utility dependence on the Lipschitz constant L can be made logarithmic. The main idea is that whenever functions are (w, k)-dispersed, they are also (w', k)-dispersed for any $w' \leq w$. By choosing w' sufficiently small, we are able to balance the Lw and $\frac{dH}{T\epsilon} \log \frac{R}{w}$ terms.

Corollary B.5. *Suppose the functions $u_1, \dots, u_T$ satisfy the conditions of Theorem 3.4 and $T \geq \frac{2Hd}{w\epsilon L}$. Then with probability at least $1 - \zeta$ the output $\hat{\boldsymbol{\rho}}$ sampled from $f_{\exp}$ satisfies:*

$$\frac{1}{T}\sum_{t=1}^{T} u_t(\hat{\boldsymbol{\rho}}) \geq \frac{1}{T}\sum_{t=1}^{T} u_t(\boldsymbol{\rho}^*) - O\left(\frac{H}{T\epsilon}\left(d \log \frac{L\epsilon RT}{2Hd}\left(+ \log \frac{1}{\zeta}\right) + \frac{Hk}{T}\right)\right.$$

Proof. If the functions $u_1, \dots, u_T$ are (w, k)-dispersed, then they are also (w', k)-dispersed for any $w' \leq w$. This bound follows from applying Theorem 3.4 using the (w', k)-dispersion with $w' = \frac{2Hd}{\epsilon LT}$. The bound on T ensures that $w' \leq w$. □

In all of our applications we show (w, k)-dispersion for $w \approx 1/\sqrt{T}$ and $k \approx \sqrt{T}$ (ignoring problem-specific parameters). In this case, the requirement on T becomes $T^{3/2} \geq \frac{2H}{\epsilon L}$, which will be satisfied for sufficiently large T.

Approximate sampling for differential privacy

Lemma B.16. *Let $u_1, \dots, u_T$ be piecewise L-Lipschitz and (w, k)-dispersed at a maximizer $\boldsymbol{\rho}^* \in \mathcal{C}$, and suppose that $\mathcal{C} \subset \mathbb{R}^d$ is convex, contained in a ball of radius R, and $B(\boldsymbol{\rho}^*, w) \subset \mathcal{C}$. For any privacy parameters $\epsilon > 0$ and $\delta > 0$, let $\hat{\boldsymbol{\rho}}$ be the output of running Algorithm 15 to sample from $g(\boldsymbol{\rho}) = \frac{T\epsilon'}{2H} \cdot \frac{1}{T}\sum_{t=1}^{T} u_t(\boldsymbol{\rho})$ with parameters $\eta = \epsilon' = \epsilon/3$ and $\zeta = \delta/(1 + e^{\epsilon})$. This procedure preserves (ϵ, δ)-differential privacy and with probability at least $1 - \delta$ we have*

$$\frac{1}{T}\sum_{t=1}^{T} u_t(\hat{\boldsymbol{\rho}}) \geq \frac{1}{T}\sum_{t=1}^{T} u_t(\boldsymbol{\rho}^*) - O\left(\frac{H}{T\epsilon}\left(d \log \frac{R}{w} + \log \frac{1}{\delta}\right) - Lw - \frac{Hk}{T}\right).$$

Proof. Let $u_1, \dots, u_T$ and $u'_1, \dots, u'_T$ be two neighboring sets of functions (that is, they differ on at most one function) and let $g(\boldsymbol{\rho}) = \frac{T\epsilon'}{2H} \cdot \frac{1}{T}\sum_{t=1}^{T} u_t(\boldsymbol{\rho})$ and $g'(\boldsymbol{\rho}) = \frac{T\epsilon'}{2H} \cdot \frac{1}{T}\sum_{t=1}^{T} u'_t(\boldsymbol{\rho})$. Let μ and μ' be the distributions with densities proportional to g and g', respectively. The distributions μ is the output distribution of the exponential mechanism when maximizing $\frac{1}{T}\sum_{t=1}^{T} u_t$ (and similarly for μ'). We know that exactly sampling from μ preserves $(\epsilon, 0)$-differential privacy and has strong utility guarantees. When we run Algorithm 15, we get approximate samples from μ and μ'. We need to show that the approximate sampling procedure still preserves (ϵ, δ)-differential privacy and has good utility.

Let $\hat{\boldsymbol{\rho}}$ and $\hat{\boldsymbol{\rho}}'$ be samples produced by Algorithm 15 when run on g and g', respectively. From Lemma B.10, we know that all approximate integration and sampling operations of Algorithm 15 succeed with probability at least $1 - \zeta$. Let $\hat{\mu}$ be the output distribution of Algorithm 15 when run on g conditioned on success for all integration and sampling operations (and similarly let $\hat{\mu}'$ be the distribution when run on g' without failures). Also by Lemma B.10, we know that $D_\infty(\hat{\mu}, \mu) \leq \eta$

and $D_\infty(\hat{\mu}', \mu') \leq \eta$. With this, for any set $E \subset \mathcal{C}$ of outcomes, we have

$$\begin{aligned}
\Pr(\hat{\boldsymbol{\rho}} \in E) &\leq \hat{\mu}(E) + \zeta && \text{(Failure probability of Algorithm 15)} \\
&\leq e^{\eta}\mu(E) + \zeta && (D_\infty(\hat{\mu}, \mu) \leq \eta) \\
&\leq e^{2\eta}\mu'(E) + \zeta && \text{(The exp. mech. preserves } \eta\text{-differential privacy)} \\
&\leq e^{3\eta}\hat{\mu}'(E) + \zeta && (D_\infty(\hat{\mu}', \mu') \leq \eta) \\
&\leq e^{3\eta}(\Pr(\hat{\boldsymbol{\rho}}' \in E) + \zeta) + \zeta && \text{(Failure probability of Algorithm 15)} \\
&= e^{\epsilon}\Pr(\hat{\boldsymbol{\rho}}' \in E) + \delta.
\end{aligned}$$

It follows that the approximate sampling procedure preserves (ϵ, δ)-differential privacy.

Next we turn to proving the utility guarantee. Let

$$E = \left\{ \boldsymbol{\rho} \in \mathcal{C} \,:\, \frac{1}{T}\sum_{t=1}^{T} u_t(\boldsymbol{\rho}) < \frac{1}{T}\sum_{t=1}^{T} u_t(\boldsymbol{\rho}^*) - \frac{2H}{T\eta}\left(d\log\frac{R}{w} + \log\frac{1}{\zeta}\right) - Lw - \frac{Hk}{|\mathcal{S}|} \right\},$$

be the set of parameter vectors with high suboptimality. By Theorem 3.4 we know that $\mu(E) \leq \zeta$. Applying Lemma B.10, we have

$$\Pr(\hat{\boldsymbol{\rho}} \in E) \leq \hat{\mu}(E) + \zeta \leq e^{\eta}\mu(E) + \zeta \leq (1 + e^{\eta})\zeta = \delta,$$

and the claim follows. □

Lower bound for differential privacy

Our privacy lower bounds follow a similar packing construction as the bounds given by De [54]. We will make use of the following simple Lemma arguing that we can pack many balls of radius r into the unit ball in d dimensions.

Lemma B.17. *For any dimension d and any radius $0 < r \leq 1/2$, there exist $t = (4r)^{-d}$ disjoint balls $B_1, \ldots, B_t$ of radius r contained in $B(0, 1)$.*

Proof. Let $\boldsymbol{\rho}_1, \ldots, \boldsymbol{\rho}_t \in B(0, 1/2)$ be any maximal set of points satisfying $\|\boldsymbol{\rho}_i - \boldsymbol{\rho}_j\|_2 \geq 2r$ for any $i \neq j$. First, we argue that $B(0, 1)$ is contained in $\bigcup_{i=1}^{t} B(\boldsymbol{\rho}_i, 2r)$. For contradiction, suppose there is some point $\boldsymbol{\rho} \in B(0, 1/2)$ that is not contained in $\bigcup_{i=1}^{t} B(\boldsymbol{\rho}_i, 2r)$. Then we must have that $\|\boldsymbol{\rho} - \boldsymbol{\rho}_i\|_2 \geq 2r$ for all r, which implies that it could be added to the list $\boldsymbol{\rho}_1, \ldots, \boldsymbol{\rho}_t$, contradicting maximality. From this, it follows that $\mathrm{Vol}(B(0, 1/2)) \leq \mathrm{Vol}(\bigcup_i B(\boldsymbol{\rho}_i, 2r))$. Using the fact that $\mathrm{Vol}(B(\cdot, r)) = r^d v_d$ and $\mathrm{Vol}(\bigcup_i B(\boldsymbol{\rho}_i, 2r)) \leq \sum_i \mathrm{Vol}(B(\boldsymbol{\rho}_i, 2r))$, this implies that $(1/2)^d v_d \leq t(2r)^d v_d$. Rearranging gives $t \geq (4r)^{-d}$.

Now consider the set of balls given by $B_i = B(\boldsymbol{\rho}_i, r)$. We know that $B_i \subset B(0, 1)$, since $\boldsymbol{\rho}_i \in B(0, 1/2)$ and $r \leq 1/2$. Moreover, since $\|\boldsymbol{\rho}_i - \boldsymbol{\rho}_j\|_2 \geq 2r$ for all $i \neq j$, we have that $B_i \cap B_j = \emptyset$ for all $i \neq j$. It follows that the set of balls $B_1, \ldots, B_t$ are disjoint and contained in $B(0, 1)$. □

With this, we are ready to prove our differential privacy lower bound.

Theorem 3.5. *For every dimension $d \geq 1$, privacy parameter $\epsilon > 0$, failure probability $\zeta > 0$, $T \geq \frac{d}{\epsilon}(\frac{\ln 2}{2} - \ln \frac{1}{\zeta})$) and ϵ-differentially private optimization algorithm $\mathcal{A}$ that takes as input a collection of T piecewise constant functions mapping $B(0,1) \subset \mathbb{R}^d$ to $[0,1]$ and outputs an approximate maximizer, there exists a multiset $\mathcal{S}$ of such functions so that with probability at least $1-\zeta$, the output $\hat{\boldsymbol{\rho}}$ of $\mathcal{A}(\mathcal{S})$ satisfies*

$$\frac{1}{T}\sum_{u\in\mathcal{S}} u(\hat{\boldsymbol{\rho}}) \leq \max_{\boldsymbol{\rho}\in B(0,1)} \frac{1}{T}\sum_{u\in\mathcal{S}} u(\boldsymbol{\rho}) - \Omega\left(\inf_{(w,k)} \frac{d}{T\epsilon}\left(\ln\frac{1}{w} - \ln\frac{1}{\zeta}\right) + \frac{k}{T}\right),$$

where the infimum is taken over all (w,k)-dispersion at the maximizer parameters satisfied by $\mathcal{S}$.

Proof. We will construct $M = 2^d$ multisets $\mathcal{S}_1, \ldots, \mathcal{S}_M$ of piecewise constant functions all satisfying the same (w,k)-dispersion parameters. We argue that for every ϵ-differentially private optimizer $\mathcal{A}$, there is at least one $\mathcal{S}_i$ such that $\mathcal{A}(\mathcal{S}_i)$ outputs a relatively suboptimal point with high probability. Next, we tune the parameters of the construction so that this suboptimality bound can be expressed in terms of the dispersion parameters w and k.

Set Construction. Let $\boldsymbol{\rho}_1, \ldots, \boldsymbol{\rho}_M$ be a collection of $M = 2^d$ points such that the balls $B(\boldsymbol{\rho}_i, 1/8)$ for $i = 1, \ldots, M$ are disjoint and contained in $B(0,1)$ (Lemma B.17 ensures that such a collection exists). Now define $u_{\text{all}}(\boldsymbol{\rho}) = \mathbb{I}\{\boldsymbol{\rho} \in \bigcup_{i=1}^M B(\boldsymbol{\rho}_i, 1/8)\}$ and $u_i(\boldsymbol{\rho}) = \mathbb{I}\{\boldsymbol{\rho} \in B(\boldsymbol{\rho}_i, r)\}$ for each $i = 1, \ldots, M$, where r is a parameter we will set later. Finally, for each index i, let $\mathcal{S}_i$ be the multiset of functions that contains N copies of u_i and $T - N$ copies of u_{all}, where N is a second parameter of the construction that we will set later.

Dispersion Parameters. For each set $\mathcal{S}_i$, we can exactly characterize the (w,k)-dispersion parameters at the maximizer. First, for $\mathcal{S}_i$, the point $\boldsymbol{\rho}_i$ is a maximizer with total utility T. On the other hand, any point outside $B(\boldsymbol{\rho}_i, r)$ has utility at most $T - N < T$. For any $w \leq r$, the ball $B(\boldsymbol{\rho}_i, w)$ is not split by any of the discontinuities of functions in $\mathcal{S}_i$, so the functions are $(w,0)$-dispersed at the maximizer. For $r < w \leq 1/8$, the ball $B(\boldsymbol{\rho}_i, w)$ is split by the discontinuities of the N copies of u_i, and so the functions are (w,N)-dispersed at the maximizer. Finally, for any $w > 1/8$, the functions are (w,T)-dispersed at the maximizer, since every function's discontinuity splits the ball. To summarize, the functions are (w,k)-dispersed at the maximizer for any w with

$$k = \begin{cases} 0 & \text{if } w < r \\ N & \text{if } r \leq w < 1/8 \\ T & \text{if } w \geq 1/8. \end{cases}$$

Suboptimality. Let $\mathcal{A}$ be any ϵ-differentially private optimizer for collections of piecewise constant functions. We first argue that running $\mathcal{A}$ on $\mathcal{S}_1$ must output a point with low utility for at least one of the other sets of functions $\mathcal{S}_i$ with high probability. Since the balls $B(\boldsymbol{\rho}_i, 1/8)$ are disjoint, we also know that the balls $B(\boldsymbol{\rho}_i, r)$ are also. Therefore, we have that $\sum_{i=1}^M \Pr(\mathcal{A}(\mathcal{S}_1) \in B(\boldsymbol{\rho}_i, r)) \leq 1$. But this implies that there exists some i such that $\Pr(\mathcal{A}(\mathcal{S}_1) \in B(\boldsymbol{\rho}_i, r)) \leq 1/M = 2^{-d}$. Given that any point outside of $B(\boldsymbol{\rho}_i, r)$ has suboptimality at least N for the set $\mathcal{S}_i$, it follows that $\mathcal{A}(\mathcal{S}_1)$ has suboptimality at least N for the functions in $\mathcal{S}_i$ with probability at least $1 - 2^{-d}$. Next, we show that this implies that $\mathcal{A}$ has low utility when run on $\mathcal{S}_i$ itself. Since $\mathcal{A}$ is ϵ-differentially private and the

sets of functions $\mathcal{S}_1$ and $\mathcal{S}_i$ differ only $2N$ functions (the N copies of u_1 in $\mathcal{S}_1$ and the N copies of u_i in $\mathcal{S}_i$), we have

$$\Pr(\mathcal{A}(\mathcal{S}_i) \in B(\boldsymbol{\rho}_i, r)) \leq e^{2\epsilon N} \Pr(\mathcal{A}(\mathcal{S}_1) \in B(\boldsymbol{\rho}_i, r)) \leq e^{2\epsilon N}/M$$

Therefore, with probability at least $1 - e^{2\epsilon N}/M$, the point $\mathcal{A}(\mathcal{S}_i)$ is N-suboptimal for $\mathcal{S}_i$.

Parameter Setting. There are two parameters in the above construction that we can set: r, the radius of the small optimal balls, and N, the number of copies of the indicator function for those small balls in each set of functions. Inuitively, we will set r to be small enough so that the dispersion parameters giving the best bound are $w = 1/8$ and $k = N$. Tuning the value of N is more involved.

Let r be small enough that $\frac{d}{\epsilon} \log \frac{1}{r} \geq \frac{d}{\epsilon} \log \frac{1}{8} + N$. For this value of r we have that

$$\inf_{w,k} \frac{d}{\epsilon} \log \frac{1}{w} + k = \frac{d}{\epsilon} \log \frac{1}{8} + N.$$

We also know that with probability at least $1 - e^{2\epsilon N}/M$, the suboptimality of algorithm $\mathcal{A}$ when run on $\mathcal{S}_i$ is at least N. Choosing the value of N trades between two competing effects: first, as we increase N, the suboptimality of $\mathcal{A}$ in the bad event that it outputs a point outside of $B(\boldsymbol{\rho}_i, r)$ get worse (formally, our suboptimality lower bound scales with N). Second, as we increase N, the datasets $\mathcal{S}_1, \ldots, \mathcal{S}_M$ become more different, and the probability of the bad event required by ϵ-differential privacy drops (formally, $1 - e^{2\epsilon N}/M$ gets smaller as N grows). We will have proved the theorem if we can find a value of N such that the probability $e^{2\epsilon N}/M \leq \zeta$ and $N = \Omega(\inf_{(w,k)} \frac{d}{\epsilon}(\log \frac{1}{w} - \log \frac{1}{\zeta}) + k)$. We will have $e^{2\epsilon N}/M \leq \zeta$ whenever $N \leq \frac{d}{\epsilon}(\frac{\ln 2}{2} - \ln \frac{1}{\zeta})$. Therefore, setting $N = \frac{d}{\epsilon}(\frac{\ln 2}{2} - \ln \frac{1}{\zeta})$ achieves the probability requirement. Finally, for this setting we have that $N = \Omega(N + N) = \Omega(\inf_{(w,k)} \frac{d}{\epsilon}(\log \frac{1}{w} - \ln \frac{1}{\zeta}) + k)$. For this setting to be justified, we must have $T \geq N = \frac{d}{\epsilon}(\frac{\ln 2}{2} - \ln \frac{1}{\zeta}))$.

Finally, this bound was on the total suboptimality. Dividing by T proves the theorem. □

Next, we show that the above lower bound can be instantiated by maximum weight independent set instances, showing that these lower bounds bind for algorithm configuration problems. In this case, the dimension of the problem is $d = 1$. To show this, we only need to construct MWIS instances for which the utility function of our greedy algorithm as a function of its parameter behaves like the indicator set for some subinterval of $[0, 1]$. The following Lemma shows that this can be achieved. For a graph x, let $u(x, \rho)$ be the total weight of the independent set returned by the algorithm parameterized by ρ.

Lemma B.18 (Gupta and Roughgarden [77])**.** *For any constants $0 < r < s < 1$ and any $t \geq 2$, there exists a MWIS instance x on $t^3 + 2t^2 + t - 2$ vertices such that $u(x, \rho) = 1$ when $\rho \in (r, s)$ and $u(x, \rho) = \frac{t^r(t^2-2)+t^{-s}(t^2+t+1)}{t^3-1}$ when $\rho \in [0, 1] \setminus (r, s)$.*

Corollary B.6. *For any constants $\frac{1}{10} < r < s < \frac{3}{20}$, there exists a MWIS instance x on 178 vertices such that $u(x, \rho) = 1$ when $\rho \in (r, s)$ and $\frac{2}{5} \leq u(x, \rho) \leq \frac{1}{2}$ when $\rho \in [0, 1] \setminus (r, s)$.*

While the Corollary B.6 does not show that the constructed instance behave exactly as indicator functions for subintervals, it demonstrates that for any interval $[r, s] \subset [\frac{2}{20}, \frac{3}{20}]$, we can construct a graph x so that the utility for any $\rho \in [r, s]$ is 1, and the utility for any $\rho \notin [r, s]$ is at most $1/2$. This additive gap is enough to instantiate Theorem 3.5 (after rescaling appropriately so that the construction is performed in the interval $[\frac{2}{20}, \frac{3}{20}]$).

B.1.6 Proofs for algorithm configuration (Section 3.1.5)

MWIS algorithm configuration

Theorem 3.6. *Suppose all vertex weights are in* $(0, 1]$ *and for each* $d^{(i)}$*, every pair of vertex weights has a* κ*-bounded joint distribution. For any* $\boldsymbol{w}$ *and* $\boldsymbol{e}$*,* $u(\boldsymbol{w}, \boldsymbol{e}, \cdot)$ *is piecewise 0-Lipschitz and for any* $\alpha \geq 1/2$*, with probability* $1 - \zeta$ *over* $\mathcal{S} \sim \bigtimes_{i=1}^{T} d^{(i)}$*,* u *is*

$$\left(\frac{1}{T^{1-\alpha}\kappa \ln n}, O\left(n^4 T^{\alpha}\sqrt{\ln \frac{n}{\zeta}}\right)\right) \textit{-dispersed}$$

with respect to $\mathcal{S}$*.*

Proof. Given a set of samples $\mathcal{S} = \left\{\left(\boldsymbol{w}^{(1)}, \boldsymbol{e}^{(1)}\right), \ldots, \left(\boldsymbol{w}^{(T)}, \boldsymbol{e}^{(T)}\right)\right\}$, Gupta and Roughgarden [77] prove that the $\sum_{t=1}^{T} u\left(\boldsymbol{w}^{(t)}, \boldsymbol{e}^{(t)}, \cdot\right)$ is piecewise constant and the boundaries between the constant pieces have the form

$$\frac{\ln\left(w_i^{(t)}\right) - \ln\left(w_j^{(t)}\right)}{\ln(d_1) - \ln(d_2)}$$

for all $t \in [T]$ and $i, j, d_1, d_2 \in [n]$, where $w_j^{(t)}$ is the weight of the j^{th} vertex of the t^{th} sample. For each unordered pair $(i, j) \in \binom{[n]}{2}$ and degrees $d_1, d_2 \in [n]$, let

$$\mathcal{B}_{i,j,d_1,d_2} = \left\{ \frac{\ln\left(w_i^{(t)}\right) - \ln\left(w_j^{(t)}\right)}{\ln(d_1) - \ln(d_2)} \;:\; t \in [T] \right\}.$$

The points in each set $\mathcal{B}_{i,j,d_1,d_2}$ are independent since they are determined by different problem instances. Since the vertex weights are supported on $(0, 1]$ and have pairwise κ-bounded joint densities, Lemma B.5 tells us that $\ln\left(w_i^{(t)}\right) - \ln\left(w_j^{(t)}\right)$ has a $\kappa/2$-bounded distribution for all $i, j \in [n]$ and $t \in [T]$. Also, since $|\ln(d_1) - \ln(d_2)| \leq \ln n$, Lemma B.7 allows us to conclude that the elements of each set $\mathcal{B}_{i,j,d_1,d_2}$ come from $\frac{\kappa \ln n}{2}$-bounded distributions. The theorem statement follows from Lemma 3.1 with $M = \max |\mathcal{B}_{i,j,d_1,d_2}| = T$ and $P = n^4/2$. □

Theorem B.5 (Differential privacy)**.** *Given a set of samples* $\mathcal{S} = \left\{\left(\boldsymbol{w}^{(1)}, \boldsymbol{e}^{(1)}\right), \ldots, \left(\boldsymbol{w}^{(T)}, \boldsymbol{e}^{(T)}\right)\right\} \sim d^T$*, suppose Algorithm 14 takes as input the function* $\sum_{t=1}^{T} u\left(\boldsymbol{w}^{(t)}, \boldsymbol{e}^{(t)}, \cdot\right)$ *and the set of intervals over which this function is piecewise constant. Suppose all vertex weights are in* $(0, 1]$ *and every pair of vertex weights has a* κ*-bounded joint distribution. Algorithm 14 returns a parameter* $\hat{\rho}$ *such that with probability at least* $1 - \zeta$ *over the draw of* $\mathcal{S}$*,*

$$\mathop{\mathbb{E}}_{(\boldsymbol{w},\boldsymbol{e})\sim d}\left[u(\boldsymbol{w}, \boldsymbol{e}, \hat{\rho})\right] \geq \max_{\rho \in [0,B]} \mathop{\mathbb{E}}_{(\boldsymbol{w},\boldsymbol{e})\sim d}\left[u(\boldsymbol{w}, \boldsymbol{e}, \rho)\right] - O\left(\frac{H}{T\epsilon}\log\frac{BT\kappa \ln n}{\zeta} + Hn^4\sqrt{\frac{\log(n/\zeta)}{T}}\right).$$

Proof. The theorem statement follows from Theorems 3.4 and 3.6 and Lemma B.19. □

Theorem B.6 (Full information online optimization)**.** *Let* $u\left(\boldsymbol{w}^{(1)}, \boldsymbol{e}^{(1)}, \cdot\right), \ldots, u\left(\boldsymbol{w}^{(T)}, \boldsymbol{e}^{(T)}, \cdot\right)$ *be the set of functions observed by Algorithm 16, where each instance* $\left(\boldsymbol{w}^{(t)}, \boldsymbol{e}^{(t)}\right)$ *is drawn from a distribution* $d^{(t)}$*. Suppose all vertex weights are in* $(0, 1]$ *and every pair of vertex weights has a* κ*-bounded joint distribution. Algorithm 16 with input parameter* $\lambda = \frac{1}{H}\sqrt{\frac{\ln\left(B\sqrt{T}\kappa \ln n\right)}{T}}$ *has regret bounded by* $\tilde{O}\left(n^4 H\sqrt{T}\right)$*.*

Proof. In Theorem 3.6, we show that with probability $1-\zeta$ over $\mathcal{S} \sim \bigtimes_{t=1}^{T} d^{(t)}$, u is

$$\left(\frac{1}{\sqrt{T}\kappa \ln n}, O\left(n^4\sqrt{T\ln(n/\zeta)}\right)\right)\text{-dispersed}$$

with respect to $\mathcal{S}$. Therefore, by Theorem 3.1, with probability at least $1-\zeta$, the expected regret of Algorithm 16 is at most $\tilde{O}\left(Hn^4\sqrt{T}\right)$. If this regret bound does not hold, then the regret is at most HT, but this only happens with probability ζ. Setting $\zeta = 1/\sqrt{T}$ gives the result. □

Theorem B.7 (Differentially private online optimization in the full information setting)**.** *Let*

$$u\left(\boldsymbol{w}^{(1)}, \boldsymbol{e}^{(1)}, \cdot\right), \ldots, u\left(\boldsymbol{w}^{(T)}, \boldsymbol{e}^{(T)}, \cdot\right)$$

be the set of functions observed by Algorithm 16, where each instance $\left(\boldsymbol{w}^{(t)}, \boldsymbol{e}^{(t)}\right)$ is drawn from a distribution $d^{(t)}$. Suppose all vertex weights are in $(0,1]$ and every pair of vertex weights has a κ-bounded joint distribution. Algorithm 16 with input parameter $\lambda = \frac{\epsilon}{4H\sqrt{2T\ln(1/\delta)}}$ is (ϵ,δ)-differentially private and has regret bounded by $\tilde{O}\left(H\sqrt{T}\left(1/\epsilon + n^4\right)\right)$.

Proof. The proof is exactly the same as the proof of Theorem B.6, except we rely on Theorem B.3 instead of Theorem 3.1 to obtain the regret bound. □

Theorem B.8 (Bandit feedback)**.** *Let $u\left(\boldsymbol{w}^{(1)}, \boldsymbol{e}^{(1)}, \cdot\right), \ldots, u\left(\boldsymbol{w}^{(T)}, \boldsymbol{e}^{(T)}, \cdot\right)$ be a sequence of functions where each instance $\left(\boldsymbol{w}^{(t)}, \boldsymbol{e}^{(t)}\right)$ is drawn from a distribution $d^{(t)}$. Suppose all vertex weights are in $(0,1]$ and every pair of vertex weights has a κ-bounded joint distribution. There is a bandit-feedback online optimization algorithm with regret bounded by $\tilde{O}\left(HT^{2/3}\left(\sqrt{B}+n^4\right)\right)$.*

Proof. In Theorem 3.6 with $\alpha = 2/3$, we show that with probability $1-\zeta$ over $\mathcal{S} \sim \bigtimes_{t=1}^{T} d^{(t)}$, u is

$$\left(\frac{1}{T^{1/3}\kappa \ln n}, O\left(n^4 T^{2/3}\sqrt{\ln(n/\zeta)}\right)\right)\text{-dispersed}$$

with respect to $\mathcal{S}$. Therefore, by Theorem 3.3 with $R = B$, with probability at least $1-\zeta$, there is a bandit-feedback algorithm with expected regret at most $\tilde{O}\left(HT^{2/3}\left(\sqrt{B}+n^4\right)\right)$. If this regret bound does not hold, then the regret is at most HT, but this only happens with probability ζ. Setting $\zeta = 1/T^{1/3}$ gives the result. □

Lemma B.19 ([77])**.** *Let $\left\{\left(\boldsymbol{w}^{(1)}, \boldsymbol{e}^{(1)}\right), \ldots, \left(\boldsymbol{w}^{(T)}, \boldsymbol{e}^{(T)}\right)\right\} \sim d^T$ be a set of samples. Then with probability at least $1-\zeta$, for all $\rho > 0$,*

$$\left|\frac{1}{T}\sum_{t=1}^{T} u\left(\boldsymbol{w}^{(t)}, \boldsymbol{e}^{(t)}, \rho\right) - \mathop{\mathbb{E}}_{(\boldsymbol{w},\boldsymbol{e})\sim d}\left[u\left(\boldsymbol{w}, \boldsymbol{e}, \rho\right)\right]\right| = O\left(H\sqrt{\frac{1}{T}\log\frac{n}{\zeta}}\right).$$

Knapsack algorithm configuration

In the knapsack problem, the input is a knapsack capacity C and a set of n items i each with a value v_i and a size s_i. The goal is to determine a set $I \subseteq \{1, \ldots, n\}$ with maximium total value $\sum_{i \in I} v_i$ such that $\sum_{i \in I} s_i \leq C$. We assume that $v_i \in (0, 1]$ for all $i \in [n]$. Gupta and Roughgarden [77] suggest the family of algorithms parameterized by $\rho \in [0, \infty)$ where each algorithm returns the better of the following two solutions:

- Greedily pack items in order of nonincreasing value v_i subject to feasibility.
- Greedily pack items in order of v_i / s_i^ρ subject to feasibility.

It is well-known that the algorithm with $\rho = 1$ achieves a 2-approximation. We consider the family of algorithms where we restrict the parameter ρ to lie in the interval $\mathcal{C} = [0, B]$ for some $B \in \mathbb{R}$. We model the distribution d over knapsack problem instances as a distribution over value-size-capacity tuples $(\boldsymbol{v}, \boldsymbol{s}, C) \in (0, 1]^n \times \mathbb{R}^n \times \mathbb{R}$. For a sample of knapsack problem instances $\mathcal{S} = \left\{ \left(\boldsymbol{v}^{(t)}, \boldsymbol{s}^{(t)}, C^{(t)} \right) \right\}_{t=1}^T$, we denote the value and size of item i under instance $\left(\boldsymbol{v}^{(t)}, \boldsymbol{s}^{(t)}, C^{(t)} \right)$ as $v_i^{(t)}$ and $s_i^{(t)}$. We use the notation $u\left(\boldsymbol{v}, \boldsymbol{s}, C, \rho\right)$ to denote the total value of the items returned by the algorithm parameterized by ρ given input $(\boldsymbol{v}, \boldsymbol{s}, C)$.

Gupta and Roughgarden [77] prove the following fact about the function u.

Lemma B.20 ([77])**.** *Given a set of samples $\left\{ \left(\boldsymbol{v}^{(t)}, \boldsymbol{s}^{(t)}, C^{(t)} \right) \right\}_{t=1}^T$, the function*

$$\sum_{t=1}^{T} u\left(\boldsymbol{v}^{(t)}, \boldsymbol{s}^{(t)}, C^{(t)}, \cdot \right)$$

is piecewise constant. It has at most Tn^2 constant pieces and the boundaries between constant pieces have the form

$$\frac{\ln\left(v_i^{(t)}\right) - \ln\left(v_j^{(t)}\right)}{\ln\left(s_i^{(t)}\right) - \ln\left(s_j^{(t)}\right)}$$

for all $t \in [T]$ and $i, j \in [n]$.

We now prove that dispersion holds under natural conditions.

Theorem B.9. *Suppose that every pair of item values has a κ-bounded joint distribution, every item size is in $[1, W]$, and the item values are independent from the item sizes. For any tuple $(\boldsymbol{v}, \boldsymbol{s}, C)$, $u(\boldsymbol{v}, \boldsymbol{s}, C, \cdot)$ is piecewise 0-Lipschitz. With probability at least $1 - \zeta$ over $\mathcal{S} \sim \times_{t=1}^T d^{(t)}$, for any $\alpha \geq 1/2$, u is $\left(\frac{1}{T^{1-\alpha} \kappa \ln W}, O\left(n^2 T^\alpha \sqrt{\ln \frac{n}{\zeta}} \right) \right)$-dispersed with respect to $\mathcal{S}$.*

Proof. Consider the following partitioning of the boundaries:

$$\mathcal{B}_{i,j} = \left\{ \frac{\ln\left(v_i^{(t)}\right) - \ln\left(v_j^{(t)}\right)}{\ln\left(s_i^{(t)}\right) - \ln\left(s_j^{(t)}\right)} \ : \ t \in [T] \right\}$$

for all $(i, j) \in \binom{[n]}{2}$. The points making up each $\mathcal{B}_{i,j}$ are all independent since they come from different samples. Since the values are supported on $(0, 1]$ and have pairwise κ-bounded joint densities,

Lemma B.5 tells us that $\ln\left(v_i^{(t)}\right) - \ln\left(v_j^{(t)}\right)$ has a $\kappa/2$-bounded distribution for all $i, j \in [n]$ and $t \in [T]$. Also, since $\left|\ln\left(s_i^{(t)}\right) - \ln\left(s_j^{(t)}\right)\right| \leq \ln W$ and the numerator of each element in $\mathcal{B}_{i,j}$ is independent from its denominator, Lemma B.6 implies that the elements of each $\mathcal{B}_{i,j}$ come from $\frac{\kappa \ln W}{2}$-bounded distributions. Applying Lemma 3.1 with $M = T$ and $P \leq n^2$ gives the result, since each bin $\mathcal{B}_{i,j}$ contains T elements and there are at most n^2 bins. □

Theorem B.10 (Differential privacy)**.** *Given a set of samples*

$$\mathcal{S} = \left\{\left(\boldsymbol{v}^{(1)}, \boldsymbol{s}^{(1)}, C^{(1)}\right), \ldots, \left(\boldsymbol{v}^{(T)}, \boldsymbol{s}^{(T)}, C^{(T)}\right)\right\} \sim d^T,$$

suppose Algorithm 14 takes as input the function $\sum_{t=1}^{T} u\left(\boldsymbol{v}^{(t)}, \boldsymbol{s}^{(t)}, C^{(t)}, \cdot\right)$ and the set of intervals over which this function is piecewise constant. Suppose that every pair of item values has a κ-bounded joint value distribution, every item size is in $[1, W]$, and the item values are independent from the item sizes. Algorithm 14 returns a parameter $\hat{\rho}$ such that with probability at least $1 - \zeta$ over the draw of $\mathcal{S}$,

$$\mathbb{E}[u\left(\boldsymbol{v}, \boldsymbol{s}, C, \hat{\rho}\right)] \geq \max_{\rho \in [0,B]} \mathbb{E}[u\left(\boldsymbol{v}, \boldsymbol{s}, C, \rho\right)] - O\left(\frac{H}{T\epsilon} \log \frac{BT\kappa \ln W}{\zeta} + Hn^2\sqrt{\frac{\log\left(n/\zeta\right)}{T}}\right).$$

Proof. The theorem statement follows from Theorems 3.4 and B.9 and Lemma B.21. □

Theorem B.11 (Full information online optimization)**.** *Let*

$$u\left(\boldsymbol{v}^{(1)}, \boldsymbol{s}^{(1)}, C^{(1)}, \cdot\right), \ldots, u\left(\boldsymbol{v}^{(T)}, \boldsymbol{s}^{(T)}, C^{(T)}, \cdot\right)$$

be the set of functions observed by Algorithm 16, where each instance $\left(\boldsymbol{v}^{(t)}, \boldsymbol{s}^{(t)}, C^{(t)}\right)$ is drawn from a distribution $d^{(t)}$. Suppose that every pair of item values has a κ-bounded joint distribution, every item size is in $[1, W]$, and the item values are independent from the item sizes. Algorithm 16 with input parameter $\lambda = \frac{1}{H}\sqrt{\frac{\ln\left(B\sqrt{T}\kappa \ln W\right)}{T}}$ has regret bounded by $\tilde{O}\left(Hn^2\sqrt{T}\right)$.

Proof. In Theorem B.9, we show that with probability $1 - \zeta$ over $\mathcal{S} \sim \times_{t=1}^{T} d^{(t)}$, u is

$$\left(\frac{1}{\sqrt{T}\kappa \ln W}, O\left(n^2\sqrt{T \ln \frac{n}{\zeta}}\right)\right)\text{-dispersed}$$

with respect to $\mathcal{S}$. Therefore, by Theorem 3.1, with probability at least $1 - \zeta$, the expected regret of Algorithm 16 is at most $\tilde{O}\left(Hn^2\sqrt{T}\right)$. If this regret bound does not hold, then the regret is at most HT, but this only happens with probability ζ. Setting $\zeta = 1/\sqrt{T}$ gives the result. □

Theorem B.12 (Differentially private online optimization in the full information setting)**.** *Let*

$$u\left(\boldsymbol{v}^{(1)}, \boldsymbol{s}^{(1)}, C^{(1)}, \cdot\right), \ldots, u\left(\boldsymbol{v}^{(T)}, \boldsymbol{s}^{(T)}, C^{(T)}, \cdot\right)$$

be the set of functions observed by Algorithm 16, where each instance $\left(\boldsymbol{v}^{(t)}, \boldsymbol{s}^{(t)}, C^{(t)}\right)$ is drawn from a distribution $d^{(t)}$. Suppose that every pair of item values has a κ-bounded joint distribution, every item size is in $[1, W]$, and the item values are independent from the item sizes. Algorithm 16 with input parameter $\lambda = \frac{\epsilon}{4H\sqrt{2T\ln(1/\delta)}}$ is (ϵ, δ)-differentially private and has regret bounded by $\tilde{O}\left(H\sqrt{T}\left(1/\epsilon + n^2\right)\right)$.

Proof. The proof is exactly the same as the proof of Theorem B.11, except we rely on Theorem B.3 instead of Theorem 3.1 to obtain the regret bound. □

Theorem B.13 (Bandit feedback)**.** *Let $u\left(\boldsymbol{v}^{(1)}, \boldsymbol{s}^{(1)}, C^{(1)}, \cdot\right), \ldots, u\left(\boldsymbol{v}^{(T)}, \boldsymbol{s}^{(T)}, C^{(T)}, \cdot\right)$ be a sequence of functions where each instance $\left(\boldsymbol{v}^{(t)}, \boldsymbol{s}^{(t)}, C^{(t)}\right)$ is drawn from a distribution $d^{(t)}$. Suppose that every pair of item values has a κ-bounded joint distribution, every item size is in $[1, W]$, and the item values are independent from the item sizes. There is a bandit-feedback online optimization algorithm with regret bounded by $\tilde{O}\left(HT^{2/3}\left(\sqrt{B}+n^2\right)\right)$.*

Proof. In Theorem B.9 with $\alpha = 2/3$, we show that with probability $1-\zeta$ over $\mathcal{S} \sim \bigtimes_{t=1}^{T} d^{(t)}$, u is

$$\left(\frac{1}{T^{1/3}\kappa \ln W}, O\left(n^2 T^{2/3}\sqrt{\ln(n/\zeta)}\right)\right)\text{-dispersed}$$

with respect to $\mathcal{S}$. Therefore, by Theorem 3.3 with $R = B$, with probability at least $1-\zeta$, there is a bandit-feedback algorithm with expected regret at most $\tilde{O}\left(HT^{2/3}\left(\sqrt{B}+n^2\right)\right)$. If this regret bound does not hold, then the regret is at most HT, but this only happens with probability ζ. Setting $\zeta = 1/T^{1/3}$ gives the result. □

Lemma B.21 ([77])**.** *Let $\left\{\left(\boldsymbol{v}^{(t)}, \boldsymbol{s}^{(t)}, C^{(t)}\right)\right\}_{t=1}^{T}$ be T knapsack problem instances sampled from d. Then with probability at least $1-\zeta$, for all $\rho \geq 0$,*

$$\left|\frac{1}{T}\sum_{t=1}^{T} u\left(\boldsymbol{v}^{(t)}, \boldsymbol{s}^{(t)}, C^{(t)}, \rho\right) - \mathop{\mathbb{E}}_{(\boldsymbol{v},\boldsymbol{s},C)\sim d}\left[u\left(\boldsymbol{v}, \boldsymbol{s}, C, \rho\right)\right]\right| = O\left(H\sqrt{\frac{\log\left(n/\zeta\right)}{T}}\right).$$

Outward rotation rounding algorithms

Algorithm 18 SDP rounding algorithm with rounding function $r : \mathbb{R} \to [-1, 1]$

Require: Matrix $A \in \mathbb{R}^{n\times n}$.

1: Solve the SDP

$$\text{maximize} \sum_{i,j\in[n]} a_{ij}\left\langle \boldsymbol{u}_i, \boldsymbol{u}_j\right\rangle \qquad \text{subject to } \boldsymbol{u}_i \in S^{n-1}$$

for the optimal embedding $U = \{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n\}$.

2: Draw $\boldsymbol{Z} \sim \mathcal{N}_n$.

3: For all $i \in [n]$, with probability $\left(1 + r\left(\left\langle \boldsymbol{Z}, \boldsymbol{u}_i\right\rangle\right)\right)/2$, set $z_i = 1$ and with probability $\left(1 - r\left(\left\langle \boldsymbol{Z}, \boldsymbol{u}_i\right\rangle\right)\right)/2$, set $z_i = -1$.

Ensure: $z_1, \ldots, z_n$.

Theorem 3.7. *For any matrix A and vector $\boldsymbol{Z}$, $u_{\text{owr}}\left(A, \boldsymbol{Z}, \cdot\right)$ is piecewise 0-Lipschitz. With probability $1-\zeta$ over $\boldsymbol{Z}^{(1)}, \ldots, \boldsymbol{Z}^{(T)} \sim \mathcal{N}_{2n}$, for any $A^{(1)}, \ldots, A^{(T)} \in \mathbb{R}^{n\times n}$ and any $\alpha \geq 1/2$, u_{owr} is*

$$\left(T^{\alpha-1}, O\left(nT^{\alpha}\sqrt{\log\frac{n}{\zeta}}\right)\right)\textit{-dispersed}$$

with respect to $\mathcal{S} = \left\{\left(A^{(t)}, \boldsymbol{Z}^{(t)}\right)\right\}_{t=1}^{T}$.

Algorithm 19 SDP rounding algorithm using γ-outward rotation

Require: Matrix $A \in \mathbb{R}^{n \times n}$

1: Solve the SDP

$$\text{maximize} \sum_{i,j \in [n]} a_{ij} \langle \boldsymbol{u}_i, \boldsymbol{u}_j \rangle \qquad \text{subject to } \boldsymbol{u}_i \in S^{n-1}$$

to obtain the optimal embedding $U = \{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n\}$.

2: Define a new embedding $\boldsymbol{u}'_i$ in $\mathbb{R}^{2n}$ as follows. The first n co-ordinates correspond to $\boldsymbol{u}_i \cos\gamma$ and the following n co-ordinates are set to 0 except the $(n+i)$th co-ordinate which is set to $\sin\gamma$.

3: Choose a random vector $\boldsymbol{Z} \in \mathbb{R}^{2n}$ according to the $2n$-dimensional Gaussian distribution.

4: For each decision variable z_i, assign $z_i = \operatorname{sign}\left(\langle \boldsymbol{u}'_i, \boldsymbol{Z} \rangle\right)$.

Ensure: $z_1, \ldots, z_n$.

Proof. Balcan et al. [18] prove that the function $\sum_{t=1}^{T} u_{\text{owr}}\left(A^{(t)}, \boldsymbol{Z}^{(t)}, \cdot\right)$ consists of $nT+1$ piecewise constant components. The discontinuities are of the form

$$\tan^{-1}\left(-\frac{\left\langle \boldsymbol{u}_i^{(j)}, \boldsymbol{Z}^{(j)}[1,\ldots,n] \right\rangle}{Z^{(j)}[n+i]}\right)$$

for each $\boldsymbol{u}_i^{(j)}$ in the optimal SDP embedding of each $A^{(j)}$. We show that the critical points are uniform random variables and thus are dispersed.

For an IQP instance A and its SDP embedding $\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n\}$, since each $\boldsymbol{u}_i$ is a unit vector, we know that $-\langle \boldsymbol{u}_i, \boldsymbol{Z}[1,\ldots,n] \rangle$ is a standard normal random variable. Therefore, $-\frac{\langle \boldsymbol{u}_i, \boldsymbol{Z}[1,\ldots,n] \rangle}{Z[n+i]}$ is a Cauchy random variable and $\tan^{-1}\left(-\frac{\langle \boldsymbol{u}_i, \boldsymbol{Z}[1,\ldots,n] \rangle}{Z[n+i]}\right)$ is a uniform random variable in the range $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$ [139, 30].

Define

$$\gamma_i^{(j)} = \tan^{-1}\left(-\frac{\left\langle \boldsymbol{u}_i^{(j)}, \boldsymbol{Z}^{(j)}[1,\ldots,n] \right\rangle}{Z^{(j)}[n+i]}\right).$$

For any two vectors $\boldsymbol{u}_i^{(j)}$ and $\boldsymbol{u}_i^{(k)}$ from different SDP embeddings, the random variables $\gamma_i^{(j)}$ and $\gamma_i^{(k)}$ are independent uniform random variables in $[-\pi/2, \pi/2]$. Therefore, we define the sets $\mathcal{B}_1, \ldots, \mathcal{B}_n$ such that $\mathcal{B}_i = \left\{\gamma_i^{(1)}, \ldots, \gamma_i^{(T)}\right\}$. Within each $\mathcal{B}_i$, the variables are independent. Therefore, by Lemma 3.1 with $P = n$, $M = \max|\mathcal{B}_i| = T$, and $\kappa = \pi$, the theorem statement holds. □

Theorem B.14 (Differential privacy)**.** *Given a set of samples $\mathcal{S} = \left\{\left(A^{(1)}, \boldsymbol{Z}^{(1)}\right), \ldots, \left(A^{(T)}, \boldsymbol{Z}^{(T)}\right)\right\} \sim \left(\mathcal{D} \times \mathcal{N}_{2n}\right)^T$, suppose Algorithm 14 takes as input the function $\sum_{t=1}^{T} u_{\text{owr}}\left(A^{(t)}, \boldsymbol{Z}^{(t)}, \cdot\right)$ and the set of intervals over which this function is piecewise constant. Algorithm 14 returns a parameter $\hat{\gamma}$ such that with probability at least $1-\zeta$ over the draw of $\mathcal{S}$,*

$$\mathop{\mathbb{E}}_{A, \boldsymbol{Z} \sim d \times \mathcal{N}_{2n}}\left[u_{\text{owr}}\left(A, \boldsymbol{Z}, \hat{\gamma}\right)\right] \geq \max_{\gamma \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]} \mathop{\mathbb{E}}_{A, \boldsymbol{Z} \sim d \times \mathcal{N}_{2n}}\left[u_{\text{owr}}\left(A, \boldsymbol{Z}, \gamma\right)\right] - O\left(\frac{H}{T\epsilon}\log\frac{T}{\zeta} + Hn\sqrt{\frac{1}{T}\log\frac{n}{\zeta}}\right).$$

Proof. The theorem statement follows from Theorems 3.4 and 3.7 and Lemma B.22. □

Theorem B.15 (Full information online optimization)**.** *Let* $u_{\text{owr}}\left(A^{(1)}, \boldsymbol{Z}^{(1)}, \cdot\right), \ldots, u_{\text{owr}}\left(A^{(T)}, \boldsymbol{Z}^{(T)}, \cdot\right)$ *be the set of functions observed by Algorithm 16, where each vector* $\boldsymbol{Z}^{(t)}$ *is drawn from* $\mathcal{N}_{2n}$*. Algorithm 16 with input parameter* $\lambda = \frac{1}{H}\sqrt{\frac{\ln(\pi\sqrt{T})}{T}}$ *has regret bounded by* $\tilde{O}\left(Hn\sqrt{T}\right)$*.*

Proof. In Theorem 3.6, we show that with probability $1-\zeta$ over $\boldsymbol{Z}^{(1)}, \ldots, \boldsymbol{Z}^{(T)} \sim \mathcal{N}_{2n}$, u_{owr} is $\left(\frac{1}{\sqrt{T}}, O\left(n\sqrt{T\log(n/\zeta)}\right)\right)$-dispersed with respect to $\mathcal{S} = \{(A^{(t)}, \boldsymbol{Z}^{(t)})\}_{t=1}^{T}$. Therefore, by Theorem 3.1, with probability at least $1-\zeta$, the expected regret of Algorithm 16 is at most $\tilde{O}\left(Hn\sqrt{T}\right)$. If this regret bound does not hold, then the regret is at most HT, but this only happens with probability ζ. Setting $\zeta = 1/\sqrt{T}$ gives the result. □

Theorem B.16 (Differentially private online optimization in the full information setting)**.** *Let*

$$u_{\text{owr}}\left(A^{(1)}, \boldsymbol{Z}^{(1)}, \cdot\right), \ldots, u_{\text{owr}}\left(A^{(T)}, \boldsymbol{Z}^{(T)}, \cdot\right)$$

be the set of functions observed by Algorithm 16, where each vector $\boldsymbol{Z}^{(t)}$ *is drawn from* $\mathcal{N}_{2n}$*. Algorithm 16 with input parameter* $\lambda = \frac{\epsilon}{4H\sqrt{2T\ln(1/\delta)}}$ *is* (ϵ, δ)*-differentially private and has regret bounded by* $\tilde{O}\left(H\sqrt{T}\left(1/\epsilon + n\right)\right)$*.*

Proof. The proof is exactly the same as the proof of Theorem B.15, except we rely on Theorem B.3 instead of Theorem 3.1 to obtain the regret bound. □

Theorem B.17 (Bandit feedback)**.** *Let* $u_{\text{owr}}\left(A^{(1)}, \boldsymbol{Z}^{(1)}, \cdot\right), \ldots, u_{\text{owr}}\left(A^{(T)}, \boldsymbol{Z}^{(T)}, \cdot\right)$ *be a sequence of functions where each vector* $\boldsymbol{Z}^{(t)}$ *is drawn from* $\mathcal{N}_{2n}$*. There is a bandit-feedback online optimization algorithm with regret bounded by* $\tilde{O}\left(HnT^{2/3}\right)$*.*

Proof. The proof is exactly the same as the proof of Theorem B.15, except we rely on Theorem 3.3 instead of Theorem 3.1 to obtain the regret bound. In this case, $\mathcal{C} = [0, \pi/2]$ and we take $\zeta = 1/T^{1/3}$.

In Theorem 3.7 with $\alpha = 2/3$, over $\boldsymbol{Z}^{(1)}, \ldots, \boldsymbol{Z}^{(T)} \sim \mathcal{N}_{2n}$, for any $A^{(1)}, \ldots, A^{(T)} \in \mathbb{R}^{n\times n}$, u_{owr} is $\left(\frac{1}{T^{1/3}}, O\left(nT^{1/3}\sqrt{\log(n/\zeta)}\right)\right)$-dispersed with respect to $\mathcal{S} = \{(A^{(t)}, \boldsymbol{Z}^{(t)})\}_{t=1}^{T}$. Therefore, by Theorem 3.3 with $R = \pi/2$, with probability at least $1-\zeta$, there is a bandit-feedback algorithm with expected regret at most $\tilde{O}\left(HnT^{2/3}\right)$. If this regret bound does not hold, then the regret is at most HT, but this only happens with probability ζ. Setting $\zeta = 1/T^{1/3}$ gives the result. □

Lemma B.22. *[[18]] Let* $\mathcal{S} = \left\{\left(A^{(1)}, \boldsymbol{Z}^{(1)}\right), \ldots, \left(A^{(T)}, \boldsymbol{Z}^{(T)}\right)\right\}$ *be* T *tuples sampled from* $d \times \mathcal{N}_{2n}$*. With probability at least* $1-\zeta$*, for all* $\gamma \in [-\pi/2, \pi/2]$*,*

$$\left|\frac{1}{T}\sum_{t=1}^{T} u_{\text{owr}}\left(A^{(t)}, \boldsymbol{Z}^{(t)}, \gamma\right) - \mathop{\mathbb{E}}_{A, \boldsymbol{Z}\sim d\times\mathcal{N}_{2n}}\left[u_{\text{owr}}\left(A, \boldsymbol{Z}, \gamma\right)\right]\right| < O\left(H\sqrt{\frac{\log\left(n/\zeta\right)}{T}}\right).$$

s-linear rounding algorithms

We make the following assumption, which is without loss of generality up to scaling, on the input matrices $A^{(1)}, \ldots, A^{(T)}$.

Assumption B.1. There exists a constant $H \in \mathbb{R}$ such that for any matrices $A^{(1)}, \ldots, A^{(T)}$ given as input to the algorithms in this chapter, $\sum_{i,j}\left|a_{ij}^{(t)}\right| \in [1, H]$ for all $t \in [T]$.

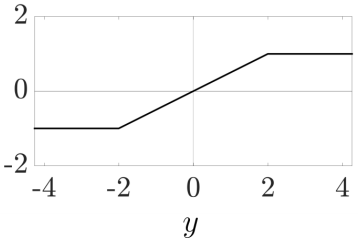


Figure B.1: A graph of the 2-linear function ϕ_2.

Theorem 3.8. *With probability $1-\zeta$ over $\boldsymbol{Z}^{(1)}, \ldots, \boldsymbol{Z}^{(T)} \sim \mathcal{N}_n$, for any matrices $A^{(1)}, \ldots, A^{(T)}$ and any $\alpha \geq 1/2$, the functions $u_{\text{slin}}\left(\boldsymbol{Z}^{(1)}, A^{(1)}, \cdot\right), \ldots, u_{\text{slin}}\left(\boldsymbol{Z}^{(T)}, A^{(T)}, \cdot\right)$ are piecewise L-Lipschitz with $L = \tilde{O}\left(MT^3n^5/\zeta^3\right)$, where $M = \max_{i,j\in[n],t\in[T]} |a_{ij}^{(t)}|$, and u_{slin} is*

$$\left(T^{\alpha-1}, O\left(nT^{\alpha}\sqrt{\log\frac{n}{\zeta}}\right)\right) \textit{-dispersed}$$

with respect to $\mathcal{S} = \left\{\left(A^{(t)}, \boldsymbol{Z}^{(t)}\right)\right\}_{t=1}^{T}$.

Proof. Balcan et al. [18] proved that $\sum_{t=1}^{T} u_{\text{slin}}\left(A^{(t)}, \boldsymbol{Z}^{(t)}, s\right)$ has the form

$$\sum_{t=1}^{T} u_{\text{slin}}\left(A^{(t)}, \boldsymbol{Z}^{(t)}, s\right) = \sum_{t=1}^{T}\left(\sum_{i=1}^{n}\left(a_{ii}^{(t)}\right)^2 + \sum_{i\neq j} a_{ij}^{(t)}\phi_s\left(\left\langle \boldsymbol{Z}^{(t)}, \boldsymbol{u}_i^{(t)}\right\rangle\right)\cdot\phi_s\left(\left\langle \boldsymbol{Z}^{(t)}, \boldsymbol{u}_j^{(t)}\right\rangle\right)\right)$$

and the function $\sum_{t=1}^{T} u_{\text{slin}}\left(A^{(t)}, \boldsymbol{Z}^{(t)}, \cdot\right)$ is made up of $Tn+1$ piecewise components of the form $\frac{a}{s^2}+\frac{b}{s}+c$ for some constants $a, b, c \in \mathbb{R}$. Let $\mathcal{B}_1, \ldots, \mathcal{B}_n$ be n sets of random variables such that $\mathcal{B}_i = \left\{\left|\left\langle \boldsymbol{u}_i^{(t)}, \boldsymbol{Z}^{(t)}\right\rangle\right| : t \in [T]\right\}$. Balcan et al. [18] proved that $\bigcup_{t=1}^{n}\mathcal{B}_t$ are all of the boundaries dividing the domain of $\sum_{t=1}^{T} u_{\text{slin}}\left(A^{(t)}, \boldsymbol{Z}^{(t)}, \cdot\right)$ into pieces over which the function is differentiable. Also, within each $\mathcal{B}_i$, the variables are all absolute values of independent standard Gaussians, since for any unit vector $\boldsymbol{u}$ and any $\boldsymbol{Z} \sim \mathcal{N}_n$, $\langle \boldsymbol{u}, \boldsymbol{Z}\rangle$ is a standard Gaussian. When Z is a Gaussian random variable, $|Z|$ is drawn from a $(4/5)$-bounded distribution. Therefore, the dispersion bound follows from Lemma 3.1 with $P = n$ and $M = \max|\mathcal{B}_i| = T$.

The main challenge in this proof is showing that for any $t \in [T]$, $u_{\text{slin}}\left(A^{(t)}, \boldsymbol{Z}^{(t)}, \cdot\right)$ is Lipschitz even when s approaches zero. We show that with probability at least $1-\zeta$, for all $t \in [T]$, $u_{\text{slin}}\left(A^{(t)}, \boldsymbol{Z}^{(t)}, \cdot\right)$ is constant on the interval $\left(0, 16MT^3n^5/\zeta^3\right)$. This way, we know that the derivative of $u_{\text{slin}}\left(A^{(t)}, \boldsymbol{Z}^{(t)}, \cdot\right)$ is zero as s goes to zero, not infinity.

Let s_0 be the smallest boundary between piecewise components of any function $u_{\text{slin}}\left(A^{(t)}, \boldsymbol{Z}^{(t)}, \cdot\right)$. In other words, for all $t \in [T]$, when $s \in (0, s_0)$, $u_{\text{slin}}\left(A^{(t)}, \boldsymbol{Z}^{(t)}, s\right)$ is differentiable and $u_{\text{slin}}\left(A^{(t)}, \boldsymbol{Z}^{(t)}, \cdot\right)$ is not differentiable at s_0. For all $s \in (0, s_0)$, all $i \in [n]$, and all $t \in [T]$, $\left|\left\langle \boldsymbol{u}_i^{(t)}, \boldsymbol{Z}^{(t)}\right\rangle\right| > s$. This means that $\phi_s\left(\left\langle \boldsymbol{u}_i^{(t)}, \boldsymbol{Z}^{(t)}\right\rangle\right) = \pm 1$. Therefore, for any $t \in [T]$, the derivative of $u_{\text{slin}}\left(A^{(t)}, \boldsymbol{Z}^{(t)}, \cdot\right)$ is zero on the interval $(0, s_0)$. In Lemma B.25, we prove that with probability $1-\zeta/2$, $s_0 \geq \frac{\zeta}{4nT}$.

We now bound the maximum absolute value of the derivative of any $u_{\text{slin}}\left(A^{(t)}, \boldsymbol{Z}^{(t)}, \cdot\right)$ for any $s > s_0$ where $u_{\text{slin}}\left(A^{(t)}, \boldsymbol{Z}^{(t)}, \cdot\right)$ is differentiable. We know that

$$\begin{aligned}\frac{d}{ds}u_{\text{slin}}\left(A^{(t)}, \boldsymbol{Z}^{(t)}, s\right) &= \frac{d}{ds}\left(\sum_{i=1}^{n}\left(a_{ii}^{(t)}\right)^2 + \sum_{i\neq j} a_{ij}^{(t)}\phi_s\left(\left\langle \boldsymbol{Z}^{(t)}, \boldsymbol{u}_i^{(t)}\right\rangle\right)\cdot\phi_s\left(\left\langle \boldsymbol{Z}^{(t)}, \boldsymbol{u}_j^{(t)}\right\rangle\right)\right) \\ &= \sum_{i\neq j} a_{ij}^{(t)}\frac{d}{ds}\left(\phi_s\left(\left\langle \boldsymbol{Z}^{(t)}, \boldsymbol{u}_i^{(t)}\right\rangle\right)\cdot\phi_s\left(\left\langle \boldsymbol{Z}^{(t)}, \boldsymbol{u}_j^{(t)}\right\rangle\right)\right).\end{aligned}$$

Therefore, we only need to bound $\left|\frac{d}{ds}\left(\phi_s\left(\left\langle \boldsymbol{Z}^{(t)}, \boldsymbol{u}_i^{(t)}\right\rangle\right)\cdot\phi_s\left(\left\langle \boldsymbol{Z}^{(t)}, \boldsymbol{u}_j^{(t)}\right\rangle\right)\right)\right|$ for all $i, j \in [n]$ and $t \in [T]$. We assume that

$$\max\left\{\left|\left\langle \boldsymbol{Z}^{(t)}, \boldsymbol{u}_i^{(t)}\right\rangle\right| : i \in [n], t \in [T]\right\} \leq \sqrt{2\ln\left(\sqrt{\frac{8}{\pi}}\frac{2nT}{\zeta}\right)},$$

which we know from Lemma B.28 happens with probability at least $1 - \zeta/2$. We also assume that $s_0 \geq \frac{\zeta}{4nT}$, which we know from Lemma B.25 also happens with probability at least $1 - \zeta/2$.

To this end, there are only three possible cases:

- **Case 1:** $\phi_s\left(\left\langle \boldsymbol{Z}^{(t)}, \boldsymbol{u}_i^{(t)}\right\rangle\right)\cdot\phi_s\left(\left\langle \boldsymbol{Z}^{(t)}, \boldsymbol{u}_j^{(t)}\right\rangle\right) = \pm 1$
- **Case 2:** $\phi_s\left(\left\langle \boldsymbol{Z}^{(t)}, \boldsymbol{u}_i^{(t)}\right\rangle\right)\cdot\phi_s\left(\left\langle \boldsymbol{Z}^{(t)}, \boldsymbol{u}_j^{(t)}\right\rangle\right) = \frac{\left\langle \boldsymbol{Z}^{(t)}, \boldsymbol{u}_i^{(t)}\right\rangle}{s}$
- **Case 3:** $\phi_s\left(\left\langle \boldsymbol{Z}^{(t)}, \boldsymbol{u}_i^{(t)}\right\rangle\right)\cdot\phi_s\left(\left\langle \boldsymbol{Z}^{(t)}, \boldsymbol{u}_j^{(t)}\right\rangle\right) = \frac{\left\langle \boldsymbol{Z}^{(t)}, \boldsymbol{u}_i^{(t)}\right\rangle\left\langle \boldsymbol{Z}^{(t)}, \boldsymbol{u}_j^{(t)}\right\rangle}{s^2}$.

In the first case, $\left|\frac{d}{ds}\left(\phi_s\left(\left\langle \boldsymbol{Z}^{(t)}, \boldsymbol{u}_i^{(t)}\right\rangle\right)\cdot\phi_s\left(\left\langle \boldsymbol{Z}^{(t)}, \boldsymbol{u}_j^{(t)}\right\rangle\right)\right)\right| = \left|\frac{d}{ds}\pm 1\right| = 0$. In the second case,

$$\begin{aligned}\left|\frac{d}{ds}\left(\phi_s\left(\left\langle \boldsymbol{Z}^{(t)}, \boldsymbol{u}_i^{(t)}\right\rangle\right)\cdot\phi_s\left(\left\langle \boldsymbol{Z}^{(t)}, \boldsymbol{u}_j^{(t)}\right\rangle\right)\right)\right| &= \left|\frac{d}{ds}\frac{\left\langle \boldsymbol{Z}^{(t)}, \boldsymbol{u}_i^{(t)}\right\rangle}{s}\right| \\ &= \left|\frac{\left\langle \boldsymbol{Z}^{(t)}, \boldsymbol{u}_i^{(t)}\right\rangle}{s^2}\right| \\ &\leq \frac{1}{s^2}\sqrt{2\ln\left(\sqrt{\frac{8}{\pi}}\frac{2nT}{\zeta}\right)} \qquad \text{(Lemma B.28)} \\ &\leq \frac{16n^2T^2}{\zeta^2}\sqrt{2\ln\left(\sqrt{\frac{8}{\pi}}\frac{2nT}{\zeta}\right)}. \qquad \text{(Lemma B.25)}\end{aligned}$$

In the third case,

$$\begin{aligned}\left|\frac{d}{ds}\left(\phi_s\left(\left\langle \boldsymbol{Z}^{(t)}, \boldsymbol{u}_i^{(t)}\right\rangle\right)\cdot\phi_s\left(\left\langle \boldsymbol{Z}^{(t)}, \boldsymbol{u}_j^{(t)}\right\rangle\right)\right)\right| &= \left|\frac{d}{ds}\frac{\left\langle \boldsymbol{Z}^{(t)}, \boldsymbol{u}_i^{(t)}\right\rangle\left\langle \boldsymbol{Z}^{(t)}, \boldsymbol{u}_j^{(t)}\right\rangle}{s^2}\right| \\ &= \left|\frac{2\left\langle \boldsymbol{Z}^{(t)}, \boldsymbol{u}_i^{(t)}\right\rangle\left\langle \boldsymbol{Z}^{(t)}, \boldsymbol{u}_j^{(t)}\right\rangle}{s^3}\right| \\ &\le \frac{2}{s^3}\cdot 2\ln\left(\sqrt{\frac{8}{\pi}}\frac{2nT}{\zeta}\right) \qquad \text{(Lemma B.28)} \\ &\le \frac{256n^3T^3}{\zeta^3}\ln\left(\sqrt{\frac{8}{\pi}}\frac{2nT}{\zeta}\right). \qquad \text{(Lemma B.25)}\end{aligned}$$

Since $\frac{16n^2T^2}{\zeta^2}\sqrt{2\ln\left(\sqrt{\frac{8}{\pi}}\frac{2nT}{\zeta}\right)} < \frac{256n^3T^3}{\zeta^3}\ln\left(\sqrt{\frac{8}{\pi}}\frac{2nT}{\zeta}\right)$, this derivative is maximized in the third case. Noting that $M = \max\left|a_{ij}^{(t)}\right|$, we have that for $s > s_0$,

$$\left|\frac{d}{ds}u_{\text{slin}}\left(A^{(t)}, \boldsymbol{Z}^{(t)}, s\right)\right| \le n^2 M\cdot\frac{256n^3T^3}{\zeta^3}\ln\left(\sqrt{\frac{8}{\pi}}\frac{2nT}{\zeta}\right) = \frac{256Mn^5T^3}{\zeta^3}\ln\left(\sqrt{\frac{8}{\pi}}\frac{2nT}{\zeta}\right).$$

□

Theorem B.18 (Differential privacy)**.** *Given a set of samples $\mathcal{S} = \left\{\left(A^{(1)}, \boldsymbol{Z}^{(1)}\right), \ldots, \left(A^{(T)}, \boldsymbol{Z}^{(T)}\right)\right\} \sim (\mathcal{D}\times\mathcal{N}_n)^T$ with $T \ge 8H^2n^2\ln\frac{8}{\zeta}$, suppose Algorithm 14 takes as input the function $\sum_{t=1}^{T} u\left(A^{(t)}, \boldsymbol{Z}^{(t)}, \cdot\right)$ and the set of intervals intersecting $\left(0, \sqrt{2\ln\left(\sqrt{8/\pi}\,(8nT/\zeta)\right)}\right)$ over which this function is piecewise constant. Algorithm 14 returns a parameter $\hat{s}$ such that with probability at least $1-\zeta$ over the draw of $\mathcal{S}$,*

$$\mathop{\mathbb{E}}_{A,\boldsymbol{Z}\sim d\times\mathcal{N}_n}\left[u_{\text{slin}}\left(A, \boldsymbol{Z}, \hat{s}\right)\right] \ge \max_{s>0}\mathop{\mathbb{E}}_{A,\boldsymbol{Z}\sim d\times\mathcal{N}_n}\left[u_{\text{slin}}\left(A, \boldsymbol{Z}, s\right)\right] - \tilde{O}\left(\frac{H}{T\epsilon} + \frac{Hn}{\sqrt{T}}\right).$$

Proof. First, in Theorem 3.8, we prove that with probability $1-\zeta/4$, the functions

$$u_{\text{slin}}(\boldsymbol{Z}^{(1)}, A^{(1)}, \cdot), \ldots, u_{\text{slin}}(\boldsymbol{Z}^{(T)}, A^{(T)}, \cdot)$$

are piecewise L-Lipschitz with $L = \frac{16384Mn^5T^3}{\zeta^3}\ln\left(\sqrt{\frac{8}{\pi}}\frac{8nT}{\zeta}\right)$ and u_{slin} is $\left(1/\sqrt{T}, O\left(n\sqrt{T\log(n/\zeta)}\right)\right)$-dispersed with respect to $\mathcal{S}$.

In Lemma B.29, we show that with probability $1-\zeta/4$, the values of s that maximize

$$\sum_{t=1}^{T} u_{\text{slin}}\left(A^{(t)}, \boldsymbol{Z}^{(t)}, \cdot\right)$$

lie within the interval $\left(0, \sqrt{2\ln\left(\sqrt{8/\pi}\,(8nT/\zeta)\right)}\right)$. Thus, we can restrict Algorithm 14 to searching for a parameter in this range.

We next show that with probability $1 - 3\zeta/4$,

$$\frac{1}{T}\left(\sum_{t=1}^{T} u_{\text{slin}}\left(\boldsymbol{Z}^{(t)}, A^{(t)}, \hat{s}\right) - \max_{s>0} u_{\text{slin}}\left(\boldsymbol{Z}^{(t)}, A^{(t)}, s\right)\right) = \tilde{O}\left(\frac{H}{T\epsilon} + \frac{Hn}{\sqrt{T}}\right) \tag{B.8}$$

If $L < H$, then this follows from Theorem 3.4. Otherwise, if $L \geq H$, it follows from Corollary B.5, assuming, as we can with probability $1 - \zeta/4$, that $\log(L) = \tilde{O}(1)$. Corollary B.5 only holds if $T \geq \frac{2H}{weL}$, which is the case when $L \geq H$ because $\frac{2H}{weL} < \frac{1}{w} = \sqrt{T} \leq T$.

In the last step of this proof, we show that since $\hat{s}$ is nearly optimal over the sample, it is nearly optimal over d as well. To do this, we call upon a result by Balcan et al. [18], which we include here as Lemma B.30. It guarantees that with probability at least $1 - \zeta/4$, for all $s > 0$, $\left|\frac{1}{T}\sum_{t=1}^{T} u_{\text{slin}}\left(A^{(t)}, \boldsymbol{Z}^{(t)}, s\right) - \mathbb{E}_{A,\boldsymbol{Z}\sim\mathcal{D}\times\mathcal{N}_n}\left[u_{\text{slin}}\left(A, \boldsymbol{Z}, s\right)\right]\right| < O\left(H\sqrt{\log\left(n/\zeta\right)/T}\right)$. Putting this together with Equation (B.8), the theorem statement holds. □

Theorem B.19 (Full information online optimization)**.** *Let* $u_{\text{slin}}\left(A^{(1)}, \boldsymbol{Z}^{(1)}, \cdot\right), \ldots, u_{\text{slin}}\left(A^{(T)}, \boldsymbol{Z}^{(T)}, \cdot\right)$ *be the set of functions observed by Algorithm 16, where* $T \geq 8H^2n^2\ln\frac{6}{\zeta}$ *and each vector* $\boldsymbol{Z}^{(t)}$ *is drawn from* $\mathcal{N}_n$*. Further, suppose we limit the parameter search space of Algorithm 16 to* $(0, \bar{s})$*, where* $\bar{s} = \sqrt{2\ln\left(\sqrt{\frac{8}{\pi}}\left(6nT/\zeta\right)\right)}$*. Algorithm 16 with input parameter* $\lambda = \frac{1}{H}\sqrt{\frac{\ln\left(\bar{s}\sqrt{T}\right)}{T}}$ *has regret bounded by* $\tilde{O}\left(Hn\sqrt{T}\right)$.

Proof. First, in Theorem 3.8, we prove that with probability $1 - \zeta/3$, the functions

$$u_{\text{slin}}(\boldsymbol{Z}^{(1)}, A^{(1)}, \cdot), \ldots, u_{\text{slin}}(\boldsymbol{Z}^{(T)}, A^{(T)}, \cdot)$$

are piecewise L-Lipschitz with $L = O\left(\frac{Mn^5T^3}{\zeta^3}\ln\left(\frac{nT}{\zeta}\right)\right)$ and u_{slin} is $\left(1/\sqrt{T}, O\left(n\sqrt{T\log(n/\zeta)}\right)\right)$-dispersed with respect to $\mathcal{S} = \left\{\left(A^{(1)}, \boldsymbol{Z}^{(1)}\right), \ldots, \left(A^{(T)}, \boldsymbol{Z}^{(T)}\right)\right\}$.

In Lemma B.29, we show that with probability $1 - \zeta/3$, the values of s that maximize

$$\sum_{t=1}^{T} u_{\text{slin}}\left(A^{(t)}, \boldsymbol{Z}^{(t)}, \cdot\right)$$

lie within the interval $\left(0, \sqrt{2\ln\left(\sqrt{8/\pi}\left(6nT/\zeta\right)\right)}\right)$. Thus, we can restrict Algorithm 14 to searching for a parameter in this range.

We now show that the expected regret of Algorithm 16 is at most $\tilde{O}\left(Hn\sqrt{T}\right)$. If $L < 1$, Theorem 3.1 guarantees that with probability at least $1 - \zeta$, the expected regret of Algorithm 16 is at most $\tilde{O}\left(Hn\sqrt{T}\right)$. Otherwise, if $L \geq 1$, we can apply Corollary B.2, which gives the same expected regret bound assuming $\log(L) = \tilde{O}(1)$, which we can assume with probability $1 - \zeta/3$. Corollary B.2 only holds when $T \geq \frac{1}{Lw}$, which is indeed with probability $1 - \zeta/3$ the case when $L \geq 1$ since $w = \sqrt{\frac{1}{T}}$.

If this regret bound does not hold, then the regret is at most HT, but this only happens with probability ζ. Setting $\zeta = 1/\sqrt{T}$ gives the result. □

Theorem B.20 (Differentially private online optimization in the full information setting)**.** *Let*

$$u_{\text{slin}}\left(A^{(1)}, \boldsymbol{Z}^{(1)}, \cdot\right), \ldots, u_{\text{slin}}\left(A^{(T)}, \boldsymbol{Z}^{(T)}, \cdot\right)$$

be the set of functions observed by Algorithm 16, where $T \geq 8H^2n^2 \ln \frac{6}{\zeta}$ *and each vector* $\boldsymbol{Z}^{(t)}$ *is drawn from* $\mathcal{N}_n$*. Let* $\epsilon, \delta > 0$ *be privacy parameters. Further, suppose we limit the parameter search space of Algorithm 16 to* $(0, \bar{s})$*, where* $\bar{s} = \sqrt{2 \ln\left(\sqrt{\frac{8}{\pi}}\left(6nT/\zeta\right)\right)}$*. Algorithm 16 with input parameter* $\lambda = \frac{\epsilon}{4H\sqrt{2T\ln(1/\delta)}}$ *is* (ϵ, δ)*-differentially private and has regret bounded by* $\tilde{O}\left(H\sqrt{T}\left(1/\epsilon + n\right)\right)$*.*

Proof. The proof is exactly the same as the proof of Theorem B.19, except we rely on Corollary B.4 instead of Corollary B.2 to obtain the regret bound. □

Lemma B.23 (Anthony and Bartlett [3])**.** *If* Z *is a standard normal random variable and* $x > 0$*, then* $\Pr[Z \geq x] \geq \frac{1}{2}\left(1 - \sqrt{1 - e^{-x^2}}\right)$*.*

Corollary B.7. *If* Z *is a standard normal random variable and* $x > 0$*, then* $\Pr[|Z| \geq x] \geq 1 - x$*.*

Proof.

$$\begin{aligned} \Pr[|Z| \leq x] &\leq \sqrt{1 - e^{-x^2}} && \text{(Lemma B.23)} \\ &\leq \sqrt{x^2} = x && \left(1 - e^{-\gamma} \leq \gamma \text{ for all } \gamma \in \mathbb{R}\right) \end{aligned}$$

□

Lemma B.24. *Suppose* $Z_1, \ldots, Z_\tau$ *are* τ *independent standard normal random variables. Then*

$$\Pr\left[\min_{i \in [\tau]} |Z_i| \leq \frac{\zeta}{2\tau}\right] \leq \zeta.$$

Proof. From Corollary B.7, we know that

$$\Pr\left[\min_{i \in [\tau]} |Z_i| \geq \frac{\zeta}{2\tau}\right] = \prod_{i=1}^{\tau} \Pr\left[|Z_i| \geq \frac{\zeta}{2\tau}\right] \geq \left(1 - \frac{\zeta}{2\tau}\right)^{\tau} \geq e^{-\zeta}.$$

The last inequality holds because for $\gamma \in [0, 3/4]$, we have that $1 - \gamma \geq e^{-2\gamma}$, which is applicable because $\frac{\zeta}{2\tau} < \frac{3}{4}$. Therefore,

$$\Pr\left[\min_{i \in [\tau]} |Z_i| \leq \frac{\zeta}{2\tau}\right] < 1 - e^{-\zeta} \leq \zeta.$$

□

Lemma B.25. *With probability at least* $1 - \zeta$*,* $\min\left\{\left|\left\langle \boldsymbol{Z}^{(t)}, \boldsymbol{u}_i^{(t)} \right\rangle\right| : i \in [n], t \in [T]\right\} \geq \frac{\zeta}{2nT}$*.*

Proof. Let $S_1, \ldots, S_n$ be n sets of random variables such that $S_i = \left\{ \left| \left\langle \boldsymbol{u}_i^{(t)}, \boldsymbol{Z}^{(t)} \right\rangle \right| : t \in [T] \right\}$. Notice that $\cup_{i=1}^n S_i$ are all of the boundaries dividing the domain of $\sum_{t=1}^T u_{\text{slin}}\left(A^{(t)}, \boldsymbol{Z}^{(t)}, \cdot\right)$ into intervals over which the function is differentiable. Also, within each S_i, the variables are all absolute values of independent Gaussians, since for any unit vector $\boldsymbol{u}$ and any $\boldsymbol{Z} \sim \mathcal{N}_n$, $\boldsymbol{u} \cdot \boldsymbol{Z}$ is a standard Gaussian. Lemma B.24 guarantees that for all $i \in [n]$, $\Pr\left[\min_{t\in[T]} \left\{ \left| \left\langle \boldsymbol{u}_i^{(t)}, \boldsymbol{Z}^{(t)} \right\rangle \right| \right\} \leq \frac{\zeta}{2nT}\right] \leq \frac{\zeta}{n}$. By a union bound, this means that with probability at least $1-\zeta$, $\min_{i\in[n],t\in[T]} \left\{ \left| \left\langle \boldsymbol{u}_i^{(t)}, \boldsymbol{Z}^{(t)} \right\rangle \right| \right\} \geq \frac{\zeta}{2nT}$. By definition of the sets $S_1, \ldots, S_n$ and the value s_0, this means that with probability at least $1-\zeta$, $s_0 \geq \frac{\zeta}{2nT}$. □

Lemma B.26. *If $T \geq 8H^2n^2 \ln \frac{1}{\zeta}$, with probability at least $1-\zeta$, there exists $s > 0$ such that*

$$\sum_{t=1}^{T} u_{\text{slin}}\left(A^{(t)}, \boldsymbol{Z}^{(t)}, s\right) \geq 0.$$

Proof. We will prove that with probability $1-\zeta$ over the draw of $\boldsymbol{Z}^{(1)}, \ldots, \boldsymbol{Z}^{(T)} \sim \mathcal{N}_n$,

$$\sum_{t=1}^{T} u_{\text{slin}}\left(A^{(t)}, \boldsymbol{Z}^{(t)}, \tilde{s}\right) \geq 0,$$

where $\tilde{s} = \frac{3}{2nT(10n+8)}$. From Lemma B.25, we know that with probability at least $1 - \frac{3}{10n+8}$, $\min\left\{ \left| \left\langle \boldsymbol{Z}^{(t)}, \boldsymbol{u}_i^{(t)} \right\rangle \right| : i \in [n], t \in [T] \right\} > \tilde{s}$. Recall that

$$\phi_s(y) = \begin{cases} \text{sign}(y) & \text{if } |y| \geq s \\ y/s & \text{if } |y| < s. \end{cases}$$

Therefore, when $\tilde{s} < \min\left\{ \left| \left\langle \boldsymbol{Z}^{(t)}, \boldsymbol{u}_i^{(t)} \right\rangle \right| : i \in [n], t \in [T] \right\}$, for all $t \in [T]$,

$$u_{\text{slin}}\left(A^{(t)}, \boldsymbol{Z}^{(t)}, \tilde{s}\right) = \sum_{i=1}^{n} a_{ii}^2 + \sum_{i\neq j} a_{ij} \,\text{sign}\left(\left\langle \boldsymbol{Z}^{(t)}, \boldsymbol{u}_i^{(t)} \right\rangle\right) \text{sign}\left(\left\langle \boldsymbol{Z}^{(t)}, \boldsymbol{u}_j^{(t)} \right\rangle\right). \tag{B.9}$$

Recall that the GW algorithm uses the rounding function $r(y) = \text{sign}(y)$. In other words, when the matrix $A^{(t)}$ is the input to Algorithm 18 and $\boldsymbol{Z}^{(t)}$ is the hyperplane drawn in Step 2, it sets $z_i = 1$ with probability $\frac{1}{2}\left(1 + \text{sign}\left(\left\langle \boldsymbol{Z}^{(t)}, \boldsymbol{u}_i^{(t)} \right\rangle\right)\right)$ and it sets $z_i = -1$ with probability $\frac{1}{2}\left(1 - \text{sign}\left(\left\langle \boldsymbol{Z}^{(t)}, \boldsymbol{u}_i^{(t)} \right\rangle\right)\right)$. In other words, it sets $z_i = \text{sign}\left(\left\langle \boldsymbol{Z}^{(t)}, \boldsymbol{u}_i^{(t)} \right\rangle\right)$. Therefore, Equation (B.9) is the objective value of the GW algorithm given the input matrix $A^{(t)}$ and hyperplane $\boldsymbol{Z}^{(t)}$. Since the GW algorithm has an expected approximation ratio of 0.878 (in expectation over the draw of the hyperplane),

$$\begin{aligned} &\mathop{\mathbb{E}}_{\boldsymbol{Z}^{(t)}\sim\mathcal{N}_n}\left[u_{\text{slin}}\left(A^{(t)}, \boldsymbol{Z}^{(t)}, \tilde{s}\right) \;\middle|\; \tilde{s} < \min\left\{ \left| \left\langle \boldsymbol{Z}^{(t)}, \boldsymbol{u}_i^{(t)} \right\rangle \right| : i \in [n], t \in [T] \right\}\right] \\ \geq\; & 0.878 \max_{\boldsymbol{z}\in\{0,1\}^n} \left\{ \sum_{i,j} a_{ij}^{(t)} z_i z_j \right\}. \end{aligned}$$

Charikar and Wirth [43] prove that $\max_{\boldsymbol{z}\in\{0,1\}^n}\left\{\sum_{i,j}a_{ij}^{(t)}z_iz_j\right\}\geq\frac{1}{n}\sum_{i,j}\left|a_{ij}^{(t)}\right|$. Therefore, using the notation E to denote the event where $\tilde{s}<\min\left\{\left|\left\langle\boldsymbol{Z}^{(t)},\boldsymbol{u}_i^{(t)}\right\rangle\right|:i\in[n],t\in[T]\right\}$, we know that

$$\mathop{\mathbb{E}}_{\boldsymbol{Z}^{(t)}\sim\mathcal{N}_n}\left[u_{\text{slin}}\left(A^{(t)},\boldsymbol{Z}^{(t)},\tilde{s}\right)\;\middle|\;E\right]\geq\frac{0.878}{n}\sum_{i,j}\left|a_{ij}^{(t)}\right|\geq\frac{4}{5n}\sum_{i,j}\left|a_{ij}^{(t)}\right|. \tag{B.10}$$

By the law of total expectation,

$$\begin{aligned}
&\mathop{\mathbb{E}}_{\boldsymbol{Z}^{(t)}\sim\mathcal{N}_n}\left[u_{\text{slin}}\left(A^{(t)},\boldsymbol{Z}^{(t)},\tilde{s}\right)\right]\\
=\;&\mathop{\mathbb{E}}_{\boldsymbol{Z}^{(t)}\sim\mathcal{N}_n}\left[u_{\text{slin}}\left(A^{(t)},\boldsymbol{Z}^{(t)},\tilde{s}\right)\;\middle|\;E\right]\cdot\Pr[E]+\mathop{\mathbb{E}}_{\boldsymbol{Z}^{(t)}\sim\mathcal{N}_n}\left[u_{\text{slin}}\left(A^{(t)},\boldsymbol{Z}^{(t)},\tilde{s}\right)\;\middle|\;\neg E\right]\cdot(1-\Pr[E])\\
\geq\;&\frac{4}{5n}\sum_{i,j}\left|a_{ij}^{(t)}\right|\cdot\Pr[E]+\mathop{\mathbb{E}}_{\boldsymbol{Z}^{(t)}\sim\mathcal{N}_n}\left[u_{\text{slin}}\left(A^{(t)},\boldsymbol{Z}^{(t)},\tilde{s}\right)\;\middle|\;\neg E\right]\cdot(1-\Pr[E])\\
\geq\;&\frac{4}{5n}\sum_{i,j}\left|a_{ij}^{(t)}\right|\cdot\Pr[E]-\sum_{i,j}\left|a_{ij}^{(t)}\right|\cdot(1-\Pr[E])\\
=\;&\sum_{i,j}\left|a_{ij}^{(t)}\right|\left(\Pr[E]\left(\frac{4}{5n}+1\right)-1\right)
\end{aligned}$$

where the second-to-last inequality follows from Equation (B.10) and the final inequality follows from the fact that with probability 1, $\left|u_{\text{slin}}\left(A^{(t)},\boldsymbol{Z}^{(t)},\tilde{s}\right)\right|\leq\sum_{i,j}\left|a_{ij}^{(t)}\right|$.

Since $\Pr[E]\geq1-\frac{3}{10n+8}$, we have that $\mathbb{E}_{\boldsymbol{Z}^{(t)}\sim\mathcal{N}_n}\left[u_{\text{slin}}\left(A^{(t)},\boldsymbol{Z}^{(t)},\tilde{s}\right)\right]\geq\frac{1}{2n}\sum_{i,j}\left|a_{ij}^{(t)}\right|\geq\frac{1}{2n}$. We now apply Hoeffding's to prove the result:

$$\begin{aligned}
&\Pr\left[\sum_{t=1}^{T}u_{\text{slin}}\left(A^{(t)},\boldsymbol{Z}^{(t)},\tilde{s}\right)\leq0\right]\\
=\;&\Pr\left[\mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T}u_{\text{slin}}\left(A^{(t)},\boldsymbol{Z}^{(t)},\tilde{s}\right)\right]-\frac{1}{T}\sum_{t=1}^{T}u_{\text{slin}}\left(A^{(t)},\boldsymbol{Z}^{(t)},\tilde{s}\right)\geq\mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T}u_{\text{slin}}\left(A^{(t)},\boldsymbol{Z}^{(t)},\tilde{s}\right)\right]\right]\\
\leq\;&\Pr\left[\mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T}u_{\text{slin}}\left(A^{(t)},\boldsymbol{Z}^{(t)},\tilde{s}\right)\right]-\frac{1}{T}\sum_{t=1}^{T}u_{\text{slin}}\left(A^{(t)},\boldsymbol{Z}^{(t)},\tilde{s}\right)\geq\frac{1}{2n}\right]\\
\leq\;&\exp\left(-\frac{2T^2}{16n^2\sum_{t=1}^{T}\left(\sum_{i,j}\left|a_{ij}^{(t)}\right|\right)^2}\right)\\
\leq\;&\exp\left(-\frac{T^2}{8n^2\cdot TH^2}\right)\\
\leq\;&\zeta
\end{aligned}$$

where the second-to-last inequality followsfrom the fact that with probability 1, for all $t\in[T]$, $\left|u_{\text{slin}}\left(A^{(t)},\boldsymbol{Z}^{(t)},\tilde{s}\right)\right|\leq\sum_{i,j}\left|a_{ij}^{(t)}\right|\leq H$. The final inequality follows from the fact that $T\geq8H^2n^2\ln\frac{1}{\zeta}$. □

Lemma B.27 (Gordon [75])**.** *Let Z be a standard normal random variable. Then* $\Pr[|Z| \geq z] \leq \frac{2}{z\sqrt{2\pi}} e^{-z^2/2}$.

Lemma B.28. *With probability at least* $1-\zeta$, $\max\left\{\left|\left\langle \boldsymbol{Z}^{(t)}, \boldsymbol{u}_i^{(t)} \right\rangle\right| : i \in [n], t \in [T]\right\} \leq \sqrt{2\ln\left(\sqrt{\frac{8}{\pi}}\frac{nT}{\zeta}\right)}$.

Proof. Let $z = \sqrt{2\ln\left(\sqrt{\frac{8}{\pi}}\frac{nT}{\zeta}\right)}$. We may assume that $n \geq 2$, which means that $z \geq 1$. Therefore, if Z is a standard Gaussian, by Lemma B.27, we know that $\Pr[|Z| \geq z] \leq \frac{2}{z\sqrt{2\pi}} e^{-z^2/2} \leq \sqrt{\frac{2}{\pi}} e^{-z^2/2}$. Let $S_1, \ldots, S_n$ be n sets of random variables such that $S_i = \left\{\left|\left\langle \boldsymbol{u}_i^{(t)}, \boldsymbol{Z}^{(t)} \right\rangle\right| : t \in [T]\right\}$. Notice that $\cup_{i=1}^n S_i$ are all of the boundaries dividing the domain of $\sum_{t=1}^T u_{\text{slin}}\left(A^{(t)}, \boldsymbol{Z}^{(t)}, \cdot\right)$ into intervals over which the function is differentiable. Also, within each S_i, the variables are all absolute values of independent Gaussians, since for any unit vector $\boldsymbol{u}$ and any $\boldsymbol{Z} \sim \mathcal{N}_n$, $\boldsymbol{u} \cdot \boldsymbol{Z}$ is a standard Gaussian. Therefore, for all $i \in [n]$, $\Pr\left[\max_{t\in[T]}\left\{\left|\left\langle \boldsymbol{Z}^{(t)}, \boldsymbol{u}_i^{(t)} \right\rangle\right|\right\} \leq z\right] \geq \left(1 - \sqrt{\frac{2}{\pi}} e^{-z^2/2}\right)^T$. By a union bound, this means that

$$\begin{aligned}
\Pr\left[\max_{i\in[n],t\in[T]}\left\{\left|\left\langle \boldsymbol{Z}^{(t)}, \boldsymbol{u}_i^{(t)} \right\rangle\right|\right\} \geq z\right] &\leq n\left(1 - \left(1 - \sqrt{\frac{2}{\pi}} e^{-z^2/2}\right)^T\right) \\
&= n\left(1 - \left(1 - \frac{\zeta}{2nT}\right)^T\right) \\
&\leq n\left(1 - e^{-\zeta/n}\right) && \left(\forall x \in (0, 3/4), e^{-2x} \leq 1 - x\right) \\
&\leq \zeta. && \left(\forall x \in \mathbb{R}, 1 - e^{-x} \leq x\right)
\end{aligned}$$

□

Lemma B.29. *If* $T \geq 8H^2n^2\ln\frac{2}{\zeta}$*, with probability at least* $1-\zeta$, $\operatorname{argmax}_{s>0}\sum_{t=1}^T u_{\text{slin}}\left(A^{(t)}, \boldsymbol{Z}^{(t)}, s\right) \leq \sqrt{2\ln\left(\sqrt{\frac{8}{\pi}}\frac{2nT}{\zeta}\right)}$.

Proof. Let $\bar{s} = \max\left\{\left|\left\langle \boldsymbol{Z}^{(t)}, \boldsymbol{u}_i^{(t)} \right\rangle\right| : i \in [n], t \in [T]\right\}$. From Lemma B.28, we know that with probability at least $1-\zeta/2$, $\bar{s} \leq \sqrt{2\ln\left(\sqrt{\frac{8}{\pi}}\frac{2nT}{\zeta}\right)}$. By definition of ϕ_s, when $s > \bar{s}$, $\sum_{t=1}^T u_{\text{slin}}\left(A^{(t)}, \boldsymbol{Z}^{(t)}, s\right) = a/s^2$ for some $a \in \mathbb{R}$. If $a \geq 0$, then $\sum_{t=1}^T u_{\text{slin}}\left(A^{(t)}, \boldsymbol{Z}^{(t)}, s\right)$ is non-increasing as s grows beyond $\bar{s}$, so the claim holds. If $a < 0$, then $\sum_{t=1}^T u_{\text{slin}}\left(A^{(t)}, \boldsymbol{Z}^{(t)}, s\right) < 0$ for all $s > \bar{s}$. However, by Lemma B.26, we know that with probability at least $1 - \zeta/2$, there exists some $s > 0$ such that $\sum_{t=1}^T u_{\text{slin}}\left(A^{(t)}, \boldsymbol{Z}^{(t)}, s\right) \geq 0$. Therefore, with probability $1-\zeta$, $\operatorname{argmax}_{s>0}\sum_{t=1}^T u_{\text{slin}}\left(A^{(t)}, \boldsymbol{Z}^{(t)}, s\right) \leq \bar{s}$. □

Lemma B.30. *[Balcan et al. [18]] Let* $\left(A^{(1)}, \boldsymbol{Z}^{(1)}\right), \ldots, \left(A^{(T)}, \boldsymbol{Z}^{(T)}\right)$ *be T tuples sampled from* $d \times \mathcal{N}_n$*. With probability at least* $1-\zeta$*, for all* $s > 0$*,*

$$\left|\frac{1}{T}\sum_{t=1}^T u_{\text{slin}}\left(A^{(t)}, \boldsymbol{Z}^{(t)}, s\right) - \mathop{\mathbb{E}}_{A,\boldsymbol{Z}\sim\mathcal{D}\times\mathcal{N}_n}\left[u_{\text{slin}}\left(A, \boldsymbol{Z}, s\right)\right]\right| = O\left(H\sqrt{\frac{\log\left(n/\zeta\right)}{T}}\right).$$

B.1.7 Proofs for distributional learning (Section 3.1.6)

We begin by recalling the definition of the pseudo-dimension of a class $\mathcal{F} = \{f : \Pi \to \mathbb{R}\}$ of real-valued functions. We say that the set $\mathcal{F}$ P-shatters a set $\mathcal{S} = \{x_1, \ldots, x_N\}$ if there exist thresholds $s_1, \ldots, s_N \in \mathbb{R}$ such that for all subsets $E \subseteq \mathcal{S}$ there exists $f \in \mathcal{F}$ such that $f(x_i) \geq s_i$ if $x_i \in E$ and $f(x_i) < s_i$ if $i \notin E$. The Pseudo-dimension of a class $\mathcal{F}$, denoted by $\mathrm{Pdim}(\mathcal{F})$ is the cardinality of the largest set $\mathcal{S}$ that is P-shattered by $\mathcal{F}$.

Theorem 3.9. *Let $\mathcal{F} = \{f_{\boldsymbol{\rho}} : \Pi \to [0,1] \ : \ \boldsymbol{\rho} \in \mathcal{C}\}$ be parameterized by $\mathcal{C} \subset \mathbb{R}^d$, where $\mathcal{C}$ lies in a ball of radius R. For any set $\mathcal{S} = \{x_1, \ldots, x_T\}$, suppose the functions $u_{x_i}(\boldsymbol{\rho}) = f_{\boldsymbol{\rho}}(x_i)$ for $i \in [T]$ are piecewise L-Lipschitz and (w,k)-dispersed. Then*

$$\hat{R}(\mathcal{F}, \mathcal{S}) \leq O\left(\min\left\{\sqrt{\frac{d}{T}\log\frac{R}{w}} + Lw + \frac{k}{T}, \sqrt{\frac{\mathrm{Pdim}(\mathcal{F})}{T}}\right\}\right).$$

Proof. The key idea is that whenever the functions $u_{x_1}, \ldots, u_{x_N}$ are (w,k)-dispersed, we know that any pair of parameters $\boldsymbol{\rho}$ and $\boldsymbol{\rho}'$ with $\|\boldsymbol{\rho} - \boldsymbol{\rho}'\|_2 \leq w$ satisfy $|f_{\boldsymbol{\rho}}(x_i) - f_{\boldsymbol{\rho}'}(x_i)| = |u_{x_i}(\boldsymbol{\rho}) - u_{x_i}(\boldsymbol{\rho}')| \leq Lw$ for all but at most k of the elements in $\mathcal{S}$. Therefore, we can approximate the functions in $\mathcal{F}$ on the set $\mathcal{S}$ with a finite subset $\hat{\mathcal{F}}_w = \{f_{\hat{\boldsymbol{\rho}}} \ : \ \hat{\boldsymbol{\rho}} \in \hat{\mathcal{C}}_w\}$, where $\hat{\mathcal{C}}_w$ is a w-net for $\mathcal{C}$. Since $\hat{\mathcal{F}}_w$ is finite, its empirical Rademacher complexity is $O((\log|\hat{\mathcal{F}}_w|/N)^{1/2})$. We then argue that the empirical Rademacher complexity of $\mathcal{F}$ is not much larger, since all functions in $\mathcal{F}$ are approximated by some function in $\hat{\mathcal{F}}_w$.

In particular, we know that there exists a subset $\hat{\mathcal{C}}_w \subset \mathcal{C}$ of size $|\hat{\mathcal{C}}_w| \leq (3R/w)^d$ (see Lemma B.13) such that for every $\rho \in \mathcal{C}$ there exists $\hat{\rho} \in \hat{\mathcal{C}}_w$ satisfying $\|\rho - \hat{\rho}\|_2 \leq w$. For any point $\rho \in \mathcal{C}$, let $\mathrm{NN}(\rho)$ denote a point in $\hat{\mathcal{C}}_w$ with $\|\rho - \mathrm{NN}(\rho)\|_2 \leq w$. Let $\hat{\mathcal{F}}_w = \{u_\rho : \Pi \to [0,1] \,|\, \rho \in \hat{\mathcal{C}} - w\}$ be the corresponding finite subset of $\mathcal{F}$.

Since $\hat{\mathcal{F}}_w$ is finite and the function range is $[0,1]$, we know that its empirical Rademacher complexity is at most

$$O\left(\sqrt{\frac{\log|\hat{\mathcal{F}}_w|}{N}}\right) = O\left(\sqrt{\frac{d\log(R/w)}{N}}\right).$$

Next, fix any $f_\rho \in \mathcal{F}$ and any vector $\sigma \in \{\pm 1\}^N$ of signs. We use (w,k)-dispersion to show that the correlation of $(f_\rho(x_1), \ldots, f_\rho(x_N))$ with σ cannot be substantially greater than the correlation of

$(f_{\mathrm{NN}(\rho)}(x_1), \ldots, f_{\mathrm{NN}(\rho)}(x_N))$ with σ.

$$\begin{aligned}
\frac{1}{N}\sum_{i=1}^{N}\sigma_i f_\rho(x_i) &= \frac{1}{N}\sum_{i=1}^{N}\sigma_i u_{x_i}(\rho) \\
&= \frac{1}{N}\sum_{i=1}^{N}\sigma_i u_{x_i}(\mathrm{NN}(\rho)) + \sum_{i=1}^{N}\sigma_i(u_{x_i}(\rho) - u_{x_i}(\mathrm{NN}(\rho))) \\
&\leq \frac{1}{N}\sum_{i=1}^{N}\sigma_i u_{x_i}(\mathrm{NN}(\rho)) + \sum_{i=1}^{N}|u_{x_i}(\rho) - u_{x_i}(\mathrm{NN}(\rho))| \\
&\leq \frac{1}{N}\sum_{i=1}^{N}\sigma_i u_{x_i}(\mathrm{NN}(\rho)) + Lw + \frac{k}{N} \\
&= \frac{1}{N}\sum_{i=1}^{N}\sigma_i f_{\mathrm{NN}(\rho)}(x_i) + Lw + \frac{k}{N}
\end{aligned}$$

Finally, we have

$$\begin{aligned}
\hat{R}(\mathcal{F}, S) &= \mathop{\mathbb{E}}_{\sigma\sim\{\pm1\}^N}\left[\sup_{f_\rho\in\mathcal{F}}\frac{1}{N}\sum_{i=1}^{N}\sigma_i f_\rho(x_i)\right] \\
&\leq \mathop{\mathbb{E}}_{\sigma\sim\{\pm1\}^N}\left[\sup_{f_\rho\in\mathcal{F}}\frac{1}{N}\sum_{i=1}^{N}\sigma_i f_{\mathrm{NN}(\rho)}(x_i)\right] + Lw + \frac{k}{N} \\
&= \mathop{\mathbb{E}}_{\sigma\sim\{\pm1\}^N}\left[\sup_{f_{\hat{\rho}}\in\hat{\mathcal{F}}_w}\frac{1}{N}\sum_{i=1}^{N}\sigma_i f_{\hat{\rho}}(x_i)\right] + Lw + \frac{k}{N} \\
&= O\left(\sqrt{\frac{d\log(R/w)}{N}} + Lw + \frac{k}{N}\right),
\end{aligned}$$

as required.

The bound on $\hat{R}(\mathcal{F}, \mathcal{S})$ in terms of the pseudo-dimension can be found in [117, 61]. □

B.1.8 Discretization-based algorithm

In this section we provide an implementation of the exponential mechanism achieving $(\epsilon, 0)$-differential privacy. It applies to multi-dimensional parameter spaces. First, we discretize the parameter space $\mathcal{C}$ using a regular grid (or any other net). We then apply the exponential mechanism to the resulting finite set of outcomes. Let $\hat{\boldsymbol{\rho}}$ be the resulting parameter. Standard guarantees for the exponential mechanism ensure that $\hat{\boldsymbol{\rho}}$ is neraly optimal over the discretized set. Therefore, the main challenge is showing that the net contains a parameter competitive with the optimal parameter in $\mathcal{C}$.

Theorem B.21. *Let $\mathcal{S} = \{x_1, \ldots, x_N\} \in \Pi$ be a collection of problem instances such that u is piecewise L-Lipschitz and (w, k)-disperse. Let $\boldsymbol{\rho}_1, \ldots, \boldsymbol{\rho}_K$ be a w-net for the parameter space $\mathcal{C}$. Let $\hat{\boldsymbol{\rho}}$ be set to $\boldsymbol{\rho}_i$ with probability proportional to $f_{\exp}^{\mathcal{S},\epsilon}(\boldsymbol{\rho}_i)$. Outputting $\hat{\boldsymbol{\rho}}$ satisfies $(\epsilon, 0)$-differential privacy and with probability at least $1 - \delta$ we have*

$$\frac{1}{N}\sum_{i=1}^{N}u(x_i, \hat{\boldsymbol{\rho}}) \geq \frac{1}{N}\sum_{i=1}^{N}u(x_i, \boldsymbol{\rho}^*) - \frac{2H}{N\epsilon}\log\frac{K}{\delta} - Lw - \frac{Hk}{N}.$$

Proof sketch. Since $\boldsymbol{\rho}_1, \ldots, \boldsymbol{\rho}_K$ is a w-net for the parameter space $\mathcal{C}$, we know there is some $\boldsymbol{\rho}_j$ within distance w of $\boldsymbol{\rho}^*$. Also, since $B(\boldsymbol{\rho}^*, w) \subset \mathcal{C}$, we know that $\boldsymbol{\rho}_j$ is a valid parameter vector. As in the proof of Theorem 3.4 we know that $\frac{1}{N}\sum_{i=1}^N u(x_i, \boldsymbol{\rho}_j) \geq \frac{1}{N}\sum_{i=1}^N u(x_i, \boldsymbol{\rho}^*) - \frac{Hk}{N} - Lw$. The result then follows from the standard analysis of the exponential mechanism, which guarantees that $\hat{\boldsymbol{\rho}}$ is competitive with the best $\boldsymbol{\rho}_j$ for $j \in \{1, \ldots, K\}$. □

This algorithm has strengths and weaknesses when compared to Algorithm 15. Recall, Algorithm 15 also applies to the multi-dimensional setting. The main strength is that this algorithm preserves pure $(\epsilon, 0)$-differential privacy. However, there are two significant disadvantages. First, it has running time exponential in the dimension since a w-net for $\mathcal{C}$ typically grows exponentially with dimension. Second, it requires knowledge of an upper bound on the dispersion parameter w in order to choose the granularity of the net. This prevents us from optimizing the utility guarantee over w as we did in Corollary B.5. Moreover, decreasing the parameter w increases the running time of the algorithm. This forces us to trade between computational cost and accuracy.

B.2 Appendix for Section 3.2

B.2.1 Online Optimization

Theorem 3.11. *Let $\mathcal{C} \subset \mathbb{R}^d$ be contained in a ball of radius R and $\ell_1, \ell_2, \cdots : \mathcal{C} \to [0,1]$ be piecewise L-Lipschitz functions that are f-dispersed with an r_0-interior minimizer. Moreover, suppose the learner gets semi-bandit feedback and, on each round t, the feedback system $A_1^{(t)}, \ldots, A_M^{(t)}$ has M feedback sets. For any $r \in (0, r_0]$, running Algorithm 5 with $\lambda = \sqrt{d\log(R/r)/(TM)}$ satisfies the following regret bound:*

$$\mathbb{E}\big[\sum_{t=1}^T \ell_t(\rho_t) - \ell_t(\rho^*)\big] \leq O\big(\sqrt{dTM\log(R/r)} + f(T,r) + TLr\big).$$

Proof of Theorem 3.11. For the majority of the proof we consider an arbitrary sequence of piecewise Lipschitz loss functions $\ell_1, \ldots, \ell_T$ with an r_0-interior minimizer. We will only suppose they are f-dispersed in the final steps of the proof.

Following the proof of the Exp3-Set algorithm of Alon et al. [2], we will upper and lower bound the quantity $\mathbb{E}[\log(W_{T+1}/W_1)]$. Our upper bound will be in terms of the learner's total expected loss, while the lower bound will be in terms of the expected total loss of the optimal parameter in hindsight. Dispersion plays a crucial role in the lower bound, since it allows us to guarantee that a set of parameters with non-trivial volume has nearly optimal total loss. Combining these bounds and then finally taking the expectation of the bound for a sequence of losses $\ell_1, \ldots, \ell_T$ that are f-dispersed will give the final bound.

Upper Bound. Consider the ratio of consecutive normalizing constants W_{t+1}/W_t. Using the definition of w_{t+1} and p_t, we have

$$\frac{W_{t+1}}{W_t} = \int_{\mathcal{C}} \frac{w_t(\rho)}{W_t} \exp(-\lambda\hat{\ell}_t(\rho))\, d\rho = \int_{\mathcal{C}} p_t(\rho)\exp(-\lambda\hat{\ell}_t(\rho))\, d\rho.$$

Next, using that $e^{-z} \leq 1 - z + z^2/2$ for all $z \geq 0$, we have

$$\frac{W_{t+1}}{W_t} \leq \int_{\mathcal{C}} p_t(\rho)\left(1 - \lambda\hat{\ell}_t(\rho) + \frac{\lambda^2}{2}\hat{\ell}_t(\rho)\right) d\rho = 1 - \lambda\int_{\mathcal{C}} p_t(\rho)\hat{\ell}_t(\rho)\, d\rho + \frac{\lambda^2}{2}\int_{\mathcal{C}} p_t(\rho)\hat{\ell}_t(\rho)^2\, d\rho.$$

Using the fact that $1 - z \leq \exp(-z)$ for all $z \geq 0$ and taking the product over $t = 1, \ldots, T$, we have

$$\frac{W_{T+1}}{W_1} \leq \exp\left(-\lambda \sum_{t=1}^{T} \int_{\mathcal{C}} p_t(\rho)\hat{\ell}_t(\rho)\, d\rho + \frac{\lambda^2}{2} \sum_{t=1}^{T} \int_{\mathcal{C}} p_t(\rho)\hat{\ell}_t(\rho)^2\, d\rho\right).$$

Taking logs, we have

$$\log(\frac{W_{T+1}}{W_1}) \leq -\lambda \sum_{t=1}^{T} \int_{\mathcal{C}} p_t(\rho)\hat{\ell}_t(\rho)\, d\rho + \frac{\lambda^2}{2} \sum_{t=1}^{T} \int_{\mathcal{C}} p_t(\rho)\hat{\ell}_t(\rho)^2\, d\rho. \tag{B.11}$$

Next, we will take the expectation of the above bound to simplify the two integrals. Recall that for each time t, we let $A_1^{(t)}, \ldots, A_M^{(t)}$ be the feedback system and for any $\rho \in \mathcal{C}$ and let $A^{(t)}(\rho)$ denote the set $A_i^{(t)}$ such that $\rho \in A_i^{(t)}$. Recall that the importance-weighted losses $\hat{\ell}_t$ were constructed to ensure that for any time t and any fixed $\rho \in \mathcal{C}$, we have $\mathbb{E}_t[\hat{\ell}_t(\rho)] = \ell_t(\rho)$. Therefore,

$$\mathbb{E}\left[\int_{\mathcal{C}} p_t(\rho)\hat{\ell}_t(\rho)\, d\rho\right] = \mathop{\mathbb{E}}_{<t}\left[\mathop{\mathbb{E}}_{t}\left[\int_{\mathcal{C}} p_t(\rho)\hat{\ell}_t(\rho)\, d\rho\right]\right] = \mathop{\mathbb{E}}_{<t}\left[\int_{\mathcal{C}} p_t(\rho)\ell_t(\rho)\, d\rho\right].$$

The integral in the final expectation is the definition of $\mathbb{E}_t[\ell_t(\rho_t)]$, which gives $\mathbb{E}\left[\int_{\mathcal{C}} p_t(\rho)\hat{\ell}_t(\rho)\, d\rho\right] = \mathbb{E}_{<t}[\mathbb{E}_t[\ell_t(\rho_t)]] = \mathbb{E}[\ell_t(\rho_t)]$. Therefore, we have

$$\mathbb{E}\left[\sum_{t=1}^{T} \int_{\mathcal{C}} p_t(\rho)\hat{\ell}_t(\rho)\, d\rho\right] = \mathbb{E}\left[\sum_{t=1}^{T} \ell_t(\rho_t)\right], \tag{B.12}$$

which is the total expected loss of the algorithm on the first T rounds.

Now we turn to simplifying the expectation of the second integral in (B.11). For any $\rho \in \mathcal{C}$ and any time t, we have

$$\mathop{\mathbb{E}}_{t}[\hat{\ell}_t(\rho)^2] = \int_{\mathcal{C}} p_t(\rho') \left(\frac{\mathbb{I}\{\rho \in A^{(t)}(\rho')\}}{p_t(A^{(t)}(\rho'))}\ell_t(\rho)\right)^2 d\rho.$$

Using the fact that $\rho \in A^{(t)}(\rho')$ if and only if $\rho' \in A^{(t)}(\rho)$, we can upper bound the integral as follows:

$$\int_{\mathcal{C}} p_t(\rho') \left(\frac{\mathbb{I}\{\rho \in A^{(t)}(\rho')\}}{p_t(A^{(t)}(\rho'))}\ell_t(\rho)\right)^2 d\rho = \left(\frac{\ell_t(\rho)}{p_t(A^{(t)}(\rho))}\right)^2 \cdot \int_{A^{(t)}(\rho)} p_t(\rho')\, d\rho = \frac{\ell_t(\rho)^2}{p_t(A^{(t)}(\rho))} \leq \frac{1}{p_t(A^{(t)}(\rho))}.$$

This implies that

$$\mathbb{E}\left[\int_{\mathcal{C}} p_t(\rho)\hat{\ell}_t(\rho)^2\, d\rho\right] = \mathop{\mathbb{E}}_{<t}\left[\mathop{\mathbb{E}}_{t}\left[\int_{\mathcal{C}} p_t(\rho)\hat{\ell}_t(\rho)^2\, d\rho\right]\right] \leq \mathbb{E}\left[\int_{\mathcal{C}} p_t(\rho)\frac{1}{p_t(A^{(t)}(\rho))}\, d\rho\right]$$

Finally, we evaluate the integral by writing it as the sum of integrals over the feedback sets $A_1^{(t)}, \ldots, A_M^{(t)}$, which is justified since these sets partition $\mathcal{C}$. In particular, we have

$$\int_{\mathcal{C}} p_t(\rho)\frac{1}{p_t(A^{(t)}(\rho))}\, d\rho = \sum_{i=1}^{M} \frac{1}{p_t(A_i^{(t)})} \cdot \int_{A_i^{(t)}} p_t(\rho)\, d\rho = M.$$

Putting it together, we have

$$\mathbb{E}\left[\sum_{t=1}^{T}\int_{\mathcal{C}} p_t(\rho)\hat{\ell}_t(\rho)^2\,d\rho\right] \le TM. \tag{B.13}$$

Taking the expectation of (B.11) and using the calculations given by (B.12) and (B.13), we have

$$\mathbb{E}\left[\log\frac{W_{T+1}}{W_1}\right] \le -\lambda\,\mathbb{E}\left[\sum_{t=1}^{T}\ell_t(\rho_t)\right] + \frac{\lambda^2}{2}TM.$$

Lower Bound. Next, let $\rho^* \in \operatorname{argmin}_{\rho\in\mathcal{C}}\sum_{t=1}^{T}\ell_t(\rho)$ be such that $B(\rho^*, r_0) \subset \mathcal{C}$ and fix any radius $r \le r_0$. Using the fact that $W_1 = \int_{\mathcal{C}} 1\,d\rho = \mathrm{Vol}(\mathcal{C})$ and the weights $w_{T+1}(\rho)$ are positive, we have

$$\frac{W_{T+1}}{W_1} = \frac{1}{\mathrm{Vol}(\mathcal{C})}\int_{\mathcal{C}} w_{T+1}(\rho)\,d\rho \ge \frac{1}{\mathrm{Vol}(\mathcal{C})}\int_{B(\rho^*,r)}\exp\left(-\lambda\sum_{t=1}^{T}\hat{\ell}_t(\rho)\right)\,d\rho.$$

Taking the log of this bounds gives

$$\log\frac{W_{T+1}}{W_1} \ge \log\frac{1}{\mathrm{Vol}(\mathcal{C})} + \log\left(\int_{B(\rho^*,r)}\exp\left(-\lambda\sum_{t=1}^{T}\hat{\ell}_t(\rho)\right)\,d\rho\right).$$

At this point it is tempting to apply dispersion to lower bound the term $\exp\bigl(-\lambda\sum_t\hat{\ell}_t(\rho)\bigr)$ in terms of $\exp\bigl(-\lambda\sum_t\hat{\ell}_t(\rho^*)\bigr)$. In particular, if at each time t the feedback system $A_1^{(t)},\ldots,A_M^{(t)}$ corresponds to the piecewise Lispchitz partitioning of $\mathcal{C}$ for the loss function ℓ_t, then the estimated loss function $\hat{\ell}_t$ has a subset of the discontinuities of ℓ_t. In this case, the estimated losses $\hat{\ell}_1, \hat{\ell}_2, \ldots$ are also f-dispersed for the same function f as the true losses. However, when the feedback system at around t does not match the piecewise Lipschitz partition, we would require a separate dispersion analysis for the sequence of estimated losses $\hat{\ell}_1, \hat{\ell}_2, \ldots$. The more serious challenge, however, is that the importance weight $1/p_t(A^{(t)}(\rho_t))$ in the definition of $\hat{\ell}_t$ causes it to take values in the range $[0, 1/p_t(A^{(t)}(\rho_t))]$, which is much larger than the true losses which take values in $[0,1]$. Moreover, the Lipschitz parameter of the estimated loss ℓ_t is $L' = L/p_t(A^{(t)}(\rho_t))$. This larger loss range and Lipschitz constant lead to a worse final regret bound. Instead, we defer applying dispersion until after taking expectations so that we can use the dispersion properties of the true losses $\ell_1, \ell_2, \ldots$ directly.

Towards this end, we first use Jensen's inequality to simplify the above bound. Let $h : \mathcal{C} \to \mathbb{R}$ be any function and $S \subset \mathcal{C}$ be any subset of the parameter space. Then, using the fact that $\log$ is concave, we can apply Jensen's inequality to obtain the following bound:

$$\begin{aligned}
\log\left(\int_S \exp(h(\rho))\,d\rho\right) &= \log\left(\mathrm{Vol}(S)\int_S \frac{1}{\mathrm{Vol}(S)}\exp(h(\rho))\,d\rho\right)\\
&= \log(\mathrm{Vol}(S)) + \log\left(\int_S \frac{1}{\mathrm{Vol}(S)}\exp(h(\rho))\,d\rho\right)\\
&\ge \log(\mathrm{Vol}(S)) + \int_S \frac{1}{\mathrm{Vol}(S)}\log(\exp(h(\rho)))\,d\rho\\
&= \log(\mathrm{Vol}(S)) + \int_S \frac{1}{\mathrm{Vol}(S)}h(\rho)\,d\rho,
\end{aligned}$$

Applying this inequality to our lower bound on $\log \frac{W_{T+1}}{W_1}$ with $h(\rho) = -\lambda \sum_{t=1}^{T} \hat{\ell}_t(\rho)$ and $S = B(\rho^*, r)$ gives

$$\log \frac{W_{T+1}}{W_1} \geq \log \frac{\mathrm{Vol}(B(\rho^*, r))}{\mathrm{Vol}(\mathcal{C})} - \lambda \int_{B(\rho^*, r)} \frac{1}{\mathrm{Vol}(B(\rho^*, r))} \sum_{t=1}^{T} \hat{\ell}_t(\rho)\, d\rho.$$

Next, since $\mathcal{C}$ is contained in a ball of radius R and the volume of a ball of radius R in $\mathbb{R}^d$ is proportional to R^d, it follows that the volume ratio is at least $(r/R)^d$. Taking expectations, we have

$$\mathbb{E}\left[\log \frac{W_{T+1}}{W_1}\right] \geq d \log \frac{r}{R} - \lambda \int_{B(\rho^*, r)} \frac{1}{\mathrm{Vol}(B(\rho^*, r))} \sum_{t=1}^{T} \ell_t(\rho)\, d\rho,$$

where we used the fact that for any fixed $\rho \in \mathcal{C}$, we have $\mathbb{E}[\hat{\ell}_t(\rho)] = \mathbb{E}_{<t}[\mathbb{E}_t[\hat{\ell}_t(\rho)]] = \ell_t(\rho)$. Finally, we will upper bound the sum of losses $\sum_{t=1}^{T} \ell_t(\rho)$ for points in the ball $B(\rho^*, r)$ in terms of the number of non-Lipschitz functions on that ball. let $D = \big|\{1 \leq t \leq T \,|\, \ell_t \text{ is not } L\text{-Lipschitz on } B(\rho^*, r)\}\big|$ be the number of functions among $\ell_1, \ldots, \ell_T$ that are not L-Lipschitz on the ball $B(\rho^*, r)$. Then for any $\rho \in B(\rho^*, r)$, we have

$$\sum_{t=1}^{T} \ell_t(\rho) \leq \sum_{t=1}^{t} \ell_t(\rho^*) + TLr + D.$$

The integral $\int_{B(\rho^*, r)} \frac{1}{\mathrm{Vol}(B(\rho^*, r))} \sum_{t=1}^{T} \ell_t(\rho)\, d\rho$ is the average total loss of the parameters $\rho \in B(\rho^*, r)$, which is at most $\sum_{t=1}^{t} \ell_t(\rho^*) + TLr + D$. Substituting this into our bound gives

$$\mathbb{E}\left[\log \frac{W_{T+1}}{W_1}\right] \geq d \log \frac{r}{R} - \lambda \sum_{t=1}^{T} \ell_t(\rho^*) - TLr - D.$$

Combined bound. Combining the upper and lower bounds and rearranging, we have

$$\sum_{t=1}^{T} \mathbb{E}[\ell_t(\rho_t)] - \ell_t(\rho^*) \leq \frac{\lambda}{2} TM + \frac{d}{\lambda} \log \frac{R}{r} + TLr + D.$$

Finally, now suppose that the functions $\ell_1, \ldots, \ell_T$ are a random sequence that satisfy f-dispersion. To bound $\mathbb{E}[D]$, let $\ell_1, \ldots, \ell_T$ be f-dispersed and ρ^* be any r_0-interior minimizer. Then we have

$$\begin{aligned}
\mathbb{E}[D] &= \mathbb{E}[\big|\{1 \leq t \leq T \,|\, \ell_t \text{ is not } L\text{-Lipschitz on } B(\rho^*, r)\}\big|] \\
&\leq \mathbb{E}[\sup_{\rho \in \mathcal{C}} \big|\{1 \leq t \leq T \,|\, \ell_t \text{ is not } L\text{-Lipschitz on } B(\rho, r)\}\big|] \\
&\leq f(T, \epsilon),
\end{aligned}$$

where this expectation is taken over the draw of the loss functions $\ell_1, \ldots, \ell_T$. This leads to the bound

$$\mathbb{E}\left[\sum_{t=1}^{T} \ell_t(\rho_t) - \ell_t(\rho^*)\right] \leq \frac{\lambda}{2} TM + \frac{d}{\lambda} \log \frac{R}{r} + TLr + f(T, r),$$

where now the expectation is taken over both the algorithm's randomness and the random sampling of the loss functions ℓ_t. The specific bounds given in the theorem statement follow by substituting the chosen value of λ. □

Optimizing Utilities and H-Bounded Losses

We note that the regret bound obtained in Theorem 3.11 for Algorithm 5 can also be used to obtain similar results in two closely related settings. First, if we instead have piecewise Lipschitz utility functions $u_1, u_2, \cdots : \mathcal{C} \to [0,1]$ and our goal is to maximize utility rather than minimize loss, we can transform this into a loss-minimization problem by minimizing the losses given by $\ell_t(\rho) = 1 - u_t(\rho)$. This transformation preserves the regret of any algorithm, the feedback system at each round, and the piecewise Lipschitz and dispersion properties of the functions. Similarly, if the losses take values in $[0, H]$ for some known maximum loss H, instead of $[0,1]$, the learner can preprocess the losses to fall in $[0,1]$ by dividing them by H. The rescaled functions take values in $[0,1]$ and have Lipschitz constant $L' = L/H$. Then expected regret of Algorithm 5 with respect to the unscaled loss functions is $O(H\sqrt{TMd\log(R/r)} + Hf(T,r) + TLr)$.

Lemma B.31. *Let $u_1, u_2, \cdots : \mathcal{C} \to [0, H]$ be a sequence of utility functions that are each piecewise L-Lipschitz and f-dispersed. Define a corresponding sequence of losses $\ell_1, \ell_2, \cdots : \mathcal{C} \to [0, H]$ given by $\ell_t(\rho) = H - u_t(\rho)$. The functions $\ell_1, \ell_2, \ldots$ are also piecewise L-Lispchitz and f-dispersed.*

Proof. First, consider any time t. Since $u_t : \mathcal{C} \to [0, H]$ is piecewise L Lipschitz, by definition we know that there is a partition $\mathcal{C}_1, \ldots, \mathcal{C}_M$ of $\mathcal{C}$ so that for each $i \in [M]$ and any $\rho, \rho' \in \mathcal{C}_i$, we have $|u_t(\rho) - u_t(\rho')| \leq L \cdot \|\rho - \rho'\|_2$. We will argue that the loss function ℓ_t is also piecewise L-lispschitz on the same partition. Fix any $i \in [M]$ and any pair of points $\rho, \rho' \in \mathcal{C}_i$. Then we have that

$$|\ell_t(\rho) - \ell_t(\rho')| = \big|(H - u_t(\rho)) - (H - u_t(\rho'))\big| = |u_t(\rho') - u_t(\rho)| \leq L \cdot \|\rho - \rho'\|_2,$$

where the last inequality follows from the fact that u_t is L-Lipschitz restricted to $\mathcal{C}_i$. It follows that ℓ_t is also piecewise L-Lipschitz and has the same piecewise Lipschitz partition. This holds for all times t.

Next, we argue that whenever the utility functions $u_1, u_2, \ldots$ are f-dispersed, so are the loss functions $\ell_1, \ell_2, \ldots$. For any time horizon T, radius $\epsilon > 0$, and parameter $\rho \in \mathcal{C}$, define

$$D_u(T, \epsilon, \rho) = \big|\{1 \leq t \leq T \,:\, u_t \text{ is not } L\text{-Lipschitz on } B(\rho, \epsilon)\}\big|$$

and

$$D_\ell(T, \epsilon, \rho) = \big|\{1 \leq t \leq T \,:\, \ell_t \text{ is not } L\text{-Lipschitz on } B(\rho, \epsilon)\}\big|.$$

Following an identical argument as in the first part, with probability 1, whenever u_t is L-Lipschitz on the ball $B(\rho, \epsilon)$, so is the function ℓ_t. From this, it follows that $D_u(T, \epsilon, \rho) = D_\ell(T, \epsilon, \rho)$ for all $T \in \mathbb{N}$, $\epsilon > 0$, and $\rho \in \mathcal{C}$. Finally, since the functions $u_1, u_2, \ldots$ were f-dispersed, we have that for all $T \in \mathbb{N}$ and all radiuses $\epsilon > 0$, we have

$$\mathbb{E}[\sup_{\rho \in \mathcal{C}} D_u(T, \epsilon, \rho)] \leq f(T, \epsilon).$$

It follows that

$$\mathbb{E}[\sup_{\rho \in \mathcal{C}} D_\ell(T, \epsilon, \rho)] = \mathbb{E}[\sup_{\rho \in \mathcal{C}} D_u(T, \epsilon, \rho)] \leq f(T, \epsilon),$$

and the loss functions $\ell_1, \ell_2, \ldots$ are also f-dispersed. □

Efficient Implementations via Interval Trees

In this section we show how to use the modified interval tree data structure of Cohen-Addad and Kanade [46] to implement the continuous Exp3-SET algorithm efficiently for one-dimensional problems with piecewise constant loss functions. In particular, the per-round cost of updating the algorithm weights and sampling from them at time t is only $O(\log(t))$, while a direct implementation has running time given by $O(t)$ instead. We also show how to use interval trees to implement the Exp3 algorithm on a set of N arms with per-round running time that is $O(\log N)$, which implies that a discretization-based algorithm in the bandit setting can be efficiently implemented even in high dimensions.

Interval Tree Summary. Cohen-Addad and Kanade [46] introduce a modified interval tree data structure used for representing piecewise constant functions mapping $\mathbb{R}$ to $\mathbb{R}$. Their data structure represents the function as a balanced tree with one leaf corresponding to each constant interval of the function. It supports two main operations called DRAW and UPDATE:

- The DRAW procedure returns a sample from the density function that is proportional to the represented piecewise constant function f.
- The UPDATE procedure takes an interval $[a, b)$ and an update u, and updates the represented piecewise function by multiplying the function values in $[a, b)$ by u. That is, if the represented function was originally $f : \mathbb{R} \to \mathbb{R}$, after executing UPDATE with interval $[a, b)$ and update u, the resulting function is
$$f'(x) = \begin{cases} f(x) & \text{if } x \notin [a, b) \\ u \cdot f(x) & \text{if } x \in [a, b). \end{cases}$$

Cohen-Addad and Kanade [46] show that the operations DRAW and UPDATE can be implemented in $O(\log P)$ time, where P is the number of constant pieces in the represented function. The data structure also makes it possible to implement a third procedure INTEGRATE in $O(\log P)$ time, which takes an interval $[a, b)$ and returns the integral of the represented represented function on the interval $[a, b)$.

Exp3-Set for Piecewise Constant One Dimensional Problems. First, we show how to efficiently implement Algorithm 5 efficiently for one-dimensional optimization problems with piecewise constant loss functions. We simply use the interval tree datastructure of Cohen-Addad and Kanade [46] to represent the weight function at each round. Pseudocode is given in Algorithm 20.

Lemma B.32. *Consider an online optimization problem with loss functions $\ell_1, \ell_2 : [0, 1] \to [0, 1]$ that are piecewise constant. Moreover, suppose that on each round t, the loss ℓ_t is constant on each of the feedback sets $A_i^{(t)}$. For such a problem, Algorithm 20 is equivalent to Algorithm 5. The overhead of sampling and updating the weights on round t takes $O(\log t)$ time.*

Proof. On each round we run UPDATE once to update the interval tree. This at most increases the number of constant intervals in the weights by 2, since the only constant intervals that might get split are the two containing the end points of the feedback set A_t. Therefore, on round t, the weight function is piecewise constant with at most $O(2t)$ intervals. It follows that the sampling, integration, and update operations all take $O(\log t)$ time, giving a total per-round cost of $O(\log t)$. □

Algorithm 20 Continuous Exp3-SET for Piecwise Constant One Dimensional Problems

Parameter: Step size $\lambda \in [0,1]$

1. Initialize W to be the interval tree representing $w(\rho) = \mathbb{I}\{\rho \in [0,1]\}$.
2. For $t = 1, \ldots, T$
   (a) Let $\rho_t \leftarrow \textsc{Draw}(W)$ and play ρ_t.
   (b) Observe feedback interval $A_t = A^{(t)}(\rho)$ and loss $\ell_t(\rho_t)$.
   (c) Let $\hat{\ell}_t \leftarrow \frac{\ell_t(\rho_t)}{p_t(A_t)}$, where $p_t(A_t) \leftarrow \frac{\textsc{Integrate}(W, A_t)}{\textsc{Integrate}(W, [0,1])}$.
   (d) Call $\textsc{Update}(W, A_t, \hat{\ell}_t)$.

B.2.2 Dispersion Tools

Theorem 3.12. *Let $\ell_1, \ell_2, \cdots : \mathbb{R} \to \mathbb{R}$ be independent piecewise L-Lipschitz functions, each having at most K discontinuities. Let $D(T, \epsilon, \rho) = \left|\{1 \le t \le T \mid \ell_t \text{ is not } L\text{-Lipschitz on } [\rho - \epsilon, \rho + \epsilon]\}\right|$ be the number of functions in $\ell_1, \ldots, \ell_T$ that are not L-Lipschitz on the ball $[\rho - \epsilon, \rho + \epsilon]$. Then we have*

$$\mathbb{E}[\sup_{\rho \in \mathbb{R}} D(T, \epsilon, \rho)] \le \sup_{\rho \in \mathbb{R}} \mathbb{E}[D(T, \epsilon, \rho)] + O(\sqrt{T \log(TK)}).$$

Proof of Theorem 3.12. For simplicity, we assume that every function has exactly K discontinuities. For each function ℓ_t, let $\alpha^{(t)} \in \mathbb{R}^K$ denote the vector whose entries are the discontinuity locations of ℓ_t. That is, ℓ_t has discontinuities at $\alpha_1^{(t)}, \ldots, \alpha_K^{(t)}$, but is otherwise L-Lispchitz. Since the functions $\ell_1, \ell_2, \ldots$ are independent, the vectors $\alpha^{(1)}, \alpha^{(2)}, \ldots$ are also independent.

For any interval $I \subset \mathbb{R}$, define the function $f_I : \mathbb{R}^K \to \{0,1\}$ by

$$f_I(\alpha) = \begin{cases} 1 & \text{if for some } i \in [K] \text{ we have } \alpha_i \in I, \text{ where } \alpha = (\alpha_1, \ldots, \alpha_K) \in \mathbb{R}^K \\ 0 & \text{otherwise} \end{cases}$$

The sum $\sum_{t=1}^T f_I(\alpha^{(t)})$ counts the number of vectors $\alpha^{(1)}, \ldots, \alpha^{(T)}$ that have a component in the interval I or, equivalently, the number of functions $\ell_1, \ldots, \ell_T$ that are not L-Lipschitz on I. We will apply VC-dimension uniform convergence arguments to the class $\mathcal{F} = \{f_I : \mathbb{R}^K \to \{0,1\} \mid I \subset \mathbb{R} \text{ is an interval}\}$. In particular, we will show that for an independent set of vectors $\alpha^{(1)}, \ldots, \alpha^{(T)}$, with high probability we have that $\frac{1}{T}\sum_{t=1}^T f_I(\alpha^{(t)})$ is close to $\mathbb{E}\left[\frac{1}{T}\sum_{t=1}^T f_I(\alpha^{(t)}\right]$ for all intervals I. This uniform convergence argument will lead to the desired bounds.

We begin by bounding the VC-dimension of $\mathcal{F}$ by $O(\log K)$. The key observation is the following connection between $\mathcal{F}$ and the class of indicator functions for intervals: let $S = \{x^{(1)}, \ldots, x^{(n)}\} \subset \mathbb{R}^K$ be any collection of n vectors in $\mathbb{R}^K$ and let $P = \{x_1^{(1)}, \ldots, x_K^{(1)}, \ldots, x_1^{(n)}, \ldots, x_K^{(n)}\} \subset \mathbb{R}$ denote the set containing the union of their combined nK component values. Now consider any pair of intervals I and I'. If the indicator functions for I and I' agree on all the points in P (i.e., the intervals contain exactly the same subset of P), then we must have that f_I and $f_{I'}$ agree on every vector in S. This is because if I and I' contain exactly the same subset of P, then for each vector $x^{(i)}$, both intervals contain the same subset of its component values. In particular, either they both contain none of the components, or they both contain at least one. In either case, we have that $f_I(x^{(i)}) = f_{I'}(x^{(i)})$. This shows that the number of distinct ways that functions from the class $\mathcal{F}$ can label the set of vectors S is at most the number of ways that indicator functions for intervals can label the set of points P.

Now suppose that the VC-dimension of $\mathcal{F}$ is V. Then there exists a set $S \subset \mathbb{R}^K$ of vectors of size V that is shattered by $\mathcal{F}$. Let $P \subset \mathbb{R}$ be the set containing the union of their combined VK components (as above). From Sauer's Lemma together with the fact that the VC-dimension of intervals is 2, we are guaranteed that indicator functions for intervals can label the set P of points in at most $(eVK)^2$ distinct ways. By the above reasoning, it follows that $\mathcal{F}$ can label the set S of vectors in at most $(eVK)^2$ distinct ways. On the other hand, since $\mathcal{F}$ shatters S, we know that it can label S in all 2^V possible ways, and it follows that $2^V \leq (eVK)^2$. Taking logs on both sides and rearranging, we have $V \leq \frac{2}{\ln 2}\ln(V) + \frac{2\ln(eK)}{\ln 2}$. Using the fact that for any $a \geq 1$ and $b \geq 0$, the inequality $y \leq a\ln(y) + b$ implies that $y \leq 4a\ln(2a) + 2b$, we further have that $V \leq \frac{8\ln(4/\ln 2)}{\ln 2} + \frac{4\ln(eK)}{\ln 2} = O(\log K)$, as required.

Applying VC-dimension uniform convergence arguments for the class $\mathcal{F}$, for any failure probability $\delta > 0$, if $x^{(1)}, \ldots, x^{(T)} \in \mathbb{R}^K$ are independent random vectors (but not necessarily identically distributed), then following holds with probability at least $1 - \delta$ simultaneously for all $f_I \in \mathcal{F}$:

$$\left|\frac{1}{T}\sum_{t=1}^{T}\ell_I(x^{(t)}) - \mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T}\ell_I(x^{(t)})\right]\right| \leq O\left(\sqrt{\frac{\mathrm{VCDim}(\mathcal{F}) + \log(1/\delta)}{T}}\right) = O\left(\sqrt{\frac{\log(K/\delta)}{T}}\right).$$

In particular, for any point ρ and any radius ϵ, we have that $D(T, \epsilon, \rho) = \sum_{t=1}^{T} f_I(\alpha^{(t)})$, where $I = [\rho - \epsilon, \rho + \epsilon]$. Therefore, uniform convergence for $\mathcal{F}$ implies that for all $T \in \mathbb{N}$ and all $\epsilon > 0$, and any failure probability $\delta > 0$, we have that with probability at least $1 - \delta$ the following holds for all $\rho \in \mathbb{R}$:

$$\left|\frac{1}{T}D(T, \epsilon, \rho) - \mathbb{E}\left[\frac{1}{T}D(T, \epsilon, \rho)\right]\right| \leq O\left(\sqrt{\frac{\log(K/\delta)}{T}}\right).$$

Multiplying both sides by T and rearranging gives

$$D(T, \epsilon, \rho) \leq \mathbb{E}[D(T, \epsilon, \rho)] + O(\sqrt{T\log(K/\delta)}).$$

Taking the supremum of both sides over $\rho \in \mathbb{R}$, we have

$$\sup_{\rho\in\mathbb{R}} D(T, \epsilon, \rho) \leq \sup_{\rho\in\mathbb{R}} \mathbb{E}[D(T, \epsilon, \rho)] + O(\sqrt{T\log(K/\delta)}).$$

This is a high probability bound on the maximum number of non-Lipschitz functions among $\ell_1, \ldots, \ell_T$ for any interval of radius ϵ. All that remains is to convert this into a bound in expectation. Let $\delta = 1/\sqrt{T}$ and let G denote the high-probability uniform convergence event above. Then we have

$$\begin{aligned}
\mathbb{E}[\sup_{\rho\in\mathbb{R}} D(T, \epsilon, \rho)] &= \mathbb{E}[\sup_{\rho\in\mathbb{R}} D(T, \epsilon, \rho) \mid G]\Pr(G) + \mathbb{E}[\sup_{\rho\in\mathbb{R}} D(T, \epsilon, \rho) \mid \overline{G}]\Pr(\overline{G}) \\
&\leq \sup_{\rho\in\mathbb{R}} \mathbb{E}[D(T, \epsilon, \rho)] + O(\sqrt{T\log(TK)}) + \sqrt{T} \\
&= \sup_{\rho\in\mathbb{R}} \mathbb{E}[D(T, \epsilon, \rho)] + O(\sqrt{T\log(TK)}),
\end{aligned}$$

where the last inequality uses the facts that $\Pr(G) \leq 1$ and $\mathbb{E}[\sup_{\rho\in\mathbb{R}} D(T, \epsilon, \rho) \mid \overline{G}]\Pr(\overline{G}) \leq T\delta = \sqrt{T}$. This argument holds for all T and ϵ, proving the claim. □

Next, we prove a weaker bound that follows from the analysis techniques of Balcan et al. [22].

Lemma B.33. *Let $\ell_1, \ell_2, \cdots : \mathbb{R} \to \mathbb{R}$ be independent piecewise L-Lipschitz functions, each having at most K discontinuities. Let $D(T, \epsilon, \rho) = \left|\{1 \le t \le T \,|\, \ell_t \text{ is not } L\text{-Lipschitz on } [\rho - \epsilon, \rho + \epsilon]\}\right|$ be the (random) number of functions in $\ell_1, \ldots, \ell_T$ that are not L-Lipschitz on the ball $[\rho - \epsilon, \rho + \epsilon]$. Moreover, let $\tilde{D}(T, \epsilon, \rho) = \left|\{(t, i) \in [T] \times [K] \,|\, \alpha_i^{(t)} \in [\rho - \epsilon, \rho + \epsilon]\}\right|$, where $\alpha^{(t)} \in \mathbb{R}^K$ is the vector of discontinuities of the loss ℓ_t. That is, $\tilde{D}(T, \epsilon, \rho)$ is the number of discontinuities of the functions $\ell_1, \ldots, \ell_T$ in the ball $[\rho - \epsilon, \rho + \epsilon]$. Then we have*

$$\mathbb{E}[\sup_{\rho \in \mathbb{R}} D(T, \epsilon, \rho)] \le \sup_{\rho \in \mathbb{R}} \mathbb{E}[\tilde{D}(T, \epsilon, \rho)] + K\sqrt{T \log(TK)}.$$

Note that, using the notation of Lemma B.33, we always have $D(T, \epsilon, \rho) \le \tilde{D}(T, \epsilon, \rho) \le KD(T, \epsilon, \rho)$. It follows that Lemma B.33 is looser than Theorem 3.12 in two ways: first, the error term is a factor K larger. Second, the upper bound of Lemma B.33 multiply-counts functions that have repeated discontinuities in the same ball, while our sharper bound does not.

Proof. For simplicity, we assume that every function ℓ_t has exactly K discontinuities. The proof techniques can be generalized to the case where each functions has at most K discontinuities.

For each time t, let $\alpha^{(t)} \in \mathbb{R}^K$ be the vector of discontinuities of ℓ_t. That is, ℓ_t has discontinuities at the points $\alpha_1^{(t)}, \ldots, \alpha_K^{(t)}$ and is otherwise L-Lipschitz. The key challenge is that the discontinuity locations $\alpha_1^{(t)}, \ldots, \alpha_K^{(t)}$ are not independent.

Fix any discontinuity index $i \in [K]$ and define $\tilde{D}_i(T, \epsilon, \rho) = \left|\{1 \le t \le T \,|\, \alpha_i^{(t)} \in [\rho - \epsilon, \rho + \epsilon]\}\right|$. That is, $\tilde{D}_i(T, \epsilon, \rho)$ counts the number of times t for which the i^{th} discontinuity $\alpha_i^{(t)}$ of ℓ_t lands in the interval of radius ϵ centered on ρ. Then we have that $\tilde{D}(T, \epsilon, \rho) = \sum_i \tilde{D}_i(T, \epsilon, \rho)$ counts the total number of discontinuities that belong to the interval of radius ϵ centered on ρ. Since the function ℓ_t is not L-Lipschitz on an interval I only when I contains some discontinuity for ℓ_t, we have

$$D(T, \epsilon, \rho) \le \sum_{i=1}^{K} \tilde{D}_i(T, \epsilon, \rho) = \tilde{D}(T, \epsilon, \rho).$$

Next we will apply uniform convergence arguments to obtain high probability bounds on each $\tilde{D}_i(T, \epsilon, \rho)$ in terms of their expectations. Fix a discontinuity index $i \in [K]$. The set of discontinuity locations $\alpha_i^{(1)}, \ldots, \alpha_i^{(T)}$ are independent and, since intervals have VC-dimension 2, applying standard uniform convergence guarantees implies that for any $\delta > 0$, with probability at least $1 - \delta$ the following holds for all ρ:

$$\tilde{D}_i(T, \epsilon, \rho) \le \mathbb{E}[\tilde{D}_i(T, \epsilon, \rho)] + O(\sqrt{T \log(1/\delta)}).$$

Setting the failure probability to be $1/(K\sqrt{T})$, taking the union bound over all K discontinuities, and summing the resulting bounds, the following holds with probability at least $1 - 1/\sqrt{T}$ for all ρ:

$$\begin{aligned}
\tilde{D}(T, \epsilon, \rho) &= \sum_{i=1}^{K} \tilde{D}_i(T, \epsilon, \rho) \\
&\le \mathbb{E}\left[\sum_{i=1}^{K} \tilde{D}_i(T, \epsilon, \rho)\right] + K \cdot O(\sqrt{T \log(KT)}) \\
&= \mathbb{E}[\tilde{D}(T, \epsilon, \rho)] + O(K\sqrt{T \log(KT)}).
\end{aligned}$$

Using the fact that $D(T,\epsilon,\rho) \le \tilde{D}(T,\epsilon,\rho)$ and taking the supremum over ρ, the following holds with probability at elast $1 - 1/\sqrt{T}$.

$$\sup_{\rho\in\mathbb{R}} D(T,\epsilon,\rho) \le \sup_{\rho\in\mathbb{R}} \mathbb{E}[\tilde{D}(T,\epsilon,\rho)] + O(K\sqrt{T\log(KT)}).$$

Let G denote the high-probility uniform convergence event above. Then we have

$$\begin{aligned}
\mathbb{E}[\sup_{\rho\in\mathbb{R}} D(T,\epsilon,\rho)] &= \mathbb{E}[\sup_{\rho\in\mathbb{R}} D(T,\epsilon,\rho) \mid G]\Pr(G) + \mathbb{E}[\sup_{\rho\in\mathbb{R}} D(T,\epsilon,\rho) \mid \overline{G}]\Pr(\overline{G}) \\
&\le \sup_{\rho\in\mathbb{R}} \mathbb{E}[\tilde{D}(T,\epsilon,\rho)] + O(K\sqrt{T\log(TK)}) + \sqrt{T} \\
&= \sup_{\rho\in\mathbb{R}} \mathbb{E}[\tilde{D}(T,\epsilon,\rho)] + O(K\sqrt{T\log(TK)}),
\end{aligned}$$

as required. □

B.2.3 Appendix for Applications

Lemma 3.5. *Consider an adversary choosing clustering instances where the t^{th} instance has symmetric distance matrix $D^{(t)} \in [0,B]^{n\times n}$ and for all $i \le j$, $d_{ij}^{(t)}$ is κ-smooth. The loss functions $\ell_1, \ell_2, \ldots$ defined above are piecewise constant and f-dispersed for $f(T,\epsilon) = 32T\epsilon n^8\kappa^2M^2 + O(\sqrt{T\log(Tn)})$ and β-dispersed for $\beta = 1/2$.*

Proof. The key insight of Balcan et al. [18] for this family of algorithms is that for a fixed distance matrix D, the function $\rho \mapsto \mathcal{A}_\rho(D)$ is piecewise constant with at most $O(n^8)$ pieces. That is, the algorithm will only output at most $O(n^8)$ different cluster trees, and each is produced for some subinterval of the parameter space. Their argument is as follows: for any pair of candidate cluster merges, say merging clusters C_1 and C_2 versus C_1' and C_2', we can determine the values of the parameter $\rho \in [0,1]$ for which the algorithm would prefer to merge (C_1, C_2) instead of merging (C_1', C_2') (i.e., the values of ρ so that the d_ρ distance between C_1 and C_2 is smaller than between C_1' and C_2'). In particular, the algorithm will merge clusters C_1 and C_2 instead of C_1' and C_2' if $\mathrm{d}_\rho(C_1,C_2) \le \mathrm{d}_\rho(C_1',C_2')$ or, equivalently, when

$$(1-\rho)\,\mathrm{d}_{\min}(C_1,C_2) + \rho\,\mathrm{d}_{\max}(C_1,C_2) \le (1-\rho)\,\mathrm{d}_{\min}(C_1',C_2') + \rho\,\mathrm{d}_{\max}(C_1',C_2').$$

Since the above inequality is linear in ρ, there is a single critical value of the parameter, given by

$$c = \frac{\mathrm{d}_{\min}(C_1',C_2') - \mathrm{d}_{\min}(C_1,C_2)}{\mathrm{d}_{\max}(C_1,C_2) - \mathrm{d}_{\min}(C_1,C_2) + \mathrm{d}_{\min}(C_1',C_2') - \mathrm{d}_{\max}(C_1',C_2')}$$

such that the relative preference of merging C_1 and C_2 or C_1' and C_2' changes only at $\rho = c$. Moreover, the definition of c only depends on a collection of 8 points: the closest and farthest pair between C_1 and C_2 and between C_1' and C_2'. In particular, every such critical parameter value c is given by

$$c = \frac{d_{rr'}^{(t)} - d_{ii'}^{(t)}}{d_{jj'}^{(t)} - d_{ii'}^{(t)} + d_{rr'}^{(t)} - d_{ss'}^{(t)}} \tag{B.14}$$

where $i, i', j, j', r, r', s, s' \in [n]$ are the indices of 8 points. Similarly to the knapsack example, we show that each critical parameter value is random and has a density function bounded by $16(\kappa B)^2$.

From this, it follows that for any interval I of radius ϵ, the expected total number of critical values summing over all instances $t = 1, \ldots, T$ that land in interval I is at most $32T\epsilon(\kappa B)^2$. This also bounds the expected number of functions $\ell_1, \ldots, \ell_T$ that are not constant on I. By Theorem 3.12, the functions are f-dispersed for $f(T, \epsilon) = 32T\epsilon(\kappa B)^2 + \sqrt{T \log(Tn)} = \tilde{O}(T\epsilon + \sqrt{T})$, also implying $\frac{1}{2}$-dispersion.

When the four distances present in the equation for c are distinct entries of the distance matrix D, then they are independent. However, it is possible that the closest and furthest pair of points between a pair of clusters can be the same, for example, when both clusters consist of just a single point. In this case, the corresponding distances are no longer independent, and we will need to modify our analysis slightly. Note a critical parameter c only arises for competing pairs of merges (C_1, C_2) and (C_1', C_2') that differ on at least one cluster (since otherwise both merges are identical). Moreover, since the set of clusters at any given round of the algorithm partition the data, any pair of clusters the algorithm encounters are either equal or disjoint. From this it follows that there are only four cases to consider depending on whether the closest and farthest pairs of points between C_1 and C_2 are the same, and whether the closest and farthest pairs of points between C_1' and C_2' are the same. That is, whether $(i, i') = (j, j')$ and whether $(s, s') = (r, r')$.

Case 1: $(i, i') \neq (j, j')$ *and* $(r, r') \neq (s, s')$. Let $X = d_{rr'} - d_{ii'}$ and $Y = d_{jj'} - d_{ss'}$. Rewriting expression for c given in (B.14), we have that $c = X/(X + Y)$. Moreover, both X and Y are the sum of two independent random variables having κ-bounded densities, so from Lemma B.37, it follows that X and Y also have densities bounded by κ. Next, since X and Y are independent, take values in $[-2M, 2M]$, and have κ-bounded densities, Lemma B.39 ensures that the ratio $X/(X + Y)$ has an $16(\kappa M)^2$ bounded density.

Case 2: $(i, i') = (j, j')$ *and* $(r, r') \neq (s, s')$. In this case, we are guaranteed that $d_{ii'} = d_{jj'}$, and the expression for c simplifies to

$$c = \frac{d_{rr'} - d_{ii'}}{d_{rr'} - d_{ss'}}$$

Defining $X = -d_{ii'}$, $Y = -d_{ss'}$, and $Z = d_{rr'}$, we have that $\beta = (X + Z)/(Y + Z)$. The variables X, Y, and Z are independent, each have κ-bounded densities, and $|Y| \leq M$ and $|Z| \leq M$ with probability 1. Applying Lemma B.40 to these random variables guarantees that the density function for β is $4(\kappa M)^2$-bounded.

Case 3: $(i, i') \neq (j, j')$ *and* $(r, r') = (s, s')$. This case is symmetric to case 2 and an identical argument applies.

Case 4: $(i, i') = (j, j')$ *and* $(r, r') = (s, s')$. In this case, the d_ρ distance between C_1 and C_2 is constant, as is the d_ρ distance between C_1' and C_2'. Therefore, for all values of ρ we will prefer to merge the same pair of clusters and there is no critical parameter value where we switch from one merge to the other.

In every case, the density of the critical parameter value β is upper bounded by $16\kappa^2 M^2$, completing the proof. □

Single Parameter Piecewise Unique Algorithms.

Next we provide a general approach for obtaining semi-bandit feedback that applies to many single-parameter algorithms. This enables semi-bandit feedback, but we still rely on problem-specific dis-

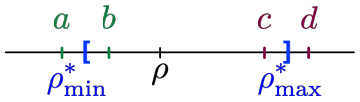


Figure B.2: Relationship between the binary search intervals $[a, b]$ and $[c, d]$ and the true interval $[\rho^*_{\min}, \rho^*_{\max}]$ on which $\mathcal{A}(x, \rho')$ outputs y_ρ.

persion analysis. This approach applies to any algorithm with a single real-valued parameter whose output is both a piecewise constant function of the parameter for any instance, and such that no output value is repeated across any distinct intervals in the piecewise decomposition. We call such an algorithm *single-parameter piecewise-unique*. Without loss of generality, we assume that the parameter space is given by $\mathcal{C} = [0, 1]$. Let $\mathcal{A} : \Pi \times [0, 1] \to \mathcal{Y}$ be an algorithm mapping problem instances $x \in \Pi$ and parameters $\rho \in [0, 1]$ to outputs in some space $\mathcal{Y}$. Given a parameter $\rho \in [0, 1]$ and a problem instance x, and an accuracy parameter $\epsilon > 0$, we will return both $\mathcal{A}(x, \rho)$, together with an interval $I = [\rho_{\min}, \rho_{\max}]$ such that for all $\rho' \in I$ we have $\mathcal{A}(x, \rho') = \mathcal{A}(x, \rho)$. Moreover, for any point $\rho' \notin [\rho_{\min} - \epsilon, \rho_{\min} + \epsilon]$, we have $\mathcal{A}(x, \rho') \neq \mathcal{A}(x, \rho)$. In other words, the interval I output by the algorithm is nearly the largest piecewise constant interval containing ρ. The high level idea of our approach is to run binary search twice to determine the upper and lower bounds $\rho_{\max}$ and $\rho_{\min}$, respectively. Each search will require that we run the algorithm $\mathcal{A}$ at most $O(\log 1/\epsilon)$ times. In cases where the algorithm parameters are specified using b bits of precision, then this procedure exactly determines the interval using $O(b)$ invocations of the base algorithm. Pseudocode is given in Algorithm 21. Steps 3 and 4 perform binary search to find the upper bound on the constant interval, while steps 5 and 6 perform binary search to find the lower bound.

Lemma B.34. *Let $\mathcal{A} : \Pi \times [0, 1] \to \mathcal{Y}$ be any single-parameter piecewise-unique algorithm and suppose y_ρ and $I = [\rho_{\min}, \rho_{\max}]$ is output by Algorithm 21 when run on $\mathcal{A}$ with problem instance $x \in \Pi$, parameter $\rho \in [0, 1]$, and target accuracy ϵ. Then Algorithm 21 runs the base algorithm $\mathcal{A}$ at most $O(\log 1/\epsilon)$ times and we have that $\mathcal{A}(x, \rho') = y_\rho$ for all $\rho' \in I$, $\rho \in I$, and for all $\rho' \notin [\rho_{\min} - \epsilon, \rho_{\max} + \epsilon]$ we have $\mathcal{A}(x, \rho') \neq y_\rho$.*

Proof. From step 1 of the algorithm, we know that $\mathcal{A}(x, \rho) = y_\rho$, by definition. Since the algorithm is single-parameter and piecewise-unique, we know that $\mathcal{A}(x, \rho')$ will output y_ρ for all ρ' belonging to some interval $[\rho^*_{\min}, \rho^*_{\max}]$ containing ρ, and it will not output y_ρ for any point outside that interval. In particular, restricted to the interval $[\rho, 1]$, there is exactly one critical parameter value, namely $\rho^*_{\max}$ below which the algorithm always outputs y_ρ and above which the algorithm always outputs something different. The binary search performed in step 3 guarantees that $\rho^*_{\max}$ is always contained in the interval $[a, b]$, yet on each iteration the length of the interval is halved. Similarly, each iteration of the binary search in step 6 guarantees that $\rho^*_{\min} \in [c, d]$, and the width of the interval halves on each iteration. Each iteration of both binary search instances requires us to run the base algorithm $\mathcal{A}$ once, and we will require $O(\log 1/\epsilon)$ iterations to guarantee the width of both intervals is less than ϵ.

Since $a \leq \rho^*_{\max}$ and $d \geq \rho^*_{\min}$, we have $[a, d] \subset [\rho^*_{\min}, \rho^*_{\max}]$ and it follows that $\mathcal{A}(x, \rho') = y_\rho$ for all $\rho' \in [a, d]$, as required. Moreover, we know that $a + \epsilon \geq b \geq \rho^*_{\max}$ and $d - \epsilon \leq c \leq \rho^*_{\min}$, implying that $\mathcal{A}(x, \rho') \neq y_\rho$ for all $\rho' \notin [a - \epsilon, d + \epsilon]$, as required. Figure B.2 depicts the relation between $[a, b]$, $[c, d]$, and $[\rho^*_{\min}, \rho^*_{\max}]$ at the end of the algorithm. □

Algorithm 21 Blackbox Bandit Feedback for Single-parameter Algorithms

Input: Algorithm $\mathcal{A} : \Pi \times [0,1] \to \mathcal{Y}$, parameter $\rho \in [0,1]$, problem instance $x \in \Pi$.

1. Let $y_\rho \leftarrow \mathcal{A}(x, \rho)$ be the output of $\mathcal{A}$ run on x with parameter ρ.
2. Let $a \leftarrow 0$ and $b \leftarrow \rho$.
3. While $b - a > \epsilon$:
   (a) Set $m \leftarrow (a+b)/2$.
   (b) If $\mathcal{A}(x, m) = y_\rho$ then set $b \leftarrow m$
   (c) Otherwise set $a \leftarrow m$.
4. Let $\rho_{\min} \leftarrow b$.
5. Let $c \leftarrow \rho$ and $d \leftarrow 1$.
6. While $d - c > \epsilon$:
   (a) Set $m \leftarrow (c+d)/2$.
   (b) If $\mathcal{A}(x, m) = y_\rho$ then set $c \leftarrow m$
   (c) Otherwise set $d \leftarrow m$.
7. Let $\rho_{\min} \leftarrow c$
8. Output y_ρ and interval $I = (\rho_{\min}, \rho_{\max})$.

B.2.4 Transformations of Bounded Densities

In this section we summarize several useful results that provide upper bounds on the density of random variables that are obtained as functions of other random variables with bounded density functions. These results allow us to reason about the distribution of discontinuity locations that arise as transformations of random problem parameters in algorithm configuration instances.

In many cases, we make use of the following result:

Theorem B.22 (Density Function Change of Variables)**.** *Let $X \in \mathbb{R}^d$ be a random vector with joint probability density function $f_X : \mathbb{R}^d \to [0, \infty)$ and let $\phi : \mathbb{R}^d \to \mathbb{R}^n$ be any bijective differentiable function. Then the random vector $Y = \phi(X)$ also has a density function $f_Y : \mathbb{R}^n \to [0, \infty)$ given by $f_Y(y) = |\det(J_{\phi^{-1}}(y))| f_X(\phi^{-1}(y))$, where $J_{\phi^{-1}}(y)$ denotes the Jacobian of ϕ^{-1} evaluated at y.*

Lemma B.35 (Lemma 6 from [22])**.** *Suppose X and Y are random variables taking values in $(0, 1]$ and suppose that their joint distribution is κ-bounded. Then the distribution of $Z = \ln(X/Y)$ is $\kappa/2$-bounded.*

Lemma B.36 (Lemma 8 from [22])**.** *Suppose X is a random variable with a κ-bounded density and suppose c is a constant. Then $Z = X/c$ has a $c\kappa$-bounded density*

Lemma B.37. *Let X and Y be two independent random variables each having densities upper bounded by κ. The random variable $U = X + Y$ has density f_U satisfying $f_U(u) \leq \kappa$ for all u.*

Proof. Let f_X and f_Y be the density functions for X and Y, respectively. The density for U is the convolution of f_X and f_Y. With this, we have

$$f_U(u) = \int_{-\infty}^{\infty} f_X(u-y) f_Y(y)\, dy \leq \int_{-\infty}^{\infty} \kappa f_Y(y)\, dy = \kappa.$$

It follows that $U = X + Y$ has a density that is upper bounded by κ. □

Lemma B.38. *Let X and Y be random variables with joint density f_{XY} that is κ-bounded and such that $|Y| \leq M$ with probability 1 and let $U = X/Y$. Then the density function f_U is κM^2-bounded.*

Proof. Consider the change of variables given by $U = X/Y$ and $V = Y$. This corresponds to the transformation function $\phi(x, y) = (x/y, y)$. The inverse of ϕ is given by $\phi^{-1}(u, v) = (uv, v)$. The Jacobian of ϕ^{-1} is

$$J_{\phi^{-1}}(u, v) = \begin{bmatrix} v & u \\ 0 & 1 \end{bmatrix},$$

whose determinant is always equal to v. Therefore, the joint density of U and V is given by

$$f_{UV}(u, v) = |v| f_{XY}(uv, v).$$

To get the marginal density for U, we integrate over v and use the fact that the density $f_{XY}(x, y) = 0$ whenever $|y| > M$. This gives

$$f_U(u) = \int_{-M}^{M} |v| f_{XY}(uv, v)\, dv \leq \kappa \int_{-M}^{M} |v|\, dv = \kappa M^2.$$

It follows that the density for U satisfies $f_U(u) \leq \kappa M^2$ for all u, as required. □

Lemma B.39. *Let X and Y be independent random variables with κ-bounded densities so that $|X| \leq M$ and $|Y| \leq M$ with probability one and define $Z = X/(X + Y)$. The random variable Z has a density function f_Z that is $4\kappa^2 M^2$-bounded.*

Proof. Consider the change of variables given by $U = X$ and $V = X + Y$. We will argue that the joint density f_{UV} is κ^2-bounded. Then, since $|X + Y| \leq 2M$ with probability 1, we can apply Lemma B.38 to ensure that the density of $Z = U/V$ is bounded by $\kappa^2(2M)^2 = 4\kappa^2 M^2$, as required.

It remains to bound the joint density of $U = X$ and $V = X + Y$. This change of variables corresponds to the transformation function $\phi(x, y) = (x, x + y)$, whose inverse is given by $\phi^{-1}(u, v) = (u, v - u)$. The Jacobian of ϕ^{-1} is given by

$$J_{\phi^{-1}}(u, v) = \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix},$$

whose determinant is always 1. It follows that the joint density for (U, V) is given by $f_{UV}(u, v) = f_{XY}(u, v - u) = f_X(u) f_Y(v - u) \leq \kappa^2$, as required. □

Lemma B.40. *Let X, Y, and Z be independent random variables with κ-bounded densities such that $|Y| \leq M$, and $|Z| \leq M$ with probability one. Then the random variable $R = \frac{X+Y}{Z+Y}$ has a density f_R that satisfies $f_R(u) \leq 4\kappa^2 M^2$.*

Proof. Consider the change of variables given by $U = X + Y$, $V = Z + Y$. We will argue that the joint density f_{UV} for U and V is κ^2-bounded. Then, since $|V| = |Z + Y| \leq 2M$ with probability 1, we can apply Lemma B.38 to ensure that the density of $R = U/V$ is bounded by $4\kappa^2 M^2$, as required.

It remains to bound the joint density of $U = X + Y$ and $V = Z + Y$. Consider the change of variables given by $U = X + Y$, $V = Z + Y$, and $W = Y$. This corresponds to the transformation

function $\phi(x, y, z) = (x + y, z + y, y)$, and has inverse $\phi^{-1}(u, v, w) = (u - w, w, v - w)$. The Jacobian of ϕ^{-1} is given by

$$J_{\phi^{-1}}(u, v, w) = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 1 \\ 0 & 1 & -1 \end{bmatrix},$$

which always has determinant given by -1. It follows that the joint density for (U, V, W) is given by

$$f_{UVW}(u, v, w) = f_{XYZ}(u - w, w, v - w) = f_X(u - w) f_Y(w) f_Z(v - w).$$

To get the joint density over only U and V we integrate over w:

$$f_{UV}(u, v) = \int_{-\infty}^{\infty} f_X(u - w) f_Y(w) f_Z(v - w)\, dw \leq \kappa^2 \int_{-\infty}^{\infty} f_Y(w)\, dw = \kappa^2,$$

as required. □

B.2.5 Discretization-based Algorithm

In this section we provide a single algorithm that has regret bounds for the full-information, semi-bandit feedback, and bandit-feedback settings. The high level idea is to reduce the problem to a finite-armed bandit and apply the Exp3-SET algorithm of Alon et al. [2], which enjoys regret bounds in all three of these settings. In particular, we choose a value of $r > 0$ and construct a r-net for the parameter space $\mathcal{C}$ (which can be done using at most $(3R/r)^d$ points when $\mathcal{C}$ is contained in a ball of radius R in d dimensions. Then we run the Exp3-SET algorithm over this finite set). Exp3-SET is guaranteed to have bounded regret compared to the best point from the r-net, and we use f-dispersion to bound the expected difference in total loss between the best discretized point and the best point in all of $\mathcal{C}$ by $TLr + f(T, r)$. We then get bounds for each of the three feedback regimes by tuning the parameters r and λ of this single algorithm.

On each round, we assume that the learner observes a feedback set A_t, together with the loss $\ell_t(\rho)$ for all points $\rho \in A_t$. In the full information setting, we take $A_t = \mathcal{C}$ to be the entire parameter space. In the bandit feedback setting, we take $A_t = \{\rho_t\}$, where ρ_t is the point played by the learner in round t. And in the semi-bandit feedback setting, we take A_t to be the piece of ℓ_t that contains the point ρ_t played by the learner. Using this notation, pseudocode is given in Algorithm 22.

Algorithm 22 Discretized Exp3-SET

Parameters: Granularity $r > 0$, step size $\lambda \in [0, 1]$

1. Let $\hat{\rho}_1, \ldots \hat{\rho}_N$ be an r-net for $\mathcal{C}$ of size $N \leq \left(\frac{3R}{r}\right)^d$.
2. Let $w_{i,1} = 1$ for all $i = 1, \ldots, N$.
3. For $t = 1, \ldots, T$
   (a) Let $p_{i,t} = w_{i,t}/W_t$, where $W_t = \sum_i w_{i,t}$.
   (b) Let $\rho_t = \hat{\rho}_i$ with probability $p_{i,t}$ and play ρ_t.
   (c) Observe feedback set A_t and $\ell_t(\hat{\rho}_i)$ for all $\hat{\rho}_i \in A_t$.
   (d) Let $\hat{\ell}_{i,t} = \frac{\mathbb{I}\{\hat{\rho}_i \in A_t\}}{q_t} \ell_t(\hat{\rho}_i)$, where $q_t = \sum_{i:\hat{\rho}_i \in A_t} p_{i,t}$.
   (e) Let $w_{i,t+1} = w_{i,t} \exp(-\lambda \hat{\ell}_{i,t})$ for all $i = 1, \ldots, N$.

Theorem B.23. *Let $\mathcal{C} \subset \mathbb{R}^d$ be contained in a ball of radius R and $\ell_1, \dots, \ell_T$ be possibly random piecewise L-Lipschitz functions that are f-dispersed and have an r_0-interior optimum. Then running Algorithm 22 with discretization parameter $r \le r_0$, the following claims hold:*

1. *Under full information, setting $r = 1/(L\sqrt{T})$ and $\lambda = \sqrt{\ln(RL\sqrt{T})/T}$ gives expected regret bounded by $O(\sqrt{dT\log(RLT)} + f(T, 1/(L\sqrt{T}))$.*
2. *Under semi-bandit feedback, if each function ℓ_t has at most M pieces, setting $r = 1/(L\sqrt{T})$ and $\lambda = \sqrt{d\ln(RL\sqrt{T})/(MT)}$ gives expected regret bounded by $O(\sqrt{dTM\log(RLT)} + f(T, 1/(L\sqrt{T}))$.*
3. *Under bandit feedback, setting $r = T^{\frac{d+1}{d+2}/2} = T^{-1/(d+2)}$ and $\lambda = \sqrt{d\ln(3RT^{\frac{1}{d+2}})/((3R)^d T^{\frac{2(d+1)}{d+2}})}$ gives expected regret bounded by $O\left(T^{\frac{d+1}{d+2}}(d(3R)^d\ln(RT) + L) + f(T, \frac{1}{T^{1/(d+2)}})\right)$.*

In each of the above bounds, the role of the dispersion parameter f is clean. For both the full-information and semi-bandit feedback settings, the only term in the bound that depends on f is the $f(T, 1/(L\sqrt{T}))$ term. Similarly, for bandit feedback, the bound depends on f only through the $f(T, 1/T^{1/(d-2)})$ term. In either case, it is clear that the regret bound after T rounds depends on the dispersion properties in neighborhoods of radius $1/(L\sqrt{T})$ and $1/T^{1/(d-2)}$, respectively. We know that the function $f(T, \epsilon)$ should be non-decreasing in the parameter ϵ, so this also shows that in one sense we require "less" dispersion in the bandit setting, since the bandit bounds depend on $f(T, 1/T^{1/(d+2)}) \le f(T, 1/\sqrt{T}))$.

Proof. Let $\ell_1, \dots, \ell_T$ be the sequence of loss functions. By assumption, with probability one there is an optimal parameter in hindsight ρ^* such that $B(\rho^*, r) \subset \mathcal{C}$. It follows that at least one of the discretization points $\hat{\rho}_1, \dots, \hat{\rho}_N$ must belong to this ball. Let $\hat{\rho}^*$ be such a point. We use f-dispersion to argue that the total loss of $\hat{\rho}^*$ is not much more than the total loss of ρ^* in expectation over the loss functions ℓ_t. Let

$$D = \left|\{1 \le t \le T \,:\, \ell_t \text{ is not } L\text{-Lipschitz on } B(\rho^*, r)\}\right|$$

be the number of loss functions that are not L-Lipschitz on $B(\rho^*, r)$. We know that $\sum_{t=1}^T \ell_t(\hat{\rho}^*) - \ell_t(\rho^*) \le TLr + D$, since D of the functions have discontinuities and the remaining $T - D < T$ are L-Lipschitz. Taking the expectation over $\ell_1, \dots, \ell_T$ and using the fact that $\mathbb{E}[D] \le f(T, r)$, we have that

$$\mathbb{E}\left[\sum_{t=1}^T \ell_t(\hat{\rho}^*) - \ell_t(\rho^*)\right] \le TLr + f(T, r). \tag{B.15}$$

Therefore, it is sufficient to show that Algorithm 22 is competitive with ρ^* in expectation over the algorithm randomness.

Algorithm 22 runs a special case of the Exp3-SET algorithm of Alon et al. [2] over the discretized set of parameters $\hat{\rho}_1, \dots, \hat{\rho}_N$. In particular, we have assumed that the feedback system takes a particular form: on each round the arms are partitioned into sets such that playing any arm from one set reveals the loss for all arms in the same set. Alon et al. [2] bound the regret of this algorithm for a broader class of feedback systems, where on every round there is an undirected graphs defined over the arms fo the bandit and playing an arm reveals not only its loss, but also the loss of all adjacent arms in the feedback graph G_t. Our special case corresponds to the setting where G_t is the union of

several cliques (in the full-information setting, G_t is the complete graph, in the semi-bandit setting, G_t is the union of M cliques, where M is the number of pieces, and in the bandit setting, G_t is the union of N cliques, each consisting of a single node). Corollary 3 of [2] bound the regret of the algorithm by

$$\mathbb{E}\left[\sum_{t=1}^{T} \ell_t(\rho_t)\right] - \sum_{t=1}^{T} \ell_t(\hat{\rho}^*) \leq \frac{1}{\lambda}\ln(N) + \frac{\lambda}{2}\sum_{t=1}^{T}\alpha(G_t),$$

where $\alpha(G_t)$ is the size of the largest independent set in the feedback graph G_t at round t and the expectation is over the randomness of the algorithm, and holds for any oblivious choice of loss functions $\ell_1, \ldots, \ell_T$ and feedback graphs $G_1, \ldots, G_T$. In particular, if $\alpha(G_t) \leq \alpha$ for some number α on all time steps, then setting $\lambda = \sqrt{\ln(N)/(T\alpha)}$ gives

$$\mathbb{E}\left[\sum_{t=1}^{T} \ell_t(\rho_t)\right] - \sum_{t=1}^{T} \ell_t(\hat{\rho}^*) \leq O(\sqrt{\ln(N)T\alpha}).$$

Finally, taking expectations over the randomness in the loss functions $\ell_1, \ldots, \ell_T$, using (B.15), the above choice of λ, and the fact that $N \leq (3R/r)^d$, we have

$$\mathbb{E}\left[\sum_{t=1}^{T} \ell_t(\rho_t) - \ell_t(\rho^*)\right] \leq O\left(\sqrt{dT\alpha\ln(R/r)} + TLr + f(T,r)\right)$$

Full information. In the full information setting, the feedback graph G_t is the complete graph, for which the largest independent set is of size $\alpha = 1$. In this case, choosing $r = 1/(L\sqrt{T})$ leads to $\lambda = \sqrt{\ln(RL\sqrt{T})/T}$ and gives expected regret bounded by $O(\sqrt{dT\log(RLT)} + f(T, 1/(L\sqrt{T}))$.

Semi-bandit feedback. Next suppose that each function ℓ_t is piecewise defined with at most M and when the learner plays a point ρ_t, they learn the loss for all points belonging to the same piece. In this case, the feedback graph G_t consists of the union of M cliques, and the largest independent set has size at most $\alpha = M$. Again, choosing $r = 1/(L\sqrt{T})$ leads to $\lambda = \sqrt{d\ln(RL\sqrt{T})/(MT)}$ and gives expected regret bound by $O(\sqrt{dTM\log(RLT)} + f(T, 1/(L\sqrt{T}))$.

Bandit feedback. Finally, consider the bandit feedback setting. In this case, the feedback graph G_t has the empty edge-set and therefore the largest independent set is of size $\alpha = N \leq (3R/r)^d$ (the total number of arms). Choosing $r = T^{\frac{d-1}{d-2}-1}$ leads to

$$\lambda = \sqrt{\frac{d\ln(3RT^{\frac{1}{d+2}})}{(3R)^d T^{\frac{2(d+1)}{d+2}}}}$$

and expected regret bounded by $O\left(T^{\frac{d+1}{d+2}}(d(3R)^d\ln(RT) + L) + f(T, \frac{1}{T^{1/(d+2)}})\right)$. □

Note, the condition that $B(\rho^*, r_0) \subset \mathcal{C}$ with probability one is only for technical convenience. We can always modify the optimization problem so that this condition is satisfied. In particular, we define an enlarged parameter space $\mathcal{C}' = \bigcup_{\rho\in\mathcal{C}} B(\rho, r_0)$ and replace the utility function u_t with its Lipschitz extension to $\mathcal{C}'$. On this modified problem we are guaranteed that there exists an optimal parameter in the r_0-interior of $\mathcal{C}'$, and the Lipschitz-extended functions are still piecewise Lipschitz and f-dispersed for the same f.

Appendix C

Appendix for Chapter 4

In this section we prove the sample complexity guarantee for learning the best algorithm from the family $\mathcal{A}_{\text{metric}}(\mathrm{d}_0, \mathrm{d}_1)$, which corresponds to learning the best convex combination of the two metrics d_0 and d_1 to use for a given clustering application. The main step of the argument is proved in Theorem 4.1, which shows that for any given clustering instance S with target clustering $\mathcal{Y}$, the function $\beta \mapsto A_\beta^{\text{metric}}(S)$ is a piecewise constant function of β with at most $O(|S|^4)$ pieces. That is, we can find $M = O(|S|^4)$ intervals $I_1, \ldots, I_M$ that partition $[0, 1]$ such that for any interval I_i and any $\beta, \beta' \in I_i$, we have $A_\beta^{\text{metric}}(S) = A_{\beta'}^{\text{metric}}(S)$. From this if also follows that the loss function $\beta \mapsto \ell(A_\beta^{\text{metric}}(S), \mathcal{Y})$ is piecewise constant with the same partition. Next, we show how to use this structure in the loss functions to prove sample complexity guarantees for learning the best value of β.

Theorem 4.2. *Consider the family $\mathcal{A}_{metric}(\mathrm{d}_0, \mathrm{d}_1)$ and let $(S_1, \mathcal{Y}_1), \ldots, (S_N, \mathcal{Y}_N)$ be an i.i.d. sample of clustering instances with target clusterings of size $N = O\big(\frac{1}{\epsilon^2}(\log n + \log \frac{1}{\delta})\big)$, where n is a bound on the number of points per instance. With probability at least $1 - \delta$, the following holds for all $\beta \in [0, 1]$*

$$\left| \frac{1}{N} \sum_{i=1}^{N} \ell(A_\beta^{metric}(S_i), \mathcal{Y}_i) - \mathop{\mathbb{E}}_{(S,\mathcal{Y})\sim\mathcal{D}} [\ell(A_\beta^{metric}(S), \mathcal{Y})] \right| \leq \epsilon.$$

Proof. For each parameter $\beta \in [0, 1]$, define a function f_β that takes as arguments a clustering instance S and a target clustering $\mathcal{Y}$ for S and outputs the loss of A_β^{metric} evaluated on S. That is, $f_\beta(S, \mathcal{Y}) = \ell(A_\beta^{\text{metric}}(S), \mathcal{Y})$. We will bound the pseudo-dimension of the class $\mathcal{F} = \{f_\beta \mid \beta \in [0, 1]\}$ by $O(\log n)$, after which standard uniform-convergence arguments imply the claim. Throughout the proof we assume that all clustering instances have at most n points.

First, suppose that the Pseudo-dimension of the class $\mathcal{F}$ is D. Then there exists a collection of clustering instances with target clusterings $(S_1, \mathcal{Y}_1), \ldots, (S_D, \mathcal{Y}_D)$ that is shattered by $\mathcal{F}$. In other words, there are witness thresholds $z_1, \ldots, z_D \in \mathbb{R}$ such that for every subset $T \subset [D]$, there exists a parameter $\beta_T \in [0, 1]$ so that $f_{\beta_T}(S_i, \mathcal{Y}_i) \geq z_i$ if and only if $i \in T$. In particular, this guarantees that the set of vectors

$$\big\{\big(f_\beta(S_1, \mathcal{Y}_1), \ldots, f_\beta(S_D, \mathcal{Y}_D)\big) \in \mathbb{R}^D \mid \beta \in [0, 1]\big\}$$

contains at least 2^D distinct elements.

Next, we argue that this set also contains at most $O(Dn^4)$ elements. For each instance $(S_i, \mathcal{Y}_i)$, Theorem 4.1 guarantees that we can find $M_i = O(n^4)$ critical parameter values $0 = c_1 < \cdots < c_{M_i} = 1$ such that for any $\beta, \beta' \in [c_{i-1}, c_i)$, we have $f_\beta(S_i, \mathcal{Y}_i) = f_{\beta'}(S_i, \mathcal{Y}_i)$. Taking the union of

these critical parameter values across all D instances, we get a partition of $[0,1]$ into $M = O(Dn^4)$ intervals $I_1, \ldots, I_M$ with the following property: For every $\beta, \beta' \in I_j$ and every instance $(S_i, \mathcal{Y}_i)$, we have $f_\beta(S_i, \mathcal{Y}_i) = f_{\beta'}(S_i, \mathcal{Y}_i)$. Let $\beta_1, \ldots, \beta_M$ be a collection of parameter values with $\beta_j \in I_j$ for all $j \in [M]$. Now let $\beta \in [0,1]$ be any parameter value. Since $I_1, \ldots, I_M$ partition $[0,1]$, we know that there is exactly one index, say j, such that $\beta \in I_j$. Therefore, we have that $f_\beta(S_i, \mathcal{Y}_i) = f_{\beta_j}(S_i, \mathcal{Y}_i)$ for all $i \in [D]$. From this, it follows that

$$\left|\left\{\left(f_\beta(S_1, \mathcal{Y}_1), \ldots, f_\beta(S_D, \mathcal{Y}_D)\right) \in \mathbb{R}^D \mid \beta \in [0,1]\right\}\right| \leq M = O(Dn^4),$$

since every parameter β has the same loss on all instances $(S_1, \mathcal{Y}_1), \ldots, (S_D, \mathcal{Y}_D)$ as one of the parameters β_j.

Combining the above arguments, we have that $2^D \leq O(Dn^4)$, which implies that $D = O(\log n)$. The final result follows by applying standard Pseudo-dimension uniform convergence results [117]. □

Appendix D

Appendix for Chapter 5

Theorem 5.1. *Let d be a metric on $\mathcal{X}$, P be a distribution on $\mathcal{X}$, and u be an L-Lipschitz utility function. Let S be a set of individuals such that there exists $\hat{\mathcal{X}} \subset \mathcal{X}$ with $P(\hat{\mathcal{X}}) \geq 1-\alpha$ and $\sup_{x \in \hat{\mathcal{X}}}(d(x, \mathrm{NN}_S(x)) \leq \beta/(2L)$. Then for any classifier $h : S \to \Delta(\mathcal{Y})$ that is EF on S, the extension $\overline{h} : \mathcal{X} \to \Delta(\mathcal{Y})$ given by $\overline{h}(x) = h(\mathrm{NN}_S(x))$ is (α, β)-EF on P.*

Proof. Let $h : S \to \Delta(\mathcal{Y})$ be any EF classifier on S and $\overline{h} : \mathcal{X} \to \Delta(\mathcal{Y})$ be the nearest neighbor extension. Sample x and x' from P. Then, x belongs to the subset $\hat{\mathcal{X}}$ with probability at least $1-\alpha$. When this occurs, x has a neighbor within distance $\beta/(2L)$ in the sample. Using the Lipschitz continuity of u, we have $|u(x, \overline{h}(x)) - u(\mathrm{NN}_S(x), h(\mathrm{NN}_S(x)))| \leq \beta/2$. Similarly, $|u(x, \overline{h}(x')) - u(\mathrm{NN}_S(x), h(\mathrm{NN}_S(x')))| \leq \beta/2$. Finally, since $\mathrm{NN}_S(x)$ does not envy $\mathrm{NN}_S(x')$ under h, it follows that x does not envy x' by more than β under $\overline{h}$. □

Lemma D.1. *Suppose $\mathcal{X} \subset \mathbb{R}^q$, $d(x, x') = \|x - x'\|_2$, and let $D = \sup_{x,x' \in \mathcal{X}} d(x, x')$ be the diameter of $\mathcal{X}$. For any distribution P over $\mathcal{X}$, $\beta > 0$, $\alpha > 0$, and $\delta > 0$ there exists $\hat{\mathcal{X}} \subset \mathcal{X}$ such that $P(\hat{\mathcal{X}}) \geq 1-\alpha$ and, if S is an i.i.d. sample drawn from P of size $|S| = O(\frac{1}{\alpha}(\frac{LD\sqrt{q}}{\beta})^q(d \log \frac{LD\sqrt{q}}{\beta} + \log \frac{1}{\delta}))$, then with probability at least $1-\delta$, $\sup_{x \in \hat{\mathcal{X}}} d(x, \mathrm{NN}_S(x)) \leq \beta/(2L)$.*

Proof. Let C be the smallest cube containing $\mathcal{X}$. Since the diameter of $\mathcal{X}$ is D, the side-length of C is at most D. Let $s = \beta/(2L\sqrt{q})$ be the side-length such that a cube with side-length s has diameter $\beta/(2L)$. It takes at most $m = \lceil D/s \rceil^q$ cubes of side-length s to cover C. Let $C_1, \ldots, C_m$ be such a covering, where each C_i has side-length s.

Let C_i be any cube in the cover for which $P(C_i) > \alpha/m$. The probability that a sample of size n drawn from P does not contain a sample in C_i is at most $(1-\alpha/m)^n \leq e^{-n\alpha/m}$. Let $I = \{i \in [m] : P(C_i) \geq \alpha/m\}$. By the union bound, the probability that there exists $i \in I$ such that C_i does not contain a sample is at most $me^{-n\alpha/m}$. Setting

$$\begin{aligned} n &= \frac{m}{\alpha} \ln \frac{m}{\delta} \\ &= O\left(\frac{1}{\alpha}\left(\frac{LD\sqrt{q}}{\beta}\right)^q \left(q \log \frac{LD\sqrt{q}}{\beta} + \log \frac{1}{\delta}\right)\right) \end{aligned}$$

results in this upper bound being δ. For the remainder of the proof, assume this high probability event occurs.

Now let $\hat{\mathcal{X}} = \bigcup_{i \in I} C_i$. For each $j \notin I$, we know that $P(C_j) < \alpha/m$. Since there at most m such cubes, their total probability mass is at most α. It follows that $P(\hat{\mathcal{X}}) \geq 1-\alpha$. Moreover, every

point $x \in \hat{\mathcal{X}}$ belongs to one of the cubes C_i with $i \in I$, which also contains a sample point. Since the diameter of the cubes in our cover is $\beta/(2L)$, it follows that $d(x, \mathrm{NN}_S(x)) \leq \beta/(2L)$ for every $x \in \hat{\mathcal{X}}$, as required. □

Lemma D.2. *Let $\mathcal{G} = \{g : \mathcal{X} \to \mathcal{Y}\}$ have Natarajan dimension d. For $g_1, g_2 \in \mathcal{G}$, let $(g_1, g_2) : \mathcal{X} \to \mathcal{Y}^2$ denote the function given by $(g_1, g_2)(x) = (g_1(x), g_2(x))$ and let $\mathcal{G}^2 = \{(g_1, g_2) : g_1, g_2 \in \mathcal{G}\}$. Then the Natarajan dimension of $\mathcal{G}^2$ is at most $2d$.*

Proof. Let D be the Natarajan dimension of $\mathcal{G}^2$. Then we know that there exists a collection of points $x_1, \ldots, x_D \in \mathcal{X}$ that is shattered by $\mathcal{G}^2$, which means there are two sequences $q_1, \ldots, q_n \in \mathcal{Y}^2$ and $q'_1, \ldots, q'_n \in \mathcal{Y}^2$ such that for all i we have $q_i \neq q'_i$ and for any subset $C \subset [D]$ of indices, there exists $(g_1, g_2) \in \mathcal{G}^2$ such that $(g_1, g_2)(x_i) = q_i$ if $i \in C$ and $(g_1, g_2)(x_i) = q'_i$ otherwise.

Let $n_1 = \sum_{i=1}^{D} \mathbb{I}\{q_{i1} \neq q'_{i1}\}$ and $n_2 = \sum_{i=1}^{D} \mathbb{I}\{q_{i2} \neq q'_{i2}\}$ be the number of pairs on which the first and second labels of q_i and q'_i disagree, respectively. Since none of the n pairs are equal, we know that $n_1 + n_2 \geq D$, which implies that at at least one of n_1 or n_2 must be $\geq D/2$. Assume without loss of generality that $n_1 \geq D/2$ and that $q_{i1} \neq q'_{i1}$ for $i = 1, \ldots, n_1$. Now consider any subset of indices $C \subset [n_1]$. We know there exists a pair of functions $(g_1, g_2) \in \mathcal{G}^2$ with $(g_1, g_2)(x_i)$ evaluating to q_i if $i \in C$ and q'_i if $i \notin C$. But then we have $g_1(x_i) = q_{i1}$ if $i \in C$ and $g_1(x_i) = q'_{i1}$ if $i \notin C$, and $q_{i1} \neq q'_{i1}$ for all $i \in [n_1]$. It follows that $\mathcal{G}$ shatters $x_1, \ldots, x_{n_1}$, which consists of at least $D/2$ points. Therefore, the Natarajan dimension of $\mathcal{G}^2$ is at most $2d$, as required. □

The following example demonstrates that the optimal randomized envy-free classifier can be arbitrarily better than the best deterministic classifier.

Example D.1. Let $S = \{x_1, x_2\}$ and $\mathcal{Y} = \{y_1, y_2, y_3\}$. Let the loss function be such that

$$\begin{array}{lll} \ell(x_1, y_1) = 0 & \ell(x_1, y_2) = 1 & \ell(x_1, y_3) = 1 \\ \ell(x_2, y_1) = 1 & \ell(x_2, y_2) = 1 & \ell(x_2, y_3) = 0 \end{array}$$

And let the utility function be such that

$$\begin{array}{lll} u(x_1, y_1) = 0 & u(x_1, y_2) = 1 & u(x_1, y_3) = \dfrac{1}{\gamma} \\ u(x_2, y_1) = 0 & u(x_2, y_2) = 0 & u(x_2, y_3) = 1 \end{array}$$

where $\gamma > 1$. Now, the only deterministic classifier with a loss of 0 is h_0 such that $h_0(x_1) = y_1$ and $h_0(x_2) = y_3$. But, this is not EF, since $u(x_1, y_1) < u(x_1, y_3)$. Furthermore, every other deterministic classifier has a total loss of at least 1, causing the optimal deterministic EF classifier to have loss of at least 1.

To show that randomized classifiers can do much better, consider the randomized classifier h_* such that $h_*(x_1) = (1 - 1/\gamma, 1/\gamma, 0)$ and $h_*(x_2) = (0, 0, 1)$. This classifier can be seen as a mixture of the classifier h_0 of 0 loss, and the deterministic classifier h_e, where $h_e(x_1) = y_2$ and $h_e(x_2) = y_3$, which has high "negative envy". One can observe that this classifier h_* is EF, and has a loss of just $1/\gamma$. Hence, the loss of the optimal randomized EF classifier is γ times smaller than the loss of the optimal deterministic one, for any $\gamma > 1$.

Theorem 5.2. *There exists a space of individuals $\mathcal{X} \subset \mathbb{R}^q$, and a distribution P over $\mathcal{X}$ such that, for every randomized algorithm $\mathcal{A}$ that extends classifiers on a sample to $\mathcal{X}$, there exists an L-Lipschitz utility function u such that, when a sample of individuals S of size $n = 4^q/2$ is drawn*

from P without replacement, there exists an EF classifier on S for which, with probability at least $1 - 2\exp(-4^q/100) - \exp(-4^q/200)$ jointly over the randomness of $\mathcal{A}$ and S, its extension by $\mathcal{A}$ is not (α, β)-EF with respect to P for any $\alpha < 1/25$ and $\beta < L/8$.

Proof. Let the space of individuals be $\mathcal{X} = [0,1]^q$ and the outcomes be $\mathcal{Y} = \{0,1\}$. We partition the space $\mathcal{X}$ into cubes of side length $s = 1/4$. So, the total number of cubes is $m = (1/s)^q = 4^q$. Let these cubes be denoted by $c_1, c_2, \ldots c_m$, and let their centers be denoted by $\mu_1, \mu_2, \ldots \mu_m$. Next, let P be the uniform distribution over the centers $\mu_1, \mu_2, \ldots \mu_m$. For brevity, whenever we say "utility function" in the rest of the proof, we mean "L-Lipschitz utility function."

To prove the theorem, we use Yao's minimax principle [148]. Specifically, consider the following two-player zero sum game. Player 1 chooses a deterministic algorithm $\mathcal{D}$ that extends classifiers on a sample to $\mathcal{X}$, and player 2 chooses a utility function u on $\mathcal{X}$. For any subset $S \subset \mathcal{X}$, define the classifier $h_{u,S} : S \to \mathcal{Y}$ by assigning each individual in S to his favorite outcome with respect to the utility function u, i.e. $h_{u,S}(x) = \arg\max_{y \in \mathcal{Y}} u(x,y)$ for each $x \in S$, breaking ties lexicographically. Define the cost of playing algorithm $\mathcal{D}$ against utility function u as the probability over the sample S (of size $m/2$ drawn from P without replacement) that the extension of $h_{u,S}$ by $\mathcal{D}$ is not (α, β)-EF with respect to P for any $\alpha < 1/25$ and $\beta < L/8$. Yao's minimax principle implies that for any randomized algorithm $\mathcal{A}$, its expected cost with respect to the worst-case utility function u is at least as high as the expected cost of any distribution over utility functions that is played against the best deterministic algorithm $\mathcal{D}$ (which is tailored for that distribution). Therefore, we establish the desired lower bound by choosing a specific distribution over utility functions, and showing that the best deterministic algorithm against it has an expected cost of at least $1 - 2\exp(-m/100) - \exp(-m/200)$.

To define this distribution over utility functions, we first sample outcomes $y_1, y_2, \ldots, y_m$ i.i.d. from Bernoulli$(1/2)$. Then, we associate each cube center μ_i with the outcome y_i, and refer to this outcome as the *favorite* of μ_i. For brevity, let $\neg y$ denote the outcome other than y, i.e. $\neg y = (1 - y)$. For any $x \in \mathcal{X}$, we define the utility function as follows. Letting c_j be the cube that x belongs to,

$$u(x, y_j) = L\left[\frac{s}{2} - \|x - \mu_j\|_\infty\right]; \quad u(x, \neg y_j) = 0. \tag{D.1}$$

See Figure D.1 for an illustration.

We claim that the utility function of Equation (D.1) is indeed L-Lipschitz with respect to any L_p norm. This is because for any cube c_i, and for any $x, x' \in c_i$, we have

$$\begin{aligned} \left|u(x, y_i) - u(x', y_i)\right| &= L\left|\|x - \mu_i\|_\infty - \|x' - \mu_i\|_\infty\right| \\ &\leq L\|x - x'\|_\infty \leq L\|x - x'\|_p. \end{aligned}$$

Moreover, for the other outcome, we have $u(x, \neg y_i) = u(x', \neg y_i) = 0$. It follows that u is L-Lipschitz within every cube. At the boundary of the cubes, the utility for any outcome is 0, and hence u is also continuous throughout $\mathcal{X}$. Because it is piecewise Lipschitz and continuous, u must be L-Lipschitz throughout $\mathcal{X}$, with respect to any L_p norm.

Next, let $\mathcal{D}$ be an arbitrary deterministic algorithm that extends classifiers on a sample to $\mathcal{X}$. We draw the sample S of size $m/2$ from P without replacement. Consider the distribution over favorites of individuals in S. Each individual in S has a favorite that is sampled independently from Bernoulli$(1/2)$. Hence, by Hoeffding's inequality, the fraction of individuals in S with a favorite of 0 is between $\frac{1}{2} - \epsilon$ and $\frac{1}{2} + \epsilon$ with probability at least $1 - 2\exp(-m\epsilon^2)$. The same holds simultaneously for the fraction of individuals with favorite 1.

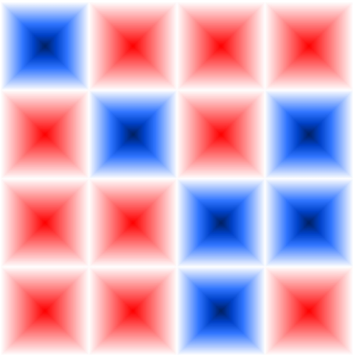


Figure D.1: Illustration of $\mathcal{X}$ and an example utility function u for $d = 2$. Red shows preference for 1, blue shows preference for 0, and darker shades correspond to more intense preference. (The gradients are rectangular to match the L_∞ norm, so, strangely enough, the misleading X pattern is an optical illusion.)

Given the sample S and the utility function u on the sample (defined by the instantiation of their favorites), consider the classifier $h_{u,S}$, which maps each individual μ_i in the sample S to his favorite y_i. This classifier is clearly EF on the sample. Consider the extension $h^{\mathcal{D}}_{u,S}$ of $h_{u,S}$ to the whole of $\mathcal{X}$ as defined by algorithm $\mathcal{D}$. Define two sets Z_0 and Z_1 by letting $Z_y = \{\mu_j \notin S \mid h^{\mathcal{D}}_{u,S}(\mu_j) = y\}$, and let y_* denote an outcome that is assigned to at least half of the out-of-sample centers, i.e., an outcome for which $|Z_{y_*}| \geq |Z_{\neg y_*}|$. Furthermore, let θ denote the fraction of out-of-sample centers assigned to y_*. Note that, since $|S| = m/2$, the number of out-of-sample centers is also exactly $m/2$. This gives us $|Z_{y_*}| = \theta\frac{m}{2}$, where $\theta \geq \frac{1}{2}$.

Consider the distribution of favorites in Z_{y_*} (these are independent from the ones in the sample since Z_{y_*} is disjoint from S). Each individual in this set has a favorite sampled independently from Bernoulli$(1/2)$. Hence, by Hoeffding's inequality, the fraction of individuals in Z_{y_*} whose favorite is $\neg y_*$ is at least $\frac{1}{2} - \epsilon$ with probability at least $1 - \exp(-\frac{m}{2}\epsilon^2)$. We conclude that with a probability at least $1 - 2\exp(-m\epsilon^2) - \exp(-\frac{m}{2}\epsilon^2)$, the sample S and favorites (which define the utility function u) are such that: (i) the fraction of individuals in S whose favorite is $y \in \{0, 1\}$ is between $\frac{1}{2} - \epsilon$ and $\frac{1}{2} + \epsilon$, and (ii) the fraction of individuals in Z_{y_*} whose favorite is $\neg y_*$ is at least $\frac{1}{2} - \epsilon$.

We now show that for such a sample S and utility function u, $h^{\mathcal{D}}_{u,S}$ cannot be (α, β)-EF with respect to P for any $\alpha < 1/25$ and $\beta < L/8$. To this end, sample x and x' from P. One scenario where x envies x' occurs when (i) the favorite of x is $\neg y_*$, (ii) x is assigned to y_*, and (iii) x' is assigned to $\neg y_*$. Conditions (i) and (ii) are satisfied when x is in Z_{y_*} and his favorite is $\neg y_*$. We know that at least a $\frac{1}{2} - \epsilon$ fraction of the individuals in Z_{y_*} have the favorite $\neg y_*$. Hence, the probability that conditions (i) and (ii) are satisfied by x is at least $(\frac{1}{2} - \epsilon)|Z_{y_*}|\frac{1}{m} = (\frac{1}{2} - \epsilon)\frac{\theta}{2}$. Condition (iii) is satisfied when x' is in S and has favorite $\neg y_*$ (and hence assigned $\neg y_*$), or, if x' is in $Z_{\neg y_*}$. We know that at least a $\left(\frac{1}{2} - \epsilon\right)$ fraction of the individuals in S have the favorite $\neg y_*$. Moreover, the size of $Z_{\neg y_*}$ is $(1 - \theta)\frac{m}{2}$. So, the probability that condition (iii) is satisfied by x' is at least

$$\frac{\left(\frac{1}{2} - \epsilon\right)|S| + |Z_{\neg y_*}|}{m} = \frac{1}{2}\left(\frac{1}{2} - \epsilon\right) + \frac{1}{2}(1 - \theta).$$

Since x and x' are sampled independently, the probability that all three conditions are satisfied is

at least

$$\left(\frac{1}{2}-\epsilon\right)\frac{\theta}{2}\cdot\left[\frac{1}{2}\left(\frac{1}{2}-\epsilon\right)+\frac{1}{2}(1-\theta)\right].$$

This expression is a quadratic function in θ, that attains its minimum at $\theta = 1$ irrespective of the value of ϵ. Hence, irrespective of $\mathcal{D}$, this probability is at least $\left[\frac{1}{2}\left(\frac{1}{2}-\epsilon\right)\right]^2$. For concreteness, let us choose ϵ to be $1/10$ (although it can be set to be much smaller). On doing so, we have that the three conditions are satisfied with probability at least $1/25$. And when these conditions are satisfied, we have $u(x, h^{\mathcal{D}}_{u,S}(x)) = 0$ and $u(x, h^{\mathcal{D}}_{u,S}(x')) = Ls/2$, i.e., x envies x' by $Ls/2 = L/8$. This shows that, when x and x' are sampled from P, with probability at least $1/25$, x envies x' by $L/8$. We conclude that with probability at least $1-2\exp(-m/100)-\exp(-m/200)$ jointly over the selection of the utility function u and the sample S, the extension of $h_{u,S}$ by $\mathcal{D}$ is not (α, β)-EF with respect to P for any $\alpha < 1/25$ and $\beta < L/8$.

To convert the joint probability into expected cost in the game, note that for two discrete, independent random variables X and Y, and for a Boolean function $\mathcal{E}(X, Y)$, it holds that

$$\Pr_{X,Y}(\mathcal{E}(X,Y)=1) = \mathbb{E}_X\left[\Pr_Y(\mathcal{E}(X,Y)=1)\right]. \tag{D.2}$$

Given sample S and utility function u, let $\mathcal{E}(u, S)$ be the Boolean function that equals 1 if and only if the extension of $h_{u,S}$ by $\mathcal{D}$ is not (α, β)-EF with respect to P for any $\alpha < 1/25$ and $\beta < L/8$. From Equation (D.2), $\Pr_{u,S}(\mathcal{E}(u, S) = 1)$ is equal to $\mathbb{E}_u[\Pr_S(\mathcal{E}(u, S) = 1)]$. The latter term is exactly the expected value of the cost, where the expectation is taken over the randomness of u. It follows that the expected cost of (any) $\mathcal{D}$ with respect to the chosen distribution over utilities is at least $1 - 2\exp(-m/100) - \exp(-m/200)$. □