Data-driven algorithm design and principled hyperparameter tuning in machine learning

Dravyansh Sharma

CMU-CS-24-120

July 2024

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

Thesis Committee:
Maria-Florina Balcan (Chair)
Tom M. Mitchell
R. Ravi
Avrim Blum (TTI-Chicago)
Tim Roughgarden (Columbia University)

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

This research was sponsored by the Toyota Technological Institute at Chicago under award number T00311601 (the Defense Advanced Research Projects Agency under cooperative agreement HR00112020003) and the National Science Foundation under award numbers 1535967, 1910321, and 1919453. 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: Learning Theory, Data-driven Algorithm Design, Hyperparameter Tuning

Abstract

For any new machine learning technique, a large body of research often follows up in order to tune the technique to work suitably for each of the numerous application areas, requiring significant scientific and engineering efforts. Moreover, this typically involves unprincipled approaches for hyperparameter selection without any guarantee on global optimality. Inspired from the recently proposed paradigm of 'data-driven algorithm design', this thesis develops first principled hyperparameter tuning techniques for several core machine learning algorithms, including semi-supervised learning, regularized linear regression, robust nearest neighbors and decision trees, with formal near-optimality guarantees in statistical and online learning settings.

Given multiple problem instances of a learning problem from some problem domain, we develop approaches to learn provably well-tuned parameters over the domain and answer questions related to the number of problem samples needed to learn a well-tuned learning algorithm. In addition, we also develop online learning techniques when the problem instances arrive in a potentially adversarial sequence. Our approaches apply to the following diverse scenarios:

- selecting graph hyperparameters in semi-supervised learning,
- setting regularization coefficients in linear regression,
- controlling the robustness vs. abstention trade-off using parameterized nearest-neighbor algorithms,
- splitting and pruning nodes for accurate and interpretable decision trees,
- meta-learning common parameters for similar tasks, and
- learning adaptively in changing environments.

In addition to providing techniques for tuning fundamental learning algorithms, we expand the applicability of data driven algorithm design to algorithm families with multiple parameters by developing better online and more computationally efficient techniques.

Acknowledgments

I am truly grateful to have Nina Balcan as my advisor, this thesis would not be possible without her guidance. When I think of Nina as a researcher the following words come to mind — passionate, practical, brilliant, innovative, assertive and efficient. I have met few people in my life with more than two of these qualities, to find them all in one person and have them guide me for over five years was a really fortunate occurrence for me. In particular, it can be hard not to be infected by Nina's endless enthusiasm and her unique combination of vigor and rigor. Her dedication to providing tireless and meticulous feedback, along with her incredible ability to quickly pin down concrete impactful steps, is truly inspirational. Going above and beyond technical and research discussions, Nina is constantly pro-active in finding all the ways in which her students can be better researchers. I also thank her for introducing me to fascinating research and amazing researchers throughout my time at CMU.

I am extremely fortunate for having Avrim Blum, Tom Mitchell, R. Ravi and Tim Roughgarden on my thesis committee, and thank them for their extremely valuable feedback and insightful research discussions. Furthermore, I really enjoyed working with and learning from Avrim, and thank him for the numerous illuminating discussions over my years as a graduate student. I also thank all my other collaborators, Travis Dick, Ameet Talwalkar, Misha Khodak, Hongyang Zhang, Steve Hanneke, Rattana Pukdee, Maxwell Jones, Anh Tuan Nguyen, Matteo Pozzi and Christopher Seiler. I greatly enjoyed working with you and learned a lot during our interactions which has in direct and indirect ways contributed to the development of this thesis. I thank Siddharth Prasad, Hedyeh Beyhaghi, Keegan Harris and Ellen Vitercik for the numerous research discussions during our reading group meetings and providing valuable feedback on my papers and presentations. I thank the students in the Computer Science Department for being welcoming and supportive colleagues. I am grateful to Deb Cavlovich, Catherine Copetas, Matthew Stewart, Jenn Landefeld, Amy Protos, Christine Martik, and all the wonderful, reliable and supportive administrative staff that made my CMU life tremendously easier. I thank my colleagues and friends with shared interests, board games, bouldering, backpacking and baking, for their support and warmth that helped me through challenging times.

Finally, I thank my family for everything. Amritansh, for always being there and for his fierce support. Mumma, for her untiring dedication. Papa, for his unwavering belief in me. I owe everything to the choices and sacrifices of my parents.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Machine learning has in the recent decades experienced a virtuous cycle of growth and adoption. But several major challenges to this development are imposed by the seemingly unprecedented march towards automation. Are we truly automating overall, or just generating alternate laborious tasks to help the machine work better? Is the rapid and often expensive deployment justified without addressing critical concerns like safety, or ensuring various important properties are met? Are various application domains uniformly able to reap the benefits of the advancement? This thesis rephrases, and attempts to initiate a new and powerful line-of-attack to address, the following tantalizing question:

Can machine learning fix its own challenge?

Assuming the widespread adoption of machine learning is here to stay, we can expect to generate and solve repeated instances of the same underlying problems. Can we leverage this availability of problems from the same domain to automatically adapt or tune our *learning techniques* to any given domain, without an excessive need of domain or application specific expertise and labour? This thesis presents principled approaches to achieve this by essentially "learning" the learning techniques, by treating the problem instances as "data" for the tuning problem. At the same time, we also ensure our approaches are sufficiently powerful to satisfy the requirements and desiderata which we have come to expect of modern machine learning systems.

More to the point, behind the scenes, typically an intensive amount of largely unprincipled effort is poured into tweaking the learning algorithms—typically by selecting real-valued "hyperparameters"—to make them work well for any given problem domain. Moreover, the increased scrutiny and lengthened list of crucial properties that come with the ubiquitous usage (e.g. robust performance under attacked inputs, ability to learn faster having seen similar tasks before, and adaptivity to unseen tasks), further magnifies this challenge. The research in this thesis takes inspiration from and builds on 'data-driven algorithm design'—a learning paradigm formally introduced by Gupta and Roughgarden [84] to the theory of computing community and further extended by Balcan et al. [7, 10, 13, 15, 16]—and makes efforts towards addressing both these concerns. Chapters 2, 3, 4 and 6 in the thesis are concerned with data-driven hyperparameter tuning in classic learning algorithms (semi-supervised learning, linear regression, decision trees and nearest-neighbor classification), while Chapters 7, 8 and 9 approach the direction of providing desirable guarantees (robustness, meta-learning and adaptivity).

The next section discusses how the thesis is organized. We also provide useful terminology and settings relevant to several chapters of the thesis, to act as a convenient reference. We will conclude this chapter with a brief discussion on related work.

1.1 Overview

As noted above, recent widespread adopton of machine learning has been accompanied by unique challenges posed by learning from data; from reducing the need for human curation of data, and finding the delicate balance between fitting observed data and generalizing to unseen data, to the challenge of learning across multiple similar tasks, and in the presence of adversaries or changing environments.

We summarize this thesis in two main themes: providing data-driven parameter selection techniques with formal guarantees for core learning problems (Chapters 2, 3, 4 and 6), and developing general tools applicable to a large class of data-driven algorithm design problems (Chapters 7, 8 and 9).

Chapter 2 considers the problem of learning with limited *labeled* data, which is often tedious and/or expensive to obtain. We focus on graph-based techniques, where the examples are connected in a graph under the implicit assumption that similar nodes likely have similar labels. We address the central challenge of how to create the graph from data, which impacts the practical usefulness of these methods significantly, but for decades has been relegated to domain-specific art and heuristics. In this chapter, we present a novel data-driven approach for learning the graph and provide strong formal guarantees in both the distributional and online learning formalizations. We expect some of the tools and techniques we develop along the way to be of general interest beyond semi-supervised learning.

Chapter 3 considers the problem of tuning the regularization parameters in popular techniques for linear regression (including the ElasticNet) across multiple problem instances, a setting that encompasses both cross-validation and multi-task hyperparameter optimization. We obtain a novel structural result for the ElasticNet which is useful to bound the structural complexity of the regularized loss functions and show generalization guarantees for tuning the ElasticNet regression coefficients in the statistical setting, and low regret guarantees in the online setting. We further extend our results to tuning classification algorithms obtained by thresholding regression fits regularized by Ridge, LASSO, or ElasticNet. Our results are the first general learning-theoretic guarantees for this important class of problems that avoid strong assumptions on the data distribution.

Chapter 4 analyzes novel and known parameterized algorithm families for learning decision trees. We consider algorithms for both classification and regression, and study approaches for top-down construction of the decision tree based on a learned splitting criterion, and pruning the built decision trees. We also study tuning of parameters in Bayesian decision tree approaches. We study the effectiveness of data-driven algorithm design for optimizing both accuracy and interpretability of the decision trees.

Chapter 5 studies the design of subsidy by a central agent in a component maintenance game relevant for civil engineering. We show that the Price of Anarchy can be very large, especially when there are a large number of components in the system. In contrast, we show that via subsidy

a central agent can improve the system performance. While good values of subsidy are easy to compute in simpler two-agent games, it is computationally hard to do so for general n-agent games under standard complexity theoretic assumptions. We show this is the case for designing subsidy allocation for a variety of relevant goals for the central agent. We further show that we can learn provably good values of subsidy in repeated games coming from the same domain.

Chapter 6 introduces the *random feature subspace* threat model, an abstraction designed to focus on the impact of non-Lipschitzness (of the model's representation mapping) on vulnerability to adversaries. Under this threat model *all* classifiers that partition the feature space into two or more classes—without an ability to abstain—are provably vulnerable to adversarial attacks for any distribution. We show that in contrast, a classifier with the ability to abstain (concretely, a thresholded nearest-neighbor algorithm) can overcome this vulnerability. We further present a novel data-driven method for learning data-specific hyperparameters in our defense algorithms to simultaneously obtain high robust accuracy and low abstention rates. We support our results with experiments on representations learned by supervised and self-supervised contrastive learning.

Chapter 7 shows how the data-driven algorithm design techniques can leverage similarity of problem instances coming from different but related domains. We present data-dependent upper bounds on the regret for a sequence of problems for a single task or domain, and optimize these bounds via meta-learning across different tasks. We instantiate our results for learning good parameters for several classic problems, and a novel robustness setting of particular relevance to meta-learning from diverse sources.

Chapter 8 extends the suite of data-driven algorithm design tools and techniques along another interesting axis, namely adaptivity to changing environment. Our approach uses a careful balance of exploration and exploitation to ensure the online learning techniques continue to work under stronger baselines, with near-optimal regret guarantees.

Chapter 9 proposes a novel computational geometry based approach for implementing the ERM algorithm for data-driven algorithm design for piecewise-structured loss functions with linear piece boundaries. We show several applications of our methodology, including pricing problems, linkage-based clustering and dynamic progamming based sequence alignment.

1.2 Preliminaries

Data-driven hyperparameter tuning. We are given a class of algorithms $\mathcal{A}$ parameterized by *hyperparameter* ρ over a set of problem instances $\mathcal{X}$, and a given utility (or loss) function $u : \mathcal{A} \times \mathcal{X} \to \mathbb{R}$ which measures the performance of any algorithm in $\mathcal{A}$ on any fixed problem instance. Suppose we need to solve repeated instances of the problem, either drawn from some problem distribution or arriving online. Can we learn a provably good value of the hyperparameter?

Distributional setting. In this setting, we assume our repeated problem instances are drawn from a fixed problem distribution $\mathcal{D}$. An important question we will be interested in is the number of 'training' problem instances that are sufficient (a.k.a. sample complexity) to learn a value of the hyperparameter ρ that is guaranteed to perform well on a random (unseen) 'test' problem instance drawn from the same distribution $\mathcal{D}$.

We state the definition of pseudo-dimension which generalizes the notion of VC dimension to real-valued functions, and is a well-known measure for hypothesis-space complexity in statistical

learning theory.

Definition 1. *Pseudo-dimension [137]. Let $\mathcal{H}$ be a set of real valued functions from input space $\mathcal{X}$. We say that $C = (x_1, \ldots, x_m) \in \mathcal{X}^m$ is pseudo-shattered by $\mathcal{H}$ if there exists a vector $r = (r_1, \ldots, r_m) \in \mathbb{R}^m$ (called "witness") such that for all $b = (b_1, \ldots, b_m) \in \{\pm 1\}^m$ there exists $h_b \in \mathcal{H}$ such that sign$(h_b(x_i) - r_i) = b_i$. Pseudo-dimension of $\mathcal{H}$ (denoted* PDIM$(\mathcal{H})$*) is the cardinality of the largest set pseudo-shattered by $\mathcal{H}$.*

Consider the set of functions $\mathcal{H}_{\mathcal{A}} = \{u(A, \cdot) \mid A \in \mathcal{A}\}$. Bounding the pseudo-dimension of $\mathcal{H}_{\mathcal{A}}$ gives a bound on the sample complexity for uniform convergence guarantees, i.e. a bound on the sample size n for which the algorithm $\hat{A}_S \in \mathcal{A}$ which minimizes the average loss on any sample S of size n drawn i.i.d. from any problem distribution $\mathcal{D}$ is guaranteed to be near-optimal with high probability [68].

Online setting. Online learning consists of a repeated game with T iterations. At iteration t, the player selects a hyperparameter value by choosing a point ρ_t from a compact decision set $\mathcal{C} \subset \mathbb{R}^d$; after the choice is committed, a bounded utility (or loss) function $u_t : \mathcal{C} \to [0, H]$ is revealed. We consider *full information* settings where $u_t(\cdot)$ is revealed over the entire domain $\mathcal{C}$, as well as partial feedback settings where the function is revealed over some subset $C_t \subseteq \mathcal{C}$ containing ρ_t. The goal of the player is to minimize the regret $R_T := \max_{\rho \in \mathcal{C}} \sum_{t=1}^{T} u_t(\rho) - \sum_{t=1}^{T} u_t(\rho_t)$, defined as the difference between the cumulative payoff using an optimal offline choice in hindsight and the online cumulative payoff accrued by the player.

We will also need the definition of *dispersion* which, informally speaking, captures how amenable a non-Lipschitz function is to online learning [15, 20].

Definition 2. *Dispersion [15]. The sequence of random loss functions $l_1, \ldots, l_T$ is β-dispersed for the Lipschitz constant L if, for all T and for all $\epsilon \geq T^{-\beta}$, we have that, in expectation, at most $\tilde{O}(\epsilon T)$ functions (the soft-O notation suppresses dependence on quantities beside ϵ, T and β, as well as logarithmic terms) are not L-Lipschitz for any pair of points at distance ϵ in the domain* $\mathbf{C}$*. That is, for all T and for all $\epsilon \geq T^{-\beta}$,*

$$\mathbb{E}\left[\max_{\substack{\rho,\rho' \in \mathbf{C} \\ ||\rho - \rho'||_2 \leq \epsilon}} \left|\{t \in [T] \mid l_t(\rho) - l_t(\rho') > L||\rho - \rho'||_2\}\right|\right] \leq \tilde{O}(\epsilon T).$$

Intuitively, a sequence of piecewise L-Lipschitz functions is well-*dispersed* if not too many functions are non-Lipschitz in the same region in $\mathcal{C}$. An assumption like this is necessary, since, even for piecewise constant functions, linear regret is unavoidable in the worst case. We will also need the following definition which captures the notion of a "smooth" distribution in several chapters. Roughly speaking the definition below captures smoothness of a distribution.

Definition 3. *A continuous probability distribution is said to be κ-bounded if the probability density function $p(x)$ satisfies $p(x) \leq \kappa$ for any x in the sample space.*

For example, the *normal* distribution $\mathcal{N}(\mu, \sigma^2)$ with mean μ and standard deviation σ is $\frac{1}{\sigma\sqrt{2\pi}}$-bounded.

1.3 Related work

Data-driven algorithm design refers to using machine learning for algorithm design, including choosing a good algorithm from a parameterized family of algorithms for given data. It is known as "hyperparameter tuning" to machine learning practitioners and typically involves a "grid search", "random search" [39] or gradient-based search, with no guarantees of convergence to a global optimum.

Data-driven algorithm design was formally introduced to the theory of computing community by Gupta and Roughgarden [84] as a learning paradigm, and was further extended by Balcan et al. [7, 10, 13, 15, 16]. The key idea is to model the problem of identifying a good algorithm from data as a statistical learning problem. The technique has found useful application in providing provably better (typically in terms of accuracy or speed) algorithms for several problems of fundamental significance in machine learning including clustering [16, 18, 24], simulated annealing [41], mixed integer programming [14, 29], low rank approximation [35] and many more [17, 25, 106, 171]. This work adapts and extends the techniques to algorithm design for core machine learning problems including semi-supervised learning [10] and regularized regression [28]. Furthermore, designing 'better' algorithms in the context modern machine learning often involves several desiderata where data-driven design can play an important role (for example, differential privacy [15, 89]). This work significantly increases the scope of data-driven design for meeting these presently prominent requirements along multiple axes—including robustness [30], meta-learning [24] and adaptive online learning [20]. There has also fundamental work on developing general tools/approaches for data-driven algorithm design that apply to a wide class of problems [15, 19, 23, 26, 84, 104, 167]. We extend this suite of tools in fundamental ways and make it significantly more powerful, in particular making them more effective in multi-parameter settings. See [7] for further context on this rapidly growing body of research.

1.4 Summary of contributions

In this thesis, we advance the theory and applicability of data-driven algorithm design along multiple axes.

Better online learning, covering multiple-parameter settings and stronger dynamic baselines. For online data-driven algorithm design, recent seminal work proposed dispersion as a sufficient condition for the existence of an online learner that achieves low expected regret guarantees. Our work significantly extends the class of algorithm design problems for which dispersion may be verified. We develop tools for establishing dispersion for single parameter problems where the dual loss function (on a fixed problem instance, as the parameter is varied) is piecewise Lipschitz, where the piece boundaries are given by roots of exponential polynomials with random coefficients [10]. We also provide a first general recipe for verifying dispersion for parameterized algorithm families with multiple parameters where the piece boundaries of the dual loss function are algebraic varieties [10], by significantly extending previous research [19] which only handled the case of one or two parameters. We further develop online learning algorithms that achieve no regret under stronger baselines of shifting experts, which capture changing

environments [20].

Design and analysis of new, useful algorithm families for ML. In addition to developing new tools and techniques for data-driven algorithm design, it is equally important to identify useful algorithm families for fundamental ML problems. We design new parameterized algorithmic families for graph-based semi-supervised learning [10], robust near-neighbor algorithms [30] and decision tree learning algorithms [11], and analyze their inherent structure to establish bounds on learning theoretic complexity. Beyond machine learning, we design a subsidy allocation family and use it to achieve various game-theoretic objectives in a component maintenance game with applications to civil engineering [32].

Adversarial robustness, applying and extending the power of data-driven algorithm design. We extend the power of data-driven algorithm design to optimize metrics beyond accuracy (e.g. clustering), profit (mechanism design) and running time (branch and bound search). We propose a class of parameterized algorithms that make non-Lipschitz networks robust to test-time adversarial attacks, by learning to abstain on potentially perturbed data. We show how data-driven algorithm design can be used to tune the parameters of the algorithm to simultaneously achieve low robust error and small abstention rate on natural data [30]. We also obtain general results on robust online data-driven algorithm design in the presence of adversarial perturbations to the observed loss function [24]. In contrast to previous works which consider bounded-norm attacks, we allow the adversary to make arbitrarily large perturbations in the feature space, provided the attack follows a smooth distribution (analogous to the "smoothed analysis" of [155]).

Better learning guarantees using multiple similar tasks. Another way in which we make data-driven algorithm design more practical is by considering a meta-learning setting which is intermediate to the (optimistic) distributional setting and the (pessimistic) online learning settings considered before [24]. We define a novel notion of task similarity which applies to the case of piecewise-Lipschitz functions that appear frequently in data-driven algorithm design, and extend the theory of online meta-learning beyond convex losses. Our learning guarantees improve with the number of tasks, and provide an improvement over the previous online learning guarantees provided the tasks are similar.

Tackling the complex hypothesis space of machine learning algorithms. While data-driven algorithm design was originally introduced to improve algorithm design, my work demonstrates the power of the paradigm for tuning core machine learning algorithms. In graph-based semi-supervised learning, we develop the first principled approaches for designing the graph, addressing a major unresolved problem in the field for over two decades [10, 152]. Key challenges involve studying the learning theoretic structure of complex concept classes induced by the family of graph algorithms, and designing efficient algorithms for online learning. We also study tuning of L1 and L2 regularization coefficients in linear regression, in both online and statistical learning settings [28, 31]. For both the above cases, we provide asymptotically tight bounds on the pseudo-dimension of algorithmic classes, essentially characterizing their learning theoretic complexity. Finally, we show how to tune several different hyperparameters when learning decision

trees using popular methods like top-down learning, Bayesian search and pruning, guaranteeing high accuracy as well as interpretability [11].

Efficient implementation. Computational efficiency is well-known as a major challenge in making data-driven algorithm design approaches practical [41, 85]. We use tools from computational geometry to develop output-sensitive algorithms for implementing the ERM in the distributional setting, when the loss function has linear piece boundaries [33]. Our work gives a first potentially efficient approach for the case of data-driven algorithm design with multiple parameters. We instantiate the result for pricing problems, linkage-based clustering and dynamic programming algorithms for sequence alignment. A key technical contribution is the development of an "execution graph" approach, that can be used to efficiently compute the refinement of the parameter space corresponding to the dual loss pieces, with each step of the algorithm (merge in clustering or a single dynamic programming update). We also develop efficient algorithms for online graph selection for semi-supervised learning [7, 152].

The work in this has thesis appeared in AISTATS 2020, NeurIPS 2021, NeurIPS 2022, NeurIPS 2023, JMLR 2023, UAI 2023, UAI 2024, EMI/PMC 2024 and the LEANOPT workshop at AAAI 2024. Part of the work in Chapter 2 received an oral presentation at NeurIPS 2021, the work in Chapter 4 received an *Outstanding Student Paper Award* and an oral presentation at UAI 2024. Part of the work in Chapter 6 was awarded an Oral presentation at the ICLR 2022 SRML workshop, and a poster based on Chapter 8 was awarded the Best Poster at the YinzOR 2019 conference.

Chapter 2

Semi-supervised learning

In recent years machine learning techniques have found gainful application in diverse settings including textual, visual, or acoustic data. A major bottleneck of the currently used approaches is the heavy dependence on expensive labeled data. At the same time advances in cheap computing and storage have made it relatively easier to store and process large amounts of unlabeled data. Therefore, an important focus of the present research community is to develop general (domain-independent) methods to learn effectively from the unlabeled data, along with a small amount of labels. Achieving this goal would significantly elevate the state-of-the-art machine intelligence, which currently lags behind the human capability of learning from a few labeled examples. Our work is a step in this direction, and provides algorithms and guarantees that enable fundamental techniques for learning from limited labeled data to provably adapt to problem domains.

Graph-based approaches have been popular for learning from unlabeled data for the past two decades [173]. Labeled and unlabeled examples form the graph nodes and (possibly weighted) edges denote the feature similarity between examples. The implicit modeling assumption needed to make semi-supervised learning possible is that the likelihood of having a particular label increases with closeness to nodes of that label [9]. The graph therefore captures how each example is related to other examples, and by optimizing a suitably regularized objective over it one obtains an efficient discriminative, nonparametric method for learning the labels. There are several well-studied ways to define and regularize an objective on the graph [54, 173], and all yield comparable results which strongly depend on the graph used. A general formulation is described as follows.

Problem formulation: Given sets L and U of labeled and unlabeled examples respectively, and a similarity metric d over the data, the goal is to use d to extrapolate labels in L to U. A graph G is constructed with $L+U$ as the nodes and weighted edges W with $w(u,v) = g(d(u,v))$ for some $g : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$. We seek labels $f(\cdot)$ for nodes u of G which minimize a regularized loss function $l(f) = \alpha \sum_{v \in L} \hat{l}(f(v), y_v) + \beta H(f, W) + \gamma \|f\|^2$, under some constraints on f. The objective H captures the *smoothness* (regularization) induced by the graph (see Table 2.1 for examples) and $\hat{l}(f(v), y_v)$ is the misclassification loss (computed here on labeled examples).

The graph G takes a central position in this formulation. However, the majority of the research effort on this problem has focused on how to design and optimize the regularized loss function $l(f)$, the effectiveness of which crucially depends on G. Indeed the graph G is expected to reflect a deep understanding of the problem structure and how the unlabeled data is expected to

Algorithm	(α, β, γ)	$H(f, W), \|\cdot\|$	Constraints on f
A. Mincut [40]	$(\infty, 1, 0)$	$f^T(D-W)f$	$f \in \{0,1\}^n$
B. Harmonic functions [174]	$(\infty, 1, 0)$	$f^T(D-W)f$	$f \in [0,1]^n$
C. Normalized cut [153]	$(\infty, 1, 0)$	$f^T(D-W)f$	$f^T\mathbf{1} = 0, f^Tf = n^2,$ $f \in [0,1]^n$
D. Label propagation [172]	$(1, \mu, 1)$	$f^T\mathcal{L}f$, $\|\cdot\|_2$	$f \in [0,1]^n$

Optimization using a quadratic objective involved in some prominent algorithms for graph-based semi-supervised learning. Here $D_{ij} := \mathbb{I}[i=j]\sum_k W_{ik}, \mathcal{L} := D^{-1/2}(D-W)D^{-1/2}$ and the objective is $l(f) = \alpha\sum_{u\in L}(f(u)-y_u)^2 + \beta H(f,W) + \gamma\|f\|^2$.

Table 2.1:

help. Despite the central role of G in the semi-supervised learning process, only some heuristics are known for setting the graph hyperparameters [175]. There is no known principled study on how to do this and prior work largely treats this as a domain-specific art. Is it possible to acquire the required domain expertise, without involving human experts?

In this work we provide an affirmative answer by introducing data-driven algorithms for building the graphs, that is techniques which learn a provably good problem-specific graph from instances of a learning problem. More precisely, we are required to solve not only one instance of the problem, but multiple instances of the underlying algorithmic problem that come from the same domain [7, 13, 84]. This approach allows us to model the problem of identifying a good algorithm from data as an online or statistical learning problem. We formulate the problem of creating the learning graph as a data-specific decision problem, where we select the graph from well-known infinite families of candidates which capture a range of ways to encode example similarity. We show learning a near-optimal graph over these families is possible in both online and distributional settings. In the process we generalize and extend results developed in the context of other data-driven learning problems, and obtain practical methods to build the graphs with strong guarantees. In particular, we show that the approach may be used for learning several parameters at once, and it is useful for learning a broader class of parameters than previously known.

2.1 Notation and definitions

We are given some labeled points L and unlabeled points U. One constructs a graph G by placing (possibly weighted) edges $w(u,v)$ between pairs of data points u, v which are 'similar', and labels for the unlabeled examples are obtained by optimizing some graph-based score. We have an oracle O which on querying provides us the labeled and unlabeled examples, and we need to pick G from some family $\mathcal{G}$ of graphs. We commit to using some algorithm $A(G, L, U)$

(abbreviated as $A_{G,L,U}$) which provides labels for examples in U, and we should pick a G such that $A(G, L, U)$ results in small error in its predictions on U. To summarize more formally,

Setup: Given data space $\mathcal{X}$, label space $\mathcal{Y}$ and an oracle O which yields a number of labeled examples $L \subset \mathcal{X} \times \mathcal{Y}$ and some unlabeled examples $U \subset \mathcal{X}$ such that $|L| + |U| = n$. We are further given a parameterized family of graph construction procedures over parameter space $\mathcal{P}$, $\mathcal{G} : \mathcal{P} \to (\mathcal{X} \times \mathcal{X} \to \mathbb{R}_{\geq 0})$, graph labeling algorithm $A : (\mathcal{X} \times \mathcal{X} \to \mathbb{R}_{\geq 0}) \times 2^{\mathcal{X} \times (\mathcal{Y} \cup \{\perp\})} \to (\mathcal{X} \to \mathcal{Y})$, a loss function $l : \mathcal{Y} \times \mathcal{Y} \to [0, 1]$ and a target labeling $\tau : U \to \mathcal{Y}$. We need to select $\rho \in \mathcal{P}$ such that corresponding graph $G(\rho)$ minimizes $\sum_U l(A_{G(\rho),L,U}(u), \tau(u))$ w.r.t. ρ.

We will consider online and distributional settings of the above. In the online setting, we will see a (potentially adversarial) sequence of semi-supervised learning problems (each with its own set of labeled and unlabeled points) and seek to minimize the regret, i.e. the loss suffered in an arbitrary online sequence of oracle queries O relative to that endured by the best parameter ρ^* in hindsight. In the distributional setting we will assume that problems supplied by O come from an underlying distribution $\mathcal{D}$ and we would like to minimize the expected loss suffered on test examples drawn from the distribution with high probability. We will present further details and notations for the respective settings in the subsequent sections.

We will now describe graph families $\mathcal{G}$ and algorithms $A_{G,L,U}$ considered in this work. We assume there is a feature based *similarity function* $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_{\geq 0}$, a metric which monotonically captures similarity between the examples. For now assume we have a single d. Definition 4 summarizes commonly used parametric methods to build a graph using the similarity function.

In this work, we will consider three parametric families of graph construction algorithms defined below. $\mathbb{I}[\cdot]$ is the indicator function taking values in $\{0, 1\}$.

Definition 4. *Graph kernels.*

a) Threshold graph, $G(r)$. Parameterized by a threshold r, we set edge weights $w(u, v) = \mathbb{I}[d(u, v) \leq r]$.

b) Polynomial kernel, $G(\tilde{\alpha})$. $w(u, v) = (\tilde{d}(u, v) + \tilde{\alpha})^d$ for fixed degree d, parameterized by $\tilde{\alpha}$.[1]

c) Gaussian RBF or exponential kernel, $G(\sigma)$. $w(u, v) = e^{-d(u,v)^2/\sigma^2}$, parameterized by bandwidth parameter σ.

Remark 1. *Another popular family of graphs used in practice is the k nearest neighbor graphs, where $k \in \{0, 1, \ldots, n-1\}$, n is the number of nodes in the graph, is the parameter. Even though k-NN graphs may result in different graphs the ones considered in the paper, learning how to build an optimal graph over the algorithm family $G(k)$ is much simpler. Online learning of the parameter k in this setting can be recognized as an instance of learning with experts advice for a finite hypothesis class (Section 3.1 of [151]), where an upper bound of $O(\sqrt{T \log n})$ is known for the Weighted Majority algorithm. Online-to-batch conversion provides generalization guarantees in the distributional setting (Section 5 of [151]).*

The threshold graph adds (unweighted) edges to G only when the examples are closer than some $r \in \mathbb{R}_{\geq 0}$, i.e. a step function of the distance. Polynomial and exponential kernels add (weighted) edges to the graph, with weights varying polynomially and exponentially (respectively) with the similarity. Note that similarity function $\tilde{d}(u, v)$ in the definition for polynomial

[1] With some notational abuse here, we have d as the integer degree of the polynomial, and $\tilde{d}(\cdot, \cdot)$ as the similarity function.

kernels increases monotonically with similarity of examples, as opposed to the other two[2]. Usually the threshold graph setting (Definition 4a) will be easier to optimize over, but it is also a small parameter family often with relatively weaker performance in practice. In the following, we will refer to this setting by the *unweighted graph* setting, and the other two settings (Definitions 4b and 4c) by the *weighted graph* setting.

Once the graph is constructed using one of the above kernels, we can assign labels using a suitable algorithm $A_{G,L,U}$. A popular and effective approach is by optimizing a quadratic objective $\frac{1}{2}\sum_{u,v} w(u,v)(f(u)-f(v))^2 = f^T(D-W)f$. Here f may either be discrete $f(u) \in \{0,1\}$ which corresponds to finding a graph mincut separating the oppositely labeled vertices [40], or f may be continuous, i.e. $f \in [0,1]$, and we can round f to obtain the labels [174]. These correspond to algorithms A and B respectively from Table 2.1. It is noted that all algorithms have comparable performance provided the graph G encodes the problem well [173].

2.2 New general dispersion-based tools

We present new techniques and generalize known tools for analyzing data-driven algorithms. Our new tools apply to a very broad class of algorithm design problems, for which we derive sufficient *smoothness* conditions to infer dispersion of a random sequence of problems, i.e. the algorithmic performance as a function of the algorithm parameters is dispersed. Balcan et al. [19] provide a general tool for verifying dispersion if non-Lipschitzness occurs along roots of (algebraic) polynomials in one and two dimensions. We improve the results along two distinct axes.

Our first result is that dispersion for one-dimensional loss functions follows when the points of discontinuity occur at the roots of exponential polynomials if the coefficients are random, lie within a finite range, and are drawn according to a bounded joint distribution.

Theorem 2.2.1. *Let $\phi(x) = \sum_{i=1}^{n} a_i e^{b_i x}$ be a random function, such that coefficients a_i are real and of magnitude at most R, and distributed with joint density at most κ. Then for any interval I of width at most ϵ, P(ϕ has a zero in I)$\leq \tilde{O}(\epsilon)$ (dependence on b_i, n, κ, R has been suppressed).*

Proof Sketch. For $n = 1$ there are no roots, so assume $n > 1$. Suppose ρ is a root of $\phi(x)$. Then $\mathbf{a} = (a_1, \ldots, a_n)$ is orthogonal to $\varrho(\rho) = (e^{b_1\rho}, \ldots, e^{b_n\rho})$ in $\mathbb{R}^n$. For a fixed ρ, the set S_ρ of coefficients $\mathbf{a}$ for which ρ is a root of $\phi(y)$ lie along an $n-1$ dimensional linear subspace of $\mathbb{R}^n$. Now ϕ has a root in any interval I of length ϵ, exactly when the coefficients lie on S_ρ for some $\rho \in I$. The desired probability is therefore upper bounded by $\max_\rho \textsc{Vol}(\cup S_y \mid y \in [\rho-\epsilon, \rho+\epsilon])/\textsc{Vol}(S_y \mid y \in \mathbb{R})$ which we will show to be $\tilde{O}(\epsilon)$. The key idea is that if $|\rho-\rho'| < \epsilon$, then $\varrho(\rho)$ and $\varrho(\rho')$ are within a small angle $\theta_{\rho,\rho'} = \tilde{O}(\epsilon)$ for small ϵ (the probability bound is vacuous for large ϵ). But any point in S_ρ is at most $\tilde{O}(\theta_{\rho,\rho'})$ from a point in $S_{\rho'}$, which implies the desired bound. □

We further go beyond single-parameter discontinuties, which occur as points along a line to general small dimensional parameter spaces $\mathbb{R}^p$, where discontinuties can occur along algebraic hypersurfaces. We employ tools from algebraic geometry to establish a bound on shattering of algebraic hypersurfaces by axis-aligned paths, which implies dispersion using a VC dimension

[2]Common choices are setting $d(u,v)$ as the Euclidean norm and $\tilde{d}(u,v)$ as the dot product, when $u, v \in \mathbb{R}^n$.

based argument. Our result is the first of its kind, a general sufficient condition for dispersion for any constant number p of parameters, and applies to a broad class of algorithm families. We will first give a bound on the ability of axis-aligned line segments to shatter a collection of algebraic hypersurfaces.

Theorem 2.2.2. *There is a constant k depending only on d and p such that axis-aligned line segments in $\mathbb{R}^p$ cannot shatter any collection of k algebraic hypersurfaces of degree at most d.*

Proof Sketch. Let $\mathbf{C}$ denote a collection of k algebraic hypersurfaces of degree at most d in $\mathbb{R}^p$. We say that a subset of $\mathbf{C}$ is *hit* by a line segment if the subset is exactly the set of curves in $\mathbf{C}$ which intersect the segment, and *hit* by a line if some segment of the line hits the subset. We can upper bound the subsets of $\mathbf{C}$ by line segments in a fixed axial direction x in two steps. Along a fixed line, Bezout's theorem bounds the number of intersections and therefore subsets hit by different line segments. The lines along x can further be shown to belong to equivalence classes corresponding to cells in the cylindrical algebraic decomposition of the projection of the hypersurfaces, orthogonal to x. Finally, we can extend this to axis-aligned segments by noting they may hit only p times as many subsets. □

Above result can be used to give a VC-dimension based argument to establish the following, which can be combined with results in [19] to establish dispersion results when discontinuities occur at roots of polynomial equations in multiple variables.

Theorem 2.2.3. *Let $l_1, \ldots, l_T : \mathbb{R}^p \to \mathbb{R}$ be independent piecewise $\mathcal{L}$-Lipschitz functions, each having discontinuities specified by a collection of at most K algebraic hypersurfaces of bounded degree. Let L denote the set of axis-aligned paths between pairs of points in $\mathbb{R}^p$, and for each $s \in L$ define $D(T, s) = |\{1 \leq t \leq T \mid l_t \text{ has a discontinuity along } s\}|$. Then we have $\mathbb{E}[\sup_{s \in L} D(T, s)] \leq \sup_{s \in L} \mathbb{E}[D(T, s)] + O(\sqrt{T \log(TK)})$.*

Proof Sketch. We relate the number of ways line segments can label vectors of K algebraic hypersurfaces of degree d to the VC-dimension of line segments (when labeling algebraic hypersurfaces), which from Theorem 2.2.2 is constant. To verify dispersion, we need a uniform-convergence bound on the number of Lipschitz failures between the worst pair of points ρ, ρ' at distance $\leq \epsilon$, but the definition allows us to bound the worst rate of discontinuties along any path between ρ, ρ' of our choice. We can bound the VC dimension of axis aligned segments against bounded-degree algebraic hypersurfaces, which will allow us to establish dispersion by considering piecewise axis-aligned paths between points ρ and ρ'. □

In the following subsections, we provide detailed proofs of the above results.

2.2.1 Dispersion for roots of exponential polynomials

In this section we will extend the applicability of the dispersion analysis technique from Appendix A.1 to exponential polynomials, i.e. functions of the form $\phi(x) = \sum_{i=1}^{n} a_i e^{b_i x}$. We will now extend the analysis to obtain similar results when using the exponential kernel $w(u, v) = e^{-||u-v||^2/\sigma^2}$. The results of Balcan et al. [19] no longer directly apply as the points of discontinuity are no longer roots of polynomials. To this end, we extend and generalize arguments from [19] below. We need to generalize Theorem A.1.1 to exponential polynomials below.

Theorem 2.2.4. *Let $\phi(x) = \sum_{i=1}^{n} a_i e^{b_i x}$ be a random function, such that coefficients a_i are real and of magnitude at most R, and distributed with joint density at most κ. Then for any interval I of width at most ϵ, P(ϕ has a zero in I)$\leq \tilde{O}(\epsilon)$ (dependence on b_i, n, κ, R suppressed).*

Proof. For $n = 1$ there are no roots, so assume $n > 1$. Suppose ρ is a root of $\phi(x)$. Then $\mathbf{a} = (a_1, \ldots, a_n)$ is orthogonal to $\varrho(\rho) = (e^{b_1\rho}, \ldots, e^{b_n\rho})$ in $\mathbb{R}^n$. For a fixed ρ, the set S_ρ of coefficients $\mathbf{a}$ for which ρ is a root of $\phi(y)$ lie along an $n-1$ dimensional linear subspace of $\mathbb{R}^n$. Now ϕ has a root in any interval I of length ϵ, exactly when the coefficients lie on S_ρ for some $\rho \in I$. The desired probability is therefore upper bounded by $\max_\rho \textsc{Vol}(\cup S_y \mid y \in [\rho-\epsilon, \rho+\epsilon])/\textsc{Vol}(S_y \mid y \in \mathbb{R})$ which we will show to be $\tilde{O}(\epsilon)$. The key idea is that if $|\rho-\rho'| < \epsilon$, then $\varrho(\rho)$ and $\varrho(\rho')$ are within a small angle $\theta_{\rho,\rho'} = \tilde{O}(\epsilon)$ for small ϵ (the probability bound is vacuous for large ϵ). But any point in S_ρ is at most $\tilde{O}(\theta_{\rho,\rho'})$ from a point in $S_{\rho'}$, which implies the desired bound (similar arguments to Theorem A.1.1).

We will now flesh out the above sketch. Indeed,

$$\begin{aligned}
\sin\theta_{\rho,\rho'} &= \sqrt{1 - \frac{\left(\langle \varrho(\rho), \varrho(\rho')\rangle\right)^2}{\|\varrho(\rho)\|\,\|\varrho(\rho')\|}} \\
&= \sqrt{1 - \frac{\left(\sum_i e^{b_i\rho} e^{b_i\rho'}\right)^2}{\sum_i e^{2b_i\rho} \sum_i e^{2b_i\rho'}}} \\
&= \sqrt{\frac{\sum_{i\neq j} e^{2(b_i\rho + b_j\rho')} - e^{(b_i+b_j)(\rho+\rho')}}{\sum_i e^{2b_i\rho} \sum_i e^{2b_i\rho'}}}.
\end{aligned}$$

Now, for $\rho' = \rho + \varepsilon$, $|\varepsilon| < \epsilon$,

$$\sin\theta_{\rho,\rho'} = \sqrt{\frac{\sum_{i\neq j} e^{2(b_i\rho + b_j\rho + b_j\varepsilon)} - e^{(b_i+b_j)(2\rho+\varepsilon)}}{\sum_i e^{2b_i\rho} \sum_i e^{2b_i\rho'}}} = \sqrt{\frac{\sum_{i\neq j} e^{2\rho(b_i+b_j)}\left(e^{2b_j\varepsilon} - e^{(b_i+b_j)\varepsilon}\right)}{\sum_i e^{2b_i\rho} \sum_i e^{2b_i\rho'}}}.$$

Using the Taylor's series approximation for $e^{2b_j\varepsilon}$ and $e^{(b_i+b_j)\varepsilon}$, we note that the largest terms that survive are quadratic in ε. $\sin\theta_{\rho,\rho'}$, and therefore also $\theta_{\rho,\rho'}$, is $\tilde{O}(\epsilon)$.

Next it is easy to show that any point in S_ρ is at most $\tilde{O}(\theta_{\rho,\rho'})$ from a point in $S_{\rho'}$. For $n = 2$, S_ρ and $S_{\rho'}$ are along lines orthogonal to ρ and ρ' and are thus themselves at an angle $\theta_{\rho,\rho'}$. Since we further assume that the coefficients are bounded by R, any point on S_ρ is within $O(R\theta_{\rho,\rho'}) = \tilde{O}(\theta_{\rho,\rho'})$ of the nearest point in $S_{\rho'}$. For $n > 2$, consider the 3-space spanned by ρ, ρ' and an arbitrary $\varsigma \in S_\rho$. S_ρ and $S_{\rho'}$ are along 2-planes in this space with normal vectors ρ, ρ' respectively. Again it is straightforward to see that the nearest point in the projection of $S_{\rho'}$ to ς is $\tilde{O}(\theta_{\rho,\rho'})$.

The remaining proof is identical to that of Theorem A.1.1 (see Theorem 18 of [19]), and is omitted for brevity.

□

We will also need the following lemma for the second step noted above, i.e. obtain a result similar to Theorem A.1.2 for exponential polynomials.

Lemma 2.2.5. *The equation $\sum_{i=1}^{n} a_i e^{b_i x} = 0$ where $a_i, b_i \in \mathbb{R}$ has at most $n-1$ distinct solutions $x \in \mathbb{R}$.*

Proof. We will use induction on n. It is easy to verify that there is no solution for $n = 1$. We assume the statement holds for all $1 \leq n \leq N$. Consider the equation $\phi_{N+1}(x) = \sum_{i=1}^{N+1} a_i e^{b_i x} = 0$. WLOG $a_1 \neq 0$ and we can write

$$\phi_{N+1}(x) = \sum_{i=1}^{N+1} a_i e^{b_i x} = a_1 e^{b_1 x} \left(1 + \sum_{i=2}^{N+1} \frac{a_i}{a_1} e^{(b_i - b_1)x}\right) = a_1 e^{b_1 x} \left(1 + g(x)\right).$$

By our induction hypothesis, $g'(0) = 0$ has at most $N-1$ solutions, and so $(1+g(x))'$ has at most $N-1$ roots. By Rolle's theorem, $(1 + g(x))$ has at most N roots, and therefore $\phi_{N+1}(x) = 0$ has at most N solutions. □

Lemma 2.2.5 implies that Theorem A.1.2 may be applied. The number of discontinuities may be exponentially high in this case. Indeed solving the quadratic objective can result in an exponential equation of the form in Lemma 2.2.5 with $O(|U|^n)$ terms.

2.2.2 Learning several metrics simultaneously

We start by getting a couple useful definitions out of the way.

Definition 5 (Homogeneous algebraic hypersurface)**.** *An algebraic hypersurface is an algebraic variety (a system of polynomial equations) that may be defined by a single implicit equation of the form $p(x_1, \ldots, x_n) = 0$, where p is a multivariate polynomial. The degree d of the algebraic hypersurface is the total degree of the polynomial p. We say that the algebraic hypersurface is homogeneous if p is a homogeneous polynomial, i.e. $p(\lambda x_1, \ldots, \lambda x_m) = \lambda^d p(x_1, \ldots, x_n)$.*

In the following we will refer to homogeneous algebraic hypersurfaces as simply algebraic hypersurfaces. We will also need the standard definition of set shattering, which we restate in our context as follows.

Definition 6 (Hitting and Shattering)**.** *Let $\mathbf{C}$ denote a set of curves in $\mathbb{R}^p$. We say that a subset of $\mathbf{C}$ is hit by a curve s if the subset is exactly the set of curves in $\mathbf{C}$ which intersect the curve s. A collection of curves $\mathcal{S}$ shatters the set $\mathbf{C}$ if for each subset C of $\mathbf{C}$, there is some element s of $\mathcal{S}$ such that s hits C.*

To extend our learning results to learning graphs built from several metrics, we will now state and prove a couple theorems involving algebraic hypersurfaces. Our results generalize significantly the techniques from [19] by bringing in novel connections with algebraic geometry.

Theorem 2.2.6. *There is a constant k depending only on d and p such that axis-aligned line segments in $\mathbb{R}^p$ cannot shatter any collection of k algebraic hypersurfaces of degree at most d.*

Proof. Let $\mathbf{C}$ denote a collection of k algebraic hypersurfaces of degree at most d in $\mathbb{R}^p$. We say that a subset of $\mathbf{C}$ is *hit* by a line segment if the subset is exactly the set of curves in $\mathbf{C}$ which intersect the segment, and *hit* by a line if some segment of the line hits the subset. We seek to upper bound the number of subsets of $\mathbf{C}$ which may be hit by axis-aligned line segments. We will first consider shattering by line segments in a fixed axial direction x. We can easily extend this to axis-aligned segments by noting they may hit only p times as many subsets.

Let L_c be a line in the x direction. The subsets of $\mathbf{C}$ which may be hit by (segments along) L_c is determined by the pattern of intersections of L_c with hypersurfaces in $\mathbf{C}$. By Bezout's theorem, there are at most $kd+1$ distinct regions of L_c due to the intersections. Therefore at most $\binom{kd+1}{2}$ distinct subsets may be hit.

Define the equivalence relation $L_{c_1} \sim L_{c_2}$ if the same hypersurfaces in $\mathbf{C}$ intersect L_{c_1} and L_{c_2}, and in the same order (including with multiplicities). To determine these equivalence classes, we will project the hypersurfaces in $\mathbf{C}$ on to a hyperplane orthogonal to the x-direction. By the Tarski-Seidenberg-Lojasiewicz Theorem, we get a semi-algebraic collection $\mathbf{C}_x$, i.e. a set of polynomial equations and constraints in the projection space. Each cell of $\mathbf{C}_x$ corresponds to an equivalence class. Using well-known upper bounds for *cylindrical algebraic decomposition* (see for example England and Davenport [73]), we get that the number of equivalence classes is at most $O\left((2d)^{2^{p-1}}k^{2^{p-1}}2^{2^{p-1}}\right)$.

Putting it all together, the number of subsets hit by any axis aligned segment is at most

$$O\left(p\binom{kd+1}{2}(2d)^{2^{p-1}}k^{2^{p-1}}2^{2^{p-1}}\right).$$

We are done as this is less than 2^k for fixed d and p and large enough k, and therefore all subsets may not be hit.

□

Theorem 2.2.7. *Let $l_1, \ldots, l_T : \mathbb{R}^p \to \mathbb{R}$ be independent piecewise $\mathcal{L}$-Lipschitz functions, each having discontinuities specified by a collection of at most K algebraic hypersurfaces of bounded degree. Let L denote the set of axis-aligned paths between pairs of points in $\mathbb{R}^p$, and for each $s \in L$ define $D(T,s) = |\{1 \le t \le T \mid l_t \text{ has a discontinuity along } s\}|$. Then we have $\mathbb{E}[\sup_{s\in L} D(T,s)] \le \sup_{s\in L} \mathbb{E}[D(T,s)] + O(\sqrt{T\log(TK)})$.*

Proof. The proof is similar to that of Theorem A.1.2 (see [19]). The main difference is that instead of relating the number of ways intervals can label vectors of discontinuity points to the VC-dimension of intervals, we instead relate the number of ways line segments can label vectors of K algebraic hypersurfaces of degree d to the VC-dimension of line segments (when labeling algebraic hypersurfaces), which from Theorem 2.2.2 is constant. To verify dispersion, we need a uniform-convergence bound on the number of Lipschitz failures between the worst pair of points α, α' at distance $\le \epsilon$, but the definition allows us to bound the worst rate of discontinuties along any path between α, α' of our choice. We can bound the VC dimension of axis aligned segments against bounded-degree algebraic hypersurfaces, which will allow us to establish dispersion by considering piecewise axis-aligned paths between points α and α'.

Let $\mathbf{C}$ denote the set of all algebraic hypersurfaces of degree d. For simplicity, we assume that every function has its discontinuities specified by a collection of exactly K algebraic hypersurfaces. For each function l_t, let $\gamma^{(t)} \in \mathbf{C}^K$ denote the ordered tuple of algebraic hypersurfaces in $\mathbf{C}$ whose entries are the discontinuity locations of l_t. That is, l_t has discontinuities along $(\gamma_1^{(t)}, \ldots, \gamma_K^{(t)})$, but is otherwise L-Lispchitz.

For any axis aligned path s, define the function $f_s : \mathbf{C}^K \to \{0, 1\}$ by

$$f_s(\gamma) = \begin{cases} 1 & \text{if for some } i \in [K], \gamma_i \text{ intersects } s, \\ 0 & \text{otherwise,} \end{cases}$$

where $\gamma = (\gamma_1, \ldots, \gamma_K) \in \mathbf{C}^K$. Now, the sum $\sum_{t=1}^T f_s(\gamma^{(t)})$ counts the number of vectors $(\gamma_1^{(t)}, \ldots, \gamma_K^{(t)})$ that intersect s or, equivalently, the number of functions $l_1, \ldots, l_T$ that are not L-Lipschitz on s. We will apply VC-dimension uniform convergence arguments to the class $\mathbf{F} = \{f_s : \mathbf{C}^K \to \{0, 1\} \mid s \text{ is an axis-aligned path}\}$. In particular, we will show that for an independent set of vectors $(\gamma_1^{(t)}, \ldots, \gamma_K^{(t)})$, with high probability we have that $\frac{1}{T}\sum_{t=1}^T f_s(\gamma^{(t)})$ is close to $\mathbb{E}[\frac{1}{T}\sum_{t=1}^T f_s(\gamma^{(t)})]$ for all paths s. This uniform convergence argument will lead to the desired bounds.

Indeed, Theorem 2.2.2 implies that VC dimension of $\mathbf{F}$ is $O(\log K)$. Now standard VC-dimension uniform convergence arguments for the class $\mathbf{F}$ imply that with probability at least $1 - \delta$, for all $f_s \in \mathbf{F}$

$$\left| \frac{1}{T}\sum_{t=1}^T f_s(\gamma^{(t)}) - \mathbb{E}\left[\frac{1}{T}\sum_{t=1}^T f_s(\gamma^{(t)})\right] \right| \leq O\left(\sqrt{\frac{\log(K/\delta)}{T}}\right), \text{ or}$$
$$\left| \sum_{t=1}^T f_s(\gamma^{(t)}) - \mathbb{E}\left[\sum_{t=1}^T f_s(\gamma^{(t)})\right] \right| \leq O\left(\sqrt{T\log(K/\delta)}\right).$$

Now since $D(T, s) = \sum_{t=1}^T f_s(\gamma^{(t)})$, we have for all s and δ, with probability at least $1 - \delta$, $\sup_{s\in L} D(T, s) \leq \sup_{s\in L} \mathbb{E}[D(T, s)] + O(\sqrt{T\log(K/\delta)})$. Taking expectation and setting $\delta = 1/\sqrt{T}$ completes the proof as it allows us to bound the expected discontinuities by $O(\sqrt{T})$ when the above high probability event fails. □

Theorem 2.2.3 above generalizes the second step of the dispersion tool from single parameter families to several hyperparameters, and uses Theorem 2.2.2 as a key ingredient. To complete the first step of in the multi-parameter setting, we can use a simple generalization of Theorem A.1.1 by showing that few zeros are likely to occur on a piecewise axis-aligned path on whose pieces the zero sets of the multivariate polynomial is the zero set of a single-variable polynomial. Putting together we get Theorem 2.2.8.

Theorem 2.2.8. *Let $l_1, \ldots, l_T : \mathbb{R}^p \to \mathbb{R}$ denote an independent sequence of losses as a function of parameters $\rho_i, i \in [p]$, when the graph is created using a polynomial kernel $w(u, v) = (\sum_{i=1}^{p-1} \rho_i \tilde{d}(u, v) + \rho_p)^d$ and labeled by optimizing the quadratic objective $\sum_{u,v} w(u, v)(f(u) - f(v))^2$. If $\tilde{d}(u, v)$ follows a κ-bounded distribution with a closed and bounded support, the sequence is $\frac{1}{2}$-dispersed, and the regret may be upper bounded by $\tilde{O}(\sqrt{T})$.*

Proof. Notice that $w(u, v)$ is a homogeneous polynomial in $\rho = (\rho_i, i \in [p])$. Further, the solutions of the quadratic objective subject to $f(u) = 1/2$ for some u are also homogeneous polynomial equations, of degree nd. Now to show dispersion, consider an axis-aligned path between any two parameter vectors ρ, ρ' such that $\|\rho - \rho'\| < \epsilon$ (notice that the definition of dispersion allows us to use any path between ρ, ρ' for counting discontinuities). To compute the

expected number of non-Lipchitzness in along this path, notice that for any fixed segment of this path, all but one variable are constant and the discontinuities are the zeros of single variable polynomial with bounded-density random coefficients, and that Theorem A.1.1 applies. Summing along these paths we get at most $\tilde{O}(p\epsilon)$ discontinuities in expectation for any $\|\rho - \rho'\| < \epsilon$. Theorem 2.2.3 now completes the proof of dispersion in this case. □

2.3 Online learning under dispersion

We consider the problem of learning the graph online. In the online setting, we are presented with instances of the problem and want to learn the best value of the parameter ρ while making predictions. We also assume we get all the labels for past instances which may be used to determine the loss for any ρ (*full information*)[3]. A choice of ρ uniquely determines the graph $G(\rho)$ (for example in single parameter families in Definition 4) and we use some algorithm $A_{G(\rho),L,U}$ to make predictions (e.g. minimizing the quadratic penalty score above) and suffer loss $l_{A(G(\rho),L,U)} := \sum_{u \in U} l(A_{G(\rho),L,U}(u), \tau(u))$ which we seek to minimize relative to the best fixed choice of ρ in hindsight. Formally, at time $t \in [T]$ we predict $p_t \in \mathcal{P}$ (the parameter space) based on labeled and unlabeled examples $(L_i, U_i), i \in [t]$ and past labels $\tau(u)$ for each $u \in U_j, j < t$ and seek to minimize

$$R_T := \sum_{t=1}^{T} l_{A(G(\rho_t),L_t,U_t)} - \min_{\rho \in \mathcal{P}} \sum_{t=1}^{T} l_{A(G(\rho),L_t,U_t)}.$$

A key difficulty in the online optimization for our settings is that the losses, as noted above, are discontinuous functions of the graph parameters ρ. We can efficiently solve this problem if we can show that the loss functions are dispersed, in fact $\frac{1}{2}$-dispersed functions may be learned with $\tilde{O}(\sqrt{T})$ regret ([15, 20]). Algorithm 1 adapts the general algorithm of [15] to data-driven graph-based learning and achieves low regret for dispersed functions. Recall that dispersion roughly says that the discontinuities in the loss function are not too concentrated. We will exploit an assumption that the embeddings are approximate, so small random perturbations to the distance metric will likely not affect learning. This mild distributional assumption allows us to show dispersion, and therefore learnability.

2.3.1 Dispersion of the loss functions

We start with showing dispersion for the unweighted graph family, with threshold parameter r. Here dispersion follows from a simple assumption that the distance $d(u, v)$ for any pair of nodes u, v follows a κ-bounded distribution[4], and observing that discontinuities of the loss (as a function of r) must lie on the set of distances $d(u, v)$ in the samples (for any optimization algorithm).

[3]We can think of each problem instance to be of a small size, so we do not need too many labels if we can learn with a reasonable number of problem instances. We improve on the label requirement further in the semi-bandit setting.

[4]A density function $f : \mathbb{R} \to \mathbb{R}$ corresponds to a κ-bounded distribution if $\max_{x \in \mathbb{R}}\{f(x)\} \leq \kappa$. For example, $\mathcal{N}(\mu, \sigma)$ is $\frac{1}{2\pi\sigma}$-bounded for any μ.

Algorithm 1 Data-driven Graph-based SSL

Input: Graphs G_t with labeled and unlabeled nodes (L_t, U_t) and node similarities $d(u,v)_{u,v \in L_t \cup U_t}$.
Hyperparameter: step size parameter $\lambda \in (0,1]$.
Output: Graph parameter ρ_t for times $t = 1, 2, \ldots, T$.
Set $w_1(\rho) = 1$ for all $\rho \in \mathbb{R}_{\geq 0}$.
for $t = 1, 2, \ldots, T$ **do**
  $W_t := \int_C w_t(\rho) d\rho$.
  Sample ρ with probability $p_t(\rho) = \frac{w_t(\rho)}{W_t}$, output as ρ_t.
  Compute average loss function $l_t(\rho) = \frac{1}{|U_t|} \sum_{u \in U} l(A_{G_t(\rho), L_t, U_t}(u), \tau(u))$.
  Set $u_t(\rho) = 1 - l_t(\rho)$ (loss is converted to utility function, $u_t(\rho) \in [0,1]$).
  For each $\rho \in C$, set $w_{t+1}(\rho) = e^{\lambda u_t(\rho)} w_t(\rho)$.

Lemma 2.3.1. *Let $\bar{l}(r) = l_{A(G(r),L,U)}$ be the loss function for graph $G(r)$ created using the threshold kernel $w(u,v) = \mathbb{I}[d(u,v) \leq r]$. Then $\bar{l}(r)$ is piecewise constant and any discontinuity occurs at $r^* = d(u,v)$ for some graph nodes u, v.*

Proof. This essentially follows from the observation that as r is increased, the graph gets a new edge only for some $r^* = d(u,v)$. Therefore no matter what the optimization algorithm is used to predict labels to minimize the loss, the loss is fixed given the graph, and has discontinuities potentially only when new edges are added. □

We can use it to show the following theorem.

Theorem 2.3.2. *Let $l_1, \ldots, l_T : \mathbb{R} \to \mathbb{R}$ denote an independent[5] sequence of losses as a function of parameter r, when the graph is created using a threshold kernel $w(u,v) = \mathbb{I}[d(u,v) \leq r]$ and labeled by applying any algorithm on the graph. If $d(u,v)$ follows a κ-bounded distribution for any u, v, the sequence is $\frac{1}{2}$-dispersed, and there is an algorithm (Algorithm 1, with suitable choice of λ) for setting r with regret upper bounded by $\tilde{O}(\sqrt{T})$.*

Proof. Assume a fixed but arbitrary ordering of nodes in each $V_t = L_t \cup U_t$ denoted by $V_t^{(i)}, i \in [n]$. Define $d_{i,j} = \{d(u,v) \mid u = V_t^{(i)}, v = V_t^{(j)}, t \in [T]\}$. Since $d_{i,j}$ is κ-bounded, the probability that it falls in any interval of length ϵ is $O(\kappa\epsilon)$. Since different problem instances are independent and using the fact that the VC dimension of intervals is 2, with probability at least $1 - \delta/D$, every interval of width ϵ contains at most $O\left(\kappa\epsilon T + \sqrt{T \log D/\delta}\right)$ discontinuities from each $d_{i,j}$ (using Lemma 2.3.1). Now a union bound over the failure modes for $d_{i,j}$ for different i, j gives $O\left(n^2 \kappa\epsilon T + n^2\sqrt{T \log n/\delta}\right)$ discontinuities with probability at least $1 - \delta$ for any ϵ-interval. Setting $\delta = 1/\sqrt{T}$, for each $\epsilon \geq 1/\sqrt{T}$ the maximum number of discontinuities in

[5]Note that the problems arriving online are adversarial. The adversary is smoothed [155] in the sense it has a distribution which it can choose as long as it has bounded density over the parameters, independent samples are drawn from adversary's distribution.

any ϵ-interval is at most $(1-\delta)O\left(\kappa n^2\sqrt{T\log n\sqrt{T}}\right)+\delta T=\tilde{O}(\epsilon T)$, in expectation, proving $\frac{1}{2}$-dispersion. □

We can show similar regret bounds via dispersion for weighted graph kernels as well. We first need a simple lemma about κ-bounded distributions. We remark that similar properties have been proved in [15, 19], in other problem contexts. Specifically, [15] show the lemma for a ratio of random variables, $Z=X/Y$, and [19] establish it for the sum $Z=X+Y$ but for independent variables X, Y.

Lemma 2.3.3. *Suppose X and Y are real-valued random variables taking values in $[m, m+M]$ and $[m', m'+M']$ for some $m, m', M, M' \in \mathbb{R}^+$ and suppose that their joint distribution is κ-bounded. Then,*

(i) $Z=X+Y$ is drawn from a $K_1\kappa$-bounded distribution, where we have $K_1 \leq \min\{M, M'\}$.

(ii) $Z=XY$ is drawn from a $K_2\kappa$-bounded distribution, where we have $K_2 \leq \min\{M/m, M'/m'\}$.

Proof. Let $f_{X,Y}(x,y)$ denote the joint density of X, Y.

(i) The case where X, Y are independent has been studied (Lemma 25 in [19]), the following is slightly more involved. The cumulative density function for Z is given by

$$\begin{aligned} F_Z(z) &= \Pr(Z \leq z) = \Pr(X+Y \leq z) = \Pr(X \leq z-Y) \\ &= \int_{m'}^{m'+M'} \int_{m}^{z-y} f_{X,Y}(x,y)dxdy. \end{aligned}$$

The density function for Z can be obtained using Leibniz's rule as

$$\begin{aligned} f_Z(z) = \frac{d}{dz}F_Z(z) &= \frac{d}{dz}\int_{m'}^{m'+M'} \int_{m}^{z-y} f_{X,Y}(x,y)dxdy \\ &= \int_{m'}^{m'+M'} \left(\frac{d}{dz}\int_{m}^{z-y} f_{X,Y}(x,y)dx\right)dy \\ &= \int_{m'}^{m'+M'} f_{X,Y}(z-y,y)dy \\ &\leq M'\kappa. \end{aligned}$$

A symmetric argument shows that $f_Z(z) \leq M\kappa$, together with above this completes the proof.

(ii) The cumulative density function for Z is given by

$$\begin{aligned} F_Z(z) &= \Pr(Z \leq z) = \Pr(XY \leq z) = \Pr(X \leq z/Y) \\ &= \int_{m'}^{m'+M'} \int_{m}^{z/y} f_{X,Y}(x,y)dxdy. \end{aligned}$$

The density function for Z can be obtained using Leibniz's rule as

$$\begin{aligned} f_Z(z) = \frac{d}{dz} F_Z(z) &= \frac{d}{dz} \int_{m'}^{m'+M'} \int_{m}^{z/y} f_{X,Y}(x,y) dx dy \\ &= \int_{m'}^{m'+M'} \left(\frac{d}{dz} \int_{m}^{z/y} f_{X,Y}(x,y) dx \right) dy \\ &= \int_{m'}^{m'+M'} \frac{1}{y} f_{X,Y}(z/y, y) dy \\ &\leq \int_{m'}^{m'+M'} \frac{1}{m'} f_{X,Y}(z/y, y) dy \\ &\leq \frac{M'}{m'} \kappa. \end{aligned}$$

Similarly we can show that $f_Z(z) \leq M\kappa/m$, together with above this completes the proof.

□

Theorem 2.3.4. *Let $l_1, \ldots, l_T : \mathbb{R} \to \mathbb{R}$ denote an independent sequence of losses as a function of parameter $\tilde{\alpha}$, when the graph is created using a polynomial kernel $w(u,v) = (\tilde{d}(u,v) + \tilde{\alpha})^d$ and labeled by optimizing the quadratic objective $\sum_{u,v} w(u,v)(f(u) - f(v))^2$. If $\tilde{d}(u,v)$ follows a κ-bounded distribution with a closed and bounded support, the sequence is $\frac{1}{2}$-dispersed, and the regret of Algorithm 1 may be upper bounded by $\tilde{O}(\sqrt{T})$.*

Proof. $w(u,v)$ is a polynomial in $\tilde{\alpha}$ of degree d with coefficient of $\tilde{\alpha}^i$ given by $c_i = D_{d,i} \tilde{d}(u,v)^{E_{d,i}}$ for $i \in [d]$. Since the support of $\tilde{d}(u,v)$ is closed and bounded, we have $m \leq \tilde{d}(u,v) \leq M$ with probability 1 for some $M > 1, m > 0$ (since $\tilde{d}(u,v)$ is a metric, $\tilde{d}(u,v) > 0$ for $u \neq v$).

To apply Theorem A.1.1, we note that we have an upper bound on the coefficients, $R < (dM)^d$. Moreover, if $f(x)$ denotes the probability density of $d(u,v)$ and $F(x)$ its cumulative density,

$$\begin{aligned} \text{Pr}(c_i \leq x_i) &= \text{Pr}\left(D_{d,i} \tilde{d}(u,v)^{E_{d,i}} \leq x_i \right) \\ &= \text{Pr}\left(\tilde{d}(u,v) \leq \left(\frac{x_i}{D_{d,i}} \right)^{1/E_{d,i}} \right) \\ &= F\left(\left(\frac{x_i}{D_{d,i}} \right)^{1/E_{d,i}} \right). \end{aligned}$$

Thus,

$$\text{Pr}(c_i \leq x_i \text{ for each } i \in [d]) = F\left(\min_i \left(\frac{x_i}{D_{d,i}} \right)^{1/E_{d,i}} \right).$$

The joint density of the coefficients is therefore $K\kappa$-bounded where K only depends on d, m. ($K \leq \max_i {D_{d,i}}^{-1/E_{d,i}} m^{-1+1/E_{d,i}}$).

Consider the harmonic solution of the quadratic objective [174] which is given by $f_U = (D_{UU} - W_{UU})^{-1} W_{UL} f_L$. For any $u \in U$, $f(u) = 1/2$ is a polynomial equation in $\tilde{\alpha}$ with degree at most nd. The coefficients of these polynomials are formed by multiplying sets of weights $w(u, v)$ of size up to n and adding the products, and are also bounded density on a bounded support (using above observation in conjunction with Lemma 2.3.3). The dispersion result now follows by an application of Theorems A.1.1 and A.1.2. The regret bound is implied by results from [15, 20]. □

Often the distance metric used for measuring similarity between the data points is a heuristic, and we can have multiple reasonable metrics. Different metrics may have their own advantages and issues and often a weighted combination of metrics, say $\sum_i \rho_i d_i(\cdot, \cdot)$, works better than any individual metric. This has been observed in practice for semi-supervised learning [12]. The combination weights ρ_i are additional graph hyperparameters. A combination of metrics has been shown to boost performance theoretically and empirically for linkage-based clustering [18]. However the argument therein crucially relies on the algorithm depending on relative distances and not the actual values, and therefore does not extend directly to our setting. We develop a first general tool for analyzing dispersion for multi-dimensional parameters (Theorem 2.2.3 above), which is used to show the following.

Theorem 2.3.5. *Let $l_1, \ldots, l_T : \mathbb{R}^p \to \mathbb{R}$ denote an independent sequence of losses as a function of parameters $\rho_i, i \in [p]$, when the graph is created using a polynomial kernel $w(u, v) = (\sum_{i=1}^{p-1} \rho_i \tilde{d}(u, v) + \rho_p)^d$ and labeled by optimizing the quadratic objective $\sum_{u,v} w(u, v)(f(u) - f(v))^2$. If $\tilde{d}(u, v)$ follows a κ-bounded distribution with a closed and bounded support, the sequence is $\frac{1}{2}$-dispersed, and the regret may be upper bounded by $\tilde{O}(\sqrt{T})$.*

2.3.2 Semi-bandit setting and efficient algorithms

Online learning with full information is usually inefficient in practice since it involves computing and working with the entire domain of hyperparameters. For our setting in particular this is computationally infeasible for weighted graphs since the number of pieces (in loss as a piecewise constant function of the parameter) may even be exponentially large. Fortunately we have a workaround provided by Balcan et al. [19] where dispersion implies learning in a semi-bandit setting as well. This setting differs from the full information online problem as follows. In each round as we select the parameter ρ_i, we only observe losses for a single interval containing ρ_i (as opposed to the entire domain). We call the set of these observable intervals the *feedback set*, and these provide a partition of the domain. The trade-off is slower convergence, the regret bound for these approaches is weaker (it is $O(\sqrt{K})$ in the size K of the feedback set instead of $O(\sqrt{\log K})$) but still converges to optimum as $\tilde{O}(1/\sqrt{T})$.

For the case of learning the unweighted threshold graph, computing the feedback set containing a given r is easy as we only need the next and previous thresholds from among the $O(n^2)$ values of pairwise distances where loss may be discontinuous in r. We present algorithms for computing the semi-bandit feedback sets (constant performance interval containing any σ) for the weighted graph setting (we will use Definition 4c for concreteness, but the algorithms easily extend to Definition 4b). We propose a hybrid combinatorial-continuous algorithm for the min-cut objective and use continuous optimization for the harmonic objective (recall objectives

Algorithm 2 Efficient Data-driven Graph-based SSL (λ)

1: **Input:** Graphs G_t with labeled and unlabeled nodes (L_t, U_t) and node similarities $d(u,v)_{u,v\in L_t\cup U_t}$.
2: **Hyperparameter:** step size parameter $\lambda \in (0,1]$.
3: **Output:** Graph parameter ρ_t for times $t = 1, 2, \ldots, T$.
4: Set $w_1(\rho) = 1$ for all $\rho \in C$
5: **for** $t = 1, 2, \ldots, T$ **do**
6: $W_t := \int_C w_t(\rho) d\rho$.
7: Sample ρ with probability $p_t(\rho) = \frac{w_t(\rho)}{W_t}$, output as ρ_t.
8: Compute the feedback set $A^{(t)}(\rho)$ containing ρ_t. For example, for the min-cut objective use Algorithm 3 to set $A^{(t)}(\rho) = \textsc{DynamicMinCut}(G_t, \rho_t, 1/\sqrt{T})$.
9: Compute average loss function $l_t(\rho) = \frac{1}{|U_t|}\sum_{u\in U} l(A_{G_t(\rho),L_t,U_t}(u), \tau(u))$.
10: Set $\hat{l}_t(\rho) = \frac{\mathbb{I}[\rho\in A^{(t)}(\rho)]}{\int_{A^{(t)}(\rho)} p_t(\rho)} l_t(\rho)$.
11: For each $\rho \in C$, set $w_{t+1}(\rho) = e^{\lambda \hat{l}_t(\rho)} w_t(\rho)$.

from Table 2.1). For the former, we describe our approach in Algorithm 3, for which we have the following runtime guarantee.

Theorem 2.3.6. *For the mincut objective and exponential kernel (Definition 4c), Algorithm 3 outputs the interval containing σ in time $O(n^3 K(n,\epsilon))$, where $K(n,\epsilon)$ is the complexity of solving an exponential equation $\phi(y) = \sum_{i=1}^{n} a_i y^{b_i} = 0$ to within error ϵ.*

Proof Sketch. Let L_1 and L_2 denote the labeled points L of different classes. To obtain the labels for U, we seek the smallest cut $(V_1, V \setminus V_1)$ of G separating the nodes in L_1 and L_2. If $L_i \subseteq V_1$, label exactly the nodes in V_1 with label i. The loss function, $l(\sigma)$ gives the fraction of labels this procedure gets right for the unlabeled set U.

It is easy to see, if the min-cut is the same for two values of σ, then so is the loss function $l(\sigma)$. So we seek the smallest amount of change in σ so that the mincut changes. Consider a fixed value of $\sigma = \sigma_0$ and the corresponding graph $G(\sigma_0)$. We can compute the max-flow on $G(\sigma)$, and simultaneously obtain a min-cut $(V_1, V \setminus V_1)$ in time $O(n^3)$. Clearly, all the edges in ∂V_1 are saturated by the flow. For each $e_i \in \partial V_1$, let f_i denote the flow that saturated e_i. Note that the f_i are distinct. Now as σ is increased, we increment each f_i by the additional capacity in the corresponding edge e_i, until an edge in $E \setminus \partial V_1$ saturates (at a faster rate than the flow through it). We now increment flows while keeping this edge saturated. The procedure stops when we can no longer find an alternate path for some flow among the unsaturated edges, which implies the existence of a new min-cut. This gives us a new critical value of σ.

Further, we can show a bound on the number of pieces. As we increase σ from σ_0 to σ_1, say edge e' saturates by increasing flow f_i. Then for any larger value of σ, capacity of e_i will exceed capacity of e'. This crucial observation allows us to bound the number of times we can find a new critical value of σ by $O(m^2)$, where m is the number of edges in G. Finally note that each time we perform step 10 of the algorithm, a new saturated edge stays saturated for all further σ until the new cut is found. So we can do this at most $O(n^2)$ times. In each loop we need to obtain the saturation condition for $O(n)$ edges. □

Algorithm 3 DYNAMICMINCUT(G, σ_0, ϵ)

Input: Graph G with unlabeled nodes, query parameter σ_0, error tolerance ϵ.
Output: Piecewise constant interval containing σ_0.
Use a max-flow algorithm to compute max-flow and min-cut $\mathbf{C}$ for $G(\sigma)$, $\sigma_h = \sigma_0$.
Compute the flow decomposition of the max-flow, $\mathbf{F}$.
Let f_e be a unique *path flow* (i.e. along an st-path) through $e \in \mathbf{C}$.
Say e is *augmentable* if flow f_e can be increased by amount $w_e(\sigma) - w_e(\sigma_h)$ for some $\sigma > \sigma_h$. e acts as the bottleneck for increasing the flow f_e.
Initialize S to $\mathbf{C}$ (a set of saturated edges).
while All edges $e \in S$ are augmentable, **do**
    Increase flow in all f_e for $e \in S$ to keep e saturated.
    Find first saturating edge $e_1 \notin S$ for some $f_{e'}$ ($e' \in S$) and σ' to within ϵ.
    Reassociate flow through e_1, e' as f_{e_1}. $f_{e'}$ will now be along an alternate path in the residual capacities graph.
    Add e_1 to S.
    Set $\sigma_h = \sigma'$.
Similarly find the start of the interval σ_l by detecting saturation while reducing flows.
return $[\sigma_l, \sigma_h]$.

For the harmonic objective, we can obtain similar efficiency (Algorithm 4). We seek points where $f_u(\sigma) = \frac{1}{2}$ for some $u \in U$ closest to given σ_0. For each u we can find the local minima of $\left(f_u(\sigma) - \frac{1}{2}\right)^2$ or simply the root of $f_u(\sigma) - \frac{1}{2}$ using gradient descent or Newton's method. The gradient computation requires $O(n^3)$ time for matrix inversion. We will later (Section 2.6) see how to make this algorithm more computationally efficient.

Theorem 2.3.7. *For the harmonic function objective (Table 2.1) and exponential kernel (Definition 4c), Algorithm 4 outputs the interval containing σ within accuracy ϵ in time $O(n^4 K_1(\epsilon))$, where $K_1(\epsilon)$ is the complexity of convergence for Newton's method ($K_1(\epsilon) = O(\log\log\frac{1}{\epsilon})$ under standard assumptions).*

Proof. The key observation is that any boundary point σ' of a piece (where the loss function is constant) has $f_u(\sigma') = \frac{1}{2}$ for some $u \in U$. This follows from continuity of $f_u(\sigma)$ (it is in fact differentiable). Algorithm 4 simply estimates the locations of these σ' closest to σ_0 for each u (by using Newton's method) to find the root of $g_u(\sigma) = (f_u(\sigma) - \frac{1}{2})^2$. For each of $O(n)$ nodes, Algorithm 4 computes the gradient and function value of $g_u(\sigma)$ in $O(n^3)$ time for different values of σ until convergence, which gives the bound on time complexity. □

2.3.3 Full details for the min-cut objective

First some notation for this section. We will use $G = (V, E)$ to denote an undirected graph with V as the set of nodes and $E \subseteq V \times V$ the weighted edges with capacity $d : E \to \mathbb{R}_{\geq 0}$. We are given special nodes $s, t \in V$ called *source* and *target* vertices. Recall the following definitions.

Algorithm 4 HARMONICFEEDBACKSET(G, σ_0, ϵ)

Input: Graph G with unlabeled nodes, query parameter σ_0, error tolerance ϵ.
Output: Piecewise constant interval containing σ_0.
Let $f_U = (D_{UU} - W_{UU})^{-1} W_{UL} f_L$ denote the harmonic objective minimizer, where $D_{ij} := \mathbb{I}[i = j] \sum_k W_{ik}$.
for all $u \in U$ **do**
    Let $g_u(\sigma) = (f_u(\sigma) - \frac{1}{2})^2$.
    $\sigma_1 = \sigma_0 - \frac{g_u(\sigma_0)}{g'_u(\sigma_0)}$, where $g'_u(\sigma)$ is given by

$$\frac{\partial g_u}{\partial \sigma} = 2\left(f_u(\sigma) - \frac{1}{2}\right)\frac{\partial f_u}{\partial \sigma},$$
$$\frac{\partial f}{\partial \sigma} = (I - P_{UU})^{-1}\left(\frac{\partial P_{UU}}{\partial \sigma} f_U - \frac{\partial P_{UL}}{\partial \sigma} f_L\right),$$
$$\frac{\partial P_{ij}}{\partial \sigma} = \frac{\frac{\partial w(i,j)}{\partial \sigma} - P_{ij} \sum_{k \in L+U} \frac{\partial w(i,k)}{\partial \sigma}}{\sum_{k \in L+U} w(i,k)},$$
$$\frac{\partial w(i,j)}{\partial \sigma} = \frac{2w(i,j)d(i,j)^2}{\sigma^3},$$

    where $P = D^{-1}W$.
    $n = 0$
    while $|\sigma_{n+1} - \sigma_n| \geq \epsilon$ **do**
        $n \leftarrow n + 1$
        $\sigma_{n+1} = \sigma_n - \frac{g_u(\sigma_n)}{g'_u(\sigma_n)}$
    $\sigma_u = \sigma_{n+1}$
$\sigma_l = \max_u\{\sigma_u \mid \sigma_u < \sigma_0\}$, $\sigma_h = \min_u\{\sigma_u \mid \sigma_u > \sigma_0\}$
return $[\sigma_l, \sigma_h]$.

Definition 7. (s,t)-flows *An (s,t)-flow (or just flow if the source and target are clear from context) is a function $f : V \times V \to \mathbb{R}_{\geq 0}$ that satisfies the conservation constraint at every vertex v except possibly s and t given by $\sum_{(u,v)\in E} f(u,v) = \sum_{(v,u)\in E} f(v,u)$. The value of flow (also refered by just flow when clear from context) is the total flow out of s, $\sum_{u\in V} f(s,u) - \sum_{u\in V} f(u,s)$.*

Definition 8. (s,t)-cut *An (s,t)-cut (or just a cut if the source and target are clear from context) is a partition of V into S, T such that $s \in S, t \in T$. We will denote the set $\{(u,v) \in E \mid u \in S, v \in T\}$ of edges in the cut by ∂S or ∂T. The capacity of the cut is the total capacity of edges in the cut.*

For convenience we also define

Definition 9. *Path flow. An (s,t)-flow is a path flow along a path $p = (s = v_0, v_1, \ldots, v_n = t)$ if $f(u,w) > 0$ iff $(u,w) = (v_i, v_{i+1})$ for some $i \in [n-1]$.*

Definition 10. *Residual capacity graph. Given a set of path flows F, the residual capacity graph (or simply the residual graph) is the graph $G' = (V, E)$ with capacities given by $c'(e) = c(e) - \sum_{f\in F} f(e)$.*

We will list without proof some well-known facts about maximum flows and minimum cuts in a graph which will be useful in our arguments.

Fact 1. *1. Let f be any feasible (s,t)-flow, and let (S,T) be any (s,t)-cut. The value of f is at most the capacity of (S,T). Moreover, the value of f equals the capacity of (S,T) if and only if f saturates every edge in the cut.*

2. Max-flow min-cut theorem. The value of maximum (value of) (s,t)-flow equals the capacity of the minimum (s,t)-cut. It may be computed in $O(VE)$ time.

3. Flow Decomposition Theorem. Every feasible (s,t)-flow f can be written as a weighted sum of directed (s,t)-paths and directed cycles. Moreover, a directed edge (u,v) appears in at least one of these paths or cycles if and only if $f(u,v) > 0$, and the total number of paths and cycles is at most the number of edges in the network. It may be computed in $O(VE)$ time.

We now have the machinery to prove the correctness and analyze the time complexity of our Algorithm 3.

Theorem 2.3.8. *For the harmonic function objective (Table 2.1) and exponential kernel (Definition 4c), Algorithm 7 outputs the interval containing σ within accuracy ϵ in time $O(n^4 K_1(\epsilon))$, where $K_1(\epsilon)$ is the complexity of convergence for Newton's method ($K_1(\epsilon) = O(\log\log\frac{1}{\epsilon})$ under standard assumptions).*

Proof. First, we briefly recall the set up of the mincut objective. Let L_1 and L_2 denote the labeled points L of different classes. To obtain the labels for U, we seek the smallest cut $(V_1, V \setminus V_1)$ of G separating the nodes in L_1 and L_2. To frame as s,t-cut we can augment the data graph with nodes s,t, and add infinite capacity edges to nodes in L_1 and L_2 respectively. If $L_i \subseteq V_1$, label exactly the nodes in V_1 with label i. The loss function, $l(\sigma)$ gives the fraction of labels this procedure gets right for the unlabeled set U.

If the min-cut is the same for two values of σ, then so is prediction on each point and thus the loss function $l(\sigma)$. So we seek the smallest amount of change in σ so that the mincut changes. Our semi-bandit feedback set is given by the intervals for which the min-cut is fixed. Consider a fixed value of $\sigma = \sigma_0$ and the corresponding graph $G(\sigma_0)$. We can compute the max-flow on $G(\sigma_0)$, and simultaneously obtain a min-cut $(V_1, V \setminus V_1)$ in time $O(VE) = O(n^3)$. All the edges in ∂V_1 are saturated by the flow. Obtain the flow decomposition of the max-flow (again $O(VE) = O(n^3)$). For each $e_i \in \partial V_1$, let f_i be a path flow through e_i from the flow decomposition (cycle flows cannot saturate, or even pass through, e_i since it is on the min-cut). Note that the f_i are distinct due to the max-flow min-cut theorem. Now as σ is increased, we increment each f_i by the additional capacity in the corresponding edge e_i, until an edge e' in $E \setminus \partial V_1$ saturates (at a faster rate than the flow through it). This can be detected by expressing f_i as a function of σ for each f_i and computing the zero of an exponential polynomial capturing the change in residual capacity of any edge $e \notin \partial V_1$. Let f_j be one of the path flows through e'. We reassign this flow to e' (it will now increase with e' as its bottleneck) and find an alternate path avoiding this edge through non-saturated edges and e_j (if one exists) along which we send the new f_j. We now increment all the path flows as before keeping their bottleneck edges saturated. The procedure stops when we can no longer find an alternate path for some e_j. But this means we must have a new cut with the saturated edges, and therefore a new min-cut. This gives us a new

critical value of σ, and the desired upper end for the feedback interval. Obtaining the lower end is possible by a symmetric procedure that decreases the path flows while keeping edges saturated.

We remark that our procedure differs from the well-known algorithms for obtaining min-cuts in a static graph. The greedy procedures for static graphs need directed edges (u, v) and (v, u) in the residual graph, and find paths through unsaturated edges through this graph to increase the flow, and cannot work with monotonically increasing path flows. We however start with a max flow and maintain the invariant that our flow equals some cut size throughout.

Finally note that each time we perform step 9 of the algorithm, a new saturated edge stays saturated for all further σ until the new cut is found. So we can do this at most $O(n^2)$ times. In each loop we need to obtain the saturation condition for $O(n)$ edges corresponding to one new path flow. □

We remind the reader that a remarkable property of finding the min-cuts dynamically in our setting is an interesting "hybrid" combinatorial and continuous set-up, which may be of independent interest. A similar dynamic, but purely combinatorial, setting for recomputing flows efficiently online over a discrete graph sequence has been studied in [3].

2.4 Distributional setting

In the distributional setting, we are presented with instances of the problem assumed to be drawn from an unknown distribution $\mathcal{D}$ and want to learn the best value of the graph parameter ρ. We also assume we get all the labels for past instances (*full information*). A choice of ρ uniquely determines the graph $G(\rho)$ and we use some algorithm $A_{G(\rho),L,U}$ to make predictions (e.g. minimizing the quadratic penalty score above) and suffers loss $l_{A(G(\rho),L,U)} := \sum_U l(A_{G(\rho),L,U}(u), \tau(u))$ which we seek to minimize relative to smallest possible loss by some graph in the hypothesis space, in expectation over the data distribution $\mathcal{D}$.

We will show a divergence in the weighted and unweighted graph learning problems. We analyze and provide asymptotically tight bounds for the pseudodimension of the set of loss functions (composed with the graph creation algorithm family and the optimization algorithm for predicting labels) parameterized by the graph family parameter ρ, i.e. $\mathcal{H}_\rho = \{l_{A(G(\rho),L,U)} \mid \rho \in \mathcal{P}\}$. For learning the unweighted threshold graphs, the pseudodimension is $O(\log n)$. However, the pseudodimension is shown to be $\Omega(n)$ for the weighted graph setting. Both these bounds are shown to be tight up to constant factors.

The online learning results above only work for smoothed but adversarial instances, while the distributional learning sample complexity results based on pseudodimension work for any types (no smoothness needed) of independent and identically distributed instances. So these results are not superseded by the online learning results, the settings are strictly speaking incomparable, and the pseudodimension results in the distributional setting provide new upper and lower bounds for the problem.

2.4.1 Pseudodimension bounds

We can efficiently learn the unweighted graph with polynomially many samples. We show this by providing a bound on the pseudodimension of the set of loss functions $\mathcal{H}_r = \{l_{A(G(r),L,U)} \mid$

$0 \leq r < \infty\}$, where $G(r)$ is specified by Definition 4a. Our bounds hold for both the min-cut and quadratic objectives (Table 2.1).

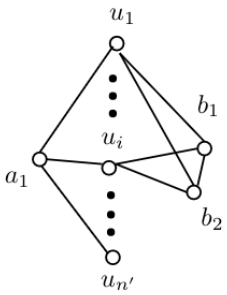
Theorem 2.4.1. *The pseudo-dimension of $\mathcal{H}_r$ is $O(\log n)$, where n is the total number of (labeled and unlabeled) data points.*

Proof. There are at most $\binom{n}{2}$ distinct distances between pairs of data points. As r is increased from 0 to infinity, the graph changes only when r corresponds to one of these distances, and so at most $\binom{n}{2} + 1$ distinct graphs may be obtained.
Thus given set $\mathcal{S}$ of m instances $(A^{(i)}, L^{(i)})$, we can partition the real line into $O(mn^2)$ intervals such that all values of r behave identically for all instances within any fixed interval. Since A and therefore its loss is deterministic once G is fixed, the loss function is a piecewise constant with only $O(n^2)$ pieces. Each piece can have a witness above or below it as r is varied for the corresponding interval, and so the binary labeling of $\mathcal{S}$ is fixed in that interval. The pseudo-dimension m satisfies $2^m \leq O(mn^2)$ and is therefore $O(\log n)$. □

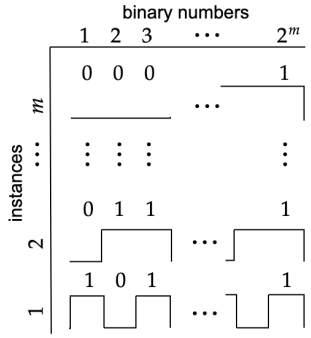
We can also show an asymptotically tight lower bound on the pseudodimension of $\mathcal{H}_r$. We do this by presenting a collection of graph thresholds and well-designed labeling instances which are shattered by the thresholds.

Theorem 2.4.2. *The pseudo-dimension of $\mathcal{H}_r$ is $\Omega(\log n)$.*

Proof Sketch. We have three labeled nodes, a_1 with label 0 and b_1, b_2 labeled 1, and $n' = O(n)$ unlabeled nodes $U = \{u_1, \ldots, u_{n'}\}$. We can show that given a sequence $\{r_1, \ldots, r_{n'}\}$ of values of r, it is possible to construct an instance with suitable true labels of U such that the loss as a function of r oscillates above and below some witness as r moves along the sequence of intervals $(r_i, r_{i+1})_{i \geq 0}$.



(a) $G(r_i)$



(b) Loss oscillations in shattered instances

Figure 2.1: Graphs $G(r)$ as r is varied, for lower bound construction for pseudodimension of $\mathcal{H}_r$. Labels are set in the instances to match bit flips in the sequence of binary numbers as shown.

At the initial threshold r_0, all unlabeled points have a single incident edge, connecting to a_1, so all predicted labels are 0. As the threshold is increased to r_i, (the distances are set so that) u_i gets connected to both nodes with label 1 and therefore its predicted label changes to 1. If the sequence of nodes u_i is alternately labeled, the loss decreases and increases alternately as all

the predicted labels turn to 1 as r is increased to $r_{n'}$. This oscillation between a high and a low value can be achieved for any subsequence of distances $r_1, \ldots, r_{n'}$, and a witness may be set as a loss value between the oscillation limits. By precisely choosing the subsequences so that the oscillations align with the bit flips in the binary digit sequence, we can construct m instances which satisfy the 2^m shattering constraints (Figure 2.1). □

For learning weighted graphs, we can show a $\Theta(n)$ bound on the pseudodimension of the set of loss functions. We show this for the loss functions for learning graphs with exponential kernels, $\mathcal{H}_\sigma = \{l_{A(G(\sigma),L,U)} \mid 0 \leq \sigma < \infty\}$, where $G(\sigma)$ is specified by Definition 4c. Our lower bound argument here is significantly more intricate.

Theorem 2.4.3. *The pseudo-dimension of $\mathcal{H}_\sigma$ is $\Theta(n)$, where n is the total number of (labeled and unlabeled) data points.*

We provide proof details for the above pseudo-dimension bounds in the following subsection.

2.4.2 Proof details

Recall that we define the set of loss functions $\mathcal{H}_r = \{l_{A(G(r),L,U)} \mid 0 \leq r < \infty\}$, where $G(r)$ is the family of threshold graphs specified by Definition 4a, and $\mathcal{H}_\sigma = \{l_{A(G(\sigma),L,U)} \mid 0 \leq \sigma < \infty\}$, where $G(\sigma)$ is the family of exponential kernel graphs specified by Definition 4c. We show lower bounds on the pseudodimension of these function classes below.

We first prove the following useful statement which helps us construct general examples with desirable properties. In particular, the following lemma guarantees that given a sequence of values of r of size $O(n)$, it is possible to construct an instance S of partially labeled points such that the cost of the output of algorithm $A(G(r), L)$ on V as a function of r oscillates above and below some threshold as r moves along the sequence of intervals (r_i, r_{i+1}). Given this powerful guarantee, we can then pick appropriate sequences of r and generate a sample set of $\Omega(\log n)$ instances that correspond to cost functions that oscillate in a manner that helps us pick $\Omega(n)$ values of r that shatters the samples.

Lemma 2.4.4. *Given integer $n > 5$ and a sequence of n' r's such that $1 < r_1 < r_2 < \cdots < r_{n'} < 2$ and $n' \leq n - 5$, there exists a real valued witness $w > 0$ and a labeling instance S of partially labeled n points, such that for $0 \leq i \leq n'/2 - 1$, $l_{A(G(r),L)} < w$ for $r \in (r_{2i}, r_{2i+1})$, and $l_{A(G(r),L)} > w$ for $r \in (r_{2i+1}, r_{2i+2})$ (where r_0 and $r_{n'+1}$ correspond to immediate left and right neighborhoods respectively of r_1 and $r_{n'}$).*

Proof. We first present a sketch of the construction. We will use binary labels a and b. We further have three points labeled a (namely a_1, a_2, a_3) and two points labeled b (say b_1, b_2). At some intial $r = r_0$, all the like-labeled points are connected in $G(r_0)$ and all the unlabeled points (namely $u_1, \ldots, u_{n'}$) are connected to a_1 as shown in Figure 2.2a. The algorithm $A(G(r), L)$ labels everything a and gets exactly half the labels right. As r is increased to r_i, u_i gets connected to b_1 and b_2 (Figure 2.2b). If the sequence u_i is alternately labeled, the loss increases and decreases alternately as all the predicted labels turn to b as r is increased to $r_{n'}$. Further increasing r may connect all the unlabeled points with true label a to a_2 and a_3 (Figure 2.2c), although this is not crucial to our argument. The rest of the proof gives concrete values of r and verifies that the construction is indeed feasible.

We will ensure all the pairwise distances are between 1 and 2, so that triangle inequality is always satisfied. It may also be readily verified that $O(\log n)$ dimensions suffice for our construction to exist. We start by defining some useful constants. We pick $r_-, r_+, r_{\max} \in (1,2)$ such that $r_- < r_1 < \cdots < r_{n'} < r_+ < r_{\max}$,

$$r_- = \frac{1+r_1}{2},$$
$$r_+ = 1 + \frac{r_{n'}}{2},$$
$$r_{\max} = 1 + \frac{r_+}{2}.$$

We will now specify the distances of the labeled points. The points with the same label are close together and away from the oppositely labeled points.

$$\begin{aligned} d(a_i, a_j) &= r_-, && 1 \le i < j \le 3, \\ d(b_1, b_2) &= r_-, && \\ d(a_i, b_j) &= r_{\max}, && 1 \le i \le 3, 1 \le j \le 2. \end{aligned}$$

Further, the unlabeled points are located as follows

$$\begin{aligned} d(a_1, u_k) &= r_-, && 1 \le k \le n', \\ d(b_i, u_k) &= r_k, && 1 \le k \le n', 1 \le i \le 2, \\ d(a_i, u_k) &= r_+, && 1 \le k \le n', 2 \le i \le 3, \\ d(u_i, u_j) &= r_{\max}, && 1 \le i < j \le n'. \end{aligned}$$

That is, all unknown points are closest to a_1, followed by b_i's, remaining a_i's and other u_i's in order. Further let the true labels of the unlabeled nodes be alternating with the index, i.e. u_k is a if and only if k is even.

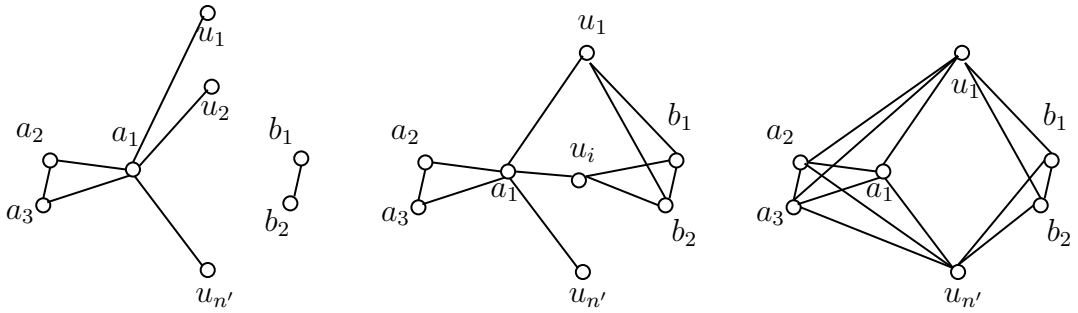
We will now compute the loss for the soft labeling algorithm $A(G(r), L)$ of [174] as r varies from r_- to r_+, starting with $r = r_0 = r_-$. We note that our construction also works for other algorithms as well, for example the min-cut based approach of [40], but omit the details.

For the graph $G(r_-)$, $A(G, L)$ labels each unknown node as a since each unknown point is a leaf node connected to a_1. Indeed if $f(a_1) = 1$, the quadratic objective attains the minimum of 0 for exactly $f(u_k) = 1$ for each $1 \le k \le n'$. This results in half the labels in the dataset being incorrectly labeled since we stipulate that half the unknown labels are of each category. This results in loss $l_{A(G(r_-),L)} =: l_{\text{high}}$ say.

Now as r is increased to r_1, the edges (b_i, u_1), $i = 1, 2$ are added with b_i labeled as $f(b_i) = 0$. This results in a fractional label of $\frac{1}{3}$ for $f(u_1)$ while $f(u_k) = 1$ for $k \neq 1$. Indeed the terms involving $f(u_1)$ in the objective are $(1 - f(u_1))^2 + 2f(u_1)^2$, which is minimized at $\frac{1}{3}$. Since u_1 has true label b, this results in a slightly smaller loss of $l_{A(G(r_1),L)} =: l_{\text{low}}$. This happens when A uses rounding, or in expectation if A uses randomized prediction with probability $f(u)$.

At the next critical point r_2, u_2 gets connected to b_i's and gets incorrectly classified as b. This increases the loss again to l_{high}. The loss function thus alternates as r is varied through the specified values, between l_{high} and l_{low}. We therefore set the witness w to something in between.

$$w = \frac{l_{\text{low}} + l_{\text{high}}}{2}.$$



(a) $G(r_-)$ (b) $G(r_i)$ (c) $G(r_+)$

Figure 2.2: Graphs $G(r)$ as r is varied, for lower bound construction for pseudodimension of $\mathcal{H}_r$.

□

Proof of Theorem 2.4.2. We will now use Lemma 2.4.4 to prove our lower bound. Arbitarily choose $n' = n - 5$ (assumed to be a power of 2 just for convenient presentation) real numbers $r_{[000...01]} < r_{[000...10]} < \cdots < r_{[111...11]}$ in $(1, 2)$. The indices are increasing binary numbers of length $m = \log n'$. We create labeling instances using Lemma 2.4.4 which can be shattered by these r values. Instance $S_i = (G_i, L_i)$ corresponds to fluctuation of i-th bit b_i in our r_b sequence, where $b = (b_1, \ldots, b_m) \in \{0, 1\}^m$, i.e., we apply the lemma by using a subset of the r_b values which correspond to the bit flips in the i-th binary digit. For example, S_1 just needs a single bit flip (at $r_{[100...00]}$). The lemma gives us both the instances and corresponding witnesses w_i.

This construction ensures $\text{sign}(l_{A(G_i(r_b),L_i)} - w_i) = b_i$, i.e. the set of instances is shattered. Thus the pseudodimension is at least $\log(n - 5) = \Omega(\log n)$.

□

Theorem 2.4.5. *The pseudo-dimension of $\mathcal{H}_\sigma$ is $O(n)$.*

Proof. There are at most $2^{|U|} \leq 2^n$ distinct cuts on any problem instance with $|U|$ unlabeled examples. As σ is increased from 0 to infinity, the number of points at which the min-cut may switch between any pair of cuts C_1 and C_2 is given by an exponential equation with $O(n^2)$ terms and therefore at most $O(n^2)$ distinct solutions in σ. Across all pairs of cuts, there are at most $O(2^{2n}n^2)$ distinct values of σ at which the min-cut may change.

Thus, given m instances we can partition the real line into $O(mn^2 2^{2n})$ intervals such that all values of σ behave identically for all problem instances and have the same value of the loss function. The pseudo-dimension m satisfies $2^m \leq O(mn^2 2^{2n})$, or $m = O(n)$. □

Theorem 2.4.6. *The pseudo-dimension of $\mathcal{H}_\sigma$ is $\Omega(n)$.*

Proof. The plan for the proof is to first construct a graph where the edge weights are carefully selected, so that we have 2^N oscillations in the loss function with σ for $N = \Omega(n)$. Then we use this construction to create $\Theta(n)$ instances, each having a subset of the oscillations so that each interval leads to a unique labeling of the instances, for a total of 2^N labelings, which would imply pseudodimension is $\Omega(n)$. We will present our discussion in terms of the min-cut objective, for simplicity of presentation.

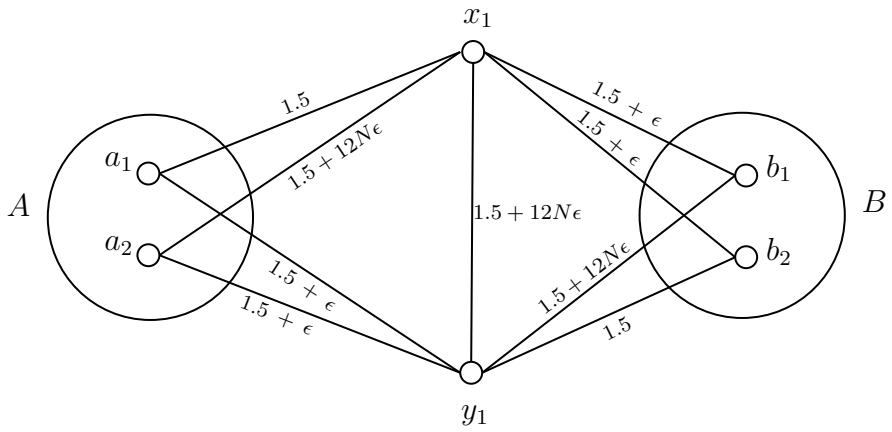


Figure 2.3: The base case of our inductive construction.

Graph construction: First a quick rough overview. We start with a pair of labeled nodes of each class, and a pair of unlabeled nodes which may be assigned either label depending on σ. We then build the graph in $N = (n-4)/2$ stages, adding two new nodes at each step with carefully chosen distances from existing nodes. Before adding the ith pair x_i, y_i of nodes, there will be 2^{i-1} intervals of σ such that the intervals correspond to distinct min-cuts which result in all possible labelings of $\{x_1, \ldots, x_{i-1}\}$. Moreover, y_j will be labeled differently from x_j in each of these intervals. The edges to the new nodes will ensure that the cuts that differ exactly in x_i will divide each of these intervals giving us 2^i intervals where distinct mincuts give all labelings of $\{x_1, \ldots, x_i\}$, and allowing an inductive proof. The challenge is that we only get to set $O(i)$ edges but seek properties about 2^i cuts, so we must do this carefully.

Let $\varsigma = e^{-1/\sigma^2}$. Notice $\varsigma \in (0,1)$, and bijectively corresponds to $\sigma \in (0,\infty)$ (due to monotonicity) and therefore it suffices to specify intervals of ς corresponding to different labelings. Further we can specify distances $d(u,v)$ between pairs of nodes u, v by specifying the squared distance $d(u,v)^2$. For the remainder of this proof we will refer to $\delta(u,v) = d(u,v)^2$ by *distance* and set values in $[1.5, 1.6]$. Consequently, $d(u,v) \in (1.22, 1.27)$ and therefore the triangle inequality is always satisfied. Notice that with this notation, the graph weights will be $w(u,v) = \varsigma^{\delta(u,v)}$.

We now provide details of the construction. We have four labeled nodes as follows. a_1, a_2 are labeled 0 and are collectively denoted by $A = \{a_1, a_2\}$, similarly b_1, b_2 are labeled 1 and $B = \{b_1, b_2\}$. Note that edges between these nodes are on all or no cut separating A, B, we set the distances to 1.6 and call this graph G_0. We further add unlabeled nodes in pairs (x_j, y_j) in *rounds* $1 \le j \le N$. In round i, we construct graph G_i by adding nodes (x_i, y_i) to G_{i-1}. The distances are set to ensure that for G_N there are 2^N unique values of ς corresponding to distinct min-cuts, each giving a unique labeling for $\{x_1, \ldots, x_n\}$ (and the complementary labeling for $\{y_1, \ldots, y_n\}$). Moreover subsets of these points also obtain the unique labeling for $\{x_1, \ldots, x_i\}$ for each G_i.

We set the distances in round 1 such that there are intervals $I_0 = (\varsigma_0, \varsigma_0') \subset (0,1)$ and $I_1 = (\varsigma_1, \varsigma_1') \subset (0,1)$ such that $\varsigma_0' < \varsigma_1$ and (x_1, y_1) are labeled $(l, 1-l)$ in interval I_l. In general, there will be 2^{i-1} intervals at the end of round $i-1$, any interval $I^{(i-1)}$ will be split into disjoint

intervals $I_0^{(i)}, I_1^{(i)} \subset I^{(i-1)}$ where labelings of $\{x_1, \ldots, x_{i-1}\}$ match that of $I^{(i-1)}$ and (x_i, y_i) are labeled $(l, 1-l)$ in $I_l^{(i)}$.

Now we set up the edges to achieve these properties. In round 1, we set the distances as follows.

$$\begin{aligned} \delta(x_1, a_1) = \delta(y_1, b_2) &= 1.5, \\ \delta(x_1, a_2) = \delta(y_1, b_1) = \delta(x_1, y_1) &= 1.5 + 12N\epsilon, \\ \delta(x_1, b_1) = \delta(x_1, b_2) = \delta(y_1, a_1) = \delta(y_1, a_2) &= 1.5 + \epsilon. \end{aligned}$$

where ϵ is a small positive quantity such that the largest distance $1.5 + 12N\epsilon < 1.6$. It is straightforward to verify that for $I_0 = (0, \frac{1}{2}^{1/\epsilon})$ we have that (x_1, y_1) are labeled $(0, 1)$ by determining the values of ς for which the corresponding cut is the min-cut (Figure 2.3). Indeed, we seek ς such that $w_{C01} = w(x_1, b_1) + w(x_1, b_2) + w(x_1, y_1) + w(y_1, a_1) + w(y_1, a_2)$ satisfies

$$w_{C01} \leq w_{C00} = w(x_1, b_1) + w(x_1, b_2) + w(y_1, b_1) + w(y_1, b_2),$$

$$w_{C01} \leq w_{C11} = w(x_1, a_1) + w(x_1, a_2) + w(y_1, a_1) + w(y_1, a_2),$$

$$w_{C01} \leq w_{C10} = w(x_1, a_1) + w(x_1, a_2) + w(x_1, y_1) + w(y_1, b_1) + w(y_1, b_2),$$

which simultaneously hold for $\varsigma < \frac{1}{2}^{1/\epsilon}$.

Moreover, we can similarly conclude that (x_1, y_1) are labeled $(1, 0)$ for the interval $I_1 = (\varsigma_1, \varsigma_1')$ where $\varsigma_1 < \varsigma_1'$ are given by the two positive roots of the equation

$$1 - 2\varsigma^{\epsilon} + 2\varsigma^{12N\epsilon} = 0.$$

We now consider the inductive step, to set the distances and obtain an inductive proof of the claim above. In round i, the distances are as specified.

$$\begin{aligned} \delta(x_i, a_1) = \delta(y_i, b_2) &= 1.5, \\ \delta(x_i, a_2) = \delta(y_i, b_1) = \delta(x_i, y_i) &= 1.5 + 12N\epsilon, \\ \delta(x_i, b_1) = \delta(x_i, b_2) = \delta(y_i, a_1) = \delta(y_i, a_2) &= 1.5 + \epsilon, \\ \delta(x_i, y_j) = \delta(y_i, x_j) &= 1.5 + 6(2j-1)\epsilon \quad (1 \leq j \leq i-1), \\ \delta(x_i, x_j) = \delta(y_i, y_j) &= 1.5 + 12j\epsilon \qquad\quad (1 \leq j \leq i-1). \end{aligned}$$

We denote the (inductively hypothesized) 2^{i-1} ς-intervals at the end of round $i-1$ by $I_{\mathbf{b}}^{(i-1)}$, where $\mathbf{b} = \{b^{(1)}, \ldots, b^{(i-1)}\} \in \{0,1\}^{i-1}$ indicates the labels of $x_j, j \in [i-1]$ in $I_{\mathbf{b}}^{(i-1)}$. Min-cuts from round $i-1$ extend to min-cuts of round i depending on how the edges incident on (x_i, y_i) are set (Figure 2.4). It suffices to consider only those min-cuts where x_j and y_j have opposite labels for each j. Consider an arbitrary such min-cut $C_{\mathbf{b}} = (A_{\mathbf{b}}, B_{\mathbf{b}})$ of G_{i-1} which corresponds to the interval $I_{\mathbf{b}}^{(i-1)}$, that is $A_{\mathbf{b}} = \{x_j \mid b^{(j)} = 0\} \cup \{y_j \mid b^{(j)} = 1\}$ and $B_{\mathbf{b}}$ contains the remaining unlabeled nodes of G_{i-1}. It extends to $C_{[\mathbf{b}\ 0]}$ and $C_{[\mathbf{b}\ 1]}$ for $\varsigma \in I_{\mathbf{b}}^{(i-1)}$ satisfying, respectively,

$$\begin{aligned} E_{\mathbf{b},0}(\varsigma) &:= \ 1 - 2\varsigma^{\epsilon} + F(C_{\mathbf{b}}; \varsigma) > 0, \\ E_{\mathbf{b},1}(\varsigma) &:= \ 1 - 2\varsigma^{\epsilon} + 2\varsigma^{12N\epsilon} + F(C_{\mathbf{b}}; \varsigma) < 0, \end{aligned}$$

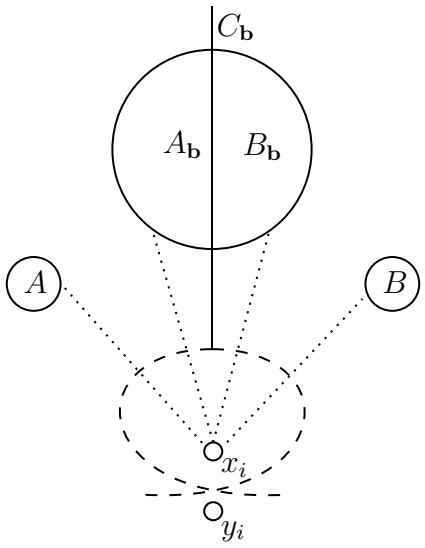


Figure 2.4: The inductive step in our lower bound construction for pseudodimension of $\mathcal{H}_\sigma$. The min-cut $C_{\mathbf{b}}$ is extended to two new min-cuts (depicted by dashed lines) for which labels of x_i, y_i are flipped, at controlled parameter intervals.

where $F(C_{\mathbf{b}}; \varsigma) = \sum_{z \in A_{\mathbf{b}}} \varsigma^{\delta(x_i,z)} - \sum_{z \in B_{\mathbf{b}}} \varsigma^{\delta(x_i,z)} = \sum_{z \in B_{\mathbf{b}}} \varsigma^{\delta(y_i,z)} - \sum_{z \in A_{\mathbf{b}}} \varsigma^{\delta(y_i,z)}$. If we show that the solutions of the above inequations have disjoint non-empty intersections with $\varsigma \in I_{\mathbf{b}}^{(i-1)}$, our induction step is complete. We will use an indirect approach for this.

For $1 \leq i \leq N$, given $\mathbf{b} = \{b^{(1)}, \ldots, b^{(i-1)}\} \in \{0,1\}^{i-1}$, let $E_{\mathbf{b},0}$ and $E_{\mathbf{b},1}$ denote the expressions (exponential polynomials in ς) in round i which determine labels of (x_i, y_i), in the case where for all $1 \leq j < i$, x_j is labeled $b^{(j)}$ (and let $E_{\phi,0}, E_{\phi,1}$ denote the expressions for round 1). Let $\varsigma_{\mathbf{b},i} \in (0,1)$ denote the smallest solution to $E_{\mathbf{b},i} = 0$. Then we need to show the $\varsigma_{\mathbf{b},i}$'s are well-defined and follow a specific ordering. This ordering is completely specified by two conditions:

(i) $\varsigma_{[\mathbf{b}\ 0],1} < \varsigma_{[\mathbf{b}],0} < \varsigma_{[\mathbf{b}],1} < \varsigma_{[\mathbf{b}\ 1],0}$, and

(ii) $\varsigma_{[\mathbf{b}\ 0\ \mathbf{c}],1} < \varsigma_{[\mathbf{b}\ 1\ \mathbf{d}],0}$

for all $\mathbf{b}, \mathbf{c}, \mathbf{d} \in \cup_{i<N}\{0,1\}^i$ and $|\mathbf{c}| = |\mathbf{d}|$.

First we make a quick observation that all $\varsigma_{\mathbf{b},i}$'s are well-defined and less than $(3/4)^{1/\epsilon}$. To do this, it will suffice to note that $E_{\mathbf{b},i}(0) = 1$ and $E_{\mathbf{b},i}(\frac{3}{4}^{1/\epsilon}) < 0$ for all $\mathbf{b}, i$, since the functions

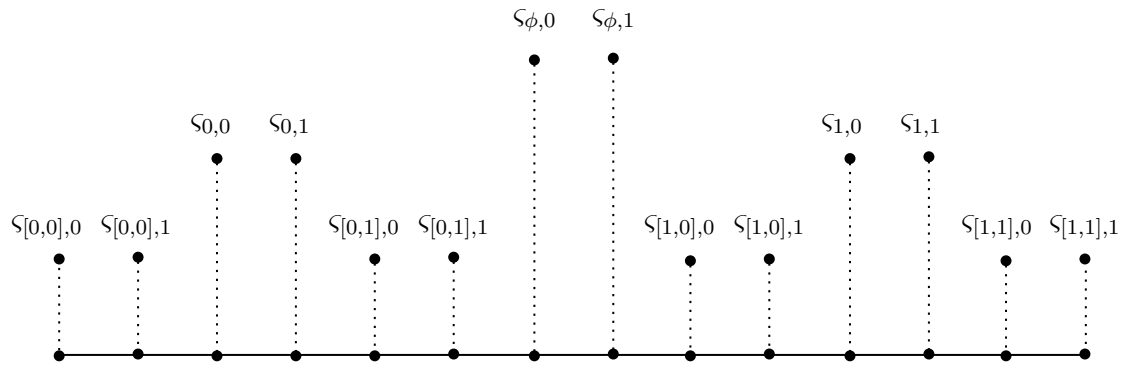


Figure 2.5: Relative positions of critical values of the parameter $\varsigma = e^{-1/\sigma^2}$.

are continuous in $(0, \frac{3}{4}^{1/\epsilon})$. This holds because

$$\begin{aligned} E_{\mathbf{b},0}\left(\frac{3}{4}^{1/\epsilon}\right) < E_{\mathbf{b},1}\left(\frac{3}{4}^{1/\epsilon}\right) &= 1 - \frac{3}{2} + \left(\frac{3}{4}\right)^{12N} + F\left(C_{\mathbf{b}}; \frac{3}{4}^{1/\epsilon}\right) \\ &\leq -\frac{1}{2} + \left(\frac{3}{4}\right)^{12N} + \sum_{j=1}^{|\mathbf{b}|}\left(\frac{3}{4}\right)^{6j}\left(1 - \left(\frac{3}{4}\right)^{6j}\right) \\ &< -\frac{1}{2} + \sum_{j=1}^{N}\left(\frac{3}{4}\right)^{6j} \\ &< 0 \end{aligned}$$

Let's now consider condition (i). We begin by showing $\varsigma_{[\mathbf{b}],0} < \varsigma_{[\mathbf{b}],1}$ for any $\mathbf{b}$. The exponential polynomials $E_{\mathbf{b},0}$ and $E_{\mathbf{b},1}$ both evaluate to 1 for $\varsigma = 0$ (since $|A_{\mathbf{b}}| = |B_{\mathbf{b}}| = |\mathbf{b}|$) and decrease monotonically (verified by elementary calculus) till their respective smallest zeros $\varsigma_{[\mathbf{b}],0}, \varsigma_{[\mathbf{b}],1}$. But then $E_{\mathbf{b},1}(\varsigma_{[\mathbf{b}],0}) = 2(\varsigma_{[\mathbf{b}],0})^{12N\epsilon} > 0$, which implies $\varsigma_{[\mathbf{b}],0} < \varsigma_{[\mathbf{b}],1}$. Now, to show $\varsigma_{[\mathbf{b}\ 0],1} < \varsigma_{[\mathbf{b}],0}$, note that $E_{[\mathbf{b}\ 0],1}(\varsigma) - E_{[\mathbf{b}],0}(\varsigma) = 2\varsigma^{12N\epsilon} + \varsigma^{12i\epsilon} - \varsigma^{(12i-6)\epsilon} = \varsigma^{(12i-6)\epsilon}(2\varsigma^{(12(N-i)+6)\epsilon} + \varsigma^{6\epsilon} - 1)$ where $1 \leq i = |\mathbf{b}| + 1 < N$. Since $\varsigma_{[\mathbf{b}],0} < \frac{3}{4}^{1/\epsilon}$, it follows that $E_{[\mathbf{b}\ 0],1}(\varsigma_{[\mathbf{b}],0}) < 0$, which implies $\varsigma_{[\mathbf{b}\ 0],1} < \varsigma_{[\mathbf{b}],0}$. Similarly, it is readily verified that $\varsigma_{[\mathbf{b}],1} < \varsigma_{[\mathbf{b}\ 1],0}$, establishing (i).

Finally, to show (ii), note that $E_{[\mathbf{b}\ 0\ \mathbf{c}],1}(\varsigma) - E_{[\mathbf{b}\ 0\ \mathbf{d}],0}(\varsigma) = 2\varsigma^{12N\epsilon} + \varsigma^{12i\epsilon} - \varsigma^{(12i-6)\epsilon} + \varsigma^{12i\epsilon}(F(C_{\mathbf{c}}; \varsigma) - F(C_{\mathbf{d}}; \varsigma)) = \varsigma^{(12i-6)\epsilon}(2\varsigma^{(12(N-i)+6)\epsilon} + \varsigma^{6\epsilon} - 1 + \varsigma^{6\epsilon}(F(C_{\mathbf{c}}; \varsigma) - F(C_{\mathbf{d}}; \varsigma)))$. Again, similar to above, we use $\varsigma_{[\mathbf{b}\ 0\ \mathbf{d}],0} < \frac{3}{4}^{1/\epsilon}$ in this expression to get $E_{[\mathbf{b}\ 0\ \mathbf{c}],1}(\varsigma_{[\mathbf{b}\ 0\ \mathbf{d}],0}) < 0$. Since the exponential polynomials decay monotonically with ς till their first roots, (ii) follows.

Problem instances: We will now show the graph instances and witnesses to establish the pseudodimension bound. Our graphs will be G_i from the above construction (padded appropriately such that the min-cut intervals do not change, if we insist each instance has exactly n nodes), and the shattering family σ_b ($b = (b_1, \ldots, b_N) \in \{0,1\}^N$) will be 2^N values of σ corresponding to the 2^N intervals of ς with distinct min-cuts in G_N described above. To obtain the witnesses, we set the labels so that only the last pair of nodes (x_i, y_i) have different labels (i.e. labels are same for all $(x_j, y_j), j < i$) and therefore the loss function oscillates 2^i times as (x_i, y_i)

are correctly and incorrectly labeled in alternating intervals. The intervals of successive G_i are nested precisely so that σ_b shatter the instances for the above labelings/witnesses. Thus, we have shown that the pseudodimension is $\Omega(N) = \Omega((n-4)/2) = \Omega(n)$. □

2.4.3 Uniform convergence

Our results above imply a uniform convergence guarantee for the distributional setting, for both weighted and unweighted graph families. We can use the pseudodimension bounds from section 2.4.1, as well as the dispersion guarantees above to bound the empirical Rademacher complexity, a useful learning theoretic complexity measure.

Definition 11. *Rademacher complexity [34]. Let $\mathcal{F} = \{f_\rho : \mathcal{X} \to [0,1], \rho \in \mathcal{C} \subset \mathbb{R}^d\}$ be a parameterized family of functions, and sample $\mathcal{S} = \{x_i, \ldots, x_T\} \subseteq \mathcal{X}$. The empirical Rademacher complexity of $\mathcal{F}$ with respect to $\mathcal{S}$ is defined as $\hat{R}(\mathcal{F}, \mathcal{S}) = \mathbb{E}_\sigma \left[\sup_{f \in \mathcal{F}} \frac{1}{T} \sum_{i=1}^{T} \sigma_i f(x_i)\right]$, where $\sigma_i \sim U(\{-1, 1\})$ are Rademacher variables.*

We will need the following theorem (slightly rephrased) from [15].

Theorem 2.4.7. *Let $\mathcal{F} = \{f_\rho : \mathcal{X} \to [0,1], \rho \in \mathcal{C} \subset \mathbb{R}^d\}$ be a parametereized family of functions, where $\mathcal{C}$ lies in a ball of radius R. For any set $\mathcal{S} = \{x_i, \ldots, x_T\} \subseteq \mathcal{X}$, suppose the functions $u_{x_i}(\rho) = f_\rho(x_i)$ for $i \in [T]$ are piecewise L-Lipschitz and β-dispersed. Then $\hat{R}(\mathcal{F}, \mathcal{S}) \leq O(\min\{\sqrt{(d/T)\log RT} + LT^{-\beta}, \sqrt{\textsc{Pdim}(\mathcal{F})/T}\})$.*

Now, combining our results above with classic results from learning theory, we can conclude that Empirical Risk Minimization has good generalization under the distribution.

2.5 Approximate semi-bandit feedback

We consider a generalization of the semi-bandit feedback setting above where only an approximation to the loss value at σ_t is revealed, along with an approximation to the piece A_t. The motivation is that approximate feedback sets may be easier to compute more efficiently, and can be used to speed up the implementation. Although our formulation below is motivated by considerations for graph parameter tuning for semi-supervised learning, we provide very general definitions and results that apply to approximate online data-driven parameter selection more generally [7].

Definition 12. *An online optimization problem with loss functions $l_1, l_2, \ldots$ is said to have (ϵ, γ)-**approximate semi-bandit feedback** with system size M if for each time $t = 1, 2, \ldots$, there is a partition $\tilde{A}_t^{(1)}, \ldots, \tilde{A}_t^{(M)}$ of the parameter space $\mathcal{P} \subset \mathbb{R}^d$, called an approximate feedback system, such that if the learner plays point $\rho_t \in \tilde{A}_t^{(i)}$, she observes the approximate feedback set $\tilde{A}_t^{(i)}$, and observes approximate loss $\tilde{l}_t(\rho)$ for all $\rho \in \tilde{A}_t^{(i)}$ such that $\sup_{\rho \in \hat{A}_t^{(i)}} |\tilde{l}_t(\rho) - l_t(\rho)| \leq \gamma$, for some (unknown) $\hat{A}_t^{(i)} \subseteq \tilde{A}_t^{(i)}$ with $\left|\mathsf{vol}\left(\tilde{A}_t^{(i)} \setminus \hat{A}_t^{(i)}\right)\right| \leq \epsilon$. Here $\mathsf{vol}(A)$ denotes the d-dimensional volume of set A. We let $\tilde{A}_t(\rho)$ denote the approximate feedback set that contains $\rho \in \mathcal{P}$.*

For example, let the parameter space $\mathcal{P}$ be one-dimensional, and in round t the learner plays point $\rho_t \in \mathcal{P}$. Now suppose the approximate loss functions are also piecewise constant with pieces $\tilde{A}_t^{(1)}, \ldots, \tilde{A}_t^{(M)}$ that partition $\mathcal{P}$, and she receives information about the constant piece

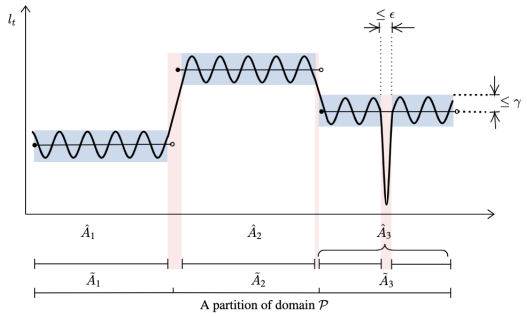


Figure 2.6: A depiction of (ϵ, γ)-approximate feedback (Definition 12) for a one dimensional loss function. Here, the true loss l_t is given by the solid curve, and approximate loss $\tilde{l}_t$ is piecewise constant.

$\tilde{A}_t(\rho_t)$ containing the played point by receiving the ends points of interval $\tilde{A}_t$ and approximate loss value $\tilde{l}_t$ for the observed piece $\tilde{A}_t$ with $|\tilde{l}_t - l_t| \leq \gamma$ for most of the interval $\tilde{A}_t$, except possibly finitely many small intervals with total length ϵ, where l_t is the true loss function. This satisfies the definition of (ϵ, γ)-approximate semi-bandit feedback. See Figure 2.6 for an illustration. This simple example captures the semi-supervised loss $l_{A(G(\rho),L,U)}$ (where in fact the true loss function is also piecewise constant [10]), but our analysis in this section applies to more general *piecewise-Lipschitz* loss functions, and for high dimensional Euclidean action space. This approximate feedback model generalizes the "exact" semibandit feedback model of Balcan et al. [19] (which in turn generalizes the standard 'full information' setting that corresponds to $M = 1$) and is useful for cases where computing the exact feedback set or loss function is infeasible or computationally expensive. Our model also generalizes the approximate loss functions of Balcan et al. [22] where positive results (data-dependent generalization guarantees) are shown for $(0, \gamma)$-approximate full-information ($M = 1$) feedback in the distributional setting. This extension is crucial for applying our techniques of efficient graph learning by computing approximate loss values for the learned graph.

Algorithm 5 APPROXIMATE CONTINUOUS EXP3-SET(λ)

Input: step size $\lambda \in [0, 1]$.
Initialize $w_1(\rho) = 1$ for all $\rho \in \mathcal{P}$.
for $t = 1, \ldots, T$ **do**
  Sample ρ_t according to $p_t(\rho) = \frac{w_t(\rho)}{\int_{\mathcal{P}} w_t(\rho) d\rho}$.
  Play ρ_t and suffer loss $l_t(\rho_t)$.
  Observe (γ, ϵ)-approximate feedback $\tilde{l}_t(\rho)$ over set $\tilde{A}_t$ with $\rho_t \in \tilde{A}_t$
  Update $w_{t+1}(\rho) = w_t(\rho) \exp(-\lambda \hat{l}_t(\rho))$, where $\hat{l}_t(\rho) = \frac{\mathbf{I}\{\rho \in \tilde{A}_t\}}{\int_{\tilde{A}_t} p_t(\rho) d\rho} \tilde{l}_t(\rho)$.

We give a general online learning algorithm in the presence of approximate semi-bandit feedback, and we show that our algorithm achieves sub-linear regret bounds. In particular, our results indicate how the approximation in the loss function impacts the regret of our algorithm and provides a way to quantify the accuracy-efficiency trade-off (better loss approximation can improve

regret in Theorem 2.5.1, but at the cost of efficiency in Theorems 2.6.3, 2.6.4).

Theorem 2.5.1. *Suppose $l_1, \ldots, l_T : \mathcal{P} \to [0,1]$ is a sequence of β-dispersed loss functions, and the domain $\mathcal{P} \subset \mathbb{R}^d$ is contained in a ball of radius R. The Approximate Continuous Exp3-Set algorithm (Algorithm 5) achieves expected regret $\tilde{O}(\sqrt{dMT\log(RT)} + T^{1-\min\{\beta,\beta'\}})$ with access to (ϵ,γ)-approximate semi-bandit feedback with system size M, provided $\gamma \leq T^{-\beta'}, \epsilon \leq \mathsf{vol}(\mathcal{B}(T^{-\beta}))T^{-\beta'}$, where $\mathcal{B}(r)$ is a d-ball of radius r.*

Proof of Theorem 2.5.1. We adapt the CONTINUOUS-EXP3-SET analysis of [2, 19]. Define weights $w_t(\rho)$ over the parameter space $\mathcal{P}$ as $w_1(\rho) = 1$ and $w_{t+1}(\rho) = w_t(\rho)\exp(-\eta\hat{l}_t(\rho))$ and normalized weights $W_t = \int_{\mathcal{P}} w_t(\rho)d\rho$. Note that $p_t(\rho) = \frac{w_t(\rho)}{W_t}$. We will give upper and lower bounds on the quantity $\mathbb{E}[\log W_{T+1}/W_1]$, i.e. the expected value of the log-ratio of normalized weights.

Using $\exp(-x) \leq 1 - x + x^2/2$ for all $x \geq 0$, we get

$$\begin{aligned}\frac{W_{t+1}}{W_t} &= \int_{\mathcal{P}} p_t(\rho)\exp(-\eta\hat{l}_t(\rho))d\rho \\ &\leq 1 - \eta\int_{\mathcal{P}} p_t(\rho)\hat{l}_t(\rho)d\rho + \frac{\eta^2}{2}\int_{\mathcal{P}} p_t(\rho)\hat{l}_t^2(\rho)d\rho.\end{aligned}$$

Computing the oscillating product and using $1 - x \leq \exp(-x)$ for all $x \geq 0$, we get

$$\frac{W_{T+1}}{W_1} \leq \exp\left(-\eta\sum_{t=1}^{T}\int_{\mathcal{P}} p_t(\rho)\hat{l}_t(\rho)d\rho + \frac{\eta^2}{2}\sum_{t=1}^{T}\int_{\mathcal{P}} p_t(\rho)\hat{l}_t^2(\rho)d\rho\right).$$

Taking logarithm and expectations on both sides we get

$$\mathbb{E}\left[\log\frac{W_{T+1}}{W_1}\right] \leq -\eta\sum_{t=1}^{T}\mathbb{E}\left[\int_{\mathcal{P}} p_t(\rho)\hat{l}_t(\rho)d\rho\right] + \frac{\eta^2}{2}\sum_{t=1}^{T}\mathbb{E}\left[\int_{\mathcal{P}} p_t(\rho)\hat{l}_t^2(\rho)d\rho\right].$$

We have, by the definitions of expectation and approximate semi-bandit feedback,

$$
\begin{aligned}
\mathbb{E}_t\left[l_t(\rho_t)\right] &= \int_{\mathcal{P}} p_t(\rho) l_t(\rho) d\rho \\
&= \sum_{i=1}^{M} \int_{\tilde{A}_t^{(i)}} p_t(\rho) l_t(\rho) d\rho \\
&= \sum_{i=1}^{M} \left[\int_{\hat{A}_t^{(i)}} p_t(\rho) l_t(\rho) d\rho + \int_{\tilde{A}_t^{(i)} \setminus \hat{A}_t^{(i)}} p_t(\rho) l_t(\rho) d\rho \right] \\
&\leq \sum_{i=1}^{M} \int_{\hat{A}_t^{(i)}} p_t(\rho) (\tilde{l}_t(\rho) + \gamma) d\rho + M\epsilon \qquad (\because\ p_t(\rho) l_t(\rho) \leq 1\ \forall\ \rho) \\
&\leq \sum_{i=1}^{M} \int_{\tilde{A}_t^{(i)}} p_t(\rho) (\tilde{l}_t(\rho) + \gamma) d\rho + M\epsilon \\
&= \int_{\mathcal{P}} p_t(\rho) \tilde{l}_t(\rho) d\rho + \gamma + M\epsilon \qquad (\because\ \int_{\mathcal{P}} p_t(\rho) d\rho = 1).
\end{aligned}
$$

Moreover,

$$
\begin{aligned}
\mathbb{E}\left[\int_{\mathcal{P}} p_t(\rho) \hat{l}_t(\rho) d\rho \right] &= \mathbb{E}_{<t} \mathbb{E}_t \left[\int_{\mathcal{P}} p_t(\rho) \hat{l}_t(\rho) d\rho \right] \\
&= \mathbb{E}_{<t} \left[\int_{\mathcal{P}} p_t(\rho) \tilde{l}_t(\rho) d\rho \right],
\end{aligned}
$$

using the definition of $\hat{l}_t$ in Algorithm 5. Plugging this in above, we get

$$
\begin{aligned}
\mathbb{E}\left[l_t(\rho_t)\right] &= \mathbb{E}_{<t} \mathbb{E}_t \left[l_t(\rho_t)\right] \\
&\leq \mathbb{E}_{<t} \left[\int_{\mathcal{P}} p_t(\rho) \tilde{l}_t(\rho) d\rho \right] + \gamma + M\epsilon \\
&= \mathbb{E}\left[\int_{\mathcal{P}} p_t(\rho) \hat{l}_t(\rho) d\rho \right] + \gamma + M\epsilon,
\end{aligned}
$$

and, further,

$$
\begin{aligned}
\mathbb{E}_t[\hat{l}_t(\rho)^2] &= \int_{\mathcal{P}} p_t(\rho') \left(\frac{\mathbf{I}[\rho \in \tilde{A}_t(\rho')]}{p_t(\tilde{A}_t(\rho'))} \tilde{l}_t(\rho) \right)^2 d\rho' \\
&= \left(\frac{\tilde{l}_t(\rho)}{p_t(\tilde{A}_t(\rho))} \right)^2 \int_{\tilde{A}_t(\rho)} p_t(\rho') d\rho' \\
&\leq \frac{1}{p_t(\tilde{A}_t(\rho))}.
\end{aligned}
$$

Therefore,

$$\mathbb{E}[\int_{\mathcal{P}} p_t(\rho)\hat{l}_t(\rho)^2 d\rho] \leq \mathbb{E}\left[\int_{\mathcal{P}} p_t(\rho) \cdot \frac{1}{p_t(\tilde{A}_t(\rho))} d\rho\right] = M.$$

Putting together, we get

$$\mathbb{E}\left[\log \frac{W_{T+1}}{W_1}\right] \leq -\eta \mathbb{E}\left[\sum_{t=1}^{T} l_t(\rho_t)\right] + \eta T(M\epsilon + \gamma) + \frac{\eta^2 MT}{2}.$$

We can also adapt the argument of [19] to obtain a lower bound for $\frac{W_{T+1}}{W_1}$ in terms of D_r, the number of L-Lipschitz violations between the worst pair of points within distance r across the T loss functions. We have

$$\begin{aligned}\frac{W_{T+1}}{W_1} &= \frac{1}{\mathsf{vol}(\mathcal{P})} \int_{\mathcal{P}} w_{T+1}(\rho) d\rho \\ &\geq \frac{1}{\mathsf{vol}(\mathcal{P})} \int_{\mathcal{B}(\rho^*, r)} w_{T+1}(\rho) d\rho.\end{aligned}$$

Taking the log and applying Jensen's inequality gives

$$\log \frac{W_{T+1}}{W_1} \geq \log \frac{\mathsf{vol}(\mathcal{B}(\rho^*, r))}{\mathsf{vol}(\mathcal{P})} - \frac{\eta}{\mathsf{vol}(\mathcal{B}(\rho^*, r))} \int_{\mathcal{B}(\rho^*, r)} \sum_{t=1}^{T} \hat{l}_t(\rho) d\rho.$$

Taking expectations w.r.t. the randomness in Algorithm 5 (but for any loss sequence $l_1, \ldots, l_t$) and using the fact that $\mathcal{P}$ is contained in a ball of radius R, we get

$$\mathbb{E}\left[\log \frac{W_{T+1}}{W_1}\right] \geq d \log \frac{r}{R} - \frac{\eta}{\mathsf{vol}(\mathcal{B}(\rho^*, r))} \sum_{t=1}^{T} \mathbb{E}\left[\int_{\mathcal{B}(\rho^*, r)} \hat{l}_t(\rho) d\rho\right].$$

Using $\mathbb{E}[\hat{l}_t(\rho)] = \tilde{l}_t(\rho)$, and noting that for any fixed t and r

$$\begin{aligned}\int_{\mathcal{B}(\rho^*, r)} \tilde{l}_t(\rho) d\rho &= \sum_{i=1}^{M} \int_{\mathcal{B}(\rho^*, r) \cap \tilde{A}_t^{(i)}} \tilde{l}_t(\rho) d\rho \\ &\leq \sum_{i=1}^{M} \int_{\mathcal{B}(\rho^*, r) \cap \hat{A}_t^{(i)}} \tilde{l}_t(\rho) d\rho + M\epsilon \\ &\leq \sum_{i=1}^{M} \int_{\mathcal{B}(\rho^*, r) \cap \hat{A}_t^{(i)}} (l_t(\rho) + \gamma) d\rho + M\epsilon \\ &\leq \sum_{i=1}^{M} \int_{\mathcal{B}(\rho^*, r) \cap \tilde{A}_t^{(i)}} \tilde{l}_t(\rho) d\rho + M\epsilon \\ &= \int_{\mathcal{B}(\rho^*, r)} l_t(\rho) d\rho + \mathsf{vol}(\mathcal{B}(\rho^*, r))\gamma + M\epsilon,\end{aligned}$$

we get that

$$\mathbb{E}\left[\log \frac{W_{T+1}}{W_1}\right] \geq d \log \frac{r}{R} - \frac{\eta}{\mathsf{vol}(\mathcal{B}(\rho^*, r))} \sum_{t=1}^{T} \int_{\mathcal{B}(\rho^*, r)} l_t(\rho) d\rho - \eta\gamma - \frac{\eta M \epsilon}{\mathsf{vol}(\mathcal{B}(\rho^*, r))}.$$

By above assumption on the number of L-Lipschitz violations we get $\sum_t l_t(\rho) \geq \sum_t l_t(\rho^*) - TLr - D_r$, or

$$\mathbb{E}\left[\log \frac{W_{T+1}}{W_1}\right] \geq d \log \frac{r}{R} - \eta \sum_{t=1}^{T} l_t(\rho^*) - \eta TLr - \eta D_r - \eta\gamma T - \frac{\eta M \epsilon T}{\mathsf{vol}(\mathcal{B}(\rho^*, r))}.$$

Combining the lower and upper bounds gives

$$\mathbb{E}\left[\sum_{t=1}^{T} l_t(\rho_t)\right] - \sum_{t=1}^{T} l_t(\rho^*) \leq \frac{d}{\eta} \log \frac{R}{r} + \frac{\eta MT}{2} + D_r + T\left(M\epsilon + 2\gamma + Lr + \frac{M\epsilon}{\mathsf{vol}(\mathcal{B}(\rho^*, r))}\right).$$

Finally, setting $r = T^{-\beta}$, $\eta = \sqrt{\frac{2d \log(RT^{\beta})}{TM}}$, $\gamma \leq T^{-\beta'}$ and $\epsilon \leq \mathsf{vol}(\mathcal{B}(r)) T^{-\beta'}$, and using that the loss sequence is β-dispersed, we get the desired regret bound

$$\begin{aligned}\mathbb{E}\left[\sum_{t=1}^{T} l_t(\rho_t) - \sum_{t=1}^{T} l_t(\rho^*)\right] &\leq O(\sqrt{dMT \log(RT)} + T^{1-\beta} + T^{1-\beta'}) \\ &= O(\sqrt{dMT \log(RT)} + T^{1-\min\{\beta, \beta'\}}).\end{aligned}$$

In particular, we have used $\mathsf{vol}(\mathcal{B}(T^{-\beta})) \leq \mathsf{vol}(\mathcal{B}(1)) \leq \frac{8\pi^2}{15}$ for any $d, T \geq 1$ and $\beta \geq 0$. □

In Theorem 2.5.1. β' measures the net impact of approximate feedback on the regret of Algorithm 5. In particular, it shows that approximation can affect regret when (γ, ϵ are such that) $\beta' < \beta$ and $\beta' < \frac{1}{2}$. The bound in Theorem 2.5.1 is good for sufficiently small γ, ϵ. However, very small γ, ϵ can come at the expense of speed. In more detail, our results in Section 2.6.2 discuss how approximate feedback can be algorithmically implemented and useful to obtain faster runtime (runtime bounds are weaker for smaller ϵ). Together, the results quantify an accuracy-efficiency trade-off, and indicate how to set the approximation parameters to improve the efficiency (of graph hyperparameter tuning) without sacrificing the accuracy.

2.6 Efficiency via sparsity and approximation

We show a formal separation in the learning-theoretic complexity of sparse and dense graph families. We further show how to approximately learn the best graphs from the sparse families efficiently using the conjugate gradient method.

2.6.1 Learning Sparse Graph Families

Let $N_k(v)$ denote the set of nodes of G which are the k-nearest neighbors of node v under the metric $d(\cdot,\cdot)$. Define k-mutual neighborhood as the set of edges for which each end-point is a k-nearest neighbor of the other, i.e. $N'_k = \{(u,v) \mid u \in N_k(v) \text{ and } v \in N_k(u)\}$ [133].

Definition 13. *Sparse graph families.*
a) Thresholded nearest neighbors, $G(k,r)$, (with $k \in \mathbb{Z}^+, r \in \mathbb{R}^+$): $w(u,v) = \mathbb{I}[d(u,v) \leq r$ and $(u,v) \in N'_k]$.
b) Gaussian nearest neighbors, $G(k,\sigma)$, (with $k \in [K]$ for $K \in \mathbb{Z}^+$, $\sigma \in \mathbb{R}^+$): $w(u,v) = e^{-\frac{d(u,v)^2}{\sigma^2}}\mathbb{I}[(u,v) \in N'_k]$.

We also define the family of loss functions $\mathcal{H}_\rho = \{l_{A(G(\rho),L,U)} \mid \rho \in \mathcal{P}\}$. For example, $\mathcal{H}_{k,r} = \{l_{A(G(k,r),L,U)} \mid (k,r) \in \mathbb{Z}^+ \times \mathbb{R}^+\}$. As a shorthand, we will often denote the loss on a fixed problem instance as a function of the graph hyperparameter ρ as simply $l(\rho)$, and refer to it as the *dual semi-supervised loss*.

We can upper bound the pseudodimension of the class of loss functions for sparse graph families, where only k nearest neighbors are connected, for tunable parameter $k \leq K$. This upper bound improves on the $O(n)$ bound from [10] since $K \leq n$, and involves a more careful argument to bound the number of possible label patterns.

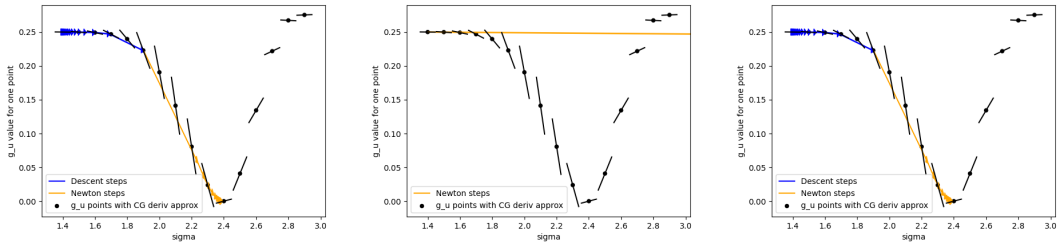
Theorem 2.6.1. *The pseudo-dimension of $\mathcal{H}_{k,\sigma}$ is $O(K + \log n)$ when the labeling algorithm A is the mincut approach of Blum and Chawla [40].*

Proof of Theorem 2.6.1. Consider an arbitrary node u in any fixed problem instance. Also fix $k \in [K]$. Since $f(d) = \exp(-d^2/\sigma^2)$ is monotonic in d for any $\sigma > 0$, the set $N_k(u)$ of k nearest neighbors of u is the same for all values σ. This is true for any u, therefore N_k and also the set of mutual nearest neighbors $N'_k(u) = \{v \in N_k(u) \mid (u,v) \in N_k\}$ is also fixed given the pairwise distances for the instance.

We can show that the label of u can flip for at most $O(K2^{2K})$ distinct values of σ for the given instance. Suppose that the label of u flips for $\sigma = \sigma_0$ (as σ is increased from 0 to infinity), say from positive to negative (WLOG). Let $S_+, S'_+ \subseteq N'_k$ for $G(k,\sigma_0^-)$ and $G(k,\sigma_0^+)$ respectively denote the positively labeled neighbors of u just before and after $\sigma = \sigma_0$. Note that σ_0 is the root of an exponential equation in at most $2k$ terms and therefore has at most $2k$ possible values (Lemma 26 in Balcan and Sharma [10]) obtained by comparing the total weights of edges in $\delta(u, N'_k \setminus S_+)$ and $\delta(u, S'_+)$, where $\delta(v,V)$ denotes the set of edges with one end-point v and the other end point in vertex set V. Over all possible pairs of S_+, S'_+ we have at most $2k\binom{2^k}{2} = O(K2^{2K})$ possibilities for σ_0.

The above bound holds for any fixed k. For all $k \in [K]$ there are at most $O(K^2 2^{2K})$ label flips for any fixed node u (as σ is varied). Summing up over all n possible choices of u and over all m problem instances, we have at most $O(mnK^2 2^K)$ intervals of σ such that the labelings of all nodes are identical for all instances, for all values of k, within a fixed interval. Using Lemma 2.3 of Balcan [7] (proof of which involves arguments similar to those used in the proof of Theorem 2.6.2), the pseudo-dimension m satisfies $2^m \leq O(mnK^2 2^K)$, or $m = O(K + \log n)$. □

The above argument gives a better sample complexity than dense graphs, for which the pseudo-dimension is known to be $\Theta(n)$ [10]. We can also give upper bounds on the pseudo-dimension



(a) Gradient Descent Only (b) Newton's Method Only (c) Our Method

Figure 2.7: An instance of a node u for graph G on a subset of the MNIST dataset, where finding local minima of $g_u(\sigma) = (f_u(\sigma) - \frac{1}{2})^2$ is challenging for both Gradient descent and Newton steps.

for $H_{k,r}$, the k-nearest neighbor graph that adds edges only in r-neighborhood, which implies existence of sample and computationally efficient algorithms for learning the best graph parameter $\rho = (k, r)$ using standard results.

Theorem 2.6.2. *The pseudo-dimension of $\mathcal{H}_{k,r}$ is $O(\log n)$ for any labeling algorithm A.*

Proof of Theorem 2.6.2. Consider any fixed problem instance with n examples. For any fixed choice of parameter k, there are at most $\frac{nk}{2}$ (unweighted) edges in $G(k, r)$ for any value of r. Therefore, as r is increased from 0 to infinity, the graph changes only when r corresponds to one of $\frac{nk}{2}$ distinct distances between pairs of data points, and so at most $\frac{nk}{2} + 1$ distinct graphs may be obtained for any k. Summing over all possible values of $k \in [n]$, we have at most $O(n^3)$ distinct graphs.

Thus given set $\mathcal{S}$ of m instances $(d^{(i)}, L^{(i)}, U^{(i)})$, we can partition the real line into $O(mn^3)$ intervals such that all values of r behave identically for all instances, and for all values of k, within any fixed interval. Since A and therefore its loss is deterministic once the graph G is fixed, the loss function is identical in each interval. Each piece can have a *witness* above or below it as r is varied for the corresponding interval, and so the binary labeling of $\mathcal{S}$ is fixed in that interval. The pseudo-dimension m satisfies $2^m \leq O(mn^3)$ and is therefore $O(\log n)$. □

Note that the lower bounds from Balcan and Sharma [10] imply that the above bound is asymptotically tight. Our bounds in this section imply upper bounds on number of problem instances needed for learning the best parameter values for the respective graph families [7] in the distributional setting.

The above argument gives a better sample complexity than dense graphs, for which the pseudo-dimension is known to be $\Theta(n)$ [10]. We can also give upper bounds on the pseudo-dimension for $H_{k,r}$, the k-nearest neighbor graph that adds edges only in r-neighborhood, which implies existence of sample and computationally efficient algorithms for learning the best graph parameter $\rho = (k, r)$ using standard results.

2.6.2 Scalability with Approximation Guarantees

We will now present and analyse an algorithm (Algorithm 7) for computing approximate semi-bandit feedback for the dual semi-supervised loss $l(\sigma)$ over $\sigma \in [\sigma_{\min}, \sigma_{\max}]$ (we assume number of nearest-neighbors k is a fixed constant in the following), where σ is the Gaussian bandwidth parameter (Def. 13). This algorithm is a scalable version of 2. Our proposed approach involves two main modifications noted below.

- Our Algorithm 7 uses approximate soft labels $f(\sigma)_\epsilon$ and gradients $\frac{\partial f}{\partial \sigma}_\epsilon$. We use the *conjugate gradient* method to compute these approximations, and provide implementations for the harmonic objective minimization approach of Zhu et al. [174], as well as the efficient algorithm of Delalleau et al. [66] with time complexity bounds (Algorithm 6 and DBLR05APPROX resp. below).
- We use the approximate gradients to locate points where $f(\sigma^*) = \frac{1}{2}$, corresponding to σ value where the predicted label flips. We use these points to find (ϵ, ϵ)-approximate feedback sets. We propose the use of smaller of Newton's step and gradient descent for better convergence to these points (line 10 in Algorithm 7; [10] use only Newton's method). We motivate this step by giving an example (from a real dataset) where the gradients are both too small and too large near the minima (Figure 2.7). This makes convergence challenging for both gradient descent and Newton's method, but the combination is effective even in this setting. We also give convergence guarantees and runtime bounds for Algorithm 7 in the presence of approximate gradients (Theorems 2.6.3, 2.6.4).

We first describe how to instantiate the sub-routine $\mathcal{A}$ to compute approximate soft labels in Algorithm 7.

Algorithm 6 computes the soft label that optimizes the harmonic function objective [174] and gradient for a given value of graph parameter σ for a fixed unlabeled node u. This is accomplished by running the conjugate gradient for given number of iterations to solve systems corresponding to the harmonic function objective and its gradient.

(Informal) DBLR05APPROX$(G, f_L, i, \sigma, \epsilon)[\tilde{U}, \lambda]$: This algorithm computes the soft label and gradient corresponding to the efficient algorithm of Delalleau et al. [66] for a graph G with parameter σ for a fixed unlabeled node $i \in U$. Unlike Algorithm 6, a matrix inverse is approximated via iterations of the CG method for the Laplacian of a small subset of unlabeled 'training' nodes $\tilde{U} \subset U$ along with a set of labeled nodes L. The labels of $i \in U \setminus \tilde{U}$ ('testing' nodes) are determined by summing the labels of each $x \in \tilde{U} \cup L$, weighted by $W_{ij}(x, i)$. The algorithm finds an ϵ approximation of $\tilde{f}_u(\sigma) \cdot \frac{\partial \tilde{f}_u}{\partial \sigma}$ using $O\left(\sqrt{\kappa(A)} \log\left(\frac{\lambda(|L_{\text{Labels}}|+|\tilde{U}_{\text{Labels}}|)}{\epsilon \sigma_{\min} \lambda_{\min}(A)}\right)\right)$ conjugate gradient iterations, where κ is the condition number, $\tilde{U} \subset U$ is a small subset, and λ is a hyperparameter.

Our main result is the following guarantee on the performance of Algorithm 7, which captures the approximation-efficiency trade-off for the algorithm. Compared to the $\tilde{O}(n^4)$ running time of the approach above, this algorithm runs in time $\tilde{O}(n^2)$ for sparse kNN graphs (i.e. k-nearest neighbors with small constant k). To achieve this speedup, we replace an $O(n^3)$ matrix inverse for a given unlabeled point with a fixed number of Conjugate Gradient iterations taking time $O(|E_G|)$, where $|E_G|$ is the number of edges for graph G corresponding to the matrix being inverted. Combined with our general algorithm for approximate data-driven algorithm design (Theorem 2.5.1), we obtain $\tilde{O}(\sqrt{T})$ expected regret for online graph parameter tuning with ap-

Algorithm 6 HARMONICAPPROXIMATION$(G, f_L, u, \sigma, \epsilon)$

1: **Input:** Graph G with labeled nodes f_L, unlabeled node u, query parameter σ, error tolerance ϵ.
2: **Output:** approximate soft label $f_{u,\epsilon}$ and approximate gradient $\frac{\partial f_u}{\partial \sigma}_\epsilon$.
3: Let $\text{CG}(A, b, t)$ represent running the conjugate gradient method for t iterations to solve $Ax = b$.
4: Let t_ϵ indicate the number of iterations sufficient for ϵ-approximation of $f_u(\sigma)\frac{\partial f_u}{\partial \sigma}$ (Theorem B.2, appendix).
5: Let $f_{U,\epsilon}(\sigma) = \text{CG}\left((I - P_{UU}), P_{UL}f_L, t_\epsilon\right)$, where $D_{ij} := \mathbb{I}[i = j]\sum_k W_{ik}, P = D^{-1}W$.
6: Let $\frac{\partial f}{\partial \sigma}_\epsilon = \text{CG}\big((I - P_{UU}), \big(\frac{\partial P_{UU}}{\partial \sigma}f_{U,\epsilon} + \frac{\partial P_{UL}}{\partial \sigma}f_L\big), t_\epsilon\big)$, where

$$\frac{\partial P_{ij}}{\partial \sigma} = \frac{\frac{\partial w(i,j)}{\partial \sigma} - P_{ij}\sum_{k\in L+U}\frac{\partial w(i,k)}{\partial \sigma}}{\sum_{k\in L+U} w(i,k)},$$
$$\frac{\partial w(i,j)}{\partial \sigma} = \frac{2w(i,j)d(i,j)^2}{\sigma^3}.$$

7: **return** $f_{u,\epsilon}(\sigma), \frac{\partial f_u}{\partial \sigma}_\epsilon$.

proximate semi-bandit feedback, provided we run Algorithm 7 with $\epsilon \leq \frac{1}{\sqrt{T}}$. For our proof, we will assume that the soft label function $f_u(\sigma)$ is convex and smooth (i.e. derivative w.r.t. σ is Lipschitz continuous) for estimating the convergence rates.

Theorem 2.6.3. *Using Algorithm 6 for computing ϵ-approximate soft labels and gradients for the harmonic objective of Zhu et al. [174], if $f_u(\sigma)$ is convex and smooth, Algorithm 7 computes (ϵ, ϵ)-approximate semi-bandit feedback for the semi-supervised loss $l(\sigma)$ in time*

$$O\left(|E_G|n\sqrt{\kappa(\mathcal{L}_{UU})}\log\left(\frac{n\Delta}{\epsilon\lambda_{\min}(\mathcal{L}_{UU})}\right)\log\log\frac{1}{\epsilon}\right),$$

where $|E_G|$ is the number of edges in graph G, $\mathcal{L}_{UU} = I - P_{UU}$ is the normalized grounded graph Laplacian (with labeled nodes grounded), $\Delta = \sigma_{\max} - \sigma_{\min}$ is the size of the parameter range and $\kappa(M) = \frac{\lambda_{\max}(M)}{\lambda_{\min}(M)}$ denotes the condition number of matrix M.

Proof. As in Balcan and Sharma [10], note that any boundary $\sigma_{\min}$ or $\sigma_{\max}$ must have some $f_u(\sigma) = \frac{1}{2}$. Algorithm 7 finds these boundary pieces by finding roots/zeros of $\left(f_u(\sigma) - \frac{1}{2}\right)^2$. As noted in Theorems 2.6.8 and 2.6.9, both Nesterov's and Newton's descent methods have quadratic convergence, so at every update step in Algorithm 7 (lines 12 and 15), we converge quadratically, leading to $\log\log(\frac{1}{\epsilon})$ update steps needed to satisfy $|\sigma^*_\epsilon - \sigma^*| < \epsilon$, where σ^* is the root with $g_u(\sigma^*) = 0$.

In Theorems 2.6.6 and 2.6.7, we assumed that $|\frac{\partial f}{\partial \sigma}| < \frac{1}{\epsilon\lambda_{\min}(G)}$ for some graph G. Consider this is not the case. We examine the Newton update, which is an upper bound on the size of the

Algorithm 7 APPROXFEEDBACKSET$(G, f_L, \sigma_0, \epsilon, \eta, \mathcal{A})$

Input: Graph G with unlabeled nodes U, labels f_L, query parameter σ_0, error tolerance ϵ, learning rate η, algorithm $\mathcal{A}$ to estimate soft labels and derivatives at any σ (e.g. Algorithm 6).
Output: Estimates for piecewise constant interval containing σ_0, and function value at σ.
Initialize $\sigma_l = \sigma_h = \sigma_0$.
for all $u \in U$ **do**
  Initialize $n = 0, \lambda_0 = 1, y_0 = \sigma_0$.
  while $|\sigma_{n+1} - \sigma_n| \geq \epsilon$ **do**
    Compute $f_{u,\epsilon}(\sigma), \frac{\partial f_u}{\partial \sigma}_\epsilon$ as $\mathcal{A}(G, f_L, u, \sigma_n, \epsilon)$
    Set $g_u(\sigma_n) = (f_{u,\epsilon}(\sigma_n) - \frac{1}{2})^2$, $g'_u(\sigma_n) = 2\left(f_{u,\epsilon}(\sigma_n) - \frac{1}{2}\right)\left(\frac{\partial f_u}{\partial \sigma}_\epsilon\right)$.
    $\xi_{\text{GD}} \leftarrow \eta g'_u(\sigma_n)$; $\xi_{\text{Newton}} \leftarrow 2\frac{g_u(\sigma_n)}{g'_u(\sigma_n)}$.
    $y_{n+1} = \sigma_n - \min\{\xi_{\text{GD}}, \xi_{\text{Newton}}\}$.
    if $\xi_{\text{GD}} < \xi_{\text{Newton}}$ **then**
      $\lambda_{n+1} = \frac{1+\sqrt{1+4\lambda_n^2}}{2}, \gamma_n = \frac{1-\lambda_n}{\lambda_{n+1}}, \sigma_{n+1} = (1-\gamma_n)y_{n+1} + \gamma_n y_n$
    else
      $\sigma_{n+1} = y_{n+1}$
    $n \leftarrow n + 1$
  $\sigma_l = \min\{\sigma_l, \sigma_{n+1}\}, \sigma_h = \max\{\sigma_h, \sigma_{n+1}\}$.
return $[\sigma_l, \sigma_h]$, $f_\epsilon(\sigma_0)$.

update step used as our update uses the minimum of Newton and Nesterov steps:

$$\begin{aligned}
2\cdot\frac{g_u(\sigma)}{g'_u(\sigma)} &= 2 \cdot \frac{(f_u(\sigma) - 1/2)^2}{2 \cdot (\partial f/\partial\sigma)(f_u(\sigma) - 1/2)} \\
&= \frac{(f_u(\sigma) - 1/2)}{(\partial f/\partial\sigma)} \\
&< \epsilon\lambda_{\min}(G)(f_u(\sigma) - 1/2) && (\because |\partial f/\partial\sigma| > 1/\epsilon\lambda_{\min}(G)) \\
&< \epsilon && (\because f_u(\sigma) \leq 1/\lambda_{\min}, \text{ cf. Thms 2.6.6, 2.6.7}).
\end{aligned}$$

Thus in this case the update step is less than ϵ, and we will terminate after one subsequent step.

As noted in Theorem 2.6.6, we need $O\left(\sqrt{\kappa(\mathcal{L}_{UU})}\log\left(\frac{n}{\epsilon'\lambda_{\min}(\mathcal{L}_{UU})}\right)\right)$ CG steps to reach an ϵ' approximation of $f\frac{\partial f}{\partial\sigma}$. Theorem 2.6.8 states that we need $\epsilon' = O\left(\frac{\epsilon}{\Delta}\right)$ to find an ϵ approximation of the root σ^*, so this takes complexity

$$O\left(\sqrt{\kappa(\mathcal{L}_{UU})}\log\left(\frac{n\Delta}{\epsilon\lambda_{\min}(\mathcal{L}_{UU})}\right)\right).$$

Running a single iteration of the conjugate gradient method requires a constant number of matrix-vector products of form Ax, where A is the weighted adjacency matrix for graph G. This computation takes $O(|E_G|)$ time. Finally, we run this algorithm for each of the n points, leading to an overall time complexity of

$$O\left(|E_G| n\sqrt{\kappa(\mathcal{L}_{UU})}\log\left(\frac{n\Delta}{\epsilon\lambda_{\min}(\mathcal{L}_{UU})}\right)\log\log\frac{1}{\epsilon}\right).$$

If G is the complete graph, $|E_G| \in O(n^2)$. If G is a kNN graph for some fixed k, then $|E_G| = kn \in O(n)$. □

Above analysis can be adapted to obtain the following guarantee for tuning σ in the efficient algorithm of Delalleau et al. [66]. While the above result guarantees a running time of $\tilde{O}(n^2)$ for kNN graphs, learning the graph can be done even more efficiently for the scalable approach of Delalleau et al. [66]. Their algorithm minimizes a proxy for the harmonic objective given by

$$\frac{1}{2}\sum_{u,v\in\tilde{U}} w(u,v)(f(u)-f(v))^2 + \lambda\sum_{w\in L}(f(w)-y_w)^2,$$

where $\tilde{U} \subset U$ and λ are hyperparameters. In particular, one chooses a small set $\tilde{U}$ with $|\tilde{U}| \ll n$ and efficiently extrapolates the harmonic labels on $\tilde{U}$ to the rest of U using a Parzen windows based extrapolation. As before, the success of this more efficient approach also depends on the choice of the graph G used. Our Algorithm 7 obtains good theoretical guarantees in this case as well, with appropriate choice of algorithm $\mathcal{A}$ (namely DBLR05APPROX).

Theorem 2.6.4. *(Informal) Given an algorithm for computing ϵ-approximate soft labels and gradients for the efficient semi-supervised learning algorithm of Delalleau et al. [66] (*DBLR05APPROX*), Algorithm 7 computes (ϵ,ϵ)-approximate semi-bandit feedback for the semi-supervised loss $l(\sigma)$ in time*

$$O\left(|E_{G_{\tilde{U}}}| n\sqrt{\kappa(\mathcal{L}_A)}\log\left(\frac{\lambda(|L_{Labels}| + |\tilde{U}_{Labels}|)\Delta}{\epsilon\sigma_{\min}\lambda_{\min}(\mathcal{L}_A)}\right)\log\log\frac{1}{\epsilon}\right),$$

where all values are defined as in DBLR05APPROX.

We empirically observe that sparsity (using only kNN edges, and nodes in $\tilde{U}$) results in well-conditioned matrices (bounded $\sqrt{\kappa(A)}$) in the considered parameter range $[\sigma_{\min}, \sigma_{\max}]$.

2.6.3 Approximate Soft Label and Gradient

The piecewise constant interval computation in Algorithm 7 needs computation of soft labels $f(\sigma)$ as well as gradients $\frac{\partial f}{\partial \sigma}$ for all unlabeled nodes. Typically, one computes a matrix inverse to exactly compute these quantities, and the exact matrix inverted is different for different approaches. In this section, we provide approximate but more efficient procedures for computing these quantities for computing soft labels using the Harmonic objective approach of [174], as well as for the scalable approach of [66]. We also provide convergence guarantees for our algorithms, in terms of the number of conjugate gradient iterations needed for obtaining an ϵ-approximation to the above quantities. Note that replacing $\text{CG}(A, b, t)$ by the computation $A^{-1}b$ recovers the algorithm from [10], which is more precise but takes longer ($O(n^3)$ time or $O(n^\omega)$, where ω is the matrix multiplication exponent, for the matrix inversion step).

Approximate Efficient Soft-labeling of [174]

We provide an approximation guarantee for Algorithm 6. We first need a simple lemma to upper bound matrix vector products for positive definite matrices.

Lemma 2.6.5. *Suppose matrix $A \in \mathbb{R}^{n\times n}$ is positive definite, with $x \in \mathbb{R}^n$. Then $\|Ax\|_2 \leq \lambda_{max}\|x\|_2$ where λ_{max} is the maximum eigenvalue of A*

Proof. The idea is to normalize vector x, then consider SVD of A. Since the vectors are orthonormal, we will be able to simplify to a form that can be upper bounded by $\lambda_{\max}\|x\|_2$. Letting $\hat{x} = \frac{x}{\|x\|}$ and $\{\phi_i\}_{i\in[n]}$ be an orthonormal basis for A, we can write $\hat{x}$ as a linear combination of $\{\phi_i\}$:

$$\hat{x} = \sum_{i\in[n]} \alpha_i \phi_i.$$

Now,

$$\begin{aligned}
\|A\hat{x}\|_2^2 &= \left\| A \sum_{i\in[n]} \alpha_i \phi_i \right\|_2^2 \\
&= \left\| \sum_{i\in[n]} \alpha_i \lambda_i \phi_i \right\|_2^2 \\
&= \sum_{i\in[n]} \alpha_i^2 \lambda_i^2 && (\phi_i \text{ orthonormal}) \\
&\leq \lambda_{\max}^2 && (\lambda_i \leq \lambda_{\max} \forall i;\ \hat{x} \text{ is a unit vector}).
\end{aligned}$$

Thus, $\|Ax\| \leq \lambda_{\max}\|x\|$ using $\hat{x} = \frac{x}{\|x\|}$. □

Equipped with this lemma, we are ready to prove our approximation guarantee for Algorithm 6.

Theorem 2.6.6. *Suppose the function $f : \mathbb{R} \to \mathbb{R}$ is convex and differentiable, and that its gradient is Lipschitz continuous with constant $L > 0$, i.e. we have that $|f'(x) - f'(y)| \leq L|x - y|$ for any x, y. Then for some $\sigma \in [\sigma_{\min}, \sigma_{\max}]$, where $\left|\frac{\partial f}{\partial \sigma}\right| < \frac{1}{\epsilon\lambda_{\min}(I-P_{UU})}$ on $[\sigma_{\min}, \sigma_{\max}]$, $\kappa(A)$ is condition number of matrix A and $\lambda_{\min}(A)$ is the minimum eigenvalue of A, we can find an ϵ approximation of $f_u(\sigma)\frac{\partial f_u}{\partial \sigma}$ achieving $\left|f_u(\sigma)\frac{\partial f_u}{\partial \sigma} - \left(f_u(\sigma)\frac{\partial f_u}{\partial \sigma}\right)_\epsilon\right| < \epsilon$, where $f_u(\sigma)$, $\frac{\partial f_u}{\partial \sigma}$ are as described in Algorithm 6 using $O\left(\sqrt{\kappa(I - P_{UU})}\log\left(\frac{n}{\epsilon\lambda_{\min}(I-P_{UU})}\right)\right)$ conjugate gradient iterations.*

Proof. A *grounded* Laplacian (aka Dirichlet Laplacian) matrix is obtained by "grounding", i.e. removing rows and columns corresponding to, some subset of graph nodes from the Laplacian matrix $L = D - W$. It is well known that the grounded Laplacian matrix is positive definite [120, 162]. In particular, $\mathcal{L}_{UU} = D_{UU} - W_{UU}$ and therefore $I - P_{UU} = D_{UU}^{-1/2}\mathcal{L}_{UU}D_{UU}^{-1/2}$ are positive definite. This implies $(I - P_{UU})^{-1}$ is also positive definite with maximum eigenvalue $\frac{1}{\lambda_{\min}}$, where $\lambda_{\min}$ is the minimum eigenvalue for $I - P_{UU}$. From here, note that all elements of $P_{UL}f_L$ are less than one as all labels are 0 or 1, and P is positive in all terms and row normalized to have rowsums of 1. Therefore,

$$\|f(\sigma)\| = \|(I - P_{UU})^{-1} P_{UL} f_L\| \leq \frac{1}{\lambda_{\min}} \|P_{UL} f_L\| \leq \frac{\sqrt{n}}{\lambda_{\min}}$$

where the first inequality holds via Lemma 2.6.5.

We now have that $\|f(\sigma)\|$ is bounded by $\frac{\sqrt{n}}{\lambda_{\min}(I-P_{UU})}$ on $[\sigma_{\min}, \sigma_{\max}]$. To find an ϵ approximation in the sense $\|f - f_\epsilon\| \leq \epsilon$, we set

$$\epsilon' = \epsilon \lambda_{\min}(I - P_{UU}) \leq \frac{\sqrt{n}\epsilon}{\max_{\sigma \in [\sigma_{\min}, \sigma_{\max}]} f(\sigma)}$$

and note

$$\|f - f_{\epsilon'}\| \leq \epsilon' \|f\| \leq \epsilon$$

We consider this process for $\frac{\partial f}{\partial \sigma}$ as well since $\frac{\partial f}{\partial \sigma}$ is bounded by $\frac{1}{\epsilon \lambda_{\min}(I-P_{UU})}$. Setting $\epsilon' = \epsilon^2 \lambda_{\min}(I - P_{UU})$,

$$\left| \frac{\partial f}{\partial \sigma} - \frac{\partial f}{\partial \sigma}_\epsilon \right| \leq \epsilon' \frac{\partial f}{\partial \sigma} \leq \epsilon$$

holds. Finally, letting

$$\epsilon' = \frac{\sqrt{n}\epsilon^2 \lambda_{\min}(I - P_{UU})}{3}$$

we achieve the desired result

$$\left| f \frac{\partial f}{\partial \sigma} - f_{\epsilon'} \frac{\partial f}{\partial \sigma}_{\epsilon'} \right| < \epsilon' f + \epsilon' \frac{\partial f}{\partial \sigma} + \epsilon'^2 < \epsilon.$$

Next we analyze the number of iterations of the CG method used. By [6], finding ϵ' approximations using the CG method on positive definite matrix G be done in

$$O(\sqrt{\kappa(G)}) \log \frac{1}{\epsilon'}$$

iterations. Here we need an $\epsilon' = \frac{\sqrt{n}\epsilon^2 \lambda_{\min}(I-P_{UU})}{3}$ approximation for matrix $I - P_{UU}$, so this takes

$$O\left(\sqrt{\kappa(I - P_{UU})} \log \left(\frac{n}{\epsilon \lambda_{\min}(I - P_{UU})} \right) \right)$$

iterations of the CG method. □

Approximate Efficient Soft-labeling of [66]

Algorithm 8 computes the soft label corresponding to the efficient algorithm of [66] and gradient for a given value of graph parameter σ for a fixed unlabeled node i, by running the conjugate gradient for given number of iterations.
We again provide an approximation guarantee for the algorithm.

Algorithm 8 NONPARAMETRICAPPROXIMATION$(G, f_L, i, \sigma, \epsilon)[\tilde{U}, \lambda]$

1: **Input:** Graph G with labeled nodes f_L and set of unlabeled nodes U, unlabeled node $i \in U$, query parameter σ, error tolerance ϵ.
2: **Hyperparameters:** Small subset $\tilde{U} \subset U$ (e.g. chosen by the greedy approach of [66], or via [166]), labeled loss regularization coefficient λ (see [66]).
3: **Output:** approximate soft label $\tilde{f}_{i,\epsilon}$ and approximate gradient $\frac{\partial \tilde{f}_i}{\partial \sigma}_\epsilon$.
4: Let $\text{CG}(A, b, t)$ represent running the conjugate gradient method for t iterations to solve equation $Ax = b$.
5: Let t_ϵ be the number of iterations sufficient for ϵ-approximation (Thm 2.6.7).
6: Let $\tilde{f}_{i,\epsilon}(\sigma) = \frac{\sum_{j \in \tilde{U} \cup L} W_{ij}(\sigma) f_j(\sigma)_\epsilon}{\sum_{j \in \tilde{U} \cup L} W_{ij}(\sigma)}$, where

$$\begin{aligned} f(\sigma)_\epsilon &= \text{CG}(A, \lambda \overrightarrow{y}, t_\epsilon), \\ A &= \lambda \Delta_L + Diag(W \mathbf{1}_n) - W, \\ (\Delta_L)_{ij} &= \delta_{ij} \delta_{i \in L}, \\ \overrightarrow{y} &= (y_1, ..., y_l, 0, ..., 0)^\top. \end{aligned}$$

7: Let $\frac{\partial \tilde{f}_i}{\partial \sigma}_\epsilon = \frac{\sum_{j \in \tilde{U} \cup L} \frac{\partial W_{ij}}{\partial \sigma} f_j(\sigma) + \sum_{j \in \tilde{U} \cup L} W_{ij}(\sigma) \frac{\partial f_j}{\partial \sigma}_\epsilon + \tilde{f}_{i,\epsilon}(\sigma) \sum_{j \in \tilde{U} \cup L} \frac{\partial W_{ij}}{\partial \sigma}}{\sum_{j \in \tilde{U} \cup L} W_{ij}}$, where

$$\begin{aligned} \frac{\partial f}{\partial \sigma}_\epsilon &= -\text{CG}(A, \frac{\partial A}{\partial \sigma} f, t_\epsilon), \\ \frac{\partial A}{\partial \sigma} &= \text{Diag}\left(\frac{\partial W}{\partial \sigma} \mathbf{1_n}\right) - \frac{\partial W}{\partial \sigma}, \\ \frac{\partial W_{ij}}{\partial \sigma} &= \frac{2 W_{ij} d_{ij}^2}{\sigma^3}. \end{aligned}$$

8: **return** $\tilde{f}_{i,\epsilon}(\sigma), \frac{\partial \tilde{f}_i}{\partial \sigma}_\epsilon$.

Theorem 2.6.7. *Suppose the function $f : \mathbb{R} \to \mathbb{R}$ is convex and differentiable, and that its gradient is Lipschitz continuous with constant $L > 0$, i.e. we have that $|f'(x) - f'(y)| \leq L|x - y|$ for any x, y. Then for some $\sigma \in [\sigma_{\min}, \sigma_{\max}]$, where $\left|\frac{\partial f}{\partial \sigma}\right| \in O\left(\frac{1}{\epsilon \lambda_{\min}(A)}\right)$ on $[\sigma_{\min}, \sigma_{\max}]$, $\kappa(A)$ is condition number of matrix A and $\lambda_{\min}(A)$ is the minimum eigenvalue of A, we can find an ϵ approximation of $\tilde{f}_u(\sigma) \cdot \frac{\partial \tilde{f}_u}{\partial \sigma}$ achieving $\left|\tilde{f}_u(\sigma) \frac{\partial \tilde{f}_u}{\partial \sigma} - \left(\tilde{f}_u(\sigma) \frac{\partial \tilde{f}_u}{\partial \sigma}\right)_\epsilon\right| < \epsilon$, where $\tilde{f}_u(\sigma), \frac{\partial \tilde{f}_u}{\partial \sigma}$ are as described in Algorithm 8 using*
$O\left(\sqrt{\kappa(A)} \log\left(\frac{\lambda(|L_{Labels}| + |\tilde{U}_{Labels}|)}{\epsilon \sigma_{\min} \lambda_{\min}(A)}\right)\right)$ conjugate gradient iterations. Here L_{Labels} and $\tilde{U}_{Labels}$ are sets of labels as described in Algorithm 8, and λ is the parameter passed into Algorithm 8.

Proof. As noted in the proof of 2.6.6, the grounded Laplacian A is positive definite. We can now

bound A^{-1} as in Theorem 2.6.6 and note that

$$A^{-1}\lambda\overrightarrow{y} \leq \frac{\lambda\sqrt{|L_{\text{Labels}}|}}{\lambda_{\min}(A)}$$

via Lemma 2.6.5 as the vector $\overrightarrow{y}$ contains at most L_{labels} elements with value 1. Note that λ is the constant passed in to Algorithm 8, and $\lambda_{\min}(A)$ is the smallest eigenvalue of A.

Next, we argue that we can find ϵ approximations of f, $\frac{\partial f}{\partial \sigma}$ with $\epsilon' = \frac{\sqrt{|L_{\text{Labels}}|}\epsilon^2\lambda_{\min}(A)}{\lambda}$ similarly to Theorem 6 as well. From here we consider $\tilde{f}(\sigma)$ and note that

$$\left|\frac{\sum_{j\in\tilde{U}\cup L} W_{ij}(\sigma)f(\sigma)}{\sum_{j\in\tilde{U}\cup L} W_{ij}} - \frac{\sum_{j\in\tilde{U}\cup L} W_{ij}(\sigma)f(\sigma)_\epsilon}{\sum_{j\in\tilde{U}\cup L} W_{ij}}\right| < \left|\frac{\sum_{j\in\tilde{U}\cup L} W_{ij}}{\sum_{j\in\tilde{U}\cup L} W_{ij}}\epsilon\right| = \epsilon$$

Finally, we show that the result holds for $\frac{\partial \tilde{f}_i}{\partial \sigma}$, noting we have proven the result for both $\tilde{f}_{i,\epsilon}$ and $\frac{\partial f}{\partial \sigma}_\epsilon$, and noting that we have exact values for W_{ij} and $\frac{\partial W_{ij}}{\partial \sigma}$

$$\begin{aligned}
\left|\frac{\partial \tilde{f}_i}{\partial \sigma}_\epsilon - \frac{\partial \tilde{f}_i}{\partial \sigma}\right| &= \frac{\sum_{j\in\tilde{U}\cup L} W_{ij}(\sigma)\epsilon + \epsilon\sum_{j\in\tilde{U}\cup L}\frac{\partial W_{ij}}{\partial\sigma}}{\sum_{j\in\tilde{U}\cup L} W_{ij}(\sigma)} \\
&= \epsilon + \epsilon\frac{\sum_{j\in\tilde{U}\cup L}\frac{\partial W_{ij}}{\partial\sigma}}{\sum_{j\in\tilde{U}\cup L} W_{ij}(\sigma)} \\
&= \epsilon + \frac{2\epsilon}{\sigma^3}\frac{\sum_{j\in\tilde{U}\cup L} e^{-\frac{d_{ij}^2}{\sigma^2}} d_{ij}^2}{\sum_{j\in\tilde{U}\cup L} e^{-\frac{d_{ij}^2}{\sigma^2}}} \\
&\leq \epsilon + \frac{2\epsilon}{\sigma^3}\sum_{j\in\tilde{U}\cup L} e^{-\frac{d_{ij}^2}{2\sigma^2}} d_{ij} && \text{(Cauchy-Schwartz inequality)} \\
&\leq \epsilon + \frac{2\epsilon}{\sigma^3}\sum_{j\in\tilde{U}\cup L} \sigma e^{-\frac{1}{2}} && \text{(max of } f(x) = xe^{-\frac{x^2}{2c^2}} \text{ at } x = c\text{)} \\
&\leq \epsilon\left(1 + \frac{2(|L_{\text{Labels}}| + |\tilde{U}_{\text{Labels}}|)}{\sigma^2}\right) \\
&\leq \epsilon\left(1 + \frac{2(|L_{\text{Labels}}| + |\tilde{U}_{\text{Labels}}|)}{\sigma_{\min}^2}\right).
\end{aligned}$$

In a similar manner to Theorem 6, we need

$$\epsilon' = \frac{\sqrt{|L_{\text{Labels}}|}\epsilon^2\lambda_{\min}(A)}{\lambda}$$

to achieve ϵ approximations of $\frac{\partial f}{\partial \sigma}$ and $\tilde{f}$. Setting

$$\epsilon'' = \frac{\epsilon^2\sigma_{\min}^2\lambda_{\min}(A)}{(2(|L_{\text{Labels}}| + |\tilde{U}_{\text{Labels}}|) + \sigma_{\min}^2)\sqrt{|L_{\text{Labels}}|}\lambda}$$

we also achieve

$$\left| \frac{\partial \tilde{f}_i}{\partial \sigma}_{\epsilon'} - \frac{\partial \tilde{f}_i}{\partial \sigma} \right| < \epsilon.$$

As a result, we obtain the desired bound

$$\left| \tilde{f}_u(\sigma) \frac{\partial \tilde{f}_u}{\partial \sigma} - \left(\tilde{f}_u(\sigma) \frac{\partial \tilde{f}_u}{\partial \sigma} \right)_{\epsilon} \right| < \epsilon.$$

Since we have that A is positive definite, via [6], This can be achieved in

$$O\left(\sqrt{\kappa(A)} \log \frac{1}{\epsilon''} \right) = O\left(\sqrt{\kappa(A)} \log \left(\frac{\lambda(|L_{\text{Labels}}| + |\tilde{U}_{\text{Labels}}|)}{\epsilon \sigma_{\min} \lambda_{\min}(A)} \right) \right)$$

iterations of the CG method. □

2.6.4 Convergence of Nesterov's Gradient Descent and Newton's Method

In this section we provide useful lemmas that provide convergence analysis for Nesterov's gradient descent and Newton's method, when working with approximate gradients. First we provide a guarantee for Nesterov's method in Theorem 2.6.8, which uses the result of [64] to analyse our algorithm.

Theorem 2.6.8. *Suppose the function $f : \mathbb{R} \to \mathbb{R}$ is convex and differentiable, and that its gradient is Lipschitz continuous with constant $L > 0$, i.e. we have that $|f'(x) - f'(y)| \leq L|x - y|$ for any x, y. Then if we run Nesterov's method to minimize $g(\sigma) = (f(\sigma) - \frac{1}{2})^2$ on some range $[\sigma_{min}, \sigma_{max}]$ where $\left|\frac{\partial f}{\partial \sigma}\right| < \frac{1}{\epsilon \lambda_{\min}(G_A)}$ using $\frac{\partial g}{\partial \sigma}$ as defined in Algorithm 7 and finding soft labels and derivatives as defined by some algorithm A, we can achieve an ϵ approximation σ^*_ϵ of the optimal result σ^* satisfying $|\sigma^*_\epsilon - \sigma^*| < \epsilon$ in $O(\log \log \frac{1}{\epsilon})$ iterations of Nesterov's method. We use $O(CG_A(\frac{\epsilon}{42(\sigma_{\max} - \sigma_{\min})}) \log \log \frac{1}{\epsilon})$ conjugate gradient iterations overall, where $CG_A(\epsilon')$ is the number of conjugate gradient iterations used by algorithm A to achieve ϵ' approximations of $f, \frac{\partial f}{\partial \sigma}$ satisfying $|f_{u,\epsilon}(\sigma) \frac{\partial f}{\partial \sigma}_{\epsilon} - f_u(\sigma) \frac{\partial f}{\partial \sigma}| < \epsilon'$.*

Proof. First, note that

$$\begin{aligned}
\left| \frac{\partial g_u}{\partial \sigma} - \frac{\partial g_u}{\partial \sigma}_{\epsilon'} \right| &= \left| 2\left(f_u(\sigma) - \frac{1}{2} \right) \left(\frac{\partial f_u}{\partial \sigma} \right) - 2\left(f_u(\sigma)_{\epsilon'} - \frac{1}{2} \right) \left(\frac{\partial f_u}{\partial \sigma}_{\epsilon'} \right) \right| \\
&\leq 4\epsilon' f_u(\sigma) \frac{\partial f_u(\sigma)}{\partial \sigma} + 2(\epsilon')^2 f_u(\sigma) \frac{\partial f_u(\sigma)}{\partial \sigma} + \epsilon' \frac{\partial f_u(\sigma)}{\partial \sigma} \\
&\leq 7 \left| f_u(\sigma) \frac{\partial f_u}{\partial \sigma} - \left(f_u(\sigma) \frac{\partial f_u}{\partial \sigma} \right)_{\epsilon} \right|.
\end{aligned}$$

Letting

$$\epsilon' = \frac{\epsilon}{42(\sigma_{\max} - \sigma_{\min})}$$

we find ϵ' approximations of f and $\frac{\partial f_u}{\partial \sigma}$ in $\text{CG}_A(\epsilon')$ steps. We can then bound

$$\left| \left(\frac{\partial g}{\partial \sigma} \right)_{\epsilon'} - \left(\frac{\partial g}{\partial \sigma} \right) \right| \leq \frac{\epsilon}{6(\sigma_{\max} - \sigma_{\min})}.$$

On compact set $[\sigma_{\min}, \sigma_{\max}]$ with this bound, we then have that

$$\left| \left\langle \left(\frac{\partial g}{\partial \sigma} \right)_{\epsilon'} - \left(\frac{\partial g}{\partial \sigma} \right), y - z \right\rangle \right| \leq \frac{\epsilon}{6} \, \forall y, z \in [\sigma_{\min}, \sigma_{\max}].$$

With this, [64] shows that Nesterov's accelerated gradient descent using an approximate gradient will converge to within ϵ of the optimal $\sigma^* \in [\sigma_{\min}, \sigma_{\max}]$ in $O(\frac{1}{\sqrt{\epsilon}})$ complexity. This yields $O(\log \log \frac{1}{\epsilon})$ steps until convergence.

Next we analyze the number of iterations of the CG method used. We called algorithm A $O(\log \log \frac{1}{\epsilon})$ times, each time using $\text{CG}_A(\epsilon') = \text{CG}_A\left(\frac{\epsilon}{42(\sigma_{\max} - \sigma_{\min})}\right)$ iterations. This yields

$$O\left(\text{CG}_A\left(\frac{\epsilon}{42(\sigma_{\max} - \sigma_{\min})}\right) \log \log \frac{1}{\epsilon}\right)$$

overall iterations of the CG method to find σ^*. □

We also provide an analysis for convergence of Newton's method using approximate gradients in Theorem 2.6.9.

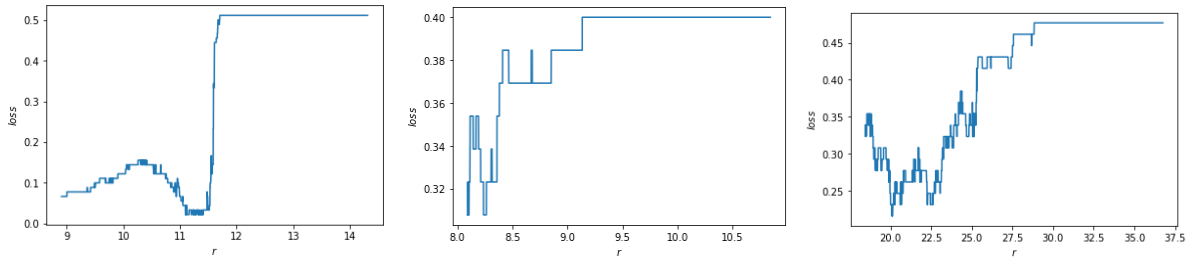
Theorem 2.6.9. *Suppose the function $f : \mathbb{R} \to \mathbb{R}$ has multiplicity 2 at optimal point x^*, with $f(x^*) = 0$. If Newton's accelerated method $x_{n+1} = x_n - 2\frac{f(x_n)}{f'(x_n)}$ coverges quadratically, then so does an epsilon approximation $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)_\epsilon}$ satisfying $|f'(x)_\epsilon - f'(x)| \leq \epsilon |f(x)| \forall x \in \mathbb{R}$*

Proof. First, quadratic convergence of accelerated Newton's method gives us $e_{n+1} \leq L e_n^2$ for some constant L, where $e_n = x^* - x_n$ is the error for the accelerated Newton's method update, and $x_{n+1} = x_n - 2\frac{f(x_n)}{f'(x_n)}$.

Using the Lagrange form of the Taylor series expansion, we see that

$$f(x_n) = f(x^*) + f'(x^*)(x^* - x_n) + (x^* - x_n) f''(\xi)$$

with ξ between x^k and x^*. Letting x^* be the optimal point with $f(x^*) = 0, f'(x^*) = 0$ by multiplicity 2, we see that $f(x_k) = (x^* - x_n) f''(\xi)$. Now to handle the ϵ-approximate case note



(a) MNIST (b) Omniglot (c) CIFAR-10

Figure 2.8: Loss for different unweighted graphs as a function of the threshold r.

that

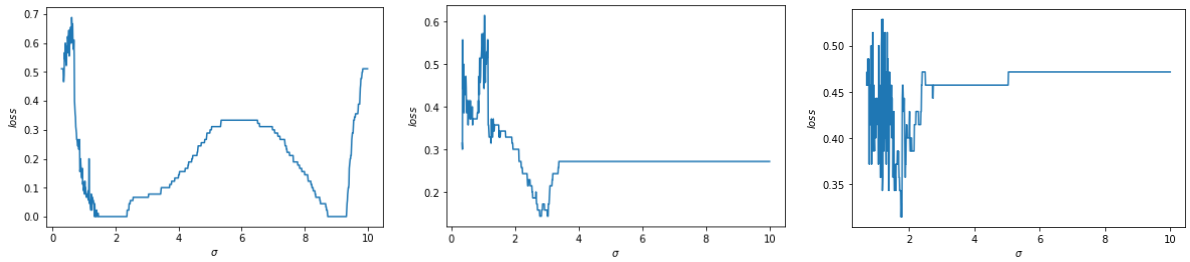
$$\begin{aligned} e_{n+1} &= x^* - x_n - 2\frac{f(x^k)}{f'(x_n)_\epsilon} \\ &\leq x^* - x_n - 2\frac{f(x^k)}{f'(x_n)(1+\epsilon)} \\ &= x^* - x_n - \frac{2f(x^k)}{f'(x_n)} + \frac{2\epsilon}{1+\epsilon}f(x_n) \\ &\leq Le_n^2 + \frac{2\epsilon}{1+\epsilon}f(x_n) \\ &\leq Le_n^2 + \frac{2\epsilon}{1+\epsilon}(x^* - x_n)^2 f''(\xi) \\ &\leq \left(L + \frac{2\epsilon}{1+\epsilon}f''(\xi)\right) e_n^2 \end{aligned}$$

as $x_n \to x^*$, we see that this is quadratic convergence if we are sufficiently close to x^* ($f''(\xi) < Cf''(x^*) \forall \xi \in [x_n, x^*]$). □

2.7 Experiments

In this section we evaluate the performance of our learning procedures when finding application-specific semi-supervised learning algorithms (i.e. graph parameters). Our experiments demonstrate that the best parameter for different applications varies greatly, and that the techniques presented in this paper can lead to large gains. We will look at image classification based on standard pixel embedding for different datasets.

Setup: We consider the task of semi-supervised classfication on image datasets. We restrict our attention to binary classification and pick two classes for each data set. We then draw random subsets of the dataset (with class restriction) of size $n = 100$ and randomly select L ($10 \leq L \leq 20$) examples for labeling. For any data subset S, we measure distance between any pairs of images using the L_2 distance between their pixel intensities. We would like to determine data-specific parameters r and σ which lead to good weighted and unweighted graphs for semi-



(a) MNIST (b) Omniglot (c) CIFAR-10

Figure 2.9: Loss for different weighted graphs as a function of the parameter σ.

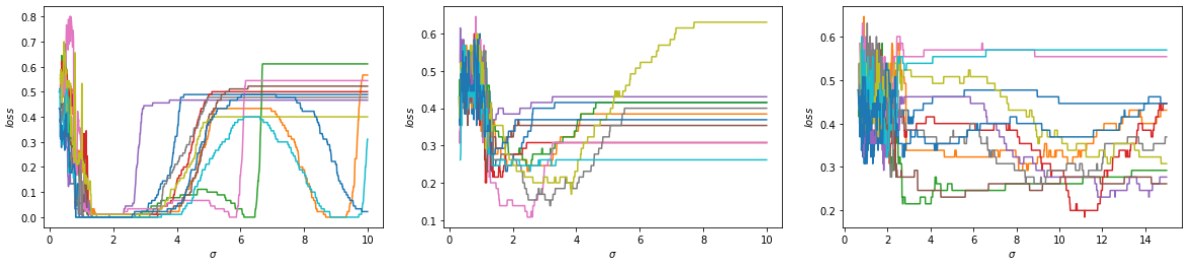
supervised learning on the datasets. We will optimize the harmonic function objective (Table 2.1) and round the fractional labels f to make our predictions.

Data sets: We use three popular benchmark datasets — MNIST, Omniglot and CIFAR-10. The MNIST dataset [107] contains images of hand-written digits from 0 to 9 as 28×28 binary images, with 5000 training examples for each class. We consider examples with labels 0 or 1. We generate a random semi-supervised learning instance from this data by sampling 100 random examples and further sampling $L = 10$ random examples from the subset for labeling. Omniglot [105] has 105×105 binary images of handwritten characters across 30 alphabets with 19,280 examples. We consider the task of distinguishing alphabets 0 and 1, and set $L = 20$ in this setting. CIFAR-10 [156] has 32×32 color images (an integer value in $[0, 255]$ for each of three colors) for object recognition among 10 classes. Again we consider objects 0 and 1 and set $L = 20$.

Results and discussion: For the MNIST dataset we get optimal parameters with near-perfect classification even with small values of L, while the error of the optimal parameter is $\sim 0.2 - 0.3$ even with larger values of L, indicating differences in the inherent difficulties of the classification tasks (like label noise and how well separated the classes are). We examine the full variation of performance of graph-based semi-supervised learning for all possible graphs $G(r)$ ($r_{\min} < r < r_{\max}$) and $G(\sigma)$ for $\sigma \in [0, 10]$ (Figures 2.8, 2.9). The losses are piecewise constant and can have large discontinuities in some cases. The optimal parameter values vary with the dataset, but we observe at least 10% gap in performance between optimal and suboptimal values within the same dataset.

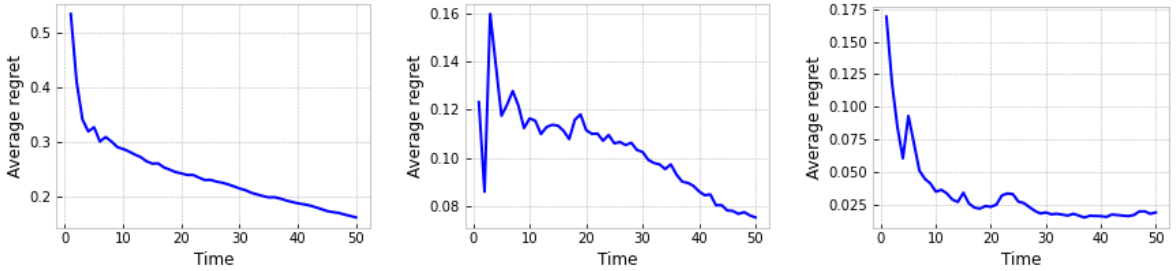
Another interesting observation is the variation of optima across subsets, indicating transductively optimal parameters may not generalize well. We plot the variation of loss with graph parameter σ for several subsets of the same size $N = 100$ for MNIST and Omniglot datasets in Figure 2.10. In MNIST we have two optimal ranges in most subsets but only one shared optimum (around $\sigma = 2$) across different subsets. This indicates that local search based techniques that estimate the optimal parameter values on a given data instance may lead to very poor performance on unseen instances. The CIFAR-10 example further shows that the optimal algorithm may not be easy to empirically discern.

We also implement our online algorithms and compute the average regret (i.e. excess error in predicting labels of unlabeled examples over the best parameter in hindsight) for finding the optimal graph parameter σ for the different datasets. To obtain smooth curves we plot the average



(a) MNIST (b) Omniglot (c) CIFAR-10

Figure 2.10: Comparing different subsets of the same problem.



(a) MNIST (b) Omniglot (c) CIFAR-10

Figure 2.11: Average regret vs. T for online learning of parameter σ

over 50 iterations for learning from 50 problem instances each ($T = 50$, Figure 2.11). We observe fast convergence to the optimal parameter regret for all the datasets considered. The starting part of these curves ($T = 0$) indicates regret for randomly setting the graph parameters, averaged over iterations, which is strongly outperformed by our learning algorithms as they learn from problem instances.

The results in this chapter are joint work with Nina Balcan [10] and Maxwell Jones [152]. Existing results have appeared in NeurIPS 2021, where the work was selected for an Oral presentation (top 1% of submissions), and in UAI 2023.

Chapter 3

Regularized regression

Ridge regression [95, 159], LASSO [157], and their generalization the ElasticNet [92] are among the most popular algorithms in machine learning and statistics, with applications to linear classification, regression, data analysis, and feature selection. Given a supervised dataset $(X, y) \in \mathbb{R}^{m\times p} \times \mathbb{R}^m$ with m datapoints and p features, these algorithms compute the linear predictor

$$\hat{\beta}^{(X,y)}_{\lambda_1,\lambda_2} = \text{argmin}_{\beta\in\mathbb{R}^p} \|y - X\beta\|_2^2 + \lambda_1\|\beta\|_1 + \lambda_2\|\beta\|_2^2 \tag{3.1}$$

Here $\lambda_1, \lambda_2 \geq 0$ are *regularization coefficients* constraining the ℓ_1 and ℓ_2 norms, respectively, of the model β. For general λ_1 and λ_2 the above algorithm is the ElasticNet, while setting $\lambda_1 = 0$ recovers Ridge and setting $\lambda_2 = 0$ recovers LASSO.

These coefficients play a crucial role across fields: in machine learning controlling the norm of β implies provable generalization guarantees and prevent over-fitting in practice [125], in data analysis their combined use yields parsimonious and interpretable models [92], and in Bayesian statistics they correspond to imposing specific priors on β [127]. In practice, λ_2 regularizes β by uniformly shrinking all coefficients, while λ_1 encourages the model vector to be sparse. This means that while they do yield learning-theoretic and statistical benefits, setting them to be too high will cause models to under-fit the data. The question of how to set the regularization coefficients becomes even more unclear in the case of the ElasticNet, as one must juggle tradeoffs between sparsity, feature correlation, and bias when setting both λ_1 and λ_2 simultaneously. As a result, there has been intense empirical and theoretical effort devoted to automatically tuning these parameters.

In this chapter, we study a variant on the above well-established and intensely studied formulation. The key distinction is that instead of a single dataset (X, y), we consider a collection of datasets or instances of the same underlying regression problem $(X^{(i)}, y^{(i)})$ and would like to learn a pair (λ_1, λ_2) that selects a model in equation (3.1) that has low loss on a validation dataset. This can be useful to model practical settings, for example where new supervised data is obtained several times or where the set of features may change frequently. We do not require all examples across datasets to be i.i.d. draws from the same data distribution, and can capture more general data generation scenarios like cross-validation and multi-task learning. Despite these advantages, we remark that our problem formulation is quite different from the standard single dataset setting. Our formulation treats the selection of regularization coefficients as *data-driven algorithm design*, which is often used to study combinatorial problems [7, 84].

Formal setup. Given data (X, y) with $X \in \mathbb{R}^{m\times p}$ and $y \in \mathbb{R}^m$, consisting of m labeled examples with p features, we seek estimators $\beta \in \mathbb{R}^p$ which minimize the regularized loss. Popular methods for regularized least-squares regression (including LASSO and ElasticNet) can be expressed as computing the solution of an optimization problem given by

$$\hat{\beta}^{(X,y)}_{\lambda,f} \in \text{argmin}_{\beta\in\mathbb{R}^p} \|y - X\beta\|_2^2 + \langle\lambda, f(\beta)\rangle$$

where $f : \mathbb{R}^p \to \mathbb{R}^d_{\geq 0}$ gives the regularization penalty for estimator β, $\lambda \in \mathbb{R}^d_{\geq 0}$ is the regularization parameter, and d is the number of regularization parameters. $d = 1$ for Ridge and LASSO, and $d = 2$ for the ElasticNet. Setting $f = f_2$ with $f_2(\beta) = \|\beta\|_2^2$ yields Ridge regression, and setting $f(\beta) = f_1(\beta) := \|\beta\|_1$ corresponds to LASSO. Also using $f_{\text{EN}}(\beta) := (f_1(\beta), f_2(\beta))$ gives the ElasticNet with regularization parameter $\lambda = (\lambda_1, \lambda_2)$. Note that we use the same λ (with some notational overloading) to denote the regularization parameters for ridge, LASSO, or ElasticNet. We write $\hat{\beta}^{(X,y)}_{\lambda,f}$ as simply $\hat{\beta}_{\lambda,f}$ when the dataset (X, y) is clear from context. On any instance $x \in \mathbb{R}^p$ from the feature space, the prediction of the regularized estimator is given by the dot product $\langle x, \hat{\beta}_{\lambda,f}\rangle$. The average squared loss over a dataset (X', y') with $X' \in \mathbb{R}^{m'\times p}$ and $y' \in \mathbb{R}^{m'}$ is given by $l_r(\hat{\beta}_{\lambda,f}, (X', y')) = \frac{1}{m'} \left\|y' - X'\hat{\beta}_{\lambda,f}\right\|_2^2$. By setting (X', y') to be the training data (X, y), we get the training loss $l_r(\hat{\beta}_{\lambda,f}, (X, y))$. We use $(X_{\text{val}}, y_{\text{val}})$ to denote a validation split.

Distributional and Online Settings. In the *distributional or statistical* setting, we receive a collection of n instances of the regression problem $P^{(i)} = (X^{(i)}, y^{(i)}, X^{(i)}_{\text{val}}, y^{(i)}_{\text{val}}) \in \mathcal{R}_{m_i,p_i,m'_i} := \mathbb{R}^{m_i\times p_i} \times \mathbb{R}^{m_i} \times \mathbb{R}^{m'_i\times p_i} \times \mathbb{R}^{m'_i}$ for $i \in [n]$ generated i.i.d. from some problem distribution $\mathcal{D}$. The problems are in the problem space given by $\Pi_{m,p} = \bigcup_{m_1\geq 0, m_2\leq m, p_1\leq p} \mathcal{R}_{m_1,p_1,m_2}$ (note that the problem distribution $\mathcal{D}$ is over $\Pi_{m,p}$). On any given instance $P^{(i)}$ the loss is given by the squared loss on the validation set, $\ell_{\text{EN}}(\lambda, P^{(i)}) = l_r(\hat{\beta}^{(X^{(i)},y^{(i)})}_{\lambda,f_{\text{EN}}}, (X^{(i)}_{\text{val}}, y^{(i)}_{\text{val}}))$. On the other hand, in the *online setting*, we receive a sequence of T instances of the ElasticNet regression problem $P^{(i)} = (X^{(i)}, y^{(i)}, X^{(i)}_{\text{val}}, y^{(i)}_{\text{val}}) \in \Pi_{m,p}$ for $i \in [T]$ online. On any given instance $P^{(i)}$, the online learner is required to select the regularization parameter $\lambda^{(i)}$ without observing $y^{(i)}_{\text{val}}$, and experiences loss given by $\ell(\lambda^{(i)}, P^{(i)}) = l_c(\hat{\beta}^{(X^{(i)},y^{(i)})}_{\lambda^{(i)},f_{EN}}, (X^{(i)}_{\text{val}}, y^{(i)}_{\text{val}}))$. The goal is to minimize the regret w.r.t. choosing the best fixed parameter in hindsight for the same problem sequence, i.e. $R_T = \sum_{i=1}^T \ell(\lambda^{(i)}, P^{(i)}) - \min_\lambda \sum_{i=1}^T \ell(\lambda, P^{(i)})$. We also define average regret as $\frac{1}{T}R_T$ and expected regret as $\mathbb{E}[R_T]$ where the expectation is over both the randomness of the loss functions and any random coins used by the online algorithm.

3.1 A structural result

Consider the class of algorithms consisting of ElasticNet regressors for different values of $\lambda = (\lambda_1, \lambda_2) \in (0, \infty)\times(0, \infty)$. We seek to solve problems of the form $P = (X, y, X_{\text{val}}, y_{\text{val}}) \in \Pi_{m,p}$, where (X, y) is the training set, $(X_{\text{val}}, y_{\text{val}})$ is the validation set with the same set of features, and m, p are upper bounds on the number of examples and features respectively in any dataset. Let $\mathcal{H}_{\text{EN}} = \{\ell_{\text{EN}}(\lambda, \cdot) \mid \lambda \in (0, \infty) \times (0, \infty)\}$ denote the set of loss functions for the class of algorithms consisting of ElasticNet regressors for different values of $\lambda \in \mathbb{R}^+ \times \mathbb{R}^+$. Additionally, we

will consider information criterion based loss functions, $\ell_{\text{EN}}^{\mathsf{AIC}}(\lambda, P) = \ell_{\text{EN}}(\lambda, P) + 2||\hat{\beta}_{\lambda,f_{\text{EN}}}^{(X,y)}||_0$ and $\ell_{\text{EN}}^{\mathsf{BIC}}(\lambda, P) = \ell_{\text{EN}}(\lambda, P) + 2||\hat{\beta}_{\lambda,f_{\text{EN}}}^{(X,y)}||_0 \log m$ [1, 149]. Let $\mathcal{H}_{\text{EN}}^{\mathsf{AIC}}$ and $\mathcal{H}_{\text{EN}}^{\mathsf{BIC}}$ denote the corresponding sets of loss functions. These criteria are popularly used to compute the squared loss on the training set, to give alternatives to cross-validation. We do not make any assumption on the relation between training and validation sets in our formulation, so our analysis can capture these settings as well. To state our structural result we will need the following definition.

Definition 14 (Piecewise structured functions, [23])**.** *A function class $H \subseteq \mathbb{R}^{\mathcal{X}}$ that maps a domain $\mathcal{X}$ to $\mathbb{R}$ is* (F, G, k)-piecewise decomposable *for a class $G \subseteq \{0,1\}^{\mathcal{X}}$ of boundary functions and a class $F \subseteq \mathbb{R}^{\mathcal{X}}$ of piece functions if the following holds: for every $h \in H$, there are k boundary functions $g_1, \ldots, g_k \in G$ and a piece function $f_{\mathbf{b}} \in F$ for each bit vector $\mathbf{b} \in \{0,1\}^k$ such that for all $x \in \mathcal{X}$, $h(x) = f_{\mathbf{b}_x}(x)$ where $\mathbf{b}_x = (g_1(x), \ldots, g_k(x)) \in \{0,1\}^k$.*

Intuitively, a real-valued function is piecewise-structured if the domain can be divided into pieces by a finite number of boundary functions (say linear or polynomial thresholds) and the function value over each piece is easy to characterize (e.g. constant, linear, polynomial). We start with a helper lemma that characterizes the solution of the ElasticNet in terms of equicorrelation sets and sign vectors (See Appendix A.2.1 for background details).

Lemma 3.1.1. *Let X be a matrix with columns in the general position, and $\lambda = (\lambda_1, \lambda_2) \in (0,\infty) \times [0,\infty)$. The ElasticNet solution $\hat{\beta}_{\lambda,f_{EN}} \in \operatorname{argmin}_{\beta\in\mathbb{R}^p} \|y - X\beta\|_2^2 + \langle \lambda, f_{EN}(\beta)\rangle$ is unique for any dataset (X, y) and satisfies*

$$\hat{\beta}_{\lambda,f_{EN}} = (X_{\mathcal{E}}^T X_{\mathcal{E}} + \lambda_2 I_{|\mathcal{E}|})^{-1} X_{\mathcal{E}}^T y - \lambda_1 (X_{\mathcal{E}}^T X_{\mathcal{E}} + \lambda_2 I_{|\mathcal{E}|})^{-1} s$$

for some $\mathcal{E} \in [p]$ and $s \in \{-1, 1\}^p$.

Proof. We start with the well-known characterization of the ElasticNet solution as the solution of a LASSO problem on a transformed dataset, obtained using simple algebra [177]. Given any dataset (X, y), the ElasticNet coefficients $\hat{\beta}_{\lambda,f_{EN}}$ are given by $\hat{\beta}_{\lambda,f_{EN}} = \frac{1}{\sqrt{1+\lambda_2}}\hat{\beta}_{\lambda}^{*1}$ where $\hat{\beta}_{\lambda}^{*}$ is the solution for a LASSO problem on a modified dataset (X^*, y^*)

$$\hat{\beta}_{\lambda}^{*} = \operatorname*{argmin}_{\beta} \|y^* - X^*\beta\|_2^2 + \lambda_1^* f_1(\beta)$$

with $X^* = \frac{1}{\sqrt{1+\lambda_2}} \begin{pmatrix} X \\ \sqrt{\lambda_2} I_p \end{pmatrix}$, $y^* = \begin{pmatrix} y \\ 0 \end{pmatrix}$, and $\lambda_1^* = \frac{\lambda_1}{\sqrt{1+\lambda_2}}$.

If the columns of X are in general position (Definition 37), then the same is true of X^*. For $\mathcal{E} \subseteq [p]$, note that $X_{\mathcal{E}}^{*T} X_{\mathcal{E}}^* = \frac{1}{1+\lambda_2}(X_{\mathcal{E}}^T X_{\mathcal{E}} + \lambda_2 I_{|\mathcal{E}|})$ and $X_{\mathcal{E}}^{*T} y^* = \frac{1}{\sqrt{1+\lambda_2}} X_{\mathcal{E}}^T y$. By Lemma A.2.2, if $\mathcal{E}$ denotes the equicorrelation set of covariates and $s \in \{-1,1\}^{|\mathcal{E}|}$ the equicorrelation sign vector for the LASSO problem, then the ElasticNet solution is given by

$$\hat{\beta}_{\lambda,f_{EN}} = c_1 - c_2\lambda_1$$

where $c_1 = \frac{1}{\sqrt{1+\lambda_2}}(X_{\mathcal{E}}^{*T} X_{\mathcal{E}}^*)^{-1} X_{\mathcal{E}}^{*T} y^* = (X_{\mathcal{E}}^T X_{\mathcal{E}} + \lambda_2 I_{|\mathcal{E}|})^{-1} X_{\mathcal{E}}^T y$, and
$c_2 = \frac{1}{1+\lambda_2}(X_{\mathcal{E}}^{*T} X_{\mathcal{E}}^*)^{-1} s = (X_{\mathcal{E}}^T X_{\mathcal{E}} + \lambda_2 I_{|\mathcal{E}|})^{-1} s$. □

[1]This corresponds to the "naive" ElasticNet solution in the terminology of [177]. They also define an ElasticNet 'estimate' given by $\sqrt{1+\lambda_2}\hat{\beta}_{\lambda}^{*}$ with nice properties, to which our analysis is easily adapted.

The following lemma helps determine the dependence of ElasticNet solutions on λ_2.

Lemma 3.1.2. *Let A be an $r \times s$ matrix. Consider the matrix $B(\lambda) = (A^T A + \lambda I_s)^{-1}$ and $\lambda > 0$.*

1. *Each entry of $B(\lambda)$ is a rational polynomial $P_{ij}(\lambda)/Q(\lambda)$ for $i, j \in [s]$ with each P_{ij} of degree at most $s-1$, and Q of degree s.*
2. *Further, for $i = j$, P_{ij} has degree $s-1$ and leading coefficient 1, and for $i \neq j$ P_{ij} has degree at most $s-2$. Also, $Q(\lambda)$ has leading coefficient 1.*

Proof. Let $G = A^T A$ be the Gramian matrix. G is symmetric and therefore diagonalizable, and the diagonalization gives the eigendecomposition $G = E\Lambda E^{-1}$. Thus we have

$$(A^T A + \lambda I_s)^{-1} = (E\Lambda E^{-1} + \lambda E E^{-1})^{-1} = E(\Lambda + \lambda I_s)^{-1} E^{-1}$$

But Λ is the diagonal matrix $\mathsf{Diag}(\Lambda_{11}, \ldots, \Lambda_{ss})$, and therefore $(\Lambda + \lambda I_s)^{-1} = \mathsf{Diag}((\Lambda_{11} + \lambda)^{-1}, \ldots, (\Lambda_{ss} + \lambda)^{-1})$. This implies the desired characterization, with $Q(\lambda) = \Pi_{i \in [s]}(\Lambda_{ii} + \lambda)$ and

$$P_{ij}(\lambda) = Q(\lambda) \sum_{k=1}^{s} \frac{E_{ik}(E^{-1})_{kj}}{\Lambda_{kk} + \lambda} = \sum_{k=1}^{s} \left(E_{ik}(E^{-1})_{kj} \Pi_{i \in [s] \setminus k}(\Lambda_{ii} + \lambda) \right),$$

with coefficient of λ^{s-1} in $P_{ij}(\lambda)$ equal to $\sum_{k=1}^{s} E_{ik}(E^{-1})_{kj} = \mathbb{I}\{i = j\}$. □

We will now formally state our key structural result which is needed to establish our generalization and online regret guarantees.

Theorem 3.1.3. *Let $\mathcal{L}$ be a set of functions $\{l_\lambda : \Pi_{m,p} \to \mathbb{R}_{\geq 0} \mid \lambda \in \mathbb{R}^+ \times \mathbb{R}_{\geq 0}\}$ that map a regression problem instance $P \in \Pi_{m,p}$ to the validation loss $\ell_{EN}(\lambda, P)$ of ElasticNet trained with regularization parameter $\lambda = (\lambda_1, \lambda_2)$. The dual class $\mathcal{L}^*$ is $(\mathcal{F}, \mathcal{G}, p3^p)$-piecewise decomposable, with $\mathcal{F} = \{f_q : \mathcal{L} \to \mathbb{R}\}$ consisting of rational polynomial functions $f_{q_1,q_2} : l_\lambda \mapsto \frac{q_1(\lambda_1,\lambda_2)}{q_2(\lambda_2)}$, where q_1, q_2 have degrees at most $2p$, and $\mathcal{G} = \{g_r : \mathcal{L} \to \{0,1\}\}$ consisting of semi-algebraic sets bounded by algebraic curves $g_r : u_\lambda \mapsto \mathbb{I}\{r(\lambda_1, \lambda_2) < 0\}$, where r is a polynomial of degree 1 in λ_1 and at most p in λ_2.*

Proof. Let $P = (X, y, X_{\text{val}}, y_{\text{val}}) \in \Pi_{m,p}$ be a regression problem instance. By using the standard reduction to LASSO [177] and well-known characterization of the LASSO solution in terms of equicorrelation sets, we can characterize the solution $\hat{\beta}_{\lambda, f_{EN}}$ of the Elastic Net as follows (Lemma 3.1.1):

$$\hat{\beta}_{\lambda, f_{EN}} = (X_{\mathcal{E}}^T X_{\mathcal{E}} + \lambda_2 I_{|\mathcal{E}|})^{-1} X_{\mathcal{E}}^T y - \lambda_1 (X_{\mathcal{E}}^T X_{\mathcal{E}} + \lambda_2 I_{|\mathcal{E}|})^{-1} s$$

for some $\mathcal{E} \in [p]$ and $s \in \{-1, 1\}^p$. Thus for any $\lambda = (\lambda_1, \lambda_2)$, the prediction $\hat{y}$ on any validation example with features $\mathbf{x} \in \mathbb{R}^p$ satisfies (for some $\mathcal{E}, s \in 2^{[p]} \times \{-1, 1\}^p$)

$$\hat{y}_j = \mathbf{x}\hat{\beta}_{\lambda, f_{EN}} = \mathbf{x}(X_{\mathcal{E}}^T X_{\mathcal{E}} + \lambda_2 I_{|\mathcal{E}|})^{-1} X_{\mathcal{E}}^T y - \lambda_1 \mathbf{x}(X_{\mathcal{E}}^T X_{\mathcal{E}} + \lambda_2 I_{|\mathcal{E}|})^{-1} s$$

For any subset $R \subseteq \mathbb{R}^2$, if the signed equicorrelation set $(\mathcal{E}, s)$ is fixed over R, then the above observation, together with Lemma 3.1.2 implies that the loss function $\ell_{\text{EN}}(\lambda, P)$ is a rational function of the form $\frac{q_1(\lambda_1,\lambda_2)}{q_2(\lambda_2)}$, where q_1 is a bivariate polynomial with degree at most $2|\mathcal{E}|$ and q_2 is univariate with degree $2|\mathcal{E}|$.

To show the piecewise structure, we need to demonstrate a set boundary functions $\mathcal{G} = \{g_1, \ldots, g_k\}$ such that for any sign pattern $\mathbf{b} \in \{0,1\}^k$, the signed equicorrelation set $(\mathcal{E}, s)$ for the region with sign pattern $\mathbf{b}$ is fixed. To this end, based on the observation above, we will consider the conditions (on λ) under which a covariate may enter or leave the equicorrelation set. We will show that this can happen only at one of a finite number of algebraic curves (with bounded degrees).

Condition for joining $\mathcal{E}$. Fix $\mathcal{E}, s$. Also fix $j \notin \mathcal{E}$. If covariate j enters the equicorrelation set, the KKT conditions (Lemma A.2.1) applied to the LASSO problem corresponding to the ElasticNet (Lemma 3.1.1) imply

$$(\mathbf{x}_j^*)^T(y^* - X_{\mathcal{E}}^*(c_1 - c_2\lambda_1^*)) = \pm\lambda_1^*,$$

where $c_1 = (X_{\mathcal{E}}^{*T}X_{\mathcal{E}}^*)^{-1}X_{\mathcal{E}}^{*T}y^*$, $c_2 = (X_{\mathcal{E}}^{*T}X_{\mathcal{E}}^*)^{-1}s$, $X^* = \frac{1}{\sqrt{1+\lambda_2}}\begin{pmatrix} X \\ \sqrt{\lambda_2}I_p \end{pmatrix}$, $y^* = \begin{pmatrix} y \\ 0 \end{pmatrix}$, and $\lambda_1^* = \frac{\lambda_1}{\sqrt{1+\lambda_2}}$. Rearranging, and simplifying, we get

$$\lambda_1^* = \frac{(\mathbf{x}_j^*)^T X_{\mathcal{E}}^*(X_{\mathcal{E}}^{*T}X_{\mathcal{E}}^*)^{-1}(X_{\mathcal{E}}^*)^T y^* - (\mathbf{x}_j^*)^T y^*}{(\mathbf{x}_j^*)^T X_{\mathcal{E}}^*(X_{\mathcal{E}}^{*T}X_{\mathcal{E}}^*)^{-1}s \pm 1}, \text{or}$$
$$\lambda_1 = \frac{\mathbf{x}_j^T X_{\mathcal{E}}(X_{\mathcal{E}}{}^T X_{\mathcal{E}} + \lambda_2 I_{|\mathcal{E}|})^{-1}X_{\mathcal{E}}{}^T y - \mathbf{x}_j^T y}{\mathbf{x}_j^T X_{\mathcal{E}}(X_{\mathcal{E}}{}^T X_{\mathcal{E}} + \lambda_2 I_{|\mathcal{E}|})^{-1}s \pm 1}.$$

Note that the terms $(\mathbf{x}_j^*)^T X_{\mathcal{E}}^* = \mathbf{x}_j^T X_{\mathcal{E}}$, $(X_{\mathcal{E}}^*)^T y^* = X_{\mathcal{E}}^T y$, and $(\mathbf{x}_j^*)^T y^* = \mathbf{x}_j^T y$ do not depend on λ_1 or λ_2 (the λ_2 terms are zeroed out since $j \notin \mathcal{E}$). Moreover, $(X_{\mathcal{E}}^{*T}X_{\mathcal{E}}^*)^{-1} = (X_{\mathcal{E}}{}^T X_{\mathcal{E}} + \lambda_2 I_{|\mathcal{E}|})^{-1}$. Using Lemma 3.1.2, we get an algebraic curve $r_{j,\mathcal{E},s}(\lambda_1, \lambda_2) = 0$ with degree 1 in λ_1 and $|\mathcal{E}|$ in λ_2 corresponding to addition of $j \notin \mathcal{E}$ given $\mathcal{E}, s$.

Condition for leaving $\mathcal{E}$. Now consider a fixed $j' \in \mathcal{E}$, given fixed $\mathcal{E}, s$. The coefficient of j' will be zero for $\lambda_1^* = \frac{(c_1)_{j'}}{(c_2)_{j'}}$, which simplifies to $\lambda_1((X_{\mathcal{E}}{}^T X_{\mathcal{E}} + \lambda_2 I_{|\mathcal{E}|})^{-1}s)_{j'} = ((X_{\mathcal{E}}{}^T X_{\mathcal{E}} + \lambda_2 I_{|\mathcal{E}|})^{-1}X_{\mathcal{E}}{}^T y)_{j'}$. Again by Lemma 3.1.2, we get an algebraic curve $r_{j',\mathcal{E},s}(\lambda_1, \lambda_2) = 0$ with degree 1 in λ_1 and at most $|\mathcal{E}|$ in λ_2 corresponding to removal of $j' \in \mathcal{E}$ given $\mathcal{E}, s$.

Putting the two together, we get $\sum_{i=0}^{p} 2^i \binom{p}{i} ((p-i)+i) = p3^p$ algebraic curves of degree 1 in λ_1 and at most p in λ_2, across which the signed equicorrelation set may change. These curves characterize the complete set of points (λ_1, λ_2) at which $(\mathcal{E}, s)$ may possibly change. Thus by setting these $p3^p$ curves as the set of boundary functions $\mathcal{G}, \mathcal{E}, s$ is guaranteed to be fixed for each sign pattern, and the corresponding loss takes the rational function form shown above. □

The exact same piecewise structure can be established for the loss function classes $\ell_{\text{EN}}^{\mathsf{AIC}}(\lambda, \cdot)$ and $\ell_{\text{EN}}^{\mathsf{BIC}}(\lambda, \cdot)$. Given this piecewise structure, a challenge to learning values of λ that minimize the loss function is that the function may not be differentiable (or may even be discontinuous, for the information criteria based losses) at the piece boundaries, making well-known gradient-based (local) optimization techniques inapplicable here. In the following, we will show that techniques from data-driven design may be used to overcome this optimization challenge.

3.1.1 A more refined structure

While the above struture is sufficient to establish a $O(p^2)$ bound on the pseudo-dimension using results from [23], we establish a more refined structure below which can be used to obtain a sharper $O(p)$ bound.

Definition 15. *A function class $\mathcal{H} \subseteq \mathbb{R}^{\mathcal{Y}}$ that maps a domain $\mathcal{Y}$ to $\mathbb{R}$ is $(\mathcal{F}, k_{\mathcal{F}}, \mathcal{G}, k_{\mathcal{G}})-$piecewise decomposable for a class $\mathcal{G}$ of boundary functions and a class $\mathcal{F} \in \mathbb{R}^{\mathcal{Y}}$ of piece functions if the following holds: for every $h \in \mathcal{H}$, (1) there are $k_{\mathcal{G}}$ functions $g^{(1)}, \ldots, g^{(k_{\mathcal{G}})} \in \mathcal{G}$ and a function $f_{\boldsymbol{b}} \in \mathcal{F}$ for each bit vector $\boldsymbol{b} \in \{0,1\}^{k_{\mathcal{G}}}$ s.t. for all $y \in \mathcal{Y}$, $h(y) = h_{\boldsymbol{b}_y}(y)$ where $\boldsymbol{b}_y = \{(g^{(1)}(y), \ldots, g^{(k_{\mathcal{G}})}(y))\} \in \{0,1\}^{k_{\mathcal{G}}}$, and (2) there is at most $k_{\mathcal{F}}$ different functions in $\mathcal{F}$.*

A key distinction from [23] is the finite bound $k_{\mathcal{F}}$ on the number of different piece functions needed to define any function in the class $\mathcal{H}$. Under this definition we give the following more refined structure for the ElasticNet loss function class by extending arguments from [28].

Theorem 3.1.4. *Let $\mathcal{H}_{EN} = \{h_{EN}(\lambda, \cdot) : \Pi_{m,p} \to \mathbb{R}_{\geq 0} \mid \lambda \in \mathbb{R}^2_{>0}\}$ the class of Elastic Net validation loss function class. Consider the dual class $\mathcal{H}^*_{EN} = \{h^*_P : \mathcal{H}_{EN} \to \mathbb{R}_{\geq 0} \mid P \in \Pi_{m,p}\}$, where $h^*_P(h_{EN}(\lambda, \cdot)) = h_{EN}(\lambda, P)$. Then $\mathcal{H}^*_{EN}$ is $(\mathcal{F}, 3^p, \mathcal{G}, p3^p)$-piecewise decomposable, where the piece function class $\mathcal{F} = \{f_q : \mathcal{H}_{EN} \to \mathbb{R}\}$ consists at most 3^p rational function $f_{q_1,q_2} : h_{EN}(\lambda, \cdot) \mapsto \frac{q_1(\lambda_1,\lambda_2)}{q_2(\lambda_1,\lambda_2)}$ of degree at most $2p$, and the boundary function class $\mathcal{G} = \{g_r : \mathcal{H}_{EN} \to \{0,1\}\}$ consists of semi-algebraic sets bounded by at most $p3^p$ algebraic curves $g_r : h_{EN}(\lambda, \cdot) \mapsto \mathbb{I}\{r(\lambda_1, \lambda_2) < 0\}$, where r is a polynomial of degree at most p.*

Proof. Given a problem instance $P = (X, y, X_{\text{val}}, y_{\text{val}}) \in \Pi_{m,p}$, from Lemma 3.1.1, for each λ, the solution $\hat{\beta}(\lambda)$ of the Elastic Net can be characterized as follow

$$\hat{\beta}(\lambda) = (X_{\mathcal{E}}^\top X_{\mathcal{E}} + \lambda_2 I_{|\mathcal{E}|})^{-1} X_{\mathcal{E}}^\top y - \lambda_1 (X_{\mathcal{E}}^\top X_{\mathcal{E}} + \lambda_2 I_{|\mathcal{E}|})^{-1} s,$$

for some $\mathcal{E} \in [p]$ and $s \in \{\pm 1\}^p$. Therefore, the prediction $\hat{y}$ on any validation example with features $\mathbf{x} \in \mathbb{R}^p$ is

$$\hat{y} = \mathbf{x}\hat{\beta}(\lambda) = \mathbf{x}[(X_{\mathcal{E}}^\top X_{\mathcal{E}} + \lambda_2 I_{|\mathcal{E}|})^{-1} X_{\mathcal{E}}^\top y - \lambda_1 (X_{\mathcal{E}}^\top X_{\mathcal{E}} + \lambda_2 I_{|\mathcal{E}|})^{-1} s].$$

This implies that: for any region $R \subset \mathbb{R}^2_{>0}$, if the equicorrelation set and sign vector $(\mathcal{E}, s)$ is fixed over R, then the solution $\hat{\beta}(\lambda)$ and the prediction y corresponding to $\mathbf{x}$ is also fixed. Consequently, within any region R where $(\mathcal{E}, s)$ remains unchanged, Lemma 3.1.2 establishes that the validation loss function $h_{\text{EN}}(\lambda, P)$ (associated with a given problem instance P) is a constant rational function of the form $\frac{q_1(\lambda_1,\lambda_2)}{q_2(\lambda_1,\lambda_2)}$, where q_1 and q_2 are polynomials of degree at most $2p$ (since $2\,|\mathcal{E}| \leq 2p$ by definition). Notably, there are at most 3^p distinct values of $(\mathcal{E}, s)$, which implies that $h_{\text{EN}}(\lambda, P)$ can take on at most 3^p different polynomial forms.

The only remaining task is to examine the semi-algebraic sets and algebraic curves that separates region R. Consider such region R, in which the equicorrelation set and sign $(\mathcal{E}, s)$ is fixed.

- *Condition for a feature enters $\mathcal{E}$*: consider a feature $j \notin \mathcal{E}$, the condition for j to enters $\mathcal{E}$ is

$$(\mathbf{x}_j^*)^\top (y^* - X_{\mathcal{E}}^*(c_1 - c_2\lambda_1^*)) = \pm\lambda_1^*$$

where $c_1 = (X_{\mathcal{E}}^{*\top} X_{\mathcal{E}}^*)^{-1}$, $c_2 = (X_{\mathcal{E}}^{*\top} X_{\mathcal{E}}^*)^{-1} s$, $X^* = \frac{1}{\sqrt{1+\lambda_2}} \begin{bmatrix} X \\ \sqrt{\lambda_2} I_p \end{bmatrix}$, $y^* = \begin{bmatrix} y \\ 0 \\ \vdots \\ 0 \end{bmatrix}$. Simplifying the equation above, we have

$$\lambda_1^* - \frac{(\mathbf{x}_j^*)^\top X_{\mathcal{E}}^* (X_{\mathcal{E}}^* X_{\mathcal{E}}^*)^{-1} (X_{\mathcal{E}}^*)^\top y^* - (x_j^*)^\top y^*}{(\mathbf{x}_j^*)^\top X_{\mathcal{E}}^* (X_{\mathcal{E}}^{*\top} X_{\mathcal{E}}^*)^{-1} s \pm 1} = 0, \text{ or}$$
$$\lambda_1 (\mathbf{x}_j^\top (X_{\mathcal{E}} X_{\mathcal{E}}^{*\top})^{-1} X_{\mathcal{E}} s \pm 1) - \mathbf{x}_j^\top X_{\mathcal{E}} (X_{\mathcal{E}}^\top X_{\mathcal{E}} + \lambda_2 I_{|\mathcal{E}|})^{-1} X_{\mathcal{E}}^\top y - \mathbf{x}_j^\top y = 0,$$

which is an algebraic curve with the RHS is a polynomial of degree at most p.

- *Condition for a feature leaves $\mathcal{E}$*: consider a feature $j' \in \mathcal{E}$. Similar to the previous case, the condition for j' to leave $\mathcal{E}$ can be described by an algebraic curve with the RHS as a polynomial of degree at most p.

Finally, notice that there are at most $\sum_{i=0}^{p} \binom{p}{i}((p-i)+i) = p3^p$ curves, across which the equicorrelation set and sign $(\mathcal{E}, s)$ might change, which concludes the proof. □

Using the GJ framework [35], one can show that if a function class $\mathcal{H}$ has its dual-class $\mathcal{H}^*$ is piece-wise decomposable (in the sense of Definition 15), and all the piece and boundary functions are rational functions with upper bounded degree, then Pdim$(\mathcal{H})$ is upper bounded.

Lemma 3.1.5. *Consider the function class $\mathcal{H} = \{h(a,\cdot) : \mathcal{X} \to \mathbb{R} \mid a \in \mathbb{R}^n\}$ be a function class parameterized by $a \in \mathbb{R}^W$. Consider the dual class $\mathcal{H}^* = \{h_x(\cdot) : \mathbb{R}^n \to \mathbb{R} \mid x \in \mathcal{X}\}$, where $h_x(a) = h(a,x)$. Assume that $\mathcal{H}^*$ is $(\mathcal{F}, k_{\mathcal{F}}, \mathcal{G}, k_{\mathcal{G}})$ piece-wise decomposable, and $\mathcal{F}$, $\mathcal{G}$ contains only rational functions in a of degree at most Δ. Then* Pdim$(\mathcal{F}) = O(n \log(\Delta(k_{\mathcal{F}} + k_{\mathcal{G}})))$.

Proof. Given an input $x \in \mathcal{X}$ and a threshold $t \in \mathbb{R}$, for any function $h(a,\cdot) \in \mathcal{H}$ corresponding to parameter a, consider the computation $\Gamma_{x,t} : \mathcal{H} \to \{0,1\}$, where

$$\Gamma_{x,t}(h(a,\cdot)) = \mathbb{I}\{h(a,x) - t \geq 0\}, \text{ for any } h(a,\cdot) \in \mathcal{H}.$$

Our goal now is to show that $\Gamma_{x,t}$ is a GJ algorithm.

From assumptions, we know that the dual class $\mathcal{H}^*$ is $(\mathcal{F}, k_{\mathcal{F}}, \mathcal{G}, k_{\mathcal{G}})$ piece-wise decomposable, where $\mathcal{F}, \mathcal{G}$ consists of rational function in a of degree at most Δ. This implies that for any $h(a,\cdot) \in \mathcal{H}$, the function $h_x(a) = h(a,x)$ is a rational function of a, of which the form is one of $k_{\mathcal{F}}$ rational functions in $\mathcal{F}$. Hence, to compute $\Gamma_{x,t}(h(a,\cdot))$, one needs to specify the closed-form of $h(a,\cdot)$, which is determined by binary-valued vector $b_a = \{g^{(1)}(a), \ldots, g^{(k_{\mathcal{G}})}(a)\}$, and can be calculated as conditional statements in the form $\mathbb{I}\{g^{(i)}(a) \geq 0\}$ for $i \in k_{\mathcal{G}}$. Therefore, we conclude that the computation of $\Gamma_{x,t}$ can be described by a GJ algorithm.

The predicate complexity of $\Gamma_{x,t}$ is the total number of functions in $\mathcal{F}$ and $\mathcal{G}$, which is equal to $k_{\mathcal{F}} + k_{\mathcal{G}}$. The degree of $\Gamma_{x,t}$ is the maximum degree of rational functions in $\mathcal{F}$ and $\mathcal{G}$, which is Δ from assumptions. Using [35], we conclude that Pdim$(\mathcal{F}) = O(n \log(\Delta(k_{\mathcal{F}} + k_{\mathcal{G}}))$. □

Theorem 3.1.6. *Let $\mathcal{H}_{EN} = \{h_{EN}(\lambda,\cdot) : \Pi \to \mathbb{R}_{\geq 0} \mid \lambda \in \mathbb{R}^2_{>0}\}$ be the Elastic Net validation loss function class that maps problem instance P to validation loss $\ell_{val}(\lambda, P)$. Then* Pdim$(\mathcal{H}_{EN})$ *is* $O(p)$.

Proof. Given a problem instance $P \in \Pi_{m,p}$ and a threshold $t \in \mathbb{R}$, for any validation loss function $h_{\text{EN}}(\lambda, \cdot) \in \mathcal{H}_{\text{EN}}$, consider the computation $\Gamma_{P,t} : \mathcal{H}_{\text{EN}} \to \{0, 1\}$, where

$$\Gamma_{P,t}(h(\lambda, \cdot)) = \mathbb{I}\{h(\lambda, P) - t \geq 0\}, \text{ for any } h(\lambda, \cdot) \in \mathcal{H}_{\text{EN}}.$$

From Theorem 3.1.4, for a given problem instance P, we know that the dual-class $\mathcal{H}^*_{\text{EN}}$ is $(\mathcal{F}, 3^p, \mathcal{G}, p3^p)-$piecewise decomposable, where $\mathcal{F}$ consists at most 3^p rational function of degree at most $2p$, and $\mathcal{G}$ consists of at most $p3^p$ algebraic curves of degree at most p. From Lemma 3.1.5, $\text{Pdim}(\mathcal{H}_{\text{EN}}) = O(2\log(2p(p+1)3^p) = O(p)$.

□

3.2 Learning to regularize the ElasticNet

We will consider the problem of learning provably good ElasticNet parameters for a given problem domain, from multiple datasets (problem instances) either available as a collection (Section 3.2.1), or arriving online (Section 3.2.3).

3.2.1 Distributional Setting

Our main result in this section is the following upper bound on the pseudo-dimension of the classes of loss functions for the ElasticNet, which implies that in our distributional setting it is possible to learn near-optimal values of λ with polynomially many problem instances.

Theorem 3.2.1. $\text{PDIM}(\mathcal{H}_{EN}) = O(p)$, $\text{PDIM}(\mathcal{H}^{AIC}_{EN}) = O(p)$ *and* $\text{PDIM}(\mathcal{H}^{BIC}_{EN}) = O(p)$.

In our setting of learning from multiple problem instances, each sample is a dataset instance, so the sample complexity is simply the number of regression problem instances needed to learn the tuning parameters to any given approximation and confidence level.

To guarantee the boundedness of the considered validation loss function classes, we will have the following assumptions for the data and regularization parameters. The first assumption is that all features and target values in the training and validation examples are bounded. The second assumption is that we only consider regularization coefficient values λ within an interval $[\lambda_{\min}, \lambda_{\max}]$. In practice, those assumptions are naturally satisfied by data normalization.

Assumption 1 (Bounded covariate and label)**.** *We assume that all the feature vectors and target values in training and validation set is upper-bounded by absolute constants R_1 and R_2, i.e.* $\max\{\|X\|_\infty, \|X_{val}\|_\infty\} \leq R_1$, *and* $\max\{\|y\|_\infty, \|y_{val}\|_\infty\} \leq R_2$.

Assumption 2 (Bounded Coefficient)**.** *We assume that* $\lambda \in [\lambda_{\min}, \lambda_{\max}]^2$ *with* $\lambda_{\min} > 0$.

Under Assumptions 2, 1, Theorem 3.2.1 immediately implies the following generalization guarantee for Elastic Net hyperparameter tuning.

Theorem 3.2.2. *Let $\mathcal{D}$ be an arbitrary distribution over the problem instance space $\Pi_{m,p}$. Under Assumptions 1, 2, the loss functions in $\mathcal{H}_{EN}$ have range bounded by some constant H. Then there exists an algorithm s.t. for any $\epsilon, \delta > 0$, given $N = O(\frac{H^2}{\epsilon^2}(p + \log(\frac{1}{\delta})))$ sample problem instances*

drawn from $\mathcal{D}$, the algorithm outputs a regularization parameter $\hat{\lambda}$ such that with probability at least $1-\delta$, $\mathbb{E}_{P\sim\mathcal{D}} h_{EN}(\hat{\lambda}, P) < \min_\lambda \mathbb{E}_{P\sim\mathcal{D}} h_{EN}(\lambda, P) + \epsilon$.

Discussion and applications. Computing the parameters which minimize the loss on the problem samples (aka Empirical Risk Minimization, or ERM) achieves the sample complexity bound in Theorem 3.2.2. Even though we only need polynomially many samples to guarantee the selection of nearly-optimal parameters, it is not clear how to implement the ERM efficiently. Note that we do not assume the set of features is the same across problem instances, so our approach can handle "feature reset" i.e. different problem instances can differ in not only the number of examples but also the number of features. Moreover, as a special case application, we consider the commonly used techniques of leave-one-out cross validation (LOOCV)[2] and Monte Carlo cross validation (repeated random test-validation splits, typically independent and in a fixed proportion). Given a dataset of size m_{tr}, LOOCV would require m_{tr} regression fits which can be inefficient for large dataset size. Alternately, we can consider draws from a distribution $\mathcal{D}_{LOO}$ which generates problem instances P from a fixed dataset $(X, y) \in \mathbb{R}^{m+1\times p} \times \mathbb{R}^{m+1}$ by uniformly selecting $j \in [m+1]$ and setting $P = (X_{-j*}, y_{-j}, X_{j*}, y_j)$. Theorem 3.2.2 now implies that $\tilde{O}(p/\epsilon^2)$ iterations are enough to determine an ElasticNet parameter $\hat{\lambda}$ with loss within ϵ (w.h.p.) of the parameter λ^* obtained from running the full LOOCV.

Remark 2. *While our result implies polynomial sample complexity, the question of learning the provably near-optimal parameter efficiently is left open.*

Boundedness of the validation loss function

We will now formally show the boundedness of the validation loss function class of Elastic Net $\mathcal{H}_{\text{EN}}$, which is essential for establishing learning guarantees. The following lemma essentially shows that under mild assumptions on the value of data and the search space of hyperparameters, the validation loss function class $\mathcal{H}_{\text{EN}}$ is uniformly bounded by some constant $H > 0$.

Lemma 3.2.3. *Under Assumptions 1 and 2, there exists a uniform constant $H > 0$ so that for all $h_{EN}(\lambda, \cdot) \in \mathcal{H}_{EN} = \{h_{EN}(\lambda, \cdot) : \Pi_{m,p} \to \mathbb{R}_{\geq 0} \mid \lambda \in [\lambda_{\min}, \lambda_{\max}]\}$, we have $\|h_{EN}(\lambda, \cdot)\|_\infty = \sup_{P\in\Pi_{m,p}} |h_{EN}(\lambda, P)| \leq H$.*

Proof. For any problem instance $P = (X, y, X_{\text{val}}, y_{\text{val}}) \in \Pi_{m,p}$, and for any $\lambda = (\lambda_1, \lambda_2) \in [\lambda_{\min}, \lambda_{\max}]^2$, consider the optimization problem for training set

$$\operatorname*{argmin}_\beta F(\beta), \tag{3.2}$$

where $F(\beta) = \frac{1}{2m}\|y - X\beta\|_2^2 + \lambda_1 \|\beta\|_1 + \lambda_2 \|\beta\|_2^2$. If we set $\beta = \vec{0}$, we have

$$F(\vec{0}) = \frac{1}{2m}\|y\|_2^2 \leq C,$$

[2]While standard LOOCV involves computing all m_{tr} train-validation splits, corresponding to leaving out a single training example in each split, we also consider $\mathcal{D}_{LOO}$ which corresponds to computing a series of training-validation splits with a uniformly random training example as the validation set.

for some constant C that only depends on R_2, due to Assumption 2. Let $\hat{\beta}_{(X,y)}(\lambda)$ be the optimal solution of 3.2, we have

$$C \geq F(\hat{\beta}_{(X,y)}(\lambda)) \geq \lambda_1 \left\|\hat{\beta}_{(X,y)}(\lambda)\right\|_1 + \lambda_2 \left\|\hat{\beta}_{(X,y)}(\lambda)\right\|_2^2.$$

Therefore, for any problem instance P, the solution of the training optimization problem $\hat{\beta}_{(X,y)}(\lambda)$ has bounded norm, i.e. $\left\|\hat{\beta}_{(X,y)}(\lambda)\right\|_1, \left\|\hat{\beta}_{(X,y)}(\lambda)\right\|_2^2 \leq \frac{C}{\lambda_{\min}}$, which implies

$$h_{\text{EN}}(\lambda, P) = \frac{1}{2m}\left\|y_{\text{val}} - \hat{\beta}_{(X,y)}(\lambda)X_{\text{val}}\right\|_2^2 \leq \frac{1}{2m}\left\|y_{\text{val}}\right\|_2^2 + \frac{1}{2m}\left\|\hat{\beta}_{(X,y)}(\lambda)X_{\text{val}}\right\|_2^2 \leq H,$$

for some constant H (that only depends on R_1, R_2 and $\lambda_{\min}$). □

3.2.2 Lower bound

Remarkably, we are able to establish a matching lower bound on the pseudo-dimension of the Elastic Net loss function class, parameterized by the regularization parameters. Note that every Elastic Net problem can be converted to an equivalent LASSO problem [177]. In fact, we show something stronger, that the pseudo-dimension of even the LASSO regression loss function class (parameterized by regression coefficient λ_1) is $\Omega(p)$, from which the above observation follows (by taking $\lambda_2 = 0$ in our construction). Our proof of the lower bound adapts the "adversarial strategy" of [115] which is used to design a worst-case LASSO regularization path. While [115] construct a single dataset to bound the number of segments in the piecewise-linear LASSO solution path, we create a collection of problem instances for which all above-below sign patterns may be achieved by selecting regularization parameters from different segments of the solution path.

Theorem 3.2.4. *Let $\mathcal{H}_{LASSO}$ be a set of functions $\{h_{LASSO}(\lambda, \cdot) : \Pi_{m,p} \to \mathbb{R}_{\geq 0} \mid \lambda \in \mathbb{R}^+\}$ that map a regression problem instance $P \in \Pi_{m,p}$ to the validation loss $h_{LASSO}(\lambda, P)$ of LASSO trained with regularization parameter λ. Then* $\text{Pdim}(\mathcal{H}_{LASSO})$ *is $\Omega(p)$.*

Proof. Our proof of the lower bound in Theorem 3.2.4 builds on the "adversarial strategy" due to [115], where a data set (X, y) is constructed with the largest possible number of segments in the LASSO regularization path, for any p. Here we will include and discuss the main results from [115] that are useful in understanding our proof.

Our approach is to construct $N = p$ problem instances such that all 2^N above-below patterns (w.r.t. witness values) for the validation loss are achieved by choosing appropriate points (λ values) on the piecewise linear regularization path of the training instance, by utilizing the property that all unsigned sparsity patterns are achieved by the construction of [115]. In more detail, recall that the *signed* sparsity pattern $\{\eta_1, \ldots, \eta_k\}$ of a piecewise-linear regularization path P for dataset (X, y) is a sequence of vectors in $\{\pm 1, 0\}^p$ corresponding to the signs of the coefficients of the LASSO fit $\hat{\beta}^{(X,y)}(\lambda)$ in consecutive pieces of P, i.e. $\eta_j = (\mathsf{sign}(\hat{\beta}_i^{(X,y)}(\lambda_j)))_{i=1}^p$ where λ_j corresponds to an interior point of the j-th piece of P. Let's further denote by $U_P = \{\overline{\eta}_j \mid 1 \leq j \leq k\}$ where $\overline{\eta}_j = (|\eta_{j1}|, \ldots, |\eta_{jp}|) \in \{0, 1\}^p$ as the *unsigned* sparsity pattern of path P.

We use the same training set (X, y) (but different validation sets) across our problem instances, namely the one with $(3^p + 1)/2$ segments constructed by Mairal and Yu (Theorem 1 of [115]). A useful property of this problem instance is that it achieves all the unsigned sparsity patterns, which follows from the following proposition.

Proposition 3.2.5 ([115]). *Consider y in $\mathbb{R}^n$ and X in $\mathbb{R}^{n\times p}$ such that $X_{\mathcal{E}}$ is full rank for each $\mathcal{E} \subseteq [p]$ and y is in the span of X. Denote by P the regularization path of the Lasso problem corresponding to (X, y), and by k the number of linear segments of P. Then, there exist y' in $\mathbb{R}^{n+1}$ and X' in $\mathbb{R}^{(n+1)\times(p+1)}$ such that the regularization path P' of the Lasso problem associated to (X', y') has $3k - 1$ linear segments. Moreover, let $\{\eta_1 = 0, \eta_2, \ldots, \eta_k\}$ denote the sequence of sparsity patterns in $\{-1, 0, 1\}^p$ of P (the coordinate-wise signs of the solutions $\hat{\beta}^{(X,y)}(\lambda)$), ordered from large to small values of λ. The sequence of sparsity patterns in $\{-1, 0, 1\}^{p+1}$ of the new path P' is the following:*

$$\left\{ \begin{bmatrix} \eta_1 \\ 0 \end{bmatrix}, \begin{bmatrix} \eta_2 \\ 0 \end{bmatrix}, \ldots, \begin{bmatrix} \eta_k \\ 0 \end{bmatrix}, \begin{bmatrix} \eta_k \\ 1 \end{bmatrix}, \begin{bmatrix} \eta_{k-1} \\ 1 \end{bmatrix}, \ldots, \begin{bmatrix} \eta_1 = 0 \\ 1 \end{bmatrix}, \begin{bmatrix} -\eta_2 \\ 1 \end{bmatrix}, \ldots, \begin{bmatrix} -\eta_k \\ 1 \end{bmatrix} \right\}.$$

Formally, one could use a simple inductive argument to establish the above claim. In the base case ($p = 1$), $X = y = [1]$ and it is easy to verify that the regularization path P_1 consists of two segments with $U_{P_1} = \{0, 1\}$. In the inductive case ($p + 1$ features), consider the first $2k$ sign patterns for the path P' in Proposition 3.2.5. Using the inductive hypothesis, it is readily verified that the number of unsigned sparsity patterns in the regularization path P' is $|U_{P'}| = 2|U_P| = 2^{p+1}$.

In other words, all subsets of the p features appear as "active sets" of coefficients along the regularization path of the training set (X, y). By carefully setting the validation sets across the p problem instances in our proof of Theorem 3.2.4, we are able to ensure that the validation loss is non-zero exactly in the subset of problems corresponding to the unsigned sparsity patterns of $\hat{\beta}^{(X,y)}(\lambda)$. Thus, the property that all 2^p unsigned sparsity patterns are achieved for certain values of λ implies that all 2^N validation loss patterns are achieved w.r.t. witnesses 0^p. □

3.2.3 Online Learning

We further extend our results to learning the regularization coefficients given an online sequence of regression problems, such as when one needs to solve a new regression problem each day. Unlike the distributional setting above, we will not assume a fixed problem distribution and our results will hold for an adversarial sequence of problem instances. We will need mild assumptions on the data, namely boundedness of feature and prediction values and 'smoothness' of predictions.

Our first assumption is that all feature values and predictions are bounded, for training as well as validation examples.

Assumption 3. *The predicted variable and all feature values are bounded by an absolute constant R, i.e. $\max\{||X^{(i)}||_{\infty,\infty}, ||y^{(i)}||_\infty, ||X^{(i)}_{val}||_{\infty,\infty}, ||y^{(i)}_{val}||_\infty\} \leq R$.*

We assume that the predicted variable y in the training set comes from a κ-bounded (i.e. smooth) distribution. Moreover, the online adversary is allowed to change the distribution as long as it is

κ-bounded. Note that our assumption also captures common data preprocessing steps, for example the jitter parameter in the popular Python library scikit-learn [135] adds a uniform noise to the y values to help model stability. The assumption is formally stated as follows:

Assumption 4 (Smooth predictions)**.** *The predicted variables $y^{(i)}$ in the training set are drawn from a joint κ-bounded distribution, i.e. for each i, the variables $y^{(i)}$ have a joint distribution with probability density bounded by κ.*

Under these assumptions, we can show that it is possible to learn the ElasticNet parameters with sublinear expected regret when the problem instances arrive online. The learning algorithm that achieves this regret is a continuous variant of the classic Exponential Weights algorithm [15, 51]. It samples points in the domain with probability inversely proportional to the exponentiated loss.

Our key contribution is to show that the loss sequence is dispersed (Definition 2) under the above assumptions. This involves establishing additional structure for the problem, specifically about the location of boundary functions in the piecewise structure from Theorem 3.1.3.

Theorem 3.2.6. *Suppose Assumptions 3 and 4 hold. Let $l_1, \ldots, l_T : C^2 \to \mathbb{R}_{\geq 0}$ denote an independent sequence of losses (e.g. fresh randomness is used to generate the validation set features in each round) as a function of the ElasticNet regularization parameter $\lambda = (\lambda_1, \lambda_2)$, $l_i(\lambda) = l_r(\hat{\beta}_{\lambda, f_{EN}}^{(X^{(i)}, y^{(i)})}, (X_{val}^{(i)}, y_{val}^{(i)}))$, for compact $C \subset \mathbb{R}^+$. The sequence of functions is $\frac{1}{2}$-dispersed, and there is an online algorithm with $\tilde{O}(\sqrt{T})$[3] expected regret. The result also holds for loss functions adjusted by information criteria AIC and BIC.*

Proof. We start with the piecewise-decomposable characterization of the dual class function in Theorem 3.1.3. On any fixed problem instance $P \in \Pi_{m,n}$, as the parameter λ is varied in the loss function $\ell_{\text{EN}}(\cdot, P)$ of ElasticNet trained with regularization parameter $\lambda = (\lambda_1, \lambda_2)$, we have the following piecewise structure. There are $k = p3^p$ boundary functions $g_1, \ldots, g_k$ for which the transition boundaries are algebraic curves $r_i(\lambda_1, \lambda_2)$, where r_i is a polynomial with degree 1 in λ_1 and at most p in λ_2. Also the piece function $f_{\mathbf{b}}$ for each sign pattern $\mathbf{b} \in \{0, 1\}^k$ is a rational polynomial function $\frac{q_1^b(\lambda_1, \lambda_2)}{q_2^b(\lambda_2)}$, where $q_1^{\mathbf{b}}, q_2^{\mathbf{b}}$ have degrees at most $2p$, and corresponds to a fixed signed equicorrelation set $(\mathcal{E}, s)$. To show online learnability, we will examine this piecewise structure more closely – in particular analyse how the structure varies when the predicted variable is drawn from a smooth distribution.

In order to show dispersion for the loss functions $\{l_i(\lambda)\}$, we will use the recipe of [19] and bound the worst rate of discontinuities between any pair of points $\lambda = (\lambda_1, \lambda_2)$ and $\lambda' = (\lambda_1', \lambda_2')$ with $||\lambda - \lambda'||_2 \leq \epsilon$ along the axis-aligned path $\lambda \to (\lambda_1', \lambda_2) \to \lambda'$. First observe that the only possible points at which $l_i(\lambda)$ may be discontinuous are

(a) (λ_1, λ_2) such that $r_i(\lambda_1, \lambda_2) = 0$ corresponding to some boundary function g_i.
(b) (λ_1, λ_2) such that $q_2^{\mathbf{b}}(\lambda_2) = 0$ corresponding to some piece function $f_{\mathbf{b}}$.

Fortunately the discontinuity of type (b) does not occur for $\lambda_2 > 0$. From the ElasticNet characterization in Lemma 3.1.1, and using Lemma 3.1.2, we know that $q_2(\lambda_2) = \Pi_{j \in [|\mathcal{E}|]}(\Lambda_j + \lambda_2)$,

[3]The $\tilde{O}(\cdot)$ notation hides dependence on logarithmic terms, as well as on quantities besides T.

where $(\Lambda_j)_{j\in[|\mathcal{E}|]}$ are non-negative eigenvalues of the positive semi-definite matrix ${X_{\mathcal{E}}^{(i)}}^T X_{\mathcal{E}}^{(i)}$. It follows that $q_2^{\mathbf{b}}$ does not have positive zeros (for any sign vector $\mathbf{b}$).

Therefore it suffices to locate boundaries of type (a). To this end, we have two subtypes corresponding to a variable entering or leaving the equicorrelation set.

Addition of $j \notin \mathcal{E}$. As observed in the proof of Theorem 3.1.3, a variate $j \notin \mathcal{E}$ can enter the equicorrelation set $\mathcal{E}$ only for (λ_1, λ_2) satisfying

$$\lambda_1 = K_0 \left(\mathbf{x}_j^T (X_{\mathcal{E}}({X_{\mathcal{E}}}^T X_{\mathcal{E}} + \lambda_2 I_{|\mathcal{E}|})^{-1} {X_{\mathcal{E}}}^T - \mathbf{x}_j^T\right) y$$

(K_0 does not depend on λ_1, λ_2 or y). For fixed λ_2, the distribution of λ_1 at which the discontinuity occurs for insertion of j is $K_1\kappa$-bounded (by Lemma A.2.7) for some constant K_1 that only depends on R, m, p and $\lambda_{\max}$. This implies an upper bound of $K_1\kappa\epsilon$ on the expected number of discontinuities corresponding to j along the segment $\lambda \to (\lambda_1', \lambda_2)$ for any $j, \mathcal{E}$.

For constant λ_1, we can use Lemma 3.1.2 and a standard change of variable argument (e.g. Theorem 22 of [19]) to conclude that the discontinuties lie at the roots of a random polynomial in λ_2 of degree $|\mathcal{E}|$, leading coefficient 1, and bounded random coefficients with $K_2\kappa$-bounded density for some constant K_2 (that only depends on R, m, p and $\lambda_{\max}$). By Theorem A.2.4, the expected number of discontinuities along the segment $(\lambda_1', \lambda_2) \to \lambda'$ is upper bounded by $K_2 K_p \kappa\epsilon$ (K_p only depends on p). This implies that the expected number of Lipschitz violations between λ and λ' along the axis aligned path is $\tilde{O}(\kappa\epsilon)$ and completes the first step of the recipe in this case ($\tilde{O}$ notation suppresses terms in R, m, p and $\lambda_{\max}$ as constants).

Removal of $j' \in \mathcal{E}$. The second case, when a variate $j' \in \mathcal{E}$ leaves the equicorrelation set $\mathcal{E}$ for (λ_1, λ_2) satisfying $\lambda_1((X_{\mathcal{E}}{}^T X_{\mathcal{E}} + \lambda_2 I_{|\mathcal{E}|})^{-1} s)_{j'} = ((X_{\mathcal{E}}{}^T X_{\mathcal{E}} + \lambda_2 I_{|\mathcal{E}|})^{-1} {X_{\mathcal{E}}}^T y)_{j'}$, also yields the same bound using the above arguments. Putting together, and noting that we have at most $p3^p$ distinct curves each with $\tilde{O}(\kappa\epsilon)$ expected number of intersections with the axis aligned path $\lambda \to \lambda'$, the total expected number of discontinuities is also $\tilde{O}(\kappa\epsilon)$. This completes the first step (S1) of the above recipe.

We use Theorem 9 of [19] to complete the second step of the recipe, which employs a VC-dimension argument for K' algebraic curves of bounded degrees (here degree is at most $p+1$) to conclude that the expected worst number of discontinuties along any axis-aligned path between any pair of points $\leq \epsilon$ apart is at most $\tilde{O}(\epsilon T) + O(\sqrt{T \log K' T})$. $K' \leq p3^p$ as shown above. This implies that the sequence of loss functions is $\frac{1}{2}$-dispersed, and further there is an algorithm (Algorithm 4 of [15]) that achieves $\tilde{O}(\sqrt{T})$ expected regret.

Finally note that loss functions with AIC and BIC have the same dual class piecewise structure, and therefore the above analysis applies. The only difference is that the value of the piece functions $f_{\mathbf{b}}$ are changed by a constant (in λ), $K_{m,p} \leq p \log m$. The piece boundaries are the same, and are therefore $\frac{1}{2}$-dispersed as above. The range of the loss functions is now $[0, K_{m,p}+1]$, so the same algorithm (Algorithm 4 of [15]) again achieves $\tilde{O}(\sqrt{T})$ expected regret. □

3.3 Regularized Kernel Regression

The Kernel Least Squares Regression ([92]) is a natural generalization of the linear regression problem, which uses a kernel to handle non-linearity. In this problem, each sample has p_1 feature, corresponding to a real-valued target. Formally, each problem instance P drawn from Π

can be described as

$$P = (X, y, X_{\text{val}}, y_{\text{val}}) \in \mathbb{R}^{m \times p_1} \times \mathbb{R}^m \times \mathbb{R}^{m' \times p_1} \times \mathbb{R}^{m'}.$$

A common issue in practice is that the relation between y and X is non-linear in the original space. To overcome this issue, we consider the mapping $\phi : \mathbb{R}^{p_1} \to \mathbb{R}^{p_2}$ which maps the original input space to a new feature space in which we hopefully can perform linear regression. Define $\phi(X) = (\phi(x_1), \ldots, \phi(x_m))_{m \times p_2}$, our goal is to find a vector $\theta \in \mathbb{R}^{p_2}$ so that the squared loss $\frac{1}{2} \|y - \phi(X)\theta\|_2^2 + R(\|\theta\|)$ is minimized, where the regularization term $R(\|\theta\|)$ is any strictly monotonically increasing function of the Hilbert space norm. It is well-known from the literature (e.g. Schölkopf et al. [148]) that under the Representer Theorem's conditions, the optimal value θ^* can be linearly represented by row vectors of $\phi(X)$, i.e., $\theta^* = \phi(X)\beta = \sum_{i=1}^m \phi(x_i)\beta_i$, where $\beta = (\beta_1, \ldots, \beta_m) \in \mathbb{R}^m$. This directly includes the ℓ_2 regularizer but does not include ℓ_1 regularization. To overcome this issue, Roth (Roth [145]) proposed an alternative approach to regularized kernel regression, which directly restricts the representation of coefficient θ via a linear combination of $\phi(x_i)$, for $i \in [m]$. The regularized kernel regression hence can be formulated as

$$\hat{\beta}_{l,\lambda}^{(X,y)} = \operatorname*{argmin}_{\beta \in \mathbb{R}^m} \frac{1}{2} \|y - K\beta\|_2^2 + \lambda_1 \|\beta\|_1 + \lambda_2 \|\beta\|_2^2,$$

where $k(x, x') = \langle \phi(x), \phi(x') \rangle$ is the kernel mapping, and the Gram matrix K satisfies $[K]_{i,j} = k(x_i, x_j)$ for all $i, j \in [m]$.

Clearly, the problem above is a linear regression problem. Formally, denote $h_{\text{KER}}(\lambda, P) = \frac{1}{2}\|y - K\hat{\beta}_{(X,y)}(\lambda)\|_2$ and let $\mathcal{H}_{\text{KER}} = \{h_{\text{KER}}(\lambda, \cdot) : \Pi_{m,p} \to \mathbb{R}_{\geq 0} \mid \lambda \in \mathbb{R}_+^2\}$. The following result is a direct corollary of Theorem 3.1.6, which gives an upper bound for the pseudo-dimension of $\mathcal{H}_{\text{KER}}$.

Corollary 3.3.1. $\text{Pdim}(\mathcal{H}_{KER}) = O(m)$.

Note that m here denotes the training set size for a single problem instance, and Corollary 3.3.1 implies a guarantee on the number of problem instances needed for learning a good regularization parameter for kernel regression via classic results [4, 7]. Our results do not make any assumptions on the m samples within a problem instance/dataset; if these samples within problem instances are further assumed to be i.i.d. draws from some data distribution (distinct from problem distribution $\mathcal{D}$), then well-known results imply that $m = O(k \log p)$ samples are sufficient to learn the optimal LASSO coefficient [164, 165], where k denotes the number of non-zero coefficients in the optimal regression fit.

The results in this chapter are based on joint work with Nina Balcan, Misha Khodak and Ameet Talwalkar [28], which appeared in NeurIPS 2022, and joint work with Nina Balcan and Anh Nguyen [31], which appeared in NeurIPS 2023.

Chapter 4

Decision Trees

4.1 Introduction

Decision trees are ubiquitous, with applications in operations research, management science, data mining, and machine learning. They are easy to use and understand models that explicitly include the decision rules used in making predictions. Each decision rule is a simple comparsion of a real-valued attribute to a threshold or a categorical attribute against a candidate set of values. Given their remarkable simplicity, decision trees are widely preferred in applications where it is important to justify algorithmic decisions with intuitive explanations [147]. However, decades of research on decision trees has resulted in a large suite of candidate approaches for building decision trees [45, 98, 114, 116, 121, 141, 142]. This raises an important question, namely how should one select the best approach to build a decision tree for the problem at hand.

Several empirical studies have been performed comparing various ways to build decision trees [74, 122, 123, 128]. Current wisdom from the literature dictates that for any problem at hand, one needs a domain expert to try out, compare and tune various methods to build the best decision trees for any given problem domain. For instance, the popular Python library Scikit-learn [135] implements both Gini impurity and entropy as candidate 'splitting criteria' (a crucial component in building the decision trees top-down by deciding which node to split into child nodes), and yet theory suggests another promising candidate [98] that achieves smaller error bounds under the Weak Hypothesis Assumption[1]. It is therefore desirable to determine which approach works better for the data coming from a given domain. With sufficient data, can we automate this tedious manual process?

In this chapter we approach this crucial question, and propose ways to build more effective decision trees automatically. Our results show provable learning theoretic guarantees and select methods over larger search spaces than what human experts would typically explore. For example, instead of comparing a small finite number of splitting criteria, we examine learnability over continuously infinite parameterized families that yield more effective decision tree learning algorithms.

We consider the problem where the learner has access to multiple related datasets $D_1, \ldots, D_N$

[1] an *a priori* assumption on the target function. Roughly speaking, it means that the decision tree node functions are already slightly correlated with the target function.

coming from the same problem domain (given by a fixed but unknown distribution $\mathcal{D}$), and the goal is to design a decision tree learning algorithm that works well over the distribution $\mathcal{D}$ using as few datasets (N, the sample complexity) as possible. This algorithm design problem is typically formulated as the selection of a hyperparameter from an infinite family. Typically finding the best hyperparameters even on a single problem sample is tedious and computationally intensive, so we would like to bound the number of samples over which we should optimize them, while learning parameters that generalize well over the distribution generating the problem samples. We take steps towards systematically unifying, automating and formalizing the process of designing decision tree learning algorithms, in a way that is adaptive to the data domain.

4.2 Preliminaries and definitions

Let $[k]$ denote the set of integers $\{1, 2, \ldots, k\}$. A (supervised) classification problem is given by a labeled dataset $D = (X, y)$ over some input domain $X \in \mathcal{X}^n$ and $y \in \mathcal{Y}^n = [c]^n$ where c denotes the number of distinct classes or categories. Let $\mathcal{D}$ be a distribution over classification problems of size n.[2] We will consider parameterized families of decision tree learning algorithms, parameterized by some parameter $\rho \in \mathcal{P} \subseteq \mathbb{R}^d$ and access to datasets $D_1, \ldots, D_N \sim \mathcal{D}^N$. We do not assume that individual data points (X_i, y_i) are i.i.d. in any dataset D_j.

We consider a finite *node function class* $\mathcal{F}$ consisting of boolean functions $\mathcal{X} \to \{0, 1\}$ which are used to label internal nodes in the decision tree, i.e. govern given any data point $x \in \mathcal{X}$ whether the left or right branch should be taken when classifying x using the decision tree. Any given data point $x \in \mathcal{X}$ corresponds to a unique leaf node determined by the node function evaluations at x along some unique root-to-leaf path. Each leaf node of the decision tree is labeled by a class in $[c]$. Given a dataset (X, y) this leaf label is typically set as the most common label for data points $x \in X$ which are mapped to the leaf node.

We denote by $T_{l \to f}$ the tree obtained by *splitting* the leaf node l, which corresponds to replacing it by an internal node labeled by f and creating two child leaf nodes. We consider a parameterized class of splitting criterion $\mathcal{G}_{\mathcal{P}}$ over some parameter space $\mathcal{P}$ consisting of functions $g_\rho : [0, 1]^c \to \mathbb{R}_{\geq 0}$ for $\rho \in \mathcal{P}$ that govern which leaf to be split next and which node function $f \in \mathcal{F}$ to be used when building the decision tree using a top-down learning algorithm which builds a decision tree by successively splitting nodes using g_ρ until the size equals input tree size t. More precisely, suppose $w(l)$ (the *weight* of leaf l) denotes the number of data points in X that map to leaf l, and suppose $p_i(l)$ denotes the fraction of data points labeled by $y = i \in [c]$ among those points that map to leaf l. The splitting function over tree T is given by

$$G_\rho(T) = \sum_{l \in \text{leaves}(T)} w(l) g_\rho\left(\{p_i(l)\}_{i=1}^c\right),$$

and we build the decision tree by successively splitting the leaf nodes using node function f which cause the maximum decrease in the splitting function. For example, the information gain criterion may be expressed using $g_\rho(\{p_i(l)\}_{i=1}^c) = -\sum_{i=1}^c p_i \log p_i$.

[2]For simplicity of technical presentation we assume that the dataset size n is fixed across problem instances, but our sample complexity results hold even without this assumption.

Algorithm 9 Top-down decision tree learner ($\mathcal{F}$, g_ρ, t)

Input: Dataset $D = (X, y)$
Parameters: Node function class $\mathcal{F}$, splitting criterion $g_\rho \in \mathcal{G}_\mathcal{P}$, tree size t
Output: Decision tree T

Initialize T to a leaf node labeled by most frequent label y in D.
while T has at most t internal nodes **do**
  $l^*, f^* \leftarrow \text{argmin}_{l \in \text{leaves}(T), f \in \mathcal{F}} G_\rho(T_{l \to f})$
  $T \leftarrow T_{l^* \to f^*}$
return T

Algorithm 9 summarizes this well-known general paradigm. We denote the tree obtained by the top-down decision tree learner on dataset D as $T_{\mathcal{F},\rho,t}(D)$. We study the 0-1 loss of the resulting decision tree classifier. If $T(x) \in [c]$ denotes the prediction of tree T on $x \in \mathcal{X}$, we define the loss on dataset $D(X, y)$ as

$$L(T, D) := \frac{1}{n} \sum_{i=1}^{n} \mathbf{I}[T(X_i) \neq y_i],$$

where $\mathbf{I}[\cdot]$ denotes the 0-1 valued indicator function.

4.3 Learning to split nodes

Given a discrete probability distribution $P = \{p_i\}$ with $\sum_{i=1}^{c} p_i = 1$, we define (α, β)-Tsallis entropy as

$$g_{\alpha,\beta}^{\text{TSALLIS}}(P) := \frac{C}{\alpha - 1} \left(1 - \left(\sum_{i=1}^{c} p_i^\alpha \right)^\beta \right),$$

where C is a normalizing constant (does not affect Algorithm 9), $\alpha \in \mathbb{R}^+, \beta \in \mathbb{Z}^+$. $\beta = 1$ corresponds to standard Tsallis entropy [160]. For example, $\alpha = 2, \beta = 1$ corresponds to Gini impurity, $\alpha = \frac{1}{2}, \beta = 2$ corresponds to the Kearns and Mansour criterion (using which error ϵ can be achieved with trees of size $\text{poly}(1/\epsilon)$, [98]) and $\lim_{\alpha \to 1} g_{\alpha,1}^{\text{TSALLIS}}(P)$ yields the (Shannon) entropy criterion. Formally, we have

Proposition 4.3.1. *The splitting criteria $g_{2,1}^{\text{TSALLIS}}(P), g_{\frac{1}{2},2}^{\text{TSALLIS}}(P)$ and $\lim_{\alpha \to 1} g_{\alpha,1}^{\text{TSALLIS}}(P)$ correspond to Gini impurity, the [98] objective and the entropy criterion respectively.*

Proof of Proposition 4.3.1. Setting $\alpha = 2, \beta = 1$ immediately yields the expression for Gini impurity. Plugging $\alpha = \frac{1}{2}, \beta = 2$ yields

$$
\begin{aligned}
g^{\text{TSALLIS}}_{\frac{1}{2},2}(P) &= \frac{C}{-\frac{1}{2}}\left(1-\left(\sum_{i=1}^{c}\sqrt{p_i}\right)^2\right) \\
&= 2C\left(\sum_{i=1}^{c}p_i + 2\sum_{i\neq j}\sqrt{p_i p_j} - 1\right) \\
&= 4C\sum_{i\neq j}\sqrt{p_i p_j}.
\end{aligned}
$$

For $c=2$, $g^{\text{TSALLIS}}_{\frac{1}{2},2}(P) = 4C\sqrt{p_1(1-p_1)}$ which matches the splitting function of [98]. Also taking the limit $\alpha \to 1$ gives

$$
\begin{aligned}
g^{\text{TSALLIS}}_{\alpha\to 1,\beta}(P) &= \lim_{\alpha\to 1}\frac{C}{\alpha-1}\left(1-\left(\sum_{i=1}^{c}{p_i}^{\alpha}\right)^{\beta}\right) \\
&= -C\beta\left(\sum_{i=1}^{c}p_i^{\alpha}\right)^{\beta-1}\left(\sum_{i=1}^{c}p_i^{\alpha}\ln p_i\right) \\
&= -C\beta\left(\sum_{i=1}^{c}p_i\ln p_i\right).
\end{aligned}
$$

For $\beta = 1$, this corresponds to the entropy criterion. □

We further show that the $g^{\text{TSALLIS}}_{\alpha,\beta}(P)$ family of splitting criteria enjoys the property of being *permissible* splitting criteria (in the sense of [98]) for any $\alpha \in \mathbb{R}^+, \beta \in \mathbb{Z}^+, \alpha \notin (1/\beta, 1)$, which implies useful desirable guarantees the top-down decision tree learner [65, 98].

Proposition 4.3.2. *(α,β)-Tsallis entropy has the following properties for any $\alpha \in \mathbb{R}^+, \beta \in \mathbb{Z}^+, \alpha \notin (1/\beta, 1)$*

1. *(Symmetry) For any $P = \{p_i\}$, $Q = \{p_{\pi(i)}$ for some permutation π over $[c]$, $g^{\text{TSALLIS}}_{\alpha,\beta}(Q) = g^{\text{TSALLIS}}_{\alpha,\beta}(P)$.*
2. *$g^{\text{TSALLIS}}_{\alpha,\beta}(P) = 0$ at any vertex $p_i = 1, p_j = 0$ for all $j \neq i$ of the probability simplex P.*
3. *(Concavity) $g^{\text{TSALLIS}}_{\alpha,\beta}(aP+(1-a)Q) \geq a g^{\text{TSALLIS}}_{\alpha,\beta}(P)+(1-a)g^{\text{TSALLIS}}_{\alpha,\beta}(Q)$ for any $a \in [0,1]$.*

Proof of Proposition 4.3.2. Properties 1 and 2 are readily verified. We further show that (α,β)-Tsallis entropy is concave for $\alpha, \beta > 0, \alpha\beta \geq 1$.

First consider the case $\alpha \geq 1$. We use the fact that the univariate function $f(x) = x^{\theta}$ is

convex for all $\theta \geq 1$. For any $a \in [0,1]$, $P = \{p_i\}_{i=1}^c, Q = \{q_i\}_{i=1}^c$,

$$\begin{aligned}
g_{\alpha,\beta}^{\text{TSALLIS}}(aP + (1-a)Q) &= \frac{C}{\alpha - 1}\left(1 - \left(\sum_{i=1}^{c}(ap_i + (1-a)q_i)^\alpha\right)^\beta\right) \\
&\geq \frac{C}{\alpha - 1}\left(1 - \left(\sum_{i=1}^{c} ap_i{}^\alpha + (1-a)q_i^\alpha\right)^\beta\right) \\
&= \frac{C}{\alpha - 1}\left(1 - \left(a\sum_{i=1}^{c} p_i{}^\alpha + (1-a)\sum_{i=1}^{c} q_i^\alpha\right)^\beta\right) \\
&\geq \frac{C}{\alpha - 1}\left(1 - \left(a\left(\sum_{i=1}^{c} p_i{}^\alpha\right)^\beta + (1-a)\left(\sum_{i=1}^{c} q_i^\alpha\right)^\beta\right)\right) \\
&= a g_{\alpha,\beta}^{\text{TSALLIS}}(P) + (1-a) g_{\alpha,\beta}^{\text{TSALLIS}}(Q).
\end{aligned}$$

It remains to consider the case $0 < \alpha \leq 1/\beta$. In this case, we apply the reverse Minkowski's inequality and use that $\alpha\beta \leq 1$ to establish concavity.

$$\begin{aligned}
g_{\alpha,\beta}^{\text{TSALLIS}}(aP + (1-a)Q) &= \frac{C}{\alpha - 1}\left(1 - \left(\sum_{i=1}^{c}(ap_i + (1-a)q_i)^\alpha\right)^\beta\right) \\
&\geq \frac{C}{\alpha - 1}\left(1 - \left(\left(\sum_{i=1}^{c}(ap_i)^\alpha\right)^{\frac{1}{\alpha}} + \left(\sum_{i=1}^{c}((1-a)q_i)^\alpha\right)^{\frac{1}{\alpha}}\right)^{\alpha\beta}\right) \\
&= \frac{C}{\alpha - 1}\left(1 - \left(a\left(\sum_{i=1}^{c} p_i{}^\alpha\right)^{\frac{1}{\alpha}} + (1-a)\left(\sum_{i=1}^{c} q_i^\alpha\right)^{\frac{1}{\alpha}}\right)^{\alpha\beta}\right) \\
&\geq \frac{C}{\alpha - 1}\left(1 - \left(a\left(\sum_{i=1}^{c} p_i{}^\alpha\right)^{\frac{1}{\alpha}\cdot\alpha\beta} + (1-a)\left(\sum_{i=1}^{c} q_i^\alpha\right)^{\frac{1}{\alpha}\cdot\alpha\beta}\right)\right) \\
&= a g_{\alpha,\beta}^{\text{TSALLIS}}(P) + (1-a) g_{\alpha,\beta}^{\text{TSALLIS}}(Q).
\end{aligned}$$

□

The above properties ensure that (α, β)-Tsallis entropy is a permissible splitting criterion whenever $\alpha \notin (1/\beta, 1)$.

We consider $\alpha \in \mathbb{R}^+$ and $\beta \in [B]$ for some positive integer B, and observe that several previously studied splitting criteria can be readily obtained by setting appropriate values of parameters α, β. We consider the problem of tuning the parameters α, β simultaneously when designing the splitting criterion, given access to multiple problem instances (datasets) drawn from some distribution $\mathcal{D}$. The goal is to find parameters $\hat{\alpha}, \hat{\beta}$ based on the training samples, so that on a random $D \sim \mathcal{D}$, the expected loss

$$\mathbb{E}_{D\sim\mathcal{D}} L(T_{\mathcal{F},(\hat{\alpha},\hat{\beta}),t}, D)$$

is minimized. We will bound the sample complexity of the ERM (Empirical Risk Minimization) principle, which given N problem samples $D_1, \ldots, D_N$ computes parameters $\hat{\alpha}, \hat{\beta}$ such that

$$\hat{\alpha}, \hat{\beta} = \text{argmin}_{\alpha>0,\beta\in[B]} \sum_{i=1}^{N} L(T_{\mathcal{F},(\alpha,\beta),t}, D_i).$$

We obtain the following guarantee on the sample complexity of learning a near-optimal splitting criterion.

Theorem 4.3.3. *Suppose $\alpha > 0$ and $\beta \in [B]$. For any $\epsilon, \delta > 0$ and any distribution $\mathcal{D}$ over problem instances with n examples, $O(\frac{1}{\epsilon^2}(t(\log|\mathcal{F}| + \log t + c\log(B+c)) + \log\frac{1}{\delta}))$ samples drawn from $\mathcal{D}$ are sufficient to ensure that with probability at least $1-\delta$ over the draw of the samples, the parameters $\hat{\alpha}, \hat{\beta}$ learned by ERM over the sample have expected loss that is at most ϵ larger than the expected loss of the best parameters $\alpha^*, \beta^* = \text{argmin}_{\alpha>0,\beta\geq 1}\mathbb{E}_{D\sim\mathcal{D}} L(T_{\mathcal{F},(\hat{\alpha},\hat{\beta}),t}, D)$ over $\mathcal{D}$. Here t is the size of the decision tree, $\mathcal{F}$ is the node function class used to label the nodes of the decision tree and c is the number of label classes.*

Proof. Our overall approach is to analyze the structure of the dual class loss function [7], that is the loss as a function of the hyperparameters α, β for a fixed problem instance (X, y). Based on this structure, we give a bound on the pseudodimension of the loss function class which implies a bound on the sample complexity using classic learning theoretic results.

Since the loss is completely determined by the final decision tree $T_{\mathcal{F},(\alpha,\beta),t}$, it suffices to bound the number of different algorithm behaviors as one varies the hyperparameters α, β in Algorithm 9. As the tree is grown according to the top-down algorithm, suppose the number of internal nodes is $\tau < t$. There are $\tau+1$ candidate leaf nodes to split and $|\mathcal{F}|$ candidate node functions, for a total of $(\tau+1)|\mathcal{F}|$ choices for (l, f). For any of $\binom{(\tau+1)|\mathcal{F}|}{2}$ pair of candidates (l_1, f_1) and (l_2, f_2), the preference for which candidate is 'best' and selected for splitting next is governed by the splitting functions $G_{\alpha,\beta}(T_{l_1\to f_1})$ and $G_{\alpha,\beta}(T_{l_2\to f_2})$. This preference flips across boundary condition given by $\sum_{l\in\text{leaves}(T_{l_1\to f_1})} w(l) g_{\alpha,\beta}(\{p_i(l)\}) = \sum_{l\in\text{leaves}(T_{l_2\to f_2})} w(l) g_{\alpha,\beta}(\{p_i(l)\})$. Most terms (all but three) cancel out on both sides as we substitute a single leaf node by an internal node on both LHS and RHS. The only unbalanced terms correspond to deleted leaves l_1, l_2 and newly introduced leaves $l_1^a, l_1^b, l_2^a, l_2^b$, i.e.

$$\sum_{l\in\{l_1^a, l_1^b, l_2\}} w(l) g_{\alpha,\beta}(\{p_i(l)\}) = \sum_{l\in\{l_2^a, l_2^b, l_1\}} w(l) g_{\alpha,\beta}(\{p_i(l)\})$$

where $g_{\alpha,\beta}(\cdot) = g_{\alpha,\beta}^{\text{TSALLIS}}(\cdot)$, the (α,β)-Tsallis entropy. For integer β, by the multinomial theorem, $(\sum_{i=1}^{c} p_i(l)^\alpha)^\beta$ consists of at most $\binom{\beta+c-1}{c}$ distinct terms. By Rolle's theorem, the number of distinct solutions of the above equation in α is $O((\beta+c)^c)$. Thus, for any fixed β and fixed partial decision tree built in τ rounds, the number of critical points of α at which the argmax in Line 3 of Algorithm 9 changes is at most $O(|\mathcal{F}|^2\tau^2(\beta+c)^c)$ and a fixed leaf node is split and labeled by a fixed f for any interval of α induced by these critical points. Over t rounds, this corresponds to at most $O(\Pi_{\tau=1}^{t}|\mathcal{F}|^2\tau^2(\beta+c)^c)$ critical points across which the algorithmic behaviour (sequence

of choices of node splits in Algorithm 9) can change as α is varied for a fixed β. Adding up over $\beta \in [B]$, we get $O(\sum_{\beta=1}^{B} |\mathcal{F}|^{2t} t^{2t} (\beta + c)^{ct})$, or at most $O(B|\mathcal{F}|^{2t} t^{2t} (B + c)^{ct})$ critical points.

This implies a bound of $O(t(\log |\mathcal{F}| + \log t + c \log(B + c))$ on the pseudodimension of the loss function class by using Lemma A.3.2. Finally, an application of Theorem A.3.1 completes the proof. □

Observe that parameter α is tuned over a continuous domain and our near-optimality guarantees hold over the entire continuous domain (as opposed to say over a finite grid of α values). Our results have implications for cross-validation since typical cross-validation can be modeled via a distribution $\mathcal{D}$ created by sampling splits from the same fixed dataset, in which case our results imply how many splits are sufficient to converge to within ϵ error of best the parameter learned by the cross validation procedure. Similar convergence guarantees have been shown for tuning the regularization coefficients of the elastic net algorithm for linear regression via cross-validation [28, 31]. Our setting is of course more general than just cross validation and includes the case where the different datasets come from related similar tasks for which we seek to learn a common good choice of hyperparameters.

While (α, β)-Tsallis entropy is well-motivated as a parameterized class of node splitting criteria as it includes several previously studied splitting criteria, and generalizes the Tsallis entropy which may be of independent interest in other applications, it involves simulatenous optimization of two parameters which can be computationally challenging. To this end, we define the following single parameter family which interpolates known node splitting methods:

$$g_\gamma(\{p_i\}) := C \left(\Pi_i p_i\right)^\gamma,$$

where $\gamma \in (0, 1]$ and C is some constant. For binary classification, the setting $\gamma = \frac{1}{2}$ and $\gamma = 1$ correspond to [98] and Gini entropy respectively, for appropriate choice of C. It is straightforward to verify that g_γ is permissible for all $\gamma \in (0, 1]$, i.e. is symmetric, zero at simplical vertices and concave. We show the following improved sample complexity guarantee for tuning γ. Note that this family is not a special case of (α, β)-Tsallis entropy, but contains additional splitting functions which may work well on given domain-specific data. Also, since it has a single parameter, it can be easier to optimize efficiently in practice.

Theorem 4.3.4. *Suppose $\gamma \in (0, 1]$. For any $\epsilon, \delta > 0$ and any distribution $\mathcal{D}$ over problem instances with n examples, $O(\frac{1}{\epsilon^2}(t(\log |\mathcal{F}| + \log t) + \log \frac{1}{\delta}))$ samples drawn from $\mathcal{D}$ are sufficient to ensure that with probability at least $1 - \delta$ over the draw of the samples, the parameter $\hat{\gamma}$ learned by ERM over the sample is ϵ-optimal, i.e. has expected loss at most ϵ larger than that of the optimal parameter over $\mathcal{D}$.*

Proof. The loss is completely determined by the final decision tree $T_{\mathcal{F},\gamma,t}$. It suffices to bound the number of different algorithm behaviors as one varies the hyperparameter γ in Algorithm 9. As the tree is grown according to the top-down algorithm, suppose the number of internal nodes is $\tau < t$. There are $\tau + 1$ candidate leaf nodes to split and $|\mathcal{F}|$ candidate node functions, for a total of $(\tau + 1)|\mathcal{F}|$ choices for (l, f). For any of $\binom{(\tau+1)|\mathcal{F}|}{2}$ pair of candidates (l_1, f_1) and (l_2, f_2), the preference for which candidate is 'best' and selected for splitting next is governed by the splitting functions $G_\gamma(T_{l_1 \to f_1})$ and $G_\gamma(T_{l_2 \to f_2})$. This preference flips across boundary condition given by $\sum_{l \in \text{leaves}(T_{l_1 \to f_1})} w(l) g_\gamma(\{p_i(l)\}) = \sum_{l \in \text{leaves}(T_{l_2 \to f_2})} w(l) g_\gamma(\{p_i(l)\})$. Most terms (all but three)

cancel out on both sides as we substitute a single leaf node by an internal node on both LHS and RHS. The only unbalanced terms correspond to deleted leaves l_1, l_2 and newly introduced leaves $l_1^a, l_1^b, l_2^a, l_2^b$, i.e.

$$\sum_{l \in \{l_1^a, l_1^b, l_2\}} w(l) g_\gamma(\{p_i(l)\}) = \sum_{l \in \{l_2^a, l_2^b, l_1\}} w(l) g_\gamma(\{p_i(l)\}).$$

Recall $g_\gamma(\{p_i\}) := C\left(\Pi_i p_i\right)^\gamma$, giving an equation in six or $O(1)$ terms. By Rolle's theorem, the number of distinct solutions of the above equation in γ is $O(1)$. Thus, the number of critical points of γ at which the argmax in Line 3 of Algorithm 9 changes is at most $O(|\mathcal{F}|^2\tau^2)$ and a fixed leaf node is split and labeled by a fixed f for any interval of γ induced by these critical points. Over t rounds, this corresponds to at most $O(\Pi_{\tau=1}^t |\mathcal{F}|^2\tau^2) = O(|\mathcal{F}|^{2t}t^{2t})$ critical points across which the algorithmic behaviour (sequence of choices of node splits in Algorithm 9) can change as γ is varied. This implies a bound of $O(t(\log|\mathcal{F}| + \log t))$ on the pseudodimension of the loss function class using Lemma A.3.2. An application of Theorem A.3.1 completes the proof. □

4.3.1 Bayesian decision tree models

Several Bayesian approaches for building a decision tree have been proposed in the literature [59, 60, 169]. The key idea is to specify a prior which induces a posterior distribution and a stochastic search is performed using Metropolis-Hastings algorithms to explore the posterior and find an effective tree. We will summarize the overall approach below and consider the problem of tuning parameters in the prior, which control the accuracy and size of the tree. Unlike most of prior research on data-driven algorithm design which study deterministic algorithms, we will analyze the learnability of parameters in a randomized algorithm.

Let $F = (f_1, \ldots, f_t)$ denote the node functions at the nodes of the decision tree T. The prior $p(F, T)$ is specified using the relationship

$$p(F, T) = p(F|T)p(T).$$

We start with a tree T consisting of a single root node. For any node τ in T, it is split with probability $p_{\text{SPLIT}}(\tau) = \sigma(1 + d_\tau)^{-\phi}$, and if split, the process is repeated for the left and right children. Here d_τ denotes the depth of node τ, and σ, ϕ are hyperparameters. The size of generated tree is capped to some upper bound t. Intuitively, σ controls the size of the tree and ϕ controls its depth. This specifies the prior $p(T)$. The conjugate prior for the node functions $F = (f_1, \ldots, f_t)$ is given by the standard Dirichlet distribution of dimension $c - 1$ (recall c is the number of label classes) with parameter $a = (a_1, \ldots, a_c), a_i > 0$. Under this prior, the label predictions are given by

$$p(y \mid X, T) = \left(\frac{\Gamma(\sum_i a_i)}{\Pi_i \Gamma(a_i)}\right)^t \prod_{j=1}^t \frac{\Pi_i \Gamma(n_{ji} + a_i)}{\Gamma(n_j + \sum_i a_i)},$$

where $n_{ji} = \sum_k \mathbf{I}(y_{jk} = i)$ counts the number of datapoints with label i at node j, $n_j = \sum_i n_{ji}$ and $i = 1, \ldots, c$. a is usually set as the vector $(1, \ldots, 1)$ which corresponds to the uniform Dirichlet prior. Finally the stochastic search of the induced posterior is done using the Metropolis-Hastings (MH) algorithm for simulating a Markov chain [59].

We consider the problem of tuning of prior hyperparameters σ, ϕ, to obtain the best expected performance of the algorithm. To this end, we define $\mathbf{z} = (\mathbf{z}_1, \ldots, \mathbf{z}_{t-1}) \in [0,1]^{t-1}$ as the randomness used in generating the tree T according to $p(T)$. Let $T_{\mathbf{z},\sigma,\phi}$ denote the resulting initial tree. Let $\mathbf{z}'$ denote the remaining randomness used in the prior and the stochastic search. Our goal is to minimize the expected loss

$$\mathbb{E}_{\mathbf{z},\mathbf{z}',\mathcal{D}} L(T_{\mathbf{z},\sigma,\phi}, D),$$

where $\mathcal{D}$ denotes the distribution according to which the data D is sampled. We prove the following generalization guarantee for learning a near-optimal prior for the Bayesian decision tree. So far, we have considered learning decision tree classifiers that classify any given data point into one of finitely many label classes. In the next subsection, we consider an extension of the setting to learning over regression data, for which decision trees are again known as useful interpretable models [45].

Theorem 4.3.5. *Suppose $\sigma, \phi > 0$. For any $\epsilon, \delta > 0$ and any distribution $\mathcal{D}$ over problem instances with n examples, $O(\frac{1}{\epsilon^2}(\log t + \log \frac{1}{\delta}))$ samples drawn from $\mathcal{D}$ are sufficient to ensure that with probability at least $1-\delta$ over the draw of the samples, the parameters $\hat{\sigma}, \hat{\phi}$ learned by ERM over the sample have expected loss that is at most ϵ larger than the expected loss of the best parameters. Here t denotes an upper bound on the size of the decision tree.*

Proof. Fix the dataset D and fix the random coins $\mathbf{z}$ used to generate the tree T. We will use the piecewise loss structure to bound the Rademacher complexity, which would imply uniform convergence guarantees by applying standard learning-theoretic results.

First, we establish a piecewise structure of the dual class loss $\ell_{\mathbf{z}}(\sigma, \phi) = \mathbb{E}_{\mathbf{z}'} L(T_{\mathbf{z},\sigma,\phi}, D)$. Notice that the expected value under the remaining randomization $\mathbf{z}'$ is fixed, once the generated tree $T_{\mathbf{z},\sigma,\phi}$ is fixed. We first give a bound on the number of pieces of distinct trees generated as σ, ϕ are varied. The decision whether a node τ_i is split is governed by whether $p_{\text{SPLIT}}(\tau) = \sigma(1+d_{\tau_i})^{-\phi} > \mathbf{z}_i$. Thus, we get at most $t-1$ 2D curves in σ, ϕ across which the splitting decision may change. The curves are clearly monotonic. We further show that any pair of curves intersect in at most one point. Indeed, if $\sigma(1+d_{\tau_i})^{-\phi} = \mathbf{z}_i$ and $\sigma(1+d_{\tau_j})^{-\phi} = \mathbf{z}_j$, then $\phi' = \log(\mathbf{z}_j/\mathbf{z}_i)/\log\left(\frac{1+d_{\tau_i}}{1+d_{\tau_j}}\right)$ and $\sigma' = \mathbf{z}_i(1+d_{\tau_i})^{\phi'}$ is the unique point provided $\phi' > 0$. Thus the set of all curves intersects in at most $\binom{t-1}{2} < t^2$ points. Since the curves are planar, the number of pieces in the dual loss function (or the number of distinct trees) is also $O(t^2)$. The above argument easily extends to a collection of N problem instances, with a total of at most $O(t^2N^2)$ pieces where distinct trees are generated across the instances.

Let $\rho_1, \ldots, \rho_m$ denote a collection of parameter values, with one parameter from each of the $m = O(N^2t^2)$ pieces induced by all the dual class functions $\ell^i_{\mathbf{z}_i}(\cdot)$ for $i \in [N]$, i.e. across problems in the sample $\{D_1, \ldots, D_N\}$ for some fixed randomizations. Let $\mathcal{F} = \{f_\rho : (D, \mathbf{z}) \mapsto l_T^{D,\mathbf{z}}(\rho) \mid \rho \in \mathbb{R}^+ \times \mathbb{R}^+\}$ be a family of functions on a given sample of instances $S = \{D_i, \mathbf{z}_i\}_{i=1}^N$. Since the function f_ρ is constant on each of the m pieces, we have the empirical Rademacher complexity,

$$\begin{aligned}\hat{R}(\mathcal{F}, S) : &= \frac{1}{N}\mathbb{E}_\sigma\left[\sup_{f_\rho \in \mathcal{F}} \sum_{i=1}^{N} \sigma_i f_\rho(D_i, \mathbf{z}_i)\right] \\ &= \frac{1}{N}\mathbb{E}_\sigma\left[\sup_{j \in [m]} \sum_{i=1}^{N} \sigma_i f_{\rho_j}(D_i, \mathbf{z}_i)\right] \\ &= \frac{1}{N}\mathbb{E}_\sigma\left[\sup_{j \in [m]} \sum_{i=1}^{N} \sigma_i v_{ij}\right],\end{aligned}$$

where $\sigma = (\sigma_1, \ldots, \sigma_m)$ is a tuple of i.i.d. Rademacher random variables, and $v_{ij} := f_{\rho_j}(D_i, \mathbf{z}_i)$. Note that $v^{(j)} := (v_{1j}, \ldots, v_{Nj}) \in [0, H]^N$, and therefore $||v^{(j)}||_2 \leq H\sqrt{N}$, for all $j \in [m]$. An application of Massart's lemma [117] gives

$$\begin{aligned}\hat{R}(\mathcal{F}, S) &= \frac{1}{N}\mathbb{E}_\sigma\left[\sup_{j \in [m]} \sum_{i=1}^{N} \sigma_i v_{ij}\right] \\ &\leq H\sqrt{\frac{2\log m}{N}} \\ &\leq H\sqrt{\frac{4\log Nt}{N}}.\end{aligned}$$

Standard Rademacher complexity bounds [Barlett et al. 2002] now imply the desired sample complexity bound.

□

4.3.2 Splitting regression trees

In the regression problem, we have $\mathcal{Y} = \mathbb{R}$ and the top-down learning algorithm can still be used but with continous splitting criteria. Popular splitting criteria for regression trees include the mean squared error (MSE) and half Poisson deviance (HPD). Let y_l denote the set of labels for data points classified by leaf node l in tree T $\overline{y_l} := \frac{1}{|y_l|}\sum_{y \in y_l} y$ is the mean prediction for node l. MSE is defined as $g_{\text{MSE}}(y_l) := \frac{1}{|y_l|}\sum_{y \in y_l}(y - \overline{y_l})^2$ and HPD as $g_{\text{HPD}}(y_l) := \frac{1}{|y_l|}\sum_{y \in y_l}(y \log \frac{y}{\overline{y_l}} - y + \overline{y_l})$. These are interpolated by the mean Tweedle deviance error with power p given by

$$g_p(y_l) := \frac{2}{|y_l|}\sum_{y \in y_l}\left(\frac{\max\{y, 0\}^{2-p}}{(1-p)(2-p)} - \frac{y\overline{y_l}}{1-p} + \frac{\overline{y_l}^{2-p}}{2-p}\right),$$

where $p = 0$ corresponds to MSE and the limit $p \to 1$ corresponds to HPD. We call this the p-Tweedle splitting criterion, and have the following sample complexity guarantee for tuning p in the multiple instance setting.

Theorem 4.3.6. *Suppose $p \in [0, 1]$. For any $\epsilon, \delta > 0$ and any distribution $\mathcal{D}$ over problem instances with n examples, $O(\frac{1}{\epsilon^2}(t(\log|\mathcal{F}| + n) + \log\frac{1}{\delta}))$ samples drawn from $\mathcal{D}$ are sufficient*

to ensure that with probability at least $1-\delta$ over the draw of the samples, the Tweedle power parameter $\hat{p}$ learned by ERM over the sample is ϵ-optimal.

Proof. The loss is completely determined by the final decision tree $T_{\mathcal{F},p,t}$. It suffices to bound the number of different algorithm behaviors as one varies the hyperparameter p in Algorithm 9. As the tree is grown according to the top-down algorithm, suppose the number of internal nodes is $\tau < t$. For any of $\binom{(\tau+1)|\mathcal{F}|}{2}$ pair of candidates (l_1, f_1) and (l_2, f_2), the preference for which candidate is 'best' and selected for splitting next is governed by the splitting functions $G_p(T_{l_1 \to f_1})$ and $G_p(T_{l_2 \to f_2})$. This preference flips across boundary condition given by $\sum_{l \in \text{leaves}(T_{l_1 \to f_1})} w(l) g_p(\{p_i(l)\}) = \sum_{l \in \text{leaves}(T_{l_2 \to f_2})} w(l) g_p(\{p_i(l)\})$. The expression simplifies and the only remaining terms correspond to deleted leaves l_1, l_2 and newly introduced leaves $l_1^a, l_1^b, l_2^a, l_2^b$, i.e. $\sum_{l \in \{l_1^a, l_1^b, l_2\}} w(l) g_p(\{p_i(l)\}) = \sum_{l \in \{l_2^a, l_2^b, l_1\}} w(l) g_p(\{p_i(l)\})$.

Recall $g_p(\{p_i\})$ gives an equation in $O(|y_l|) = O(n)$ terms. By Rolle's theorem, the number of distinct solutions of the above equation in p is $O(n)$. Thus, the number of critical points of p at which the argmax in Line 3 of Algorithm 9 changes is at most $O(|\mathcal{F}|^2 \tau^2 n)$ and a fixed leaf node is split and labeled by a fixed f for any interval of p induced by these critical points. Over t rounds, this corresponds to at most $O(\Pi_{\tau=1}^{t} |\mathcal{F}|^2 \tau^2 n) = O(|\mathcal{F}|^{2t} t^{2t} n^t)$ critical points across which the algorithmic behaviour (sequence of choices of node splits in Algorithm 9) can change as p is varied. This implies a bound of $O(t(\log|\mathcal{F}| + \log t + n)) = O((\log|\mathcal{F}| + n))$ on the pseudodimension of the loss function class using Lemma A.3.2, since $t \leq n$. An application of Theorem A.3.1 completes the proof. □

Since $t < n$, this indicates that tuning regression parameters typically (for sufficiently small B, c) has a larger sample complexity upper bound.

4.4 Learning to prune

Some leaf nodes in a decision tree learned via the top-down learning algorithm may involve nodes that overfit to a small number of data points. This overfitting problem in decision tree learning is typically resolved by pruning some of the branches and reducing the tree size [45]. The process of growing trees to size t and pruning back to smaller size t' tends to produce more effective decision trees than learning a tree of size t' top-down. We study the mininum cost-complexity pruning algorithm here, which involves a tunable complexity parameter $\tilde{\alpha}$, and establish bounds on the sample complexity of tuning $\tilde{\alpha}$ given access to repeated problem instances from dataset distribution $\mathcal{D}$.

The cost-complexity function for a tree T is given by

$$R(T, D) := L(T, D) + \tilde{\alpha}|\text{leaves}(T)|.$$

More leaf nodes correspond to higher flexibility of the decision tree in partitioning the space into smaller pieces and therefore greater ability to fit the training data. $\tilde{\alpha} \in [0, \infty)$ controls how strongly we penalize this increased complexity of the tree. The mininum cost-complexity pruning algorithm computes a subtree $T_{\tilde{\alpha}}$ of T which minimizes the cost-complexity function. When $\tilde{\alpha} = 0$, this selects T and when $\tilde{\alpha} = \infty$ a single node tree is selected.

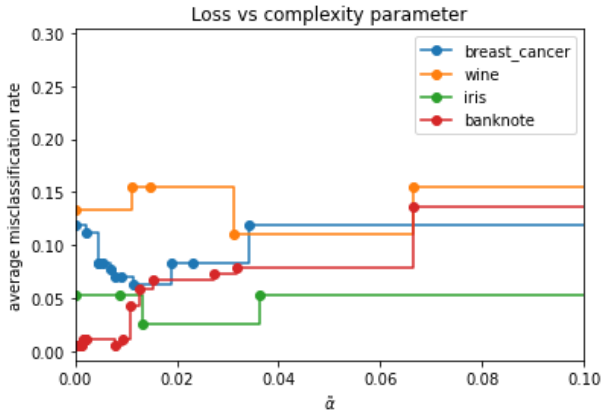


Figure 4.1: The loss of pruned tree as a function of the mininum cost-complexity pruning parameter $\tilde{\alpha}$ is piecewise constant with at most t pieces. The optimal complexity parameter $\tilde{\alpha}$ varies with dataset.

Given a leaf node l of T labeled by $i \in [c]$, the cost-complexity measure is defined to be $R(l, D) = \frac{w(l) - p_i(l)}{w(l)} + \tilde{\alpha}$. Denote by T_t, the branch of tree T rooted at node t and $R(T_t, D) := \sum_{l \in \text{leaves}(T_t)} R(l, D) + \tilde{\alpha}|\text{leaves}(T_t)|$. The mininum cost-complexity pruning algorithm successively deletes weakest links which minimize $\frac{R(t,D) - R(T_t,D)}{|\text{leaves}(T_t)| - 1}$ over internal nodes t of the currently pruned tree.

We have the following result bounding the sample complexity of tuning $\tilde{\alpha}$ from multiple data samples.

Theorem 4.4.1. *Suppose $\tilde{\alpha} \in \mathbb{R}_{\geq 0}$ and t denote the size of the unpruned tree. For any $\epsilon, \delta > 0$ and any distribution $\mathcal{D}$ over problem instances with n examples, $O(\frac{1}{\epsilon^2}(\log t + \log \frac{1}{\delta}))$ samples drawn from $\mathcal{D}$ are sufficient to ensure that with probability at least $1 - \delta$ over the draw of the samples, the mininum cost-complexity pruning parameter learned by ERM over the sample is ϵ-optimal.*

Proof. Fix a dataset D. Then there are critical values of $\tilde{\alpha}$ given by $\tilde{\alpha}_0 = 0 < \tilde{\alpha}_1 < \tilde{\alpha}_2 \cdots < \infty$ such that the optimal pruned tree T_k is fixed for over any interval $[\tilde{\alpha}_k, \tilde{\alpha}_{k+1})$ for $k \geq 0$. Furthermore, the optimal pruned trees form a sequence of nested sub-trees $T_0 = T \supset T_1 \supset \ldots$ ([45], Chapter 10). Thus, the behavior of the min cost-complexity pruning algorithm is identical over at most t intervals, and the loss function is piecewise constant with at most t pieces. The rest of the argument is similar to the proof of Theorem 4.3.3, and we obtain a pseudo-dimension bound of $O(\log t)$ using Lemma A.3.2. An application of Theorem A.3.1 implies the stated sample complexity. □

Minimum cost-complexity pruning [45] can be implemented using a simple dynamic program to find the sequence of trees that minimize $R(T, D)$ for any given fixed $\tilde{\alpha}$, which takes quadratic time to implement in the size of T [42]. Faster pruning approaches are known that directly prune nodes for which the reduction in error or splitting criterion when splitting the node is not statistically significant. This includes Critical Value Pruning [121, 122] and Pessimistic Error

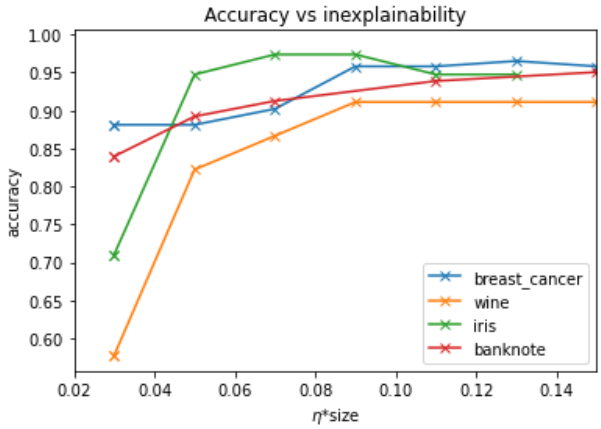


Figure 4.2: Accuracy vs $\eta * |\text{leaves}(T)|$ as the pruning parameter $\tilde{\alpha}$ is varied, for $\eta = 0.01$.

Pruning [140]. Principled statistical learning guarantees are known for the latter [116], and here we will consider the problem of tuning the confidence parameter in pessimistic pruning, which we describe below.

Suppose $\mathcal{X} \subseteq \mathbb{R}^a$, i.e. each data point consists of a real features or attributes. For any internal node h of T, if e_h denotes the fraction of data points that are misclassified among the n_h data points that are classified via the sub-tree rooted at h, and e_l denotes the fraction of misclassified data points if h is replaced by a leaf node, then the pessimistic pruning test of [116] is given by

$$e_l \leq e_h + c_1 \sqrt{\frac{t_h \log a + c_2}{n_h}},$$

where c_1 and c_2 are parameters, and t_h denotes the size of the sub-tree rooted at h. We consider the problem of tuning c_1, c_2 given repeated data samples, and bound the sample complexity of tuning in the following theorem.

Theorem 4.4.2. *Suppose $c_1, c_2 \in \mathbb{R}_{\geq 0}$ and t denote the size of the unpruned tree. For any $\epsilon, \delta > 0$ and any distribution $\mathcal{D}$ over problem instances with n examples, $O(\frac{1}{\epsilon^2}(\log t + \log \frac{1}{\delta}))$ samples drawn from $\mathcal{D}$ are sufficient to ensure that with probability at least $1 - \delta$ over the draw of the samples, the pessimistic pruning parameters learned by ERM over the sample is ϵ-optimal.*

Proof. For a fixed dataset D, the c_1, c_2 parameter space can be partitioned by at most t algebraic curves of degree 3 that determine the result of the pessimistic pruning test. We use a general result on the pseudodimension bound in data-driven algorithm design due to [35] when the loss can be computed by evaluating rational expressions to obtain a $O(\log t)$ on the pseudodimension. The result is stated below for convenience.

In this theorem, our above arguments show that there is a GJ algorithm, i.e. an algorithm which only computes and compares rational (ratios of polynomials) functions of its inputs, for computing the loss function. Here the number of real parameters $n = 2$, the maximum degree of any computed expression is $\Delta = 3$ and the total number of distinct predicates that need to be evaluated to compute the loss for any value of the parameters is $\Gamma = t$. Plugging into Theorem

A.3.3 yields a bound of $O(\log t)$ on the pseudo-dimension, and the result follows from Theorem A.3.1. □

We have studied parameter tuning in two distinct parameterized approaches for decision tree pruning. However, several other pruning methods are known in the literature [74, 75], and it is an interesting direction for future research to design approaches to select the best method based on data. We conclude this section with a remark about another interesting future direction, namely extending our results to tree ensembles.

Remark 3 (Extension to tree ensembles)**.** *Extension of our approaches to tree ensembles is an interesting question, although this comes at the expense of making the model less interpretable. We still need to choose splitting and pruning methods used in building the individual trees. If we learn a uniform splitting criterion for all trees, our sample complexity arguments are straightforward to extend to this case and would imply an additional $O(n_t)$ factor in the sample complexity, where n_t is the number of trees in the random forest (in the case of pruning, our arguments would imply an $O(\log n_t)$ term). There are interesting further questions here, including learning a combination of splitting/pruning criteria across different trees and tuning the number of trees nt as a hyperparameter (which impacts both accuracy and interpretability).*

4.5 Optimizing the Interpretability versus Accuracy trade-off

Decision trees are often regarded as one of the preferred models when the model predictions need to be interpretable. Complex or large decision trees can however not only overfit the data but also hamper model interpretability. So far we have considered parameter tuning when building or pruning the decision tree with the goal of optimizing accuracy on unseen "test" datasets on which the decision tree is built using the learned hyperparameters. We will consider a modified objective here which incorporates model complexity in the test objective. That is, we seek to find hyperparameters $\alpha, \beta, \tilde{\alpha}$ based on the training samples, so that on a random $D \sim \mathcal{D}$, the expected loss

$$L_\eta := \mathbb{E}_{D\sim\mathcal{D}} L(T, D) + \eta|\text{leaves}(T)|$$

is minimized, where $\eta \geq 0$ is the complexity coefficient. This objective has been studied in a recent line of work which designs techniques for provably optimal decision trees with high interpretability [96, 111]. Note that, while the objective is similar to min cost-complexity pruning, there the regularization term $\tilde{\alpha}|\text{leaves}(T)|$ is added to the training objective to get the best generalization accuracy on test data. In contrast, we add the regularization term to the test objective itself and η here is a fixed parameter that governs the balance between accuracy and interpetability that the learner aims to strike.

Our approach here is to combine tunable splitting and pruning to optimize the accuracy-interpetability trade-off. We set (α, β)-Tsallis entropy as the splitting criterion and min cost-complexity pruning with parameter $\tilde{\alpha}$ as the pruning algorithm. We show the following upper bound on the sample complexity when simultaneously learning to split and prune.

Dataset	Best (α^*, β^*)	Acc(α^*, β^*)	Acc(Gini)	Acc(Entropy)	Acc(KM96)
Iris	(0.5,1)	96.00 ± 1.85	92.99 ± 1.53	93.33 ± 1.07	94.67 ± 2.70
Banknote	(2.45,2)	98.32 ± 0.52	97.01 ± 0.59	97.30 ± 1.62	97.00 ± 1.79
Breast cancer	$(0.5, 3)$	94.69 ± 0.77	92.92 ± 1.29	93.01 ± 1.05	93.27 ± 1.16
Wine	(2.15,6)	96.57 ± 1.88	89.14 ± 3.18	92.57 ± 2.38	93.71 ± 2.26

Table 4.1: A comparison of the performance of different splitting criteria. The first column indicates the best (α, β) parameters for each dataset over the grid considered in Figure 4.3. Acc denotes test accuracy along with a 95% confidence interval.

Theorem 4.5.1. *Suppose $\alpha > 0, \beta \in [B], \tilde{\alpha} \geq 0$. For any $\epsilon, \delta > 0$ and any distribution $\mathcal{D}$ over problem instances with n examples, $O(\frac{1}{\epsilon^2}(t(\log|\mathcal{F}| + \log t + c\log(B+c)) + \log\frac{1}{\delta}))$ samples drawn from $\mathcal{D}$ are sufficient to ensure that with probability at least $1-\delta$ over the draw of the samples, the parameters learned by ERM for L_η are ϵ-optimal.*

Proof. As argued in the proof of Theorems 4.3.3, we have an upper bound of $O(B|\mathcal{F}|^{2t}t^{2t}(B+c)^{ct})$ on the number of distinct algorithmic behavior of the top-down learning algorithm in growing a tree of size t as the parameters α, β are varied. Further, as argued in the proof of Theorem 4.4.1, for each of these learned trees, there are at most t distinct pruned trees as $\tilde{\alpha}$ is varied. Overall, this corresponds to $O(B|\mathcal{F}|^{2t}t^{2t+1}(B+c)^{ct})$ distinct behaviors, which implies the claimed sample complexity bound using standard tools from learning theory and data-driven algorithm design (Lemma A.3.2, Theorem A.3.1). □

4.6 Experiments

We examine the significance of the novel splitting techniques and the importance of designing data-driven decision tree learning algorithms via hyperparameter tuning for various benchmark datasets. We only perform small-scale simulations that can be run on a personal computer and include code in the supplementary material for reproducibility. The datasets used are from the UCI repository, are publicly available and are briefly described below.

Iris [79] consists of three classes of the iris plant and four real-valued attributes. A total of 150 instances, 50 per class. *Wine* [109] has three classes of wines, 13 real attributes and 178 data points in all. *Breast cancer (Wisconsin diagnostic)* contains 569 instances, with 30 features, and two classes, malignant and benign [168]. The *Banknote Authentication* dataset [112] also involves binary classification and has 1372 data points and five real attributes. These datasets are selected to capture a variety of attribute sizes and number of data points.

We first study the effect of choice of (α, β) parameters in the Tsallis entropy based splitting criterion. For each dataset, we perform 5-fold cross validation for a large grid of parameters depicted in Figure 4.3 and measure the accuracy on held out test set consisting of 20% of the datapoints (i.e. training datasets are just random subsets of the 80% of the dataset used for learning the parameters). We implement a slightly more sophisticated variant of Algorithm 9 which grows the tree to maximum depth of 5 (as opposed to a fixed size t). We do not use any pruning here. There is a remarkable difference in the optimal parameter settings for different datasets.

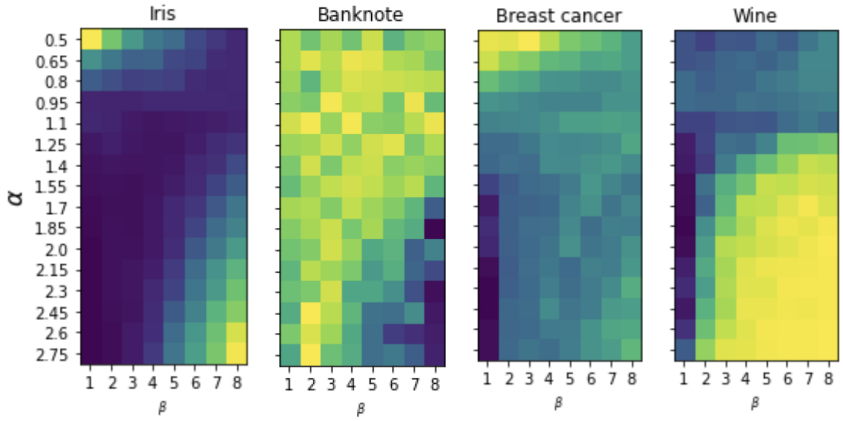


Figure 4.3: Average test accuracy (proportional to brightness, yellow is highest) of (α, β)-Tsallis entropy based splitting criterion as the parameters are varied, across datasets. We observe that different parameter settings work best for each dataset, highlighting the need to learn data-specific values.

Moreover, we note in Table 4.1, that carefully chosen values of (α, β) significantly outperform standard heuristics like Gini impurity or entropy based splitting, or even specialized heuristics like [98] for which worst-case error guarantees (assuming weak learning) are known. This further underlines the significance of data-driven algorithm design for decision tree learning.

We further study the impact of tuning the complexity paramter $\tilde{\alpha}$ in the minimum cost-complexity pruning algorithm. The test error varies with $\tilde{\alpha}$ in a data dependent way and different data could have different optimal parameter as depicted in Figure 4.1. We use Gini impurity as the splitting criterion. Furthermore, we observe that on a single instance, the average test error is a piecewise constant function with at most t pieces which motivates the sample complexity bound in Theorem 4.4.1.

We also examine the interpetability-accuracy trade-off as given by our regularized objective with complexity coefficient η. In Figure 4.2, we plot the interpetability-accuracy frontier as the pruning parameter $\tilde{\alpha}$ is varied. Here we fix the splitting criterion as the Gini impurity. For a given dataset, this frontier can be pushed by a careful choice of the splitting criterion (Theorem 4.5.1).

4.6.1 Interpretability-accuracy frontier

We study the effect of varying α (for fixed $\beta = 1$) and β (for fixed $\alpha = 1.5$) on the Iinterpretability-accuracy trade-off. We fix $\eta = 0.01$, and obtain the plot by varying the amount of pruning by changing the complexity parameter $\tilde{\alpha}$ in min-cost complexity pruning.

We perform this study for Iris and Wine datasets in Figure 4.4. We observe that for a given accuracy, the best (smallest) explanation (size) could be obtained for different different splitting criteria (corresponding to setting of α, β). In particular, different criteria can dominate in different regimes of size and η. Therefore, simultaneously tuning splitting criterion and pruning as in Theorem 4.5.1 is well-motivated.

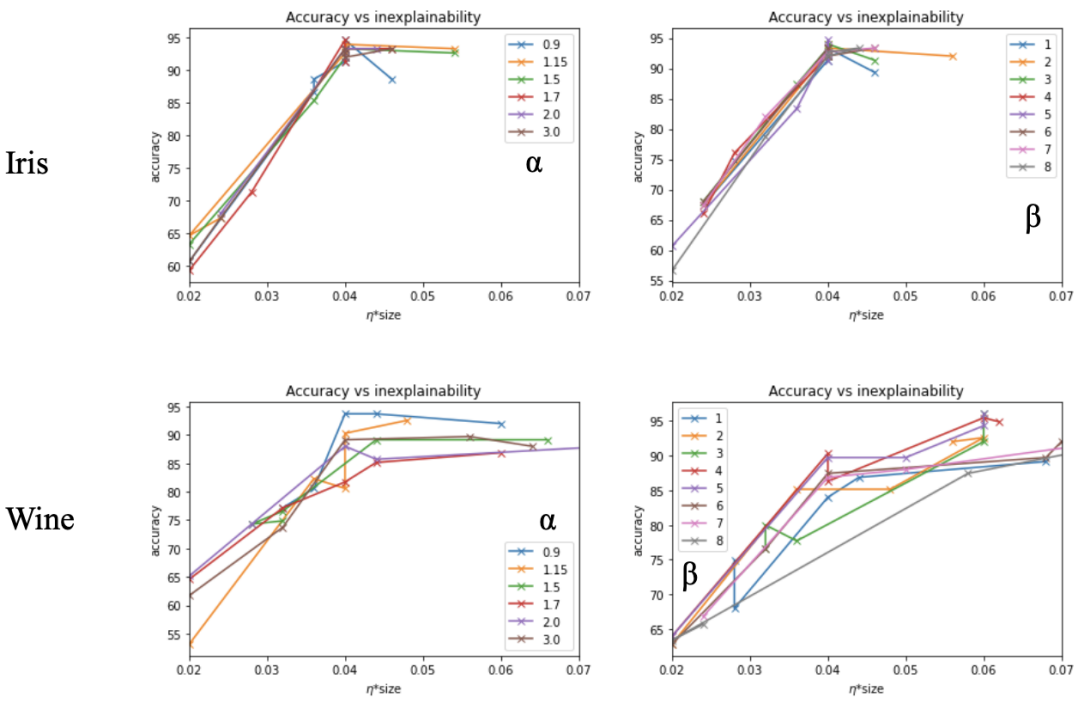


Figure 4.4: Accuracy-interpretability frontier for different α or different β, as the pruning parameter $\tilde{\alpha}$ is varied.

4.6.2 (α, β)-Tsallis entropy

We consider several additional datasets from the UCI repository and examine the best setting of (α, β) in the splitting criterion. The results are depicted in Figure 4.5 and summarized below.

Seeds [55] involves 3 classes of wheat, and has 210 instances with 7 attributes each. The splitting criterion proposed by [98] seems to work best here. Note that the original work only studied binary classification, and seeds involves three label classes and therefore our experiment involves a natural generalization of [98] to $g^{\text{TSALLIS}}_{\frac{1}{2},2}(\cdot)$.

Cryotherapy [103] has 90 instances with 7 real or integral attributes and contains the binary label of whether a wart was suffessfully treated using cryotherapy. Here $\alpha = 0.5$ with $\beta = 4$ is one of the best settings, indicating usefulness of varying the β exponent in the KM96 criterion.

Glass identification [81] involves classification into six types of glass defined in terms of their oxide content. There are 214 instances with 9 real-valued features. Interestingly, the best performance is observed when both α and β are larger than their typical values in popular criteria. For example, $(\alpha, \beta) = 2.45, 6$ works well here.

Algerian forest fires involves binary classification with 12 attributes and 244 instances. Gini entropy by itself does poorly, but augmented with the β-parameter the performance improves significantly and beats other candidate approaches for $\beta = 8$.

Human activity detection using smartphones [143] is a 6-way classification dataset consisting of smartphone accelerometer and gyroscope readings corresponding to different activities, with 10299 instances with 561 features. Smaller values of α work better on this dataset, and the

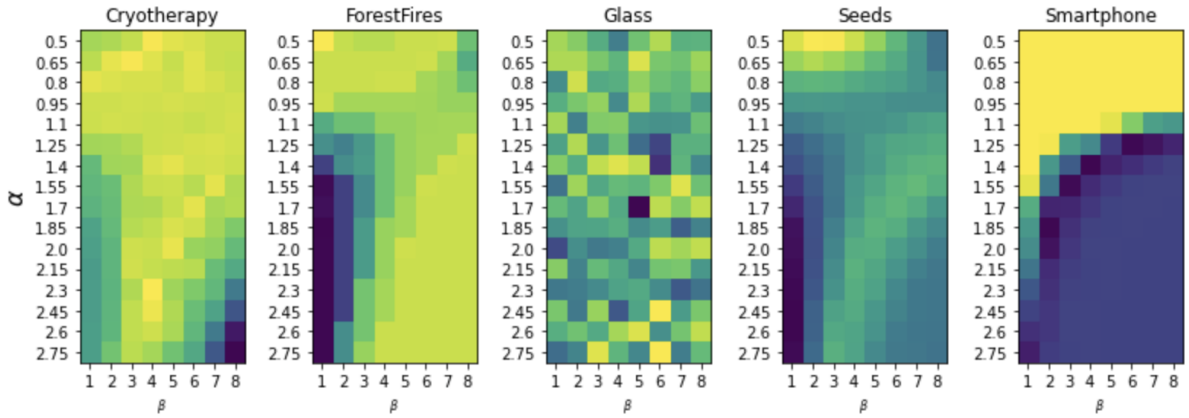


Figure 4.5: Average test accuracy (proportional to brightness) of (α, β)-Tsallis entropy based splitting criterion across additional datasets.

dependence on β is weaker.

4.6.3 Pruning experiments

We will examine the effectiveness of learning to prune by comparing the accuracy of pruning using the learned parameter $\tilde{\alpha}$ in the mininum cost-complexity pruning algorithm family with other baseline methods studied in the literature. Prior literature on empirical studies on pruning methods has shown that different pruning methods can work best for different datasets [74, 122]. This indicates that a practitioner should try out several pruning methods in order to obtain the best result for given domain-specific data. Here we will show that a well-tuned pruning from a single algorithm family can be competitive, and allows us to automate this process of manual selection of the pruning algorithm.

We perform our experiments on benchmark datasets from the UCI repository, including Iris, Wine, Breast Cancer and Digits datasets. We split the datasets into train-test sets, using 80% instances for training and 20% for testing. In each case, we build the tree using entropy as the splitting criterion. We compare the mean accuracy on the test sets over 50 different splits for the following methods:

- Unpruned, that is no pruning method is used.
- $\tilde{\alpha}^*$ in MCCP. Min-cost complexity pruning using the best parameter $\tilde{\alpha}^*$ for the dataset.
- REP, Reduced error pruning method of [140].
- TDP, Top-down pessimistic pruning method of [139].
- BUP, Bottom-up pessimistic pruning method of [116].

We report our findings in Table 4.2. We observe that the learned pruning method has a better mean test accuracy than other baseline methods on the tested datasets.

Dataset	Acc(Unpruned)	Acc($\tilde{\alpha}^*$) in MCCP	Acc(REP)	Acc(TDP)	Acc(BUP)
Iris	80.03	97.37	96.67	90.00	93.33
Digits	84.44	89.42	86.67	83.61	88.89
Breast cancer	87.72	93.71	92.98	91.23	92.11
Wine	80.56	94.44	91.67	88.89	86.11

Table 4.2: A comparison of the mean test accuracy of decision trees obtained using different pruning methods.

4.7 Conclusion

We consider the problem of automatically designing decision tree learning algorithms by data-driven selection of hyperparameters. Previous extensive research has observed that different ways to split or prune nodes when building a decision tree work best for data coming from different domain. We present a novel splitting criterion called (α, β)-Tsallis entropy which interpolates popular previously known methods into a rich infinite class of algorithms. We consider the setting where we have repeated access to data from the same domain and provide formal bounds on the sample complexity of tuning the hyperparameters for the ERM principle. We extend our study to learning regression trees, selecting pruning parameters, and optimizing over the interpetability-accuracy trade-off. Empirical simulations validate our theoretical study and highlight the significance and usefulness of learning decision tree algorithms.

Our work presents several directions for future research. While our results provide guarantees on sample efficiency, the problem of computationally efficient optimization of the sample accuracy is left open. Another direction for future research is designing and analyzing a potentially more powerful algorithm family for pruning, and extending our results to tree ensembles. We also remark that we focus on upper bounds on sample complexity, and providing corresponding lower bounds is an interesting avenue for further research.

The results in this chapter are based on joint work with Nina Balcan, which appeared as an Oral presentation at UAI 2024 and was awarded the Outstanding Student Paper Award.

Chapter 5

Subsidy design in games

5.1 Introduction

Critical infrastructure maintenance typically involves joint responsibility shared among multiple stakeholders, and failure of coordination can lead to disastrous consequences. For example, the Internet backbone consists of many networks with different owners, connected together in a certain way. Smaller networks themselves consist of independent nodes connected using a specific topology. In 2013, an outage at Google caused a 40% drop in the worldwide web traffic, and in 2021, Facebook was down for several hours. With an increasingly connected physical and digital world, it is a major challenge to coordinate large-scale systems consisting of several disjointly owned components. As a result, one crucially needs a *central planner*—who has the ability to allocate shared resources to avoid major catastrophic failures—to ensure smooth operation of the overall system. Ideally, the central agent would ensure a judicious use of the common resources which are to be allocated to appropriate stakeholders to incentivize them to do their part. Identifying the optimal resource allocations can be hard, but very often the central agent manages multiple similar systems or has access to relevant historical data. Could one take advantage of this data availability to improve the allocation?

Coordination in infrastructure projects poses multiple challenges when different pieces are owned by different agents. In systems requiring all of multiple components to simultaneously work, the failure of any single component can bring the entire system down. As a countermeasure, critical systems often have in place some amount of redundancy in terms of the components needed for the system to function. But this could introduce volunteer's dilemma, where the agents with knowledge of their redundancy can choose to not invest the due maintenance cost in hope that a different agent would put in the cost instead. Furthermore, selfish agents could choose to ignore or deliberately not collect important information about their own component, if public knowledge of that information comes at increased personal cost to them. With these various strategic aspects at play, a good amount of literature is devoted to identifying and circumventing such issues as individual agents [38, 47].

A common lesson from major failures and the primary recommendation for avoiding large-scale failure is ensuring an active role by the top management in coordination and resource allocation [86, 136]. For example, in public-private partnership infrastructure projects a cen-

tral government agency typically decides the allotment of common resources among the various project stakeholders. Resources are typically scarce, and therefore a judicious allocation is crucial to ensure that all the critical components essential for the project function properly. The challenge can be particularly severe in systems with a large number of components.

5.2 Formal notation and setup

Let $G = \langle N, (S_i), (\text{cost}_i) \rangle$ denote a game, where N is a set of n agents (or players), S_i is the finite action space of agent $i \in N$, and cost_i is the cost function of agent i. The joint action space of the agents is $S = S_1 \times \cdots \times S_n$. Given joint action $s = (s_1, \ldots, s_n) \in S$ let s_{-i} denote the actions of all agents except agent i, i.e. $s_{-i} = (s_1, \ldots, s_{i-1}, s_{i+1}, \ldots, s_n)$. The cost function $\text{cost}_i : S \to \mathbb{R}_{\geq 0}$ of agent i (which the agent seeks to minimize) is a function of the joint action $s \in S$. The *social cost* function of the game is the sum of cost functions of all the agents in the game, $\text{cost} = \sum_{i=1}^{n} \text{cost}_i$. The optimal social cost is $\mathsf{OPT} = \min_{s \in S} \text{cost}(s)$. Given a joint action s, the best response of agent i is the set of actions $\text{BR}_i(s)$ that minimizes its cost given s_{-i} the actions of other agents, i.e., $\text{BR}_i(s_{-i}) = \operatorname{argmin}_{a \in S_i} \text{cost}_i(a, s_{-i})$. A joint action $s \in S$ is a *(pure) Nash equilibrium* (or NE) if no agent can benefit from unilaterally deviating to another action, in other words every agent is simultaneously playing a best response action in s, i.e., $s_i \in \text{BR}_i(s_{-i})$ for every $i \in N$. A Nash equilibrium is said to be *global* if it also minimizes the *social cost* among all Nash equilibria. We say a Nash equilibrium is a *local* equilibrium if it is not global.

Component maintenance game [110]. We will consider a specific game defined as follows. Each agent is associated with a component c_i which has a binary state $x_i \in \{0, 1\}$ where $x_i = 0$ corresponds to a broken component and $x_i = 1$ corresponds to a functioning component. The action space of each agent is also binary, $S_i = \{0, 1\}$, where action $s_i = 1$ indicates that the agent repaired the component (denoted RE), and $s_i = 0$ denotes that the agent did nothing (denoted DN). The state x_i of c_i is updated after action s_i as $x_i' = \max\{x_i, s_i\}$. This corresponds to 'perfect repair', i.e. if an agent picks the RE action, their component is guaranteed to work, and otherwise it stays as is. For a tuple of actions s, we will denote the updated state by $\mathbf{x}'(s)$, or simply $\mathbf{x}'$ when s is evident from context. The state u of the system is a fixed binary function of the component states, $u = \phi(\mathbf{x})$, where $\mathbf{x} = (x_1, \ldots, x_n)$ and $\phi : \{0, 1\}^n \to \{0, 1\}$[1]. $u = 0$ denotes a failure of the system. Let $u' = \phi(\mathbf{x}')$ denote the state of the system after the agents' actions. The cost for agent i is given by, $\text{cost}_i = C_i s_i + 1 - u'$ for repair cost $C_i \in \mathbb{R}$. The actions depend on the *belief* about the state $\mathbf{x}$ of the components, which we model by a distribution θ over $\{0, 1\}^n$. We will assume here that θ is a product distribution, and all agents share the same common belief about the state of the components. The system failure probability under this belief is $P_\phi(\theta) = 1 - \mathbb{E}_{\mathbf{x} \sim \theta}[\phi(\mathbf{x}')]$, and the expected cost of action s_i to agent i, given other agents' actions are s_{-i}, is $l_i(s_i, s_{-i}, \theta) = \mathbb{E}_{\mathbf{x} \sim \theta}[\text{cost}_i] = C_i s_i + 1 - \mathbb{E}_{\mathbf{x} \sim \theta}[\phi(\mathbf{x}')]$. Similarly, the expected social cost is defined as $l(s, \theta) = \mathbb{E}_{\mathbf{x} \sim \theta}[\text{cost}] = \sum_i l_i(s_i, s_{-i}, \theta)$ for $s = (s_1, \ldots, s_n)$.

We now imagine that each agent j has the ability, for free, to inspect their own component to determine its state. The catch, however, is that this state is revealed to all agents. The revelation of the state would result in an updated common belief $\tilde{\theta}$ and therefore an updated cost function

[1]The component maintenance game intuitively corresponds to monotone boolean functions ϕ, but our results easily extend to general boolean functions.

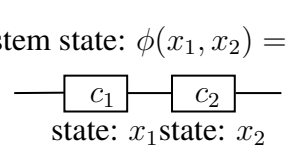


Figure 5.1: A two component series system.

Agent $1\backslash 2$	DN ($s_2 = 0$)	RE ($s_2 = 1$)
DN ($s_1 = 0$)	$0.75, 0.75$	$0.5, 0.8$
RE ($s_1 = 1$)	$0.8, 0.5$	$0.3, 0.3$

Table 5.1: Cost matrix for a 2-series system.

for all agents. As a result, the set of Nash equilibria can now change and the agent may suffer higher personal cost in the new equilibria, inducing the agent to avoid inspecting their own component for selfish reasons. We will use the terminology *prior* (respectively *posterior*) game and equilibria to refer to the game before (respectively after) the inspection of any fixed component c_j. We formalize the setup and define Value of Information metric to capture this behavior.

Component inspection game, and Value of Information [110]. Suppose the inspection of component c_j reveals state $y_j \in \{0, 1\}$. We will assume perfect inspection, i.e. $y_j = x_j$. Denote posterior belief after the inspection of component c_j by $\tilde{\theta}^{j,y_j}$. If agent i switches action from s_i to $\tilde{s}_i^{j,y_j}$ after the inspection (and other agents switch from s_{-i} to $\tilde{s}_{-i}^{j,y_j}$), the value of information about inspection of c_j for agent i and posterior y_j is given by $\text{VoI}_{i,j}(s_i, s_{-i}, \tilde{s}_i^{j,y_j}, \tilde{s}_{-i}^{j,y_j}) := l_i(s_i, s_{-i}, \theta) - l_i(\tilde{s}_i^{j,y_j}, \tilde{s}_{-i}^{j,y_j}, \tilde{\theta}^{j,y_j})$. The expected value of information is given by $\text{VoI}_{i,j}(s_i, s_{-i}, \tilde{s}_i^{j,*}, \tilde{s}_{-i}^{j,*}) := l_i(s_i, s_{-i}, \theta) - \mathbb{E}_{y_j} l_i(\tilde{s}_i^{j,y_j}, \tilde{s}_{-i}^{j,y_j}, \tilde{\theta}^{j,y_j})$, where $\tilde{s}^{j,*}$ is the collection of states $\tilde{s}^{j,0}, \tilde{s}^{j,1}$. Typically we will assume that the joint actions s and $\tilde{s}^{j,y_j}$ are Nash equilibria. We want the value of information to be non-negative for each agent i, when inspecting any component j. This is to not have any undesirable information avoidance behavior among the agents, where agents may choose to ignore freely available information (about the inspected component state) for selfish reasons (to reduce personal cost, for example by choosing to not repair their broken component), which could lead to sub-optimal social cost or undesirable system state.

5.3 Subsidy to reduce cost and tackle information avoidance

We first present a motivating example where subsidy for component repair costs can help improve social cost by steering the system to a better equilibrium.

Example 1 (A 2-series system)**.** *Consider the two component series system depicted in Fig 5.1. Suppose that the components are independent and the failure probabilities for components c_1 and c_2 are both 0.5, and the repair cost is $C_1 = C_2 = 0.3$. Then the cost matrix for the component maintenance game is given in Table 5.1. Notice that both DN-DN and RE-RE are Nash equilibria for the game, but RE-RE has a smaller social cost. If the central authority provides a subsidy of $0.05 + \epsilon$ for the repair action, for any $\epsilon > 0$, then it would incentivize the agents to switch their actions from DN to RE, and the only Nash equilibrium is RE-RE. Note that the cost reduction in RE-RE (0.3 to $0.25 - \epsilon$ for each agent) equals the subsidy provided by the central agent in this case, and the social cost + subsidy for RE-RE is preserved, while ruling out the sub-optimal equilibrium DN-DN.*

Formally, we have the following definition.

Definition 16 (Subsidy scheme)**.** *A subsidy scheme $\mathbb{S}$ is defined as a set of functions $subs_i : S_i \to \mathbb{R}_{\geq 0}$ where $subs_i(s_i)$ gives the subsidy offered by the central agent to agent i given agent action s_i. In a subsidized game using scheme $\mathbb{S}$, the cost of agent i is given by the difference function $cost_i^{\mathbb{S}} := cost_i - subs_i$, and total subsidy provided for joint action $s \in S$ is $subs(s) = \sum_i subs_i(s_i)$. In the component maintenance game, we consider subsidy schemes 'for repair', i.e. $subs_i(0, s_{-i}) = 0$ for all agents i.*

In the component maintenance game, we will consider subsidy schemes that incentivize repair, i.e. our subsidy functions subs_i will be of the form $s_i^* s_i$ for some constant s_i^*, where s_i denotes the action of agent i (recall $s_i = 1$ for repair and $s_i = 0$ for doing nothing). We also say that a subsidy scheme is *uniform* if the scheme is identical for all agents and actions, i.e. $s_i^* = c_{\text{subs}}$ for all agents i for some constant $c_{\text{subs}} \geq 0$. We consider two types of subsidies which a central agent, whose goal is to altruistically maximize social welfare, can offer to reduce the repair cost of certain components.

Conditional vs. unconditional subsidies. The central agent may offer an *unconditional* subsidy which effectively reduces the cost of repair for the components, or may be *conditional on inspection* in order to encourage agents to inspect their components, even when the information about the state of an agent's component results is something the agent might want to avoid (in the absence of subsidy). Note that the subsidy still is a payment to reduce the repair cost, and not a payment for inspection. Formally, if a component inspection game involving inspection of some component j, a general subsidy consists of three functions for each agent given by subs_i, subs_i^1, subs_i^0, corresponding to prior, posterior with component j intact, and posterior with component j damaged respectively. For simplicity, we will say that the agent provides subsidy conditional on $y_j = k$ to denote that subs_i^k is the only non-zero function in the conditional scheme, and conditional on inspection to denote that subs_i is a zero function and $\text{subs}_i^1 = \text{subs}_i^0$.

Price of Anarchy measures the reduction in system efficiency (social cost) due to selfish behavior of the agents [130, 146]. We define Price of Anarchy (PoA) in the presence of subsidy along the lines of [49] as the ratio of the sum of total social cost and subsidy in the worst case equilibrium, to the optimal social cost.

Definition 17 (Price of Anarchy under subsidy)**.** *Let $\mathbb{S} = \{subs_i\}$ denote the subsidy scheme. Let $\mathcal{S}_{NE}(\mathbb{S}) \subseteq S$ denote the subset of states corresponding to Nash equilibria when the cost for agent i is $cost_i - subs_i$. Suppose $\mathsf{OPT} \neq 0$. Then the Price of Anarchy under subsidy $\mathbb{S}$ is given by*

$$PoA(\mathbb{S}) = \frac{\max_{s \in \mathcal{S}_{NE}(\mathbb{S})} cost^{\mathbb{S}}(s) + subs(s)}{\mathsf{OPT}}.$$

We also define a related metric for studying the effectiveness of subsidy scheme $\mathbb{S}$,

$$\widetilde{PoA}(\mathbb{S}) = \frac{\max_{s \in \mathcal{S}_{NE}(\mathbb{S})} cost^{\mathbb{S}}(s) + subs(s)}{\min_{s \in \mathcal{S}_{NE}} cost(s)},$$

where $\mathcal{S}_{NE}$ denotes the set of Nash equilibria in the component maintenance game (in the absence of any subsidy), provided $\min_{s \in \mathcal{S}_{NE}} cost(s) \neq 0$.

By setting zero subsidies (i.e. $\text{subs}_i(s) = 0$ for each i, s) we recover the usual Price of Anarchy, PoA [130]. Note that finding the subsidy scheme that optimizes $\text{PoA}(\mathbb{S})$ or $\widetilde{\text{PoA}}(\mathbb{S})$ corresponds

to the same optimization problem. In some games, it will be easier to show absolute bounds on $\widetilde{\text{PoA}}(\mathbb{S})$. Note that $\widetilde{\text{PoA}}(\mathbb{S}) = \text{PoA}(\mathbb{S})/\text{PoS}$, where PoS is the usual Price of Stability in the unsubsidized game [130].

Addition of subsidy effectively changes the set of states that correspond to a Nash equilibrium. The goal of a central agent is to ensure suboptimal (local) Nash equilibria are not included in the set $\mathcal{S}_{NE}$ when the subsidy is applied. The subsidy provided by the central agent could for example come from taxes collected from the agents, and therefore it makes sense to add the total subsidy provided by the central agent to the social cost in the definition of $\text{PoA}(\mathbb{S})$.

The central agent designing the subsidy scheme can have several different objectives for the scheme. In this work, we consider the following three objectives:

1. *PoA*$(\mathbb{S})$ *is minimized*. This means that the social cost in the worst-case Nash equilibrium after the subsidy is provided (with the total subsidy added) is not much worse than the social cost of the best joint action (or the best Nash equilibrium if we minimize $\widetilde{\text{PoA}}(\mathbb{S})$) when no subsidy is offered. As the result, the harmful effects of selfish behavior and lack of coordination among the agents is minimized.
2. *The system is guaranteed to work in any Nash equilibrium*. This is desirable if the central agent is keen on guaranteeing system functionality and willing to provide the needed subsidy for it. This can differ from minimizing the price of anarchy as the optimal social cost could correspond to doing nothing and letting the system stay broken, for example when the repair costs are too large.
3. *The value of information for each agent is non-negative* when a single component is inspected. This corresponds to minimizing the tendency of agents to avoid seeking freely available information about the state of their component to avoid an increase in their personal cost, potentially at the expense of the social cost.

Depending on the game, the appropriate subsidy scheme could vary depending on the objective of the central agent. We will illustrate this for a two agent series game in Section 5.3.1, where we will obtain the optimal subsidy schemes for each objective. Similarly, optimal schemes can be derived for the two-agent parallel game, the interested reader is directed to Appendix B.2. However, for general n-agent games, we show in Section 5.3.2 that computing the optimal total subsidy is NP hard, under the above objectives.

Definition 18 (Price of Information Avoidance in the component inspection game)**.** *Let* $\mathbb{S} = \{subs_i\}$ *denote the subsidy scheme. Consider the component inspection game for inspection of component* j*. Let* $\mathcal{S}_{NE}(\mathbb{S}), \mathcal{S}^0_{NE}(\mathbb{S}), \mathcal{S}^1_{NE}(\mathbb{S}) \subseteq S$ *denote the subset of states corresponding to Nash equilibria when the cost for agent* i *is* $cost_i - subs_i$ *for prior, posteriors* $y_j = 0$ *and* $y_j = 1$ *respectively. Let* $VoI_j(\mathbb{S}) = \min_{i, s\in\mathcal{S}_{NE}(\mathbb{S}), s'\in\mathcal{S}^0_{NE}(\mathbb{S})\cup\mathcal{S}^1_{NE}(\mathbb{S})} VoI_{i,j}(s_i, s'_i)$ *denote the least value of information for any agent* i *for equilibria under* $\mathbb{S}$*. Then the Price of Information Avoidance is given by*

$$PoIA(j) = \frac{\min_{\mathbb{S}|VoI_j(\mathbb{S})\geq 0} \max_{s\in\mathcal{S}_{NE}(\mathbb{S})} cost(s)}{\min_{\mathbb{S}} \max_{s\in\mathcal{S}_{NE}(\mathbb{S})} cost(s)} = \frac{\min_{\mathbb{S}|VoI_j(\mathbb{S})\geq 0} PoA(\mathbb{S})}{\min_{\mathbb{S}} PoA(\mathbb{S})}.$$

Recall that Price of Anarchy captures the effect of selfish behavior on social cost, relative to best centralized (co-ordinated) action under unselfish behavior, and is minimized (equals 1)

when selfish behavior does not impact social cost. Similarly, Price of Information Avoidance corresponds to the effect of information avoidance, relative to the best centralized action when agents do not avoid information for selfish reasons. Also, it equals 1 if there is never any negative value of information under any subsidizing policy. Otherwise it is at least 1, and captures the sacrifice in social cost to avoid negative VoI.

5.3.1 Optimal subsidy in two-agent series game

Suppose we have two agents $N = \{1, 2\}$ with components c_1, c_2 connected in series. Let p_1, p_2 denote the (prior) probability that the components c_1, c_2 will work (respectively). We will use the notation $\overline{p} := 1 - p$ for conciseness. This corresponds to the prior game in Table 5.2, where DN denotes 'do nothing' ($s_i = 0$) and RE denotes 'repair' ($s_i = 1$). We will now consider the above three objectives. For each objective, we will note that subsidy helps and we will obtain the optimal subsidy in the two-agent series game.

Minimizing the price of anarchy with subsidy.

To demonstrate the significance of subsidy for reducing the social cost in the two-agent series games, we will first show a lower bound on the Price of Anarchy in the absence of subsidy. The following proposition indicates that the Price of Anarchy can be very high when the probabilities of the components functioning (p_1, p_2) are small. Moreover, for n agents connected in series, the price of anarchy can increase exponentially with number of agents n.

Proposition 5.3.1. *In the two-agent series prior game (defined above and cost matrix noted in the first row of Table 5.2), the Price of Anarchy in the absence of subsidy is at least PoA* $\geq \frac{2}{p_1+p_2}$, *for some repair costs* C_1, C_2. *More generally, for n agents, PoA* $\geq \tilde{H}/\tilde{G}^n$ *for some repair costs* $C_1, \ldots, C_n$, *where* $\tilde{H}$ *and* $\tilde{G}$ *are the harmonic and geometric means, respectively, of the prior probabilities* $p_1, \ldots, p_n$.

By Proposition 5.3.1, if probabilities of component working $p_1 = p_2 = \epsilon \ll 1$, then Price of Anarchy can be as large as $\frac{1}{\epsilon}$ (or as large as $\epsilon^{-(n-1)}$ for general n). We remark that our lower bounds also apply to the ratio PoA/PoS, i.e. when we compare the worst-case Nash equilibrium with the *global* Nash equilibrium instead of OPT. We will now show that a constant price of anarchy can be achieved in this game using subsidy, more precisely, that $\widetilde{\text{PoA}}(\mathbb{S}) = 1$ for some subsidy scheme $\mathbb{S}$. In fact, we characterize the total subsidy needed to guarantee this for any game parameters. The proof involves carefully considering cases for the parameters resulting in different sets of Nash equilibria and computing necessary and sufficient amounts of subsidy in each case (see Appendix).

Theorem 5.3.2. *Consider the two-agent series component maintenance game with* $C_1, C_2 > 0$ *and* $0 < p_1, p_2 < 1$. *Let* $s^* = \mathbf{I}\{(C_1, C_2) \in [\overline{p_1}p_2, \overline{p_1}] \times [\overline{p_2}p_1, \overline{p_2}]\} \cdot \min\{C_1 - \overline{p_1}p_2, C_2 - \overline{p_2}p_1\}$, *where* $\mathbf{I}\{\cdot\}$ *denotes the 0-1 valued indicator function. Then there exists a subsidy scheme* $\mathbb{S}$ *with total subsidy* s *for any* $s > s^*$ *such that* $\widetilde{PoA}(\mathbb{S}) = 1$. *Moreover, a total subsidy of at least* s^* *is necessary for any subsidy scheme* $\mathbb{S}$ *that guarantees* $\widetilde{PoA}(\mathbb{S}) = 1$.

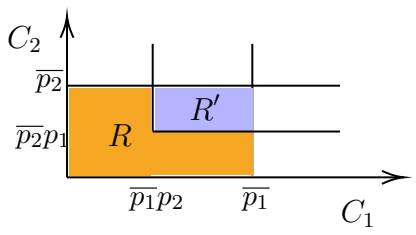


Figure 5.2: Cost region R with system functioning in any NE ($\phi(\mathbf{x}') = 1$, shaded orange) in a two-agent series game (Theorem 5.3.3). PoA $\neq$ PoS in region R' (Theorem 5.3.2).

Guaranteeing that the system functions in any NE.

As seen in Example 1, the agents may be in an equilibrium (e.g. DN-DN) such that the system is not guaranteed to function. This problem can also be remedied using subsidy. We will quantify the optimal subsidy needed to guarantee that the system functions, i.e. $\phi(\mathbf{x}') = 1$, where $\mathbf{x}'$ denotes the component states after the agents' actions.

Theorem 5.3.3. *Consider the two-agent series component maintenance game with $C_1, C_2 > 0$ and $0 < p_1, p_2 < 1$. Define $R \subset \mathbb{R}^+ \times \mathbb{R}^+$ as the set of cost pairs that satisfy $C_1 \leq \overline{p_1} \wedge C_2 \leq \overline{p_2} \wedge (C_1 \leq \overline{p_1}p_2 \vee C_2 \leq \overline{p_2}p_1)$, depicted in Figure 5.2. Let $s^* = \min_{(x,y)\in R} ||(C_1, C_2) - (x, y)||_1$, where $|| \cdot ||_1$ denotes the L_1-norm. Then there exists a subsidy scheme $\mathbb{S}$ with total subsidy s for any $s > s^*$ such that the system functions in any NE. Moreover, a total subsidy of at least s^* is necessary for any subsidy scheme $\mathbb{S}$ that guarantees that the system functions in any NE.*

The proof is deferred to Appendix B.1. In constrast to Theorem 5.3.2, here the central agent needs to provide subsidy in the states where the price of anarchy may be 1 without subsidy but the system is not guaranteed to function as agents may choose to do nothing.

Ensuring value of information is non-negative for each agent.

[110] exhibit several examples, including 2-agent series games, where the value of information for certain agents can be negative, when actions in the prior and posterior games are selected according to some Nash equilibria. We will demostrate the use of subsidy in tackling this undesirable information avoidance behavior.

In addition to the prior game, we will now consider posterior games where component c_1 is inspected, and its state y_1 is revealed on inspection ($y_1 = 1$ corresponds to c_1 is functioning, and $y_1 = 0$ corresponds to c_1 is broken). Table 5.2 summarizes prior and posterior costs for the two agents for each action pair (recall that DN denotes 'do nothing' or $s_i = 0$, and RE denotes 'repair' or $s_i = 1$). [110] show that the expected value of information (VoI) is non-negative for all agents if a global NE is selected.

Theorem 5.3.4 ([110])**.** *In the two-agent series game described above, if component $c_j, j \in \{1, 2\}$ is inspected, and the prior and posterior actions $s, \tilde{s}$ are selected from global equilibria, then the expected value of information $\text{VoI}_{i,j}(s, \tilde{s})$ is non-negative for each agent $i \in \{1, 2\}$.*

Therefore, if we avoid suboptimal local equilibria in this setting then expected VoI is guaranteed to be non-negative. Combined with Theorem 5.3.2 above, this implies that negative Value

Conditions	DN-DN	DN-RE	RE-DN	RE-RE
Prior	$\overline{p_1 p_2}, \overline{p_1 p_2}$	$\overline{p_1}, \overline{p_1} + C_2$	$\overline{p_2} + C_1, \overline{p_2}$	C_1, C_2
$y_1 = 1$	$\overline{p_2}, \overline{p_2}$	$0, C_2$	$\overline{p_2} + C_1, \overline{p_2}$	C_1, C_2
$y_1 = 0$	$1, 1$	$1, 1 + C_2$	$\overline{p_2} + C_1, \overline{p_2}$	C_1, C_2

Table 5.2: Matrix for cost-pairs (agent 1, agent 2) when component c_1 is inspected for the two-agent series system. Here $\overline{p_1 p_2} = 1 - p_1 p_2$ and $\overline{p_i} = 1 - p_i$.

of Information may be avoided via subsidizing repair costs in the two-agent series game. In more detail, we can employ a conditional subsidy scheme, with prior subsidy subs_i according to the scheme $\mathbb{S}$ in Theorem 5.3.2, and posterior subsidy subs_i^y when agent 1 is inspected employs the subsidy scheme from Theorem 5.3.2 by setting the parameter $p_1 = y$. Theorem B.1.1 in Appendix B.1 characterizes the optimal unconditional subsidy for ensuring the Value of Information is non-negative for each agent.

5.3.2 NP-Hardness in general n-agent maintenance and inspection games

We will now consider the problem of computing the best subsidy scheme in general component maintenance games, i.e. when a general boolean function ϕ governs system failure. We will prove computational hardness results in this section and assume familiarity with fundamental concepts in complexity theory [5]. We will do this by reducing the VERTEX-COVER problem (one of Karp's original NP-complete problems [97]) to the decision-problem version of computing the best subsidy, for the different objectives that the central agent may care about. Recall that VERTEX-COVER is the following decision problem, specified by a graph $\mathcal{G} = (V, E)$ and an integer k.

VERTEX-COVER: Does a given graph $\mathcal{G} = (V, E)$ admit a vertex cover[2] of size k?

Price of Anarchy under Subsidy.

Finding the best subsidy in a given component maintanence game (CMG) to minimize the price of anarchy is an optimization problem. Here we will study the hardness of a corresponding decision problem stated below.

CMG-POAS: Given a CMG G and a subsidy budget n^*, does there exist subsidy scheme $\mathbb{S}$ with non-zero subsidy provided to n^* agents such that $\text{PoA}(\mathbb{S}) = 1$?

Note that here we define the best subsidy scheme to be the one that provides subsidy to fewest agents. This meaningfully models situations when subsidy consists of allocation of indivisible resources, for example the central agent providing commitment to intervene and assist but with a bandwidth constraint on the number of agents thart could receive the assistance. As mentioned above, we will do a Karp reduction from the VERTEX-COVER problem. The key idea is to construct a component maintenance game G given any graph $\mathcal{G}$, with agents corresponding to graph nodes, and the system state function ϕ corresponding to the graph edges. We show that the Nash

[2]A set of vertices such that all edges of the the graph have at least one endpoint in the set.

equilibria of G roughly correspond to minimal vertex covers of $\mathcal{G}$. To ensure PoA$(\mathbb{S}) = 1$, we provide subsidy to the agents in a smallest vertex cover of $\mathcal{G}$.

Theorem 5.3.5. CMG-POAS *is NP-Hard.*

Proof. We will reduce the VERTEX-COVER problem to CMG-POAS. Given an instance $\mathcal{G}, k$ of the VERTEX COVER problem, we create a corresponding CMG-POAS problem as follows. Introduce an agent i for every vertex $i \in V$ and consider the (2-CNF) formula $\phi(\mathbf{x}) = \bigwedge_{(i,j)\in E}(x_i \vee x_j)$, where the clauses consist of component states x_i, x_j for all pairs i, j of agents corresponding to edges in E. Set the probability distribution θ to be the constant distribution with the entire probability mass on 0^n (i.e. all the components are guaranteed to fail without repair). Set repair cost $C_i = 1$ for all components i. Then the cost function for agent i for joint action $s = (s_i, s_{-i})$ is given by

$$\begin{aligned} l_i(s_i, s_{-i}, \theta) = \mathbb{E}_{\mathbf{x}\sim\theta}[\text{cost}_i] &= C_i s_i + P_\phi(\theta) \\ &= 1 \cdot s_i + 1 - \phi(\mathbf{x}') \\ &= s_i + 1 - \phi(s), \end{aligned}$$

where $x'_i = \max\{0, s_i\} = s_i$ denotes the component state after agent i takes action s_i.

We first proceed to characterize the set of Nash equilibria of this game. Note that $s = 0^n$ is a NE for this game, since any repair action by any agent increases the agent's cost by 1 if the repair does not change the state ϕ of the system, and by 0 otherwise (since ϕ is monotonic, $\phi(s)$ can only change from 0 to 1 on repair). Therefore no agent has any incentive to switch from DN to RE. We will now show that the remaining NEs for the game correspond to minimal vertex covers of $\mathcal{G}$.

Suppose $K \subseteq [n]$ be a set of agents for which the corresponding nodes in $\mathcal{G}$ constitute a minimal vertex cover. Let $s_{K'} = (s_1, \ldots, s_n)$ where $s_i = \mathbf{I}[i \in K']$, and $\mathbf{I}[\cdot]$ is the 0-1 valued indicator function, denote the joint action where agents in set $K' \subseteq [n]$ choose repair. Clearly, $\phi(s_K) = 1$. If $i \in K$, agent i does not reduce cost by switching from RE to DN since K is a minimal cover therefore not repairing component i causes the system to fail. As noted above, switching from DN to RE never improves an agent's cost in this game. Further, if K is the set of agents corresponding to a non-minimal vertex cover, then there must be some agent that can reduce its cost by switching from RE to DN (and system continues to function). Finally, if $K' \neq \emptyset$ is the set of agents with one or more agents short of a vertex cover, then any agent in K' can reduce its cost by switching from RE to DN. This establishes that besides 0^n, only possible NE must correspond to a minimal vertex cover. In particular, this implies that $\mathsf{OPT} = k^*$, where k^* is the size of the smallest vertex cover K^* of $\mathcal{G}$, and the corresponding NE is s_{K^*}.

To complete the reduction, we consider the game defined above with subsidy budget $n^* = k$. We will show a bijection between the YES and NO instances of the two decision problems to complete the proof.

If there exists a vertex cover of size k, then the smallest vertex cover K^* has size $k^* \leq k$. We design a subsidy scheme with subsidy allocated to $k^* \leq n^*$ agents, allocating subsidy of $1 + \frac{1}{2n}$ for repair (the total subsidy is no more than $k^* + \frac{1}{2}$) to all agents in the minimum cover K^*, a and subsidy of 0 otherwise. As argued above, the only candidate NE without subsidy are 0^n and s_K corresponding to some minimal vertex cover K. The social cost for 0^n is n (except the trivial case $k^* = 0$) and that for s_{K^*} is k^* which is smaller. If we provide subsidy in our scheme $\mathbb{S}$ to the

agents in K^* then 0^n is no longer an NE. In particular, every subsidized agent in K^* would now always choose repair at subsidized cost $-\frac{1}{2n}$ over doing nothing (even when the system stays broken after the repair). Thus, the social cost plus subsidy is $k^*(-\frac{1}{2n}) + k^*(1+\frac{1}{2n}) = k^*$, and the price of anarchy for the subsidy scheme is 1 (any agent outside of K^* will prefer to do nothing to reduce their cost).

On the other hand, suppose that $\mathcal{G}$ has no vertex cover of size k. Any minimal vertex cover of $\mathcal{G}$ therefore has size at least $k+1$. Suppose $\mathbb{S}$ is a subsidy scheme with subsidy allocated to most k agents. We will show that $\text{PoA}(\mathbb{S}) > 1$. Indeed, let K be an arbitrary minimal vertex cover of $\mathcal{G}$. By pigeonhole principle, at least one agent in K does not receive subsidy. Let K' denote the (possibly empty) set of agents that receive subsidy greater than 1. As argued above, these agents will always prefer the repair action. Thus, $s_{K'}$ is a Nash equilibrium (agents without subsidy never have incentive to switch from DN to RE in this game) in the subsidized game. Note that $\phi(s_{K'}) = 0$ since at least one vertex is missed in any minimal vertex cover K, and therefore we must have an uncovered edge. Now $\mathsf{OPT} \leq \text{cost}(s_K) = |K| < n$. Therefore, $\text{PoA}(\mathbb{S}) > 1$ as total cost plus subsidy is at least n for $s_{K'}$. □ □

We make a couple of remarks about the above result. Our proof implies a similar hardness result if the decision problem is stated for $\widetilde{\text{PoA}}(\mathbb{S})$ instead of $\text{PoA}(\mathbb{S})$. We further note that our reduction from VERTEX-COVER does not involve any negative literals in the boolean system function ϕ, i.e. applies to monotone boolean functions where if a component is repaired from broken to a working state, then it cannot cause the overall system to go from working to broken.

Guaranteeing that the system functions in any NE.

Here we consider a more challenging optimization objective of finding the optimal subsidy scheme with the least total value across all agents that receive the subsidy, and the goal of the central agent is to disburse sufficient subsidy to guarantee that the system functions in any Nash equilibrium. The decision problem in this case is stated as follows.

CMG-SYSTEM: Given a component maintenance game (CMG) and subsidy budget s^*, does some subsidy scheme $\mathbb{S}$ with total subsidy at most s^* guarantee that the system functions in any NE (i.e. $\phi(s) = 1$ for any $s \in \mathcal{S}_{NE}(\mathbb{S})$)?

We give a Karp mapping from any vertex cover instance $\mathcal{G}, k$ to a CMG-SYSTEM instance and use a slightly different game instance for the reduction. We show that the system is guaranteed to function in any NE iff the subsidy budget is one less than the size of a smallest vertex cover. See Appendix B.3 for a proof of the following result.

Theorem 5.3.6. CMG-SYSTEM *is NP-Hard.*

Value of Information.

We also show NP-hardness of the problem of determining the minimal subsidy needed to avoid negative Value of Information in the component inspection game. Formally, the decision problem is stated below.

CIG-VOI: Given a component inspection game (CIG) and subsidy budget s^*, does some subsidy scheme $\mathbb{S}$ with total subsidy at most s^* guarantee that no agent has negative value of

information when a single component j is inspected (i.e. for fixed inspected component j, $\text{VoI}_{i,j}(s, \tilde{s}^{j,y_j}) \geq 0$ for any $s \in \mathcal{S}_{NE}(\mathbb{S})$, $\tilde{s}^{j,y_j} \in \mathcal{S}_{NE}^{y_j}(\mathbb{S})$, for each agent $i \in [n]$, and each posterior component state $y_j \in \{0, 1\}$)?

Our proof (Appendix B.3) again involves a reduction from the VERTEX-COVER problem, but we use a different subsidy budget and examine the non-negativity of VoI when potentially different equilibria are selected in the prior and posterior games.

Theorem 5.3.7. CIG-VOI *is NP-Hard.*

5.3.3 A Bayesian view

It is possible to view the component maintenance game from a Bayesian perspective. A Bayesian game [91, 132] is defined as a tuple $B = \langle N, S = (S_1 \times \cdots \times S_n), T = (T_1 \times \cdots \times T_n), \theta, (\text{cost}_i) \rangle$, where N is a set of n agents (or players), S_i is the finite action space of agent $i \in N$, T_i is the type space of player i, θ is a common prior over types and $\text{cost}_i : S \times T \to \mathbb{R}_{\geq 0}$ is the cost function of agent i. In the component maintenance game, the types of the agents are their component states x_i. Bayesian games are typically analysed from three perspectives—**ex-ante**, where the agents do not know the actual type for any agent, **interim**, where the agents know their own actual type but not for any other agent, and **ex-post**, where the agents know every agents' type.

A pure-strategy Bayesian Nash Equilibrium (BNE) from the *ex-ante* perspective in a finite game is simply a pure-strategy Nash Equilibrium with the cost matrix given by the expected costs for types drawn according to θ [132]. This corresponds to the 'prior' game above in the component maintenance game. In the above formulation for component inspection, in the 'posterior' game a single agent's component is inspected and all the agents are assumed to know the type (state) of the inspected agent (component). This does not fit neatly with the *interim* state where *every* agent learns their own private type, or the *ex-post* state where *every* agent learns every agents' type. Studying the effect of providing subsidy in these settings is an interesting direction for future work.

5.4 Data-driven subsidy in repeated games

Above computational hardness results motivate us to consider a beyond worst-case approach to finding a good subsidy for a given game. Specifically, we will consider the data-driven algorithm design paradigm introduced by [83], and further studied by [13, 15]. In this framework, we will assume access to multiple games coming from the same domain (e.g. infrastructure management in similar counties) and determine a good value of subsidy for unseen game instances from the same domain. We will consider games drawn i.i.d. from an (arbitrary, unknown) game distribution, or games arriving in an online sequence. In the former we and will be interested in having a small *sample complexity* of the number of game samples needed to generalize well to an unseen sample from the same distribution. For the latter, we will study regret relative to the best possible subsidy scheme over the online sequence, in hindsight.

5.4.1 Sample complexity for subsidy schemes

In this section, we define the notion of *sample complexity* for designing sample-based subsidy schemes for cost minimization games. The sample complexity for (uniform convergence of) a given set of subsidy schemes measures how many samples are sufficient to ensure the expected social cost of any subsidy scheme in the set approximately matches its average cost over the game samples with high probability, for any given approximation and confidence level. In particular, if there is a subsidy scheme in the set with small social cost over a sufficiently large set of game samples, then that scheme will almost certainly have low cost in expectation over the distribution over games from which the samples are generated. Guarantees on sample complexity are a central topic in computational learning theory [4, 161].

Definition 19 (Sample complexity)**.** *The sample complexity for a class $\mathcal{S}$ of subsidy schemes is a function $\mathcal{N} : \mathbb{R}_{\geq 0} \times (0,1) \to \mathbb{Z}_{\geq 1}$ defined such that for any $\epsilon > 0$, any $\delta \in (0,1)$, any sample size $N \in \mathbb{Z}_{\geq 1}$, and any distribution $\mathcal{D}$ over the games, with probability at least $1-\delta$ over the draw of a set $S \sim \mathcal{D}^N$, for any scheme $\mathbf{S}$ in $\mathcal{S}$, the difference between the average cost of $\mathbf{S}$ over S and the expected cost of $\mathbf{S}$ over $\mathcal{D}$ is at most ϵ, whenever $N \geq \mathcal{N}(\epsilon, \delta)$. In other words,*

$$\Pr_{S \sim \mathcal{D}^N}\left[\exists \mathbf{S} \in \mathcal{S} \text{ s.t. } \left|\frac{1}{N}\sum_{v \in S} \textsc{cost}_{\mathbf{S}}(v) - \mathbb{E}[\textsc{cost}_{\mathbf{S}}(v)]\right| > \epsilon\right] < \delta.$$

Note that the existence of a single $\mathbf{S} \in \mathcal{S}$ that violates the ϵ-approximation for its expected cost is sufficient to cause the 'failure' event which happens with probability δ. In other words, with probability $1-\delta$, all schemes $\mathbf{S}$ must observe *uniform convergence* of sample cost to expected cost (for sufficiently large sample). The $1-\delta$ high probability condition is needed because it is always possible that (with a very small but non-zero probability) the set of samples S, no matter how large, is completely unrepresentative of the distribution $\mathcal{D}$ over the games. Clearly, $\mathcal{N}(\epsilon, \delta)$ should grow as δ or ϵ shrinks since we need to ensure that the difference between the average and expected cost of each subsidy in $\mathcal{S}$ is at most $\mathcal{N}(\epsilon, \delta)$ with probability at least $1-\delta$. The sample complexity $\mathcal{N}(\epsilon, \delta)$ of class $\mathcal{S}$ of course also depends on the specific subsidy class $\mathcal{S}$. According to classic computational learning theory, the more "complex" the subsidy class $\mathcal{S}$ is, the more challenging it is to bound the difference between the average and expected cost of every subsidy in $\mathcal{S}$, i.e. richer subsidy classes $\mathcal{S}$ have larger sample camplexity $\mathcal{N}(\epsilon, \delta)$.

For an arbitrary class $\mathcal{S}$, a bound on the sample complexity allows the subsidy scheme designer to relate the expected cost of a scheme in $\mathcal{S}$ which achieves minimum average cost over the set of samples to the expected cost of an optimal scheme in $\mathcal{S}$, using classic arguments from learning theory [125]. More precisely, for a set of samples S from the distribution over buyers' values, let $\hat{S}$ be the scheme in $\mathcal{S}$ that minimizes average cost over the set of samples and let $\mathbf{S}^*$ be the scheme in $\mathcal{S}$ that minimizes expected cost over $\mathcal{D}$. Finally, let P be the minimum cost achievable by any scheme in $\mathcal{S}$ over the support of the distribution $\mathcal{D}$. For any $\delta \in (0,1)$, with probability at least $1-\delta$ over the draw of a set of $N \geq \mathcal{N}(\epsilon, \delta)$ samples S from $\mathcal{D}$, the difference between the expected cost of $\hat{S}$ over $\mathcal{D}$ and the expected cost of $\mathbf{S}^*$ over $\mathcal{D}$ is at most 2ϵ. Therefore, so long as there is a good sample complexity $\mathcal{N}(\epsilon, \delta)$ bound for subsidy scheme class $\mathcal{S}$, the scheme designer can be confident that an optimal scheme over the set of observed samples competes with an optimal scheme in $\mathcal{S}$.

Pseudo-dimension. Pseudo-dimension [137] is a well-known learning theoretic measure of complexity of a class of functions (it generalizes the Vapnik-Chervonenkis or VC dimension to real-valued functions), and is useful in obtaining bounds on the sample complexity of fitting functions from that class to given data.

Definition 20 (Pseudo-dimension)**.** *Let $\mathcal{H}$ be a set of real valued functions from input space $\mathcal{X}$. We say that $C = (x_1, \ldots, x_m) \in \mathcal{X}^m$ is pseudo-shattered by $\mathcal{H}$ if there exists a vector $r = (r_1, \ldots, r_m) \in \mathbb{R}^m$ (called "witness") such that for all $b = (b_1, \ldots, b_m) \in \{\pm 1\}^m$ there exists $h_b \in \mathcal{H}$ such that sign$(h_b(x_i) - r_i) = b_i$. Pseudo-dimension of $\mathcal{H}$ is the cardinality of the largest set pseudo-shattered by $\mathcal{H}$.*

For a function class with range $[0, H]$ and pseudo-dimension d, a sample complexity bound of $\mathcal{N}(\epsilon, \delta) = O(\frac{H^2}{\epsilon^2}(d + \log \frac{1}{\delta}))$ is well-known [4, 7]. We conclude this section with a useful general lemma from data-driven algorithm design for giving upper bounds on the pseudo-dimension of certain loss function classes.

Lemma 5.4.1. *(Lemma 2.3, [7]) Suppose that for every game $G \in \mathbf{G}$, the function $L_G(\sigma) : \mathbb{R} \to \mathbb{R}$ is piecewise constant with at most N pieces. Then the family $\{L_\sigma(\cdot)\}$ over games in $\mathbf{G}$ has pseudo-dimension $O(\log N)$.*

5.4.2 Sample complexity for subsidizing games drawn from a distribution

Learning uniform subsidies. We start with some initial results on learning a good value of the subsidy even in the absence of considerations about value of information, or possible equilibria in posterior games. We will consider uniform subsidy $\sigma \in \mathbb{R}_{\geq 0}$ conditional on repair (i.e. reduces cost of repair for all agents that choose to repair). A simple loss objective in this uniform subsidy setting is given by

$$L_{\text{prior}}(\sigma) := \max_{s \in \mathcal{S}_{NE}(\sigma)} \text{cost}^\sigma(s) + n_s \sigma,$$

where n_s is the number of agents that choose repair in joint state s, $\mathcal{S}_{NE}(\sigma)$ and $\text{cost}^\sigma(s)$ denote the set of Nash equilibria and (respectively) the updated total cost, when a uniform subsidy of σ is applied. We further assume all the repair costs as well as subsidy budget is no more than H, i.e. $\sigma, C_i \leq H$ for each $i \in [n]$. Therefore, $L_{\text{prior}}(\sigma) \leq (2H + 1)n$.

Suppose the central agent (learner) who needs to set the subsidy has repeated instances of this game (e.g. cost matrices) drawn from a distribution. Can we learn a good value of uniform subsidy s^*, that has small expected loss over the distribution? We will use standard machinery for data-driven algorithm design [7].

Theorem 5.4.2. *For any $\epsilon, \delta > 0$ and any distribution $\mathcal{D}$ over component maintenance games with n agents, $O(\frac{n^2 H^2}{\epsilon^2}(n + \log \frac{1}{\delta}))$ samples of the game drawn from $\mathcal{D}$ are sufficient to ensure that with probability at least $1 - \delta$ over the draw of the samples, the best value of uniform subsidy over the sample $\hat{s^*}$ has expected loss $L_{prior}(\hat{s^*})$ that is at most ϵ larger than the expected loss of the best value of subsidy over $\mathcal{D}$.*

Proof. Consider any fixed component maintenance game G. Observe that if actions of all agents except agent i i.e. s'_{-i} is fixed (2^{n-1} possibilities), then the agent i will repair its component provided the cost under subsidy $C_i - s^* + 1 - \mathbb{E}_\theta \phi(\mathbf{x}'(1, s'_{-i}))$ is smaller than $1 - \mathbb{E}_\theta \phi(\mathbf{x}'(0, s'_{-i}))$. That is we have at most 2^{n-1} critical values of s^* where the preferred action of agent i may change. Over n agents, we have at most $n2^{n-1}$ such points. Moreover, the loss is piecewise constant in any fixed piece.

Given the piecewise-constant structure with a bound on the total number of pieces, the sample complexity bound follows from standard learning theoretic arguments. In more detail, by Lemma 5.4.1, this implies that the pseudo-dimension of the loss function class parameterized by the subsidy value is at most $O(\log(n2^{n-1})) = O(n)$ and the sample complexity result follows classic bounds [4, 7]. □ □

Remark 4. *Note that a naive bound of $O(2^n)$ could be derived on the pseudo-dimension of any n player game, where each player has 2 possible actions. This is because there are 2^n distinct states and therefore at most 2^{2^n} possible distinct state subsets which could correspond to a Nash Equilibrium. The critical subsidy values s^* correspond to values at which the set of NE changes, and for any pair of state subsets exactly one could be the set of NE for all values of subsidy above (respectively below) some critical value s^*. The loss is again piecewise constant, and by Lemma 5.4.1 we have that the pseudo-dimension is $O(2^n)$. The above proof makes use of cost matrix of the component maintenance game to obtain the exponentially better upper bound of $O(n)$.*

Learning non-uniform subsidies. We are further able to obtain a sample complexity bound even for the non-uniform subsidy scheme defined above, where the central agent can provide a different subsidy to each agent depending on their component cost, failure probability and how critical the component is to overall system functionality. The subsidy scheme consists of a vector of multiple real-valued parameters, one for each agent.

$$L_{\text{prior}}(\mathbb{S}) := \max_{s \in \mathcal{S}_{NE}(\mathbb{S})} \text{cost}^{\mathbb{S}}(s) + \text{subs}(s),$$

We assume that $\text{subs}_i(s), C_i \leq H$ for each $i \in [n]$, and therefore $L_{\text{prior}}(\mathbb{S}) \leq (2H+1)n$. Again, we are able to give a polynomial sample complexity for the number of games needed to learn a good value of subsidy with high probability over the draw of game samples coming from some fixed but unknown distribution (proof in Appendix B.4).

Theorem 5.4.3. *For any $\epsilon, \delta > 0$ and any distribution $\mathcal{D}$ over component maintenance games with n agents, $O(\frac{n^2H^2}{\epsilon^2}(n^2 + \log\frac{1}{\delta}))$ samples of the game drawn from $\mathcal{D}$ are sufficient to ensure that with probability at least $1-\delta$ over the draw of the samples, the best vector of subsidies over the sample $\hat{s^*}$ has expected loss L_{prior} that is at most ϵ larger than the expected loss of the best vector of subsidies over $\mathcal{D}$.*

A similar sample complexity bound can also be given for learning conditional subsidies from game samples, by minimizing a loss based on the social cost in the posterior game. See Theorem B.4.1 in Appendix B.4. Note that minimization of L_{prior} corresponds to minimization of $\text{PoA}(\mathbb{S})$. To guarantee that the system functions, we can simply add a regularization term $\lambda(1-\phi(s))$, for sufficiently large $\lambda > (2H+1)n$.

In learning terminology, our results imply a bound on the number of sample games in the 'training set' to do well on an unseen 'test' game instance from the same distribution. We note that optimization over the training set is still computationally hard, but we can avoid solving the hard problem over and over again for repeated test instances.

5.4.3 No-regret when subsidizing in an online sequence of games

In the online setting, we receive a sequence of games at times (rounds) $t = 1, \ldots, T$. In each round t, the central agent must set a value of the (uniform) subsidy σ_t, with potentially some feedback on previous rounds but no knowledge of the game parameters (costs/priors) of the current or future rounds. This is more pessimistic (but potentially also more realistic) than the distributional setting above. In particular, the sequence of games may be adversarially picked. The performance of the algorithm is measured by the difference in the cumulative loss for the selected subsidy values and the cumulative loss of the best fixed value of subsidy in hindsight, also known as regret (R_T).

$$R_T := \sum_{t=1}^{T} L_{\text{prior}}(\sigma_t) - \min_{\sigma \in [0,H]} L_{\text{prior}}(\sigma).$$

'No-regret' corresponds to R_T being sublinear in T, and the average regret R_T/T approaches zero for large T in this case. We will impose a mild assumption on the repair costs C_i to obtain good results in the online setting. We will assume that the costs are not known exactly, but come from some smooth distribution. Formally,

Assumption 5. *We assume that the probability distributions generating the costs have κ-bounded probability density, i.e. $\max_{x \in \mathbb{R}} f_i(x) \leq \kappa$ for some $\kappa \in \mathbb{R}^+$, where f_i denotes the probability density function for cost C_i.*

The adversary designing the sequence of games may select any bad distribution as long as it is smooth. Under this assumption, our analysis above and tools from [19] can be used to show that the online sequence of loss functions is *dispersed* [15]. Dispersion, informally speaking, captures how amenable a non-Lipschitz function is to online learning. As noted in [15, 20], dispersion is a sufficient condition for learning piecewise Lipschitz functions online, even in changing environments. A formal definition is included below.

Definition 21 (Dispersion, [15, 19])**.** *The sequence of random loss functions $L_1, \ldots, L_T$ is β-dispersed for the Lipschitz constant ℓ if, for all T and for all $\epsilon \geq T^{-\beta}$, we have that, in expectation, at most $\tilde{O}(\epsilon T)$ functions (here $\tilde{O}$ suppresses dependence on quantities beside ϵ, T and β, as well as logarithmic terms) are not ℓ-Lipschitz for any pair of points at distance ϵ in the domain $\mathbf{C}$. That is, for all T and $\epsilon \geq T^{-\beta}$,*

$$\mathbb{E}\left[\max_{\substack{\rho,\rho' \in \mathbf{C} \\ ||\rho - \rho'||_2 \leq \epsilon}} \left|\{t \in [T] \mid L_t(\rho) - L_t(\rho') > \ell ||\rho - \rho'||_2\}\right|\right] \leq \tilde{O}(\epsilon T).$$

Under Assumption 5, we have the following guarantee about online learning of subsidy in a sequence of games, namely one can predict good values of subsidy (with $\tilde{O}\left(\sqrt{\frac{n}{T}}\right)$ average expected regret over T online rounds). We establish $\frac{1}{2}$-dispersion of the sequence of loss functions

under the above assumption and use results from [19] to obtain the regret bound. Our main result is stated below (proof in Appendix B.4).

Theorem 5.4.4. *Suppose Assumption 5 holds. Let $L_1, \ldots, L_T : [0, H] \to [0, (2H+1)N]$ denote an independent sequence of losses as a function of the subsidy value σ, in an online sequence of T component maintenance games. Then sequence of functions is $\frac{1}{2}$-dispersed and there is an online algorithm with $\tilde{O}(\sqrt{nT})$ expected regret.*

5.5 Discussion

We study the problem of resource allocation for infrastructure maintenance in systems with privately owned components via a game-theoretic model. The abstract model captures the typical organization of engineering systems, computer networks, or project pipelines. We identify useful metrics related to price of anarchy, shared 'system' costs and value of information that a central agent may care about and examine the challenge in optimally allocating resources to optimize these metrics. Some metrics might be of interest in analyzing other games involving shared costs or information.

Our work employs a data-driven approach, which has not been previously employed in the literature of subsidy or taxation design which has largely focused on designing approximations for worst-case game parameters. An interesting further question is to extend this idea of analyzing 'typical' games to potentially obtain better domain-specific subsidy schemes in other interesting games [49, 119]. Also our learning-based approach allows modeling more realistic settings, where the game parameters or even set of agents may change over time.

The results in this chapter are based on joint work with Nina Balcan and Matteo Pozzi, which will appear at EMI/PMC 2024.

Chapter 6

Robustness

Deep networks are highly susceptible to adversarial attack, and a key challenge to robustness is that they typically involve non-Lipschitz maps from inputs to feature embeddings (as small input perturbations can often produce large movements in the network's final-layer feature space). This chapter will consider an attack model that abstracts this challenge, to help understand its intrinsic properties: the adversary may move data an arbitrary distance in feature space but only in random low-dimensional subspaces. Such adversaries can defeat any algorithm that must classify any input it is given. However, by allowing the algorithm to abstain on unusual inputs, such adversaries can be overcome when classes are reasonably well-separated in feature space. A key tool in designing a good algorithm will be using data-driven methods to select algorithm parameters to optimize over accuracy-abstention trade-offs. The results provide new robustness guarantees for nearest-neighbor style algorithms, and also have application to contrastive learning, where we empirically demonstrate the ability of such algorithms to obtain high robust accuracy with low abstention rates.

Formal threat model. Let $\mathbf{x}$ be an n_1-dimensional test input for classification. The input is embedded into an n_2-dimensional feature space using an abstract mapping F. Our threat model is that the adversary may corrupt $F(\mathbf{x})$ such that the modified feature vector is any point in a random n_3-dimensional affine subspace denoted by $\mathcal{S} + \{F(\mathbf{x})\}$. For example, if $n_3 = 1$ then $\mathcal{S} + \{F(\mathbf{x})\}$ is a random line through $F(\mathbf{x})$, and the adversary may select an arbitrary point on that line; if $n_3 = 2$ then $\mathcal{S} + \{F(\mathbf{x})\}$ is a random 2-dimensional plane through $F(\mathbf{x})$, and the adversary may select an arbitrary point in that plane. The adversary is given access to everything including the algorithm's classification function, F, $\mathbf{x}$, $\mathcal{S}$ and the true label of $\mathbf{x}$.

Our framework can be thought of as a kind of smoothed analysis [155] in its combination of random and adversarial components. However, a key distinction is that in smoothed analysis, the adversary moves first, and randomness is added to its decision afterwards. In our model, in contrast, first a random restriction is applied to the space of perturbations the adversary may choose from, and then the adversary may move arbitrarily in that random subspace. Thus, the adversary in our setting has more power, because it can make its decision after the randomness has been applied.

6.1 Robustness via abstention

Remarkably, abstention is necessary for robust prediction under our model. A classifier without abstention must label the entire feature space. For a simple linear decision boundary, a perturbation in any direction (except parallel to the boundary) can cross the boundary with an appropriate magnitude (Figure 6.1, mid). The left and right figures show that if we try to 'bend' the decision boundary to 'protect' one of the classes, the other class is still vulnerable. This intuition may be formalized and generalized (beyond linear classifiers to *any* classifier), and one could show that there must be at least one vulnerable class irrespective of how you may try to shape the class boundaries, where the adversary succeeds in a large fraction of directions.

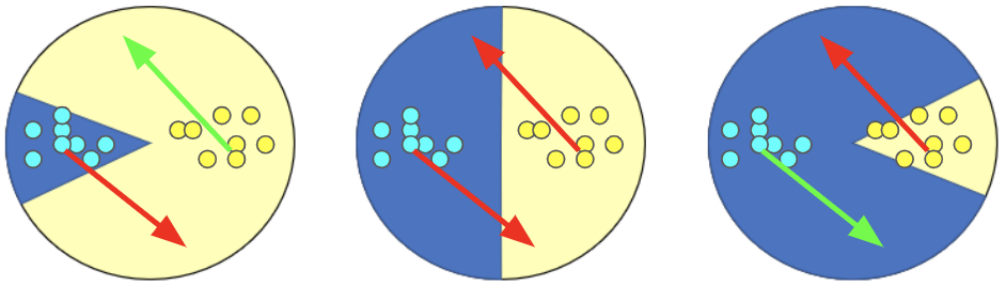


Figure 6.1: A simple example to illustrate that abstention is necessary in our model. Bending the decision boundary to avoid adversarial examples for one class makes it harder to defend the other class.

Theorem 6.1.1. *For any classifier that partitions $\mathbb{R}^{n_2}$ into two or more classes, any data distribution $\mathcal{D}$, any $\delta > 0$ and any feature embedding F, there must exist at least one class y^*, such that for at least a $1-\delta$ probability mass of examples $\mathbf{x}$ from class y^* (that is, $\mathbf{x}$ is drawn from $\mathcal{D}_{\mathcal{X}|y^*}$), for a random unit-length vector $\mathbf{v}$, with probability at least $1/2-\delta$ for some $\Delta_0 > 0$, $F(\mathbf{x}) + \Delta_0\mathbf{v}$ is not labeled y^* by the classifier. In other words, there must be at least one class y^* such that for at least $1-\delta$ probability mass of points $\mathbf{x}$ of class y^*, the adversary wins with probability at least $1/2-\delta$.*

Proof. Define r_δ to be a radius such that in the feature space, for every class y, at least a $1-\delta$ probability mass of examples $F(\mathbf{x})$ of class y lie within distance r_δ of the origin. Define R such that for a ball of radius R, if we move the ball by a distance r_δ, at least a $1-\delta$ fraction of the volume of the new ball is inside the intersection with the old ball. Now, let $\mathcal{B}$ be the ball of radius R centered at the origin in feature space. Let $\mathsf{vol}(\mathcal{B})$ denote the volume of $\mathcal{B}$ and let $\mathsf{vol}_y(\mathcal{B})$ denote the volume of the subset of $\mathcal{B}$ that is assigned label y by the classifier. Let y^* be any label such that $\mathsf{vol}_{y^*}(\mathcal{B})/\mathsf{vol}(\mathcal{B}) \leq 1/2$. Now by the definition of y^*, a point $\mathbf{z}$ picked uniformly at random from $\mathcal{B}$ has probability at least $1/2$ of being classified differently from y^*. This implies that, by the definition of R, if $F(\mathbf{x})$ is within distance r_δ of the origin, then a point $\mathbf{z}_x$ that is picked uniformly at random in the ball $\mathcal{B}_x$ of radius R centered at $F(\mathbf{x})$ has probability at least $1/2-\delta$ of being classified differently from y^*. This immediately implies that if we choose a random unit-length vector $\mathbf{v}$, then with probability at least $1/2-\delta$, there exists $\Delta_0 > 0$ such that

$F(\mathbf{x}) + \Delta_0 \mathbf{v}$ is classified differently from y^*, since we can think of choosing $\mathbf{v}$ by first sampling $\mathbf{z}_x$ from $\mathcal{B}_x$ and then defining $\mathbf{v} = (\mathbf{z}_x - F(\mathbf{x}))/\|\mathbf{z}_x - F(\mathbf{x})\|_2$. Moreover, since the classifier has no abstention region, being classified differently from y^* implies a win by the adversary. So, the theorem follows from the fact that, by the definition of r_δ, at least $1 - \delta$ probability mass of examples $F(\mathbf{x})$ from class y^* are within distance r_δ of the origin in feature space. □

We remark that our lower bound applies to any classifier and exploits the fact that a classifier without abstention must label the entire feature space. For a simple linear decision boundary, a perturbation in any direction (except parallel to the boundary) can cross the boundary with an appropriate magnitude (Figure 6.1, mid). The left and right figures show that if we try to 'bend' the decision boundary to 'protect' one of the classes, the other class is still vulnerable. Our argument formalizes and generalizes this intuition, and shows that there must be at least one vulnerable class irrespective of how you may try to shape the class boundaries, where the adversary succeeds in a large fraction of directions.

We propose a simple parameterized nearest-neighbor algorithm (to be applied to feature embeddings of deep non-Lipschitz networks) which is able to overcome the above lower bound, and attain small robust error in our model.

Algorithm 10 ROBUSTCLASSIFIER(τ, σ)

1: **Input:** A test example $F(\mathbf{x})$ (potentially an adversarial example), a set of training examples $F(\mathbf{x}_i)$ and their labels y_i, $i \in [m]$, a threshold parameter τ, a separation parameter σ.
2: **Preprocessing:** Delete training examples $\mathbf{x}_i$ corresponding to features $F(\mathbf{x}_i)$ if $\min_{j \in [m], y_i \neq y_j} \mathsf{dist}(F(\mathbf{x}_i), F(\mathbf{x}_j)) < \sigma$.
3: **Output:** A predicted label of $F(\mathbf{x})$, or "don't know".
4: **if** $\min_{i \in [m]} \mathsf{dist}(F(\mathbf{x}), F(\mathbf{x}_i)) < \tau$ **then**
5: **return** $y_{\text{argmin}_{i \in [m]} \mathsf{dist}(F(\mathbf{x}), F(\mathbf{x}_i))}$;
6: **else**
7: **return** "don't know".

We prove an upper bound on the robust error of Algorithm 10, which is defined as the probability the adversary succeeds under our threat model. Formally, $\mathcal{E}^{\mathbf{x}}_{\text{adv}}(f) := \mathbb{E}_{S \sim \mathcal{S}} I\{\exists \mathbf{e} \in S + F(\mathbf{x}) \text{ s.t. } f(\mathbf{e}) \neq \mathbf{y} \text{ and } f(\mathbf{e}) \text{ does not abstain}\}$ the robust error of a given classifier f for classifying instance $\mathbf{x}$.

We illustrate the analysis in the simpler case of $n_3 = 1$. Suppose we have a training example $\mathbf{x}'$ of another class, and suppose $F(\mathbf{x})$ and $F(\mathbf{x}')$ are at distance D in the feature space. That is, $\mathsf{dist}(F(\mathbf{x}), F(\mathbf{x}')) = D$. If $\tau = o(D)$, the probability that the adversary can move $F(\mathbf{x})$ to within distance τ of $F(\mathbf{x}')$ is at most the ratio of the surface area of a sphere of radius τ to the surface area of a sphere of radius D, which is at most

$$\left(\frac{\tau}{D}\right)^{n_2 - 1} \leq \left(\frac{\tau}{r}\right)^{n_2 - 1}$$

if the feature space is n_2-dimensional. See Figure 6.2. By a union bound, for the case $n_3 = 1$, the robust error is at most

$$m \left(\frac{\tau}{r}\right)^{n_2 - 1}.$$

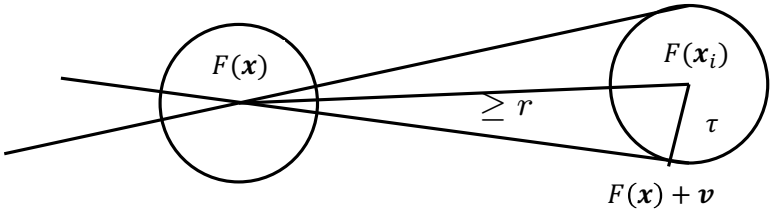


Figure 6.2: Adversarial misclassification for nearest-neighbor predictor. Here $F(\mathbf{x})$ is the test point and $F(\mathbf{x}_i)$ is a training point from a different class. For $n_3 = 1$, the adversary succeeds for the directions inside the depicted cone around $F(\mathbf{x}_i)$.

The argument above may be generalized to general $n_3 > 1$ (see Theorem 2, [30]) using the following Random Projection Theorem [63, 163].

Theorem 6.1.2 (Random Projection Theorem)**.** *Let* $\mathbf{z}$ *be a fixed unit length vector in* d*-dimensional space and* $\widetilde{\mathbf{z}}$ *be the projection of* $\mathbf{z}$ *onto a random* k*-dimensional subspace. For* $0 < \delta < 1$*,*

$$\Pr\left[\left|\|\widetilde{\mathbf{z}}\|_2^2 - \frac{k}{d}\right| \geq \delta\frac{k}{d}\right] \leq e^{-\frac{k(\delta^2-\delta^3)}{4}}.$$

In fact, we are able to use a high dimensional geometric argument to prove the following (asymptotically better for fixed n_3 and large n_2) guarantee via a tighter bound on the probability mass of the region of adversarial success.

Theorem 6.1.3. *If* $\tau = o(r)$*, the robust error* $\mathcal{E}^{\mathbf{x}}_{\text{adv}}(f)$ *of* ROBUSTCLASSIFIER$(\tau, 0)$ *in Algorithm 10 for classifying* $\mathbf{x}$ *is at most* $\mathcal{O}\left(\frac{m}{n_2-n_3}\left(\frac{\tau}{r}\right)^{n_2-n_3}\frac{1}{B(n_3/2,(n_2-n_3)/2)}\right)$*, where* $B(\cdot,\cdot)$ *is the Beta function. The Beta function is given by* $B(r_1, r_2) = \int_0^1 t^{r_1-1}(1-t)^{r_2-1}dt$*, for* $r_1, r_2 \in \mathbb{R}^+$*, and is closely related to binomial coefficients.*

Proof. We drop $F(\cdot)$ from the notation for simplicity. Let $\mathbf{x}$ be the origin. Let $\mathbf{x}'$ be a training point of another class, and R be a random n_3-dimensional linear subspace. Scale all distances by a factor of $\frac{1}{\mathsf{dist}(\mathbf{x},\mathbf{x}')}$. By rotational symmetry, we assume WLOG that R is given by $x_{n_3+1} = x_{n_3+2} = \cdots = x_{n_2} = 0$, and $\mathbf{x}'$ is the uniformly random unit vector $(z_1, \ldots, z_{n_2})$. Indeed, for a fixed direction from $\mathbf{x}$, the set of subspaces for which the projection of $\mathbf{x}'$ lies along that direction is constrained by one vector each in the range space and kernel space, and is therefore in bijection to the set of subspaces associated with another fixed direction (Figure 6.3).

The adversary can win only if the distance between $\mathbf{x}'$ with the closest vector Proj$[\mathbf{x}']$ in R, that is with $(z_1, \ldots, z_{n_3}, 0, \ldots, 0)$, is at most $\frac{\tau}{\mathsf{dist}(\mathbf{x},\mathbf{x}')} \leq \frac{\tau}{r}$. We can now apply Lemma C.1.1, which gives a bound on the fraction of the surface of the sphere at some fixed small distance from the orthogonal space, to get that the adversary succeeds by perturbing $\mathbf{x}$ to a point within $\mathcal{B}(\mathbf{x}', \tau)$ with probability at most

$$\frac{2(\tau/r)^{n_2-n_3}}{n_2-n_3} \cdot \frac{A(n_2-n_3-1)A(n_3-1)}{A(n_2-1)},$$

where $A(n)$ is the surface-area of the unit n-sphere embedded in $\mathbb{R}^{n+1}$. We have a closed form $A(n) = \frac{2\pi^{\frac{n+1}{2}}}{\Gamma\left(\frac{n+1}{2}\right)}$, where $\Gamma(z) = \int_{t=0}^{\infty} t^{z-1}e^{-t}dt$ is the gamma function.

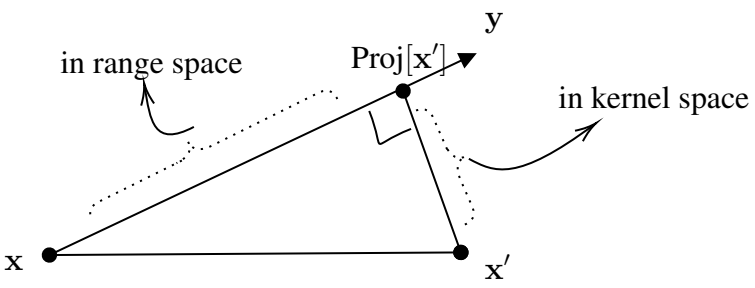


Figure 6.3: Rotational symmetry of adversarial subspaces. Let $\mathbf{y}$ be a random direction from test point $\mathbf{x}$, and Proj$[\mathbf{x}']$ be the projection of training point $\mathbf{x}'$ on to $\mathbf{xy}$. For any adversarial space with Proj$[\mathbf{x}']$ as the projection of $\mathbf{x}'$ on the space, we must have $\mathbf{xy}$ in the range space and $\mathbf{x}'$Proj$[\mathbf{x}']$ in the nullspace.

Noting that $B(z_1, z_2) = \frac{\Gamma(z_1)\Gamma(z_2)}{\Gamma(z_1+z_2)}$, together with a union bound over all training points from a different class, gives the result. □

6.1.1 Outlier Removal and Improved Upper Bound

The guarantees above are good when the test points are far from training points from other classes in the feature space. This empirically holds true for good data and perfect embeddings—a so-called neural collapse phenomenon that the trained network converges to representations such that all points of class k get embedded close to a single point μ_k in the feature space [134]. But for noisy data and good-but-not-perfect embeddings, the condition may not hold. In Theorem 6.1.4 we show that we obtain almost the same upper bound on failure probability as above by exploiting the outlier removal threshold σ. Intuitively, outlier removal artificially induces well-separateness in the feature space, by deleting training examples that are close to other examples with a different label.

Our results will be good for distributions for which the induced distribution $\mathcal{D}_\sigma$ after the preprocessing step of Algorithm 10 satisfies the following property with small $N = \sum_y |\mathcal{B}^y|$.

Definition 22. *A distribution $\mathcal{D}$ is σ-separably $\{\mathcal{B}^y\}$-coverable if all points in the support of the marginal distribution $\mathcal{D}_{F(\mathcal{X})|y}$ over $\mathbb{R}^{n_2}$ can be covered by balls in the set $\mathcal{B}^y = \{\mathbb{B}_1^y, \ldots, \mathbb{B}_{N_y}^y\}$, of radius $\tau/2$ such that*

$$\min_{\substack{F(\mathbf{x})\in\mathbb{B}_i^y, F(\mathbf{x}')\in\mathbb{B}_j^{y'}, \\ y\neq y'}} \mathsf{dist}(F(\mathbf{x}), F(\mathbf{x}')) \geq \sigma.$$

In addition, we will assume that a test point $(\mathbf{x}, y)$ from the natural distribution $\mathcal{D}$ has the property that $\mathbf{x}$ is covered by some ball in $\mathcal{B}^y$ with high probability.

Theorem 6.1.4. *Suppose the distribution $\mathcal{D}_\sigma$ induced by the preprocessing step of Algorithm 10 is σ-separably $\{\mathcal{B}^y\}$-coverable with finite $N = \sum_y |\mathcal{B}^y|$. Let $\Pr_{\mathbf{x},y\sim\mathcal{D}}[\mathbf{x} \in \cup_{\mathbb{B}_i\in\mathcal{B}^y}\mathbb{B}_i] \geq 1-\gamma$. If $\tau = o(\sigma)$, the robust error of Algorithm 10 on any test point $\mathbf{x} \sim \mathcal{D}_{F(\mathcal{X})}$ is at most*

$$\mathcal{O}\left(N\left(\frac{c\tau}{(\sigma+\tau/2)\sqrt{1-\frac{n_3}{n_2}}}\right)^{n_2-n_3} + Nc_0^{n_2-n_3} + \gamma\right),$$

where $c > 0$ and $0 < c_0 < 1$ are absolute constants.

Proof. Let $\mathbf{x}, y \sim \mathcal{D}$. We will bound the probability the adversary succeeds for a test point $\mathbf{x}$ covered by $\cup_y \mathcal{B}^y$, that is $\Pr[\text{adversary succeeds on } \mathbf{x} \mid \mathbf{x} \in \cup_{\mathbb{B}_i \in \mathcal{B}^y} \mathbb{B}_i]$. Let $\mathbf{x}_i$ be a training point that survives the preprocessing step of Algorithm 10, and belongs to a different class than $\mathbf{x}$. By the covering assumption, $\mathbf{x}_i \in \mathbb{B}_j^{y'}$ for some $y' \neq y$ and $\mathbb{B}_j^{y'} \in \mathcal{B}^{y'}$. Let $\mathbf{c}$ denote the center of $\mathbb{B}_j^{y'}$. By the σ-separable property, we have $\mathsf{dist}(\mathbf{x}, \mathbf{c}) \geq \sigma + \tau/2$. Moreover, to succeed by perturbing close to any training point in $\mathbb{B}_j^{y'}$, the adversary must perturb to a point at distance at most $\tau + \tau/2 = 3\tau/2$ from $\mathbf{c}$ (by triangle inequality).

Using the same argument as in Theorem 6.1.3, the adversary succeeds in causing misclassification by perturbing $\mathbf{x}$ close to a point in $\mathbb{B}_j^{y'}$ with probability at most

$$\left(\frac{c\tau}{(\sigma + \tau/2)\sqrt{1 - \frac{n_3}{n_2}}} \right)^{n_2 - n_3} + c_0^{n_2 - n_3}$$

over the randomness of the adversarial subspace, for absolute constants $c > 0$ and $0 < c_0 < 1$. By a union bound, the adversary's success probability is at most N times the above quantity, conditioned on $\mathbf{x} \in \cup_{\mathbb{B}_i \in \mathcal{B}^y} \mathbb{B}_i$. Finally by assumption $\Pr_{\mathbf{x}, y \sim \mathcal{D}}[\mathbf{x} \notin \cup_{\mathbb{B}_i \in \mathcal{B}^y} \mathbb{B}_i] \leq \gamma$, and using the law of total probability we get the desired upper bound. □

6.1.2 A More General Adversary with Bounded Density

We extend our results in Theorem 6.1.3 to a more general class of adversaries, which have a bounded density over the space of linear subspaces of a fixed dimension n_3 and the adversary can perturb a test feature vector arbitrarily in the sampled adversarial subspace. Specifically, a distribution is said to be *κ-bounded* if the corresponding probability density $f(x)$ satisfies, $\sup_x f(x) \leq \kappa$. For example, the standard normal distribution $\mathcal{N}(\mu, \sigma)$ is $\frac{1}{\sqrt{2\pi}\sigma}$-bounded.

Theorem 6.1.5. *Consider the setting of Theorem 6.1.3 , with an adversary having a κ-bounded distribution over the space of linear subspaces of a fixed dimension n_3 for perturbing the test point. If $\mathbb{E}(\tau, r)$ denotes the bound on error rate in Theorem 6.1.3 for* RobustClassifier$(\tau, 0)$ *in Algorithm 10, then the error bound of the same algorithm against the κ-bounded adversary is $\mathcal{O}(\kappa \mathbb{E}(\tau, r))$.*

Proof. To argue upper bounds on failure probability, we consider the set of adversarial subspaces which can allow the adversary to perturb the test point x close to a training point x'. Let $\mathcal{S}(x', \tau)$ denote the subset of linear subspaces of dimension n_3 such that for any $S \in \mathcal{S}(x', \tau)$ there exists $v \in S$ with $x + v \in \mathcal{B}(x', \tau)$. Note that we can upper bound the fraction of the total probability space occupied by $\mathcal{S}(x', \tau)$ by $\frac{1}{m}\mathbb{E}(\tau, r)$, where constants in n_2, n_3 have been suppressed. If we show that $\mathcal{S}(x', \tau)$ is a measurable set, we can use the κ-boundedness of the adversary distribution to claim that the failure probability for misclassifying as x' is upper bounded by $\kappa \mathsf{vol}(\mathcal{S}) \frac{1}{m}\mathbb{E}(\tau, r) = \mathcal{O}\left(\frac{\kappa}{m}\mathbb{E}(\tau, r)\right)$, since the volume of the complete adversarial space $\mathcal{S}$ is a constant in n_2, n_3. In Lemma 6.1.6, we make the stronger claim that $\mathcal{S}(x', \tau)$ is convex. We can then use a union bound on the training points to get a bound on the total failure probability as $\mathcal{O}\left(\kappa \mathbb{E}(\tau, r)\right)$. □

The following lemma establishes a useful convexity property for the adversarial linear subspaces.

Lemma 6.1.6. *Let $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^{n_2}, \tau \in \mathbb{R}^+$ and $\mathcal{S}(x', \tau)$ denote the subset of linear subspaces of dimension n_3 such that for any $S \in \mathcal{S}(x', \tau)$ there exists $v \in S$ with $x + v \in \mathcal{B}(x', \tau)$. The set $\mathcal{S}(x', \tau)$ is convex.*

Proof. Let $S, S' \in \mathcal{S}(x', \tau)$. Then we have $v \in S, v' \in S'$ such that $x + v, x + v' \in \mathcal{B}(x', \tau)$. Let $S^* = \alpha S + (1 - \alpha)S'$, $\alpha \in [0, 1]$. Pick $v^* = \alpha v + (1 - \alpha)v' \in S^*$. $x + v^*$ must lie in $\mathcal{B}(x', \tau)$ by convexity of $\mathcal{B}(x', \tau)$. □

6.2 Small abstention rate

If natural data has the property that for every label class, one can cover most of the probability mass of the class with not too many (potentially overlapping) balls of at least some minimal probability mass, then our algorithm will have a low abstention rate.

Definition 23. *A distribution $\mathcal{D}$ is (δ, β, N)-coverable if at least a $1 - \delta$ fraction of probability mass of the marginal distribution $\mathcal{D}_{F(\mathcal{X})}$ over $\mathbb{R}^{n_2}$ can be covered by N balls $\mathbb{B}_1$, $\mathbb{B}_2$, ... $\mathbb{B}_N$ of radius $\tau/2$ and of mass $\Pr_{\mathcal{D}_{F(\mathcal{X})}}[\mathbb{B}_k] \geq \beta$.*

Intuitively, if a set of balls cover (most of) the distribution and we sample enough points from the distribution, we should get at least one sample from each ball and our algorithm will not abstain on the covered points. Formally, we show the following guarantee on the abstention rate on distributions that are (δ, β, N)-coverable w.r.t. threshold τ.

Theorem 6.2.1. *Suppose that $F(\mathbf{x}_1), ..., F(\mathbf{x}_m)$ are m training instances i.i.d. sampled from marginal distribution $\mathcal{D}_{F(\mathcal{X})}$. If the distribution $\mathcal{D}$ is (δ, β, N)-coverable, for sufficiently large $m \geq \frac{1}{\beta} \ln \frac{N}{\gamma}$, with probability at least $1 - \gamma$ over the sampling, we have $\Pr[\cup_{i=1}^{m} \mathbb{B}(F(\mathbf{x}_i), \tau)] \geq 1 - \delta$.*

Proof. Fix ball $\mathbb{B}_i$ in the cover from Definition 23. Let B_i denote the event that no point is drawn from ball $\mathbb{B}_i$ over the m samples. Since successive draws are independent, and by Definition 23 $\Pr_{\mathcal{D}_{F(\mathcal{X})}}[\mathbb{B}_i] \geq \beta$, we have that $\Pr[B_i] \leq (1 - \beta)^m \leq \exp(-\beta m)$. Further, by a union bound over N balls $\Pr[\cup_i B_i] \leq N \exp(-\beta m) \leq \gamma$, for $m \geq \frac{1}{\beta} \ln \frac{N}{\gamma}$.

Therefore, with probability at least $1 - \gamma$ for all $k \in [N]$ there is at least a sample $F(\mathbf{x}_{i_k}) \in \{F(\mathbf{x}_1), F(\mathbf{x}_2), ..., F(\mathbf{x}_m)\}$ such that $F(\mathbf{x}_{i_k}) \in \mathbb{B}_k$. This implies $\cup_{i=1}^{m} \mathbb{B}(F(\mathbf{x}_i), \tau) \supseteq \cup_{k=1}^{N} \mathbb{B}_k$, since $\mathbb{B}_k$ is a ball of radius $\tau/2$. So with probability at least $1 - \gamma$ over the sampling, we have $\Pr[\cup_{i=1}^{m} \mathbb{B}(F(\mathbf{x}_i), \tau)] \geq \Pr[\cup_{k=1}^{N} \mathbb{B}_k] \geq 1 - \delta$. □

Note that in the special case that the N balls are disjoint and each has probability mass $\beta = 1/N$, then $m = \Omega(N \log N)$ samples are also necessary to get a point inside each ball, by a standard coupon-collector analysis.

Theorem 6.2.1 implies that if we have a covering with N balls, each with probability mass at least β and large enough sample size m, with probability at least $1-\gamma$ over the sampling, we have $\Pr[\cup_{i=1}^{m} \mathbb{B}(F(\mathbf{x}_i), \tau)] \geq 1 - \delta$. Therefore, with high probability, the algorithm will output "don't

know" only for a δ fraction of natural data. Below we give an example of a distribution where our algorithm will simultaneously achieve low robust error and low natural abstention rates.

Example distribution where Algorithm 10 is robust with low abstention rate. Our example will consist of well-separated data in the feature space. Suppose $\mathcal{D}_{F(\mathcal{X})|y}$ for each label class y consists of the uniform distribution over N_y n_2-balls of radius $\tau/2$ centered at axis-aligned unit vectors $\{\mathbf{e}_j \mid j \in S_y\}$, where $S_y \subset [n_2]$ is the set of axes with balls labeled by y, with $\tau < 1/3$ and $S_y \cap S_{y'} = \emptyset$ for $y \neq y'$. Further let $m = n_2 \log \frac{n_2}{\gamma}$ for some absolute constant $\gamma \in (0, 1)$, so this distribution is (δ, β, N)-coverable with $\delta = 0$, $\beta = 1/N$ and $N = n_2$. If $n_3 = 1$, by the argument presented above, the robust error of Algorithm 10 is bounded by $O(n_2 \log n_2 \tau^{n_2 - 1}) = o(1)$. Thus, in this setting, our algorithm enjoys low robust error without abstaining too much (for sufficiently large n_2).

Our result above also implies another bound on the abstention rate on natural data based on the *doubling dimension* [48] of the data distribution (Theorem 8, [30]).

6.3 Robustness vs. abstention trade-off

Given an embedding function F and a classifier f_τ which outputs either a predicted class if the nearest neighbor is within distance τ of a test point or abstains from predicting if not (see Algorithm 10), we want to evaluate the performance of f_τ on a test set $\mathcal{T}$ against an adversary which can perturb a test feature vector in a random n_3-dimensional subspace $S \sim \mathcal{S}$. To this end, we define

Definition 24 (Robust error)**.** *Let $\mathcal{E}_{\text{adv}}(\tau, S) := \frac{1}{|\mathcal{T}|} \sum_{(\mathbf{x},\mathbf{y}) \in \mathcal{T}} \mathbf{1}\{\exists \mathbf{e} \in S + F(\mathbf{x}) \subseteq \mathbb{R}^{n_2}$ such that $f_\tau(\mathbf{e}) \neq \mathbf{y}$ and $f_\tau(\mathbf{e})$ does not abstain$\}$ denote the robust error on the test set $\mathcal{T}$, for n_3-dimensional perturbation subspace S and threshold setting τ in Algorithm 10. Also define average robust error as $\mathcal{E}_{\text{adv}}(\tau) := \mathbb{E}_{S \sim \mathcal{S}} [\mathcal{E}_{\text{adv}}(\tau, S)]$ for distribution $\mathcal{S}$ over n_3-dimensional subspaces (assumed to be the uniform distribution unless stated otherwise) and estimated robust error over a set $\hat{S}$ of subspaces as $\hat{\mathcal{E}}_{\text{adv}}(\tau, \hat{\mathcal{S}}) := \frac{1}{|\hat{\mathcal{S}}|} \sum_{S \in \hat{\mathcal{S}}} \mathcal{E}_{\text{adv}}(\tau, S)$. Let $\hat{\mathcal{S}}$ consist of multiple samples drawn from $\mathcal{S}$, and for conciseness, we will often denote $\hat{\mathcal{E}}_{\text{adv}}(\tau, \hat{\mathcal{S}})$ by $\hat{\mathcal{E}}_{\text{adv}}(\tau)$ and $\hat{\mathcal{S}}$ will be implicit from context.*

$\hat{\mathcal{E}}_{\text{adv}}(\tau)$ gives an easier-to-compute surrogate to $\mathcal{E}_{\text{adv}}(\tau)$, by drawing subspaces in $\hat{\mathcal{S}}$ according to $\mathcal{S}$. For an abstentive classifier, the robust error can be trivially minimized by abstaining everywhere. We will therefore also need to control the abstention rate on unperturbed data.

Definition 25 (Natural abstention rate)**.** *Define the abstention rate on the unperturbed test set $\mathcal{T}$ as $\mathcal{D}_{\text{nat}}(\tau) := \frac{1}{|\mathcal{T}|} \sum_{(\mathbf{x},\mathbf{y}) \in \mathcal{T}} \mathbf{1}\{f_\tau(F(\mathbf{x}))$ abstains$\}$.*

$\mathcal{E}_{\text{adv}}(\tau)$ and $\mathcal{D}_{\text{nat}}(\tau)$ are both monotonic in τ; while the former is non-decreasing, the latter is non-increasing (Lemma 6.3.1).

Lemma 6.3.1. *Robust error $\mathcal{E}_{\text{adv}}(\tau, S)$ is monotonically non-decreasing in τ for any S. Further, natural abstention rate $\mathcal{D}_{\text{nat}}(\tau)$ is monotonically non-increasing in τ.*

Lemma 6.3.1 further implies that $\mathcal{E}_{\text{adv}}(\tau)$ and $\hat{\mathcal{E}}_{\text{adv}}(\tau)$ are also monotonic non-decreasing in τ. The robust error $\mathcal{E}_{\text{adv}}(\tau)$ is optimal at $\tau = 0$, but this implies that we abstain from prediction

Algorithm 11 Exponential Forecaster Algorithm [15]

Input: step size parameter $\lambda \in (0, 1]$.
Output: thresholds τ_t for times $t = 1, 2, \ldots, T$.
Set $w_1(\tau) = 1$ for all $\tau \in [0, \tau_{\max}]$.
for $t = 1, 2, \ldots, T$ **do**
    $W_t := \int_{[0,\tau_{\max}]} w_t(\tau) d\tau$.
    Sample τ with probability proportional to $w_t(\tau)$, that is, with probability $p_t(\tau) = \frac{w_t(\tau)}{W_t}$. Output the sampled τ as τ_t.
    Observe $l_t(\cdot)$. Set $u_t(\tau) := 1 - \frac{l_t(\tau)}{1+c}$.
    For each $\tau \in [0, \tau_{\max}]$, set $w_{t+1}(\tau) = e^{\lambda u_t(\tau)} w_t(\tau)$.

all the time (that is, $\mathcal{D}_{\text{nat}}(0) = 1$). Conversely, we can minimize the abstention rate by not abstaining, that is, $\mathcal{D}_{\text{nat}}(\infty) = 0$ corresponding to vanilla nearest-neighbor, but this maximizes the robust error. This motivates us to consider the following objective function which combines robust error and natural abstention rate.

Definition 26 (Robust Chow's objective)**.** *Define $l(\tau) := \mathcal{E}_{\text{adv}}(\tau) + c\mathcal{D}_{\text{nat}}(\tau)$ as the robust Chow's objective, where c is a positive constant and denotes the cost of abstention. Define $\hat{l}(\tau) := \hat{\mathcal{E}}_{\text{adv}}(\tau) + c\mathcal{D}_{\text{nat}}(\tau)$ as the estimated robust Chow's objective.*

Definition 26 may be viewed as an adversarial version of Chow's objective for abstentive classifiers [61], which uses natural risk instead of adversarial risk. If, for example, we are willing to take a one percent increase of the abstention rate for a two percent drop in the error rate, we could set c to $\frac{1}{2}$. For a single test set $\mathcal{T}$, the abstention rate $\mathcal{D}_{\text{nat}}(\tau)$ can change at (at most) $|\mathcal{T}|$ 'critical' values of τ corresponding to nearest neighbor distances. Given oracle access to $\mathcal{E}_{\text{adv}}(\tau)$, we can minimize $l(\tau)$ over the given test sample with at most $|\mathcal{T}|$ evaluations. Suppose, however, the test data arrives sequentially in batches of size b, potentially from related tasks with different data distributions, and we need to figure out how to set the threshold τ for unseen tasks. As we will show, techniques from data-driven algorithm design [15, 24] can help approach this multi-task robustness setting.

Formally, we define our online learning setting as follows. Consider a game consisting of T rounds. In each round $t = 1, \ldots, T$, the learner is presented with a new test batch $\mathcal{T}_t$ of size b. In Theorem 6.3.2, we show no regret can be achieved for online learning of the threshold τ using test batches of size b (consisting of unperturbed points) on which the learner chooses abstention threshold τ_t, that is, predicting using classifier f_{τ_t}. Let l_t (resp. $\hat{l}_t$) be the (resp. estimated) robust Chow's objective on the test set $\mathcal{T}_t$. The learner suffers loss $l_t(\tau_t)$ and observes $l_t(\tau)$. The goal of the learner is to minimize total expected regret, defined as $R_T := \mathbb{E}\left[\sum_{t=1}^{T} l_t(\tau_t) - \min_\tau \sum_{t=1}^{T} l_t(\tau)\right]$, where the expectation is over the randomness of the loss functions as well as learner's internal randomness.

Our main result is the following theorem (Theorem 6.3.2) in the above setting. Our proof strategy is to show that the sequence of loss functions $l_t(\tau)$ is *dispersed* (Definition 2) provided data distribution $\mathcal{D}$ is smooth.

Theorem 6.3.2. *Consider the online learning setting described above. Assume $\tau \in [0, \tau_{\max}]$ with $\tau_{\max} = o(r)$, $r > 1$, and each test batch $\mathcal{T}_t$ is sampled from a data distribution $\mathcal{D}$ that has κ-bounded density. If τ_t is set using a continuous version of the multiplicative updates algorithm, Algorithm 11, for T rounds of the online game, then with probability at least $1 - \delta$, the total expected regret of the learner for the loss sequence $l_t(\tau)$ is bounded by $O\left(\sqrt{n_2 T \log\left(\frac{\kappa m b \tau_{\max} T}{\delta}\right)}\right)$. Here b is the batch size and r is the smallest distance between points of different labels.*

Proof. We show the sequence of loss functions $l_t(\tau)$ is (w, k)-*dispersed* (in the sense of [15]) in two steps. We first argue that the robust error part of the loss $l(\tau)$ is Lipschitz, and we further show that the natural abstention rate is piecewise constant with dispersed discontinuities.

A key challenge is to analyze the adversary success probability and show that $\mathcal{E}_{\text{adv}}(\tau)$ is Lipschitz for sufficiently small τ. In Lemma C.2.2 (see Appendix C.2 for a proof), we show that $\mathcal{E}_{\text{adv}}(\tau)$ is L-Lipschitz, where $L = O\left(m\tau_{\max}^{n_2-n_3-1}/r^{n_2-n_3}\right)$. Intuitively, for any test point the probability the adversary succeeds by perturbing to within a distance τ and $\tau + \Delta$ of a fixed training point can be upper bounded using arguments similar to our proof of bounds on robust error in Section C.1. A union bound over training points then gives the bound on L. Note that $\mathcal{D}_{\text{nat}}(\tau)$ is piecewise constant. This is because, for any set $\mathcal{T}_t$ of test points, we have at most $|\mathcal{T}_t|$ points corresponding to distances of the test points to the nearest training point, where the function value decreases by $\frac{1}{|\mathcal{T}_t|}$. Together with L-Lipschitzness of $\mathcal{E}_{\text{adv}}(\tau)$, this implies $l(\tau)$ is piecewise L-Lipschitz.

In Lemma C.2.5 we show that, for batch size b, $\mathcal{D}_{\text{nat}}(\tau)$ has $O(\kappa b m w \tau_{\max}^{n_2-1})$ discontinuities in expectation (over the data distribution) in any interval of width w. Note that if a discontinuity occurs within the interval $I = [\tau, \tau + w]$, then there must exist a test point $\mathbf{x}$ in the test set $\mathcal{T}$ for which the nearest-neighbor training point is at distance $\tau' \in I$. That is, the training point lies within $\mathcal{B}(\mathbf{x}, \tau + w) \setminus \mathcal{B}(\mathbf{x}, \tau)$. The proof involves bounding the fraction of points at distance $d \in [\tau, \tau + w]$ for any test point, using smoothness of the data distribution, and using a union bound over the b test points. See Appendix C.2 for a formal argument. Since $\mathcal{E}_{\text{adv}}(\tau)$ is Lipschitz continuous, $l(\tau)$ has at most $O\left(\kappa b m w \tau_{\max}^{n_2-1}\right)$ discontinuities in expectation in any w-interval.

Using a standard argument based on the VC-dimension of 1D intervals (for example, Theorem 7 in [19]), the maximum number of discontinuities in any interval of width w is $k = O\left(\kappa b m w \tau_{\max}^{n_2-1} T + \sqrt{T \log \frac{b}{\gamma}}\right)$ with high probability $1 - \gamma$. In other words, $l(\tau)$ is (w, k)-Lipschitz with high probability over the data distribution. This allows us to use a continuous version of standard Exponential Weights update introduced by [15] as our online algorithm (which we include as Algorithm 11 for completeness), for which they show an upper bound $O\left(\sqrt{T \log \frac{R}{w}} + k + wLT\right)$ on the expected regret if the sequence of loss functions is (w, k)-dispersed with L-Lipschitz pieces, where R is a bound on the diameter of the continuous domain ($R = \tau_{\max}$ in our setting).

	Contrastive	Prior Work		Ours ($\tau = 3.0$)			Ours ($\tau = 2.0$)		
		$\mathcal{E}_{\text{nat}}$	$\mathcal{E}_{\text{adv}}$	$\mathcal{E}_{\text{nat}}$	$\mathcal{E}_{\text{adv}}$	$\mathcal{D}_{\text{nat}}$	$\mathcal{E}_{\text{nat}}$	$\mathcal{E}_{\text{adv}}$	$\mathcal{D}_{\text{nat}}$
$(\sigma = 0)$	Self-supervised	8.9%	100.0%	15.4%	40.7%	2.2%	14.3%	26.2%	28.7%
	Supervised	5.6%	100.0%	5.7%	60.5%	0.0%	5.7%	33.4%	0.0%
$(\sigma = 0.9\tau)$	Self-supervised	8.9%	100.0%	**7.2%**	**9.4%**	**12.9%**	10.0%	17.7%	29.9%
	Supervised	5.6%	100.0%	6.2%	18.9%	0.0%	5.6%	22.0%	0.1%
$(\sigma = \tau)$	Self-supervised	8.9%	100.0%	1.1%	1.2%	33.4%	2.1%	3.1%	49.9%
	Supervised	5.6%	100.0%	1.9%	2.8%	10.6%	**4.1%**	**4.8%**	**3.3%**

Table 6.1: Natural error $\mathcal{E}_{\text{nat}}$ and robust error $\mathcal{E}_{\text{adv}}$ on the CIFAR-10 data set when $n_3 = 1$ and the 512-dimensional representations are learned by contrastive learning, where $\mathcal{D}_{\text{nat}}$ represents the fraction of each algorithm's output of "don't know" on the natural data. We report values for $\sigma \approx \tau$ as they tend to give a good abstention-error trade-off w.r.t. σ. Bold values correspond to parameter settings that minimize $\mathcal{E}_{\text{adv}} + \mathcal{D}_{\text{nat}}$ over the grid. Prior work [58, 102] uses no abstention.

Formally, we can apply Theorem C.2.6 with $w = \frac{1}{\kappa b m \tau_{\max}^{n_2-1}\sqrt{T}}$ to get the desired regret bound.

$$
\begin{aligned}
R_T &= O\left(\sqrt{T \log \frac{R}{w}} + k + wLT\right) \\
&\leq O\left(\sqrt{T \log \frac{\tau_{\max}}{(\kappa b m \tau_{\max}^{n_2-1}\sqrt{T})^{-1}}} + O\left(\sqrt{T} + \sqrt{T \log \frac{b}{\delta}}\right) + \frac{O(m\tau_{\max}^{n_2-n_3-1}/r^{n_2-n_3})}{\kappa b m \tau_{\max}^{n_2-1}\sqrt{T}} \cdot T\right) \\
&\leq O\left(\sqrt{T \log\left(\frac{\kappa m b \tau_{\max}^{n_2} T}{\delta}\right)}\right),
\end{aligned}
$$

where the first inequality holds with probability at least $1 - \delta$. □

A similar no-regret learning guarantee can also be given for the estimated robust Chow's objective $\hat{l}(\tau)$. In practice $l(\tau)$ can be hard to compute, but as discussed above the learner can more easily estimate this loss by computing $\hat{l}(\tau)$. The key difference in the proof is that the estimated robust error $\hat{\mathcal{E}}_{\text{adv}}(\tau)$ is piecewise constant, while $\mathcal{E}_{\text{adv}}(\tau)$ was shown to be Lipschitz for small τ.

In this work we restrict our attention to the *full information* setting where entire function $l_t(\tau)$ is available to the learner after the prediction in round t. It is an interesting future question to model the adversary with bandit feedback where only $l_t(\tau_t)$ is revealed to the learner. The test sets $\mathcal{T}_t$ may be adversarial as long as they are generated by smooth but possibly different data distributions (in the sense of Theorem 6.3.2). Our experiments in Section 6.4 indicate Algorithm 10 can be made more effective by tuning both parameters τ and σ together. Effective tuning of data-driven algorithms with multiple parameters is an interesting research direction. Finally, we perform the analysis for tuning our relatively simple thresholded nearest-neighbor approach, but data-driven algorithm design may prove useful for selecting the best data-specific robust approach from candidate algorithms more generally.

Remark 5. *A simple goal for setting τ is to fix an upper limit d^* on $\mathcal{D}_{\text{nat}}(\tau)$, corresponding to a maximum abstention rate allowed on the natural data. It is straightforward to search for an optimal τ^* such that $\mathcal{D}_{\text{nat}}(\tau^*) = \max_{\tau, \mathcal{D}_{\text{nat}}(\tau) \leq d^*} \mathcal{D}_{\text{nat}}(\tau)$—simply use the nearest neighbor distances (to training examples) for the test points to compute the abstention rate at any τ, and do a binary search for d^*. For $\tau < \tau^*$ we have a higher abstention rate, and when $\tau > \tau^*$ we have a higher robust error rate. For more sophisticated goals, for example minimizing objectives that depend on both $\mathcal{E}_{\text{adv}}(\tau)$ and $\mathcal{D}_{\text{nat}}(\tau)$, we may not be able to perform a binary search, though a linear search would still suffice. Here we have considered a setting where we have multiple test sets, conceptually coming from different but related tasks in some domain, and rather than separately performing this parameter tuning on each task, we want instead to learn a common value of τ that works well across all the tasks.*

6.3.1 A simple intuitive example with exact calculation demonstrating significance of data-driven algorithm design

The significance of data-driven design in this setting is underlined by the following two observations. Firstly, as noted above, optimization for τ across problem instances is difficult due to the non-Lipschitz nature of $\mathcal{D}_{\text{nat}}(\tau)$ and the intractability of characterizing the objective function $l(\tau)$ exactly due to $\mathcal{E}_{\text{adv}}(\tau)$. Secondly, the optimal value of τ can be a complex function of the data geometry and sampling rate. We illustrate this by exact computation of optimal τ for a simple intuitive setting: consider a binary classification problem where the features lie uniformly on two one-dimensional manifolds embedded in two-dimensions (that is, $n_2 = 2$, see Figure 6.4). Assume that the adversary perturbs in a uniformly random direction ($n_3 = 1$). Further assume that our training set consists of $2m$ examples, m from each class. In this toy setting, we show that the optimal threshold varies with data-specific factors.

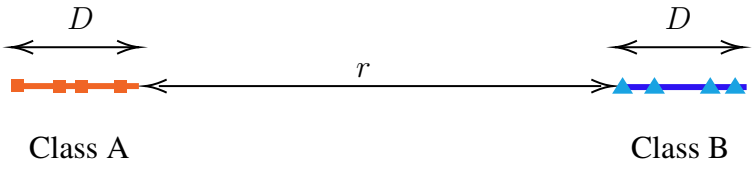


Figure 6.4: A simple example where we compute the optimal value of the abstention threshold exactly. Classes A and B are both distributed respectively on segments of length D, embedded collinear and at distance r in $\mathbb{R}^2$.

Formal setting: We set the feature and adversary dimensions as $n_2 = 2, n_3 = 1$. Examples of class A are all located on the segment $S_A = [(0, 0), (D, 0)]$, similarly instances of class B are located on $S_B = [(D + r, 0), (2D + r, 0)]$ (where $[\mathbf{a}, \mathbf{b}] := \{\alpha\mathbf{a} + (1 - \alpha)\mathbf{b} \mid \alpha \in [0, 1]\}$). The data distribution returns an even number of samples, $2m$, with $m > 0$ points each drawn uniformly from S_A and S_B. For this setting, we show that the optimal value of the threshold is a function of both the geometry (D, r) and the sampling rate (m).

Theorem 6.3.3. *Let $\tau^* := \text{argmin}_{\tau \in \mathbb{R}^+} l(\tau)$. For the setting considered above, if we further*

assume $D = o(r)$ *and* $m = \omega\left(\log\left(\frac{2\pi cr}{D}\right)\right)$*, then there is a unique value of* τ^* *in* $[0, D/2)$*. Further,*

$$\tau^* = \begin{cases} \Theta\left(\frac{D\log((\pi crm)/D)}{m}\right), & \text{if } \frac{1}{m} < \frac{\pi cr}{D}; \\ 0, & \text{if } \frac{\pi cr}{D} \leq \frac{1}{m}. \end{cases}$$

Proof. We compute the robust error $\mathcal{E}_{\text{adv}}(\tau)$ and abstention rate $\mathcal{D}_{\text{nat}}(\tau)$ as functions of τ. Even with $D = o(r)$, the exact computation of the robust error as a simple closed form is difficult without further assuming $\tau = o(r)$ as well. Fortunately, by Lemma 6.3.4, we only need to consider $\tau \leq D$. For this case, indeed $\tau = o(r)$. We compute the abstention and robust error rates in Lemmas 6.3.5 and 6.3.6, respectively. This gives us, for $\tau \leq D$,

$$\begin{aligned} l(\tau) =& \frac{\tau}{\pi r}\left(1 - \frac{m+3}{m+1}\cdot\Theta\left(\frac{D}{r}\right)\right) - \Theta\left(\left(\frac{\tau}{r}\right)^3\right) \\ &+ \frac{c}{m+1}\left[2\left(1-\frac{\tau}{D}\right)^{m+1} + (m-1)\mathbb{I}_{\tau\leq D/2}\left(1-\frac{2\tau}{D}\right)^{m+1}\right]. \end{aligned}$$

For $\tau \leq D/2$,

$$\begin{aligned} l'(\tau) =& \frac{1}{\pi r}\left(1 - \frac{m+3}{m+1}\cdot\Theta\left(\frac{D}{r}\right)\right) - \Theta\left(\frac{1}{r}\left(\frac{\tau}{r}\right)^2\right) \\ &- \frac{2c}{D}\left[\left(1-\frac{\tau}{D}\right)^{m} + (m-1)\left(1-\frac{2\tau}{D}\right)^{m}\right]. \end{aligned}$$

We need to consider two cases.
Case 1. $\frac{\pi cr}{D} \leq \frac{1}{m}$. In this case $l'(0) = \frac{1}{\pi r} - \frac{2cm}{D} \geq 0$. Since $l''(\tau) \geq 0$, so we must have the only minimum at $\tau = 0$.

Case 2. $\frac{1}{m} < \frac{\pi cr}{D}$. $l'(0) = \frac{1}{\pi r} - \frac{2cm}{D} < 0$. Also $l'(D/2) = \frac{1}{\pi r} - \frac{2c}{D2^m} > 0$ since $m > \log\left(\frac{2\pi cr}{D}\right)$. But $l''(\tau) \geq 0$, so we must have a unique local minimum in $(0, D/2)$, which is the global minimum. Further, define y as $\tau = \frac{D}{m}\log y$. Now if $y = 2^{o(m)}$, we have $\frac{\tau}{D} = o(1)$, or

$$\left(1-\frac{\tau}{D}\right)^m = \exp\left(m\log\left(1-\frac{\tau}{D}\right)\right) = y^{-1-o(1)}.$$

If $y > 1$, for $y = \frac{2\pi crm}{D}$,

$$\begin{aligned} l'(\tau) &= \frac{1}{\pi r} - \frac{2c}{D}\left[\left(\frac{D}{2\pi crm}\right)^{1+o(1)} + (m-1)\left(\frac{D}{2\pi crm}\right)^{2+o(1)}\right] \\ &> \frac{1}{\pi r} - \frac{2c}{D}\left[\left(\frac{D}{2\pi crm}\right)^{1} + (m-1)\left(\frac{D}{2\pi crm}\right)^{1}\right] \\ &= \frac{1}{\pi r} - \frac{2c}{D}\left[\frac{D}{2\pi cr}\right] = 0, \end{aligned}$$

and for $y = \left(\frac{2\pi cr(m-1)}{D}\right)^{1/4}$,

$$\begin{aligned} l'(\tau) &= \frac{1}{\pi r} - \frac{2c}{D}\left[\left(\frac{D}{2\pi cr(m-1)}\right)^{\frac{1}{4}+o(1)} + (m-1)\left(\frac{D}{2\pi crm}\right)^{\frac{1}{2}+o(1)}\right] \\ &< \frac{1}{\pi r} - \frac{2c}{D}\left[\left(\frac{D}{2\pi cr(m-1)}\right)^{1} + (m-1)\left(\frac{D}{2\pi cr(m-1)}\right)^{1}\right] \\ &= \frac{-1}{\pi r(m-1)} < 0. \end{aligned}$$

Together, we get that $\tau^* = \Theta\left(\frac{D\log((\pi crm)/D)}{m}\right)$ in this case. □

Lemma 6.3.4. *In the setting of Theorem 6.3.3, $l(\tau)$ is monotonically non-decreasing for $\tau > D$.*

Proof. Note that $\mathcal{D}_{\text{nat}}(\tau) = 0$ for $\tau > D$ as long as $m > 0$, since any test point of a class must be within D of every training point of that class. Hence, it suffices to note that $\mathcal{E}_{\text{adv}}(\tau)$ is monotonically non-decreasing in τ (increasing the threshold can only increase the ability of the adversary to successfully perturb to the opposite class). □

Lemma 6.3.5. *In the setting of Theorem 6.3.3, the abstention rate is given by*

$$\mathcal{D}_{\text{nat}}(\tau) = \frac{1}{m+1}\left[2\mathbb{I}_{\tau\le D}\left(1-\frac{\tau}{D}\right)^{m+1} + (m-1)\mathbb{I}_{\tau\le D/2}\left(1-\frac{2\tau}{D}\right)^{m+1}\right].$$

Proof. Note that for $\tau \ge D$, if $m > 0$, we never abstain on any test point. So we will assume $\tau \le D$ in the following. Consider a test point $\mathbf{x} = (x, 0)$ sampled from class A (class B is symmetric, so the overall abstention rate is the same is that of points drawn from class A). Let $\text{nbd}_{\mathbf{x}}(\tau)$ denote the intersection of a ball of radius τ around x with S_A. For x to be classified as 'don't know', we must have no training point sampled from $\text{nbd}_{\mathbf{x}}(\tau)$. This happens with probability $\left(1 - \frac{|\text{nbd}_{\mathbf{x}}(\tau)|}{D}\right)^m$, where $|\text{nbd}_{\mathbf{x}}(\tau)|$ is the size of $\text{nbd}_{\mathbf{x}}(\tau)$ and is given by

$$|\text{nbd}_{\mathbf{x}}(\tau)| = \begin{cases} \min\{x+\tau, D\}, & x < \tau; \\ \min\{2\tau, D\}, & \tau \le x \le D-\tau; \\ \min\{D-x+\tau, D\}, & x > D-\tau. \end{cases}$$

Averaging over the distribution of test points $\mathbf{x}$, we get

$$\begin{aligned} \mathcal{D}_{\text{nat}}(\tau) &= \frac{1}{D}\int_0^D \left(1 - \frac{|\text{nbd}_{\mathbf{x}}(\tau)|}{D}\right)^m dx \\ &= \frac{1}{m+1}\left[2\left(1-\frac{\tau}{D}\right)^{m+1} + (m-1)\mathbb{I}_{\tau\le D/2}\left(1-\frac{2\tau}{D}\right)^{m+1}\right]. \end{aligned}$$

□

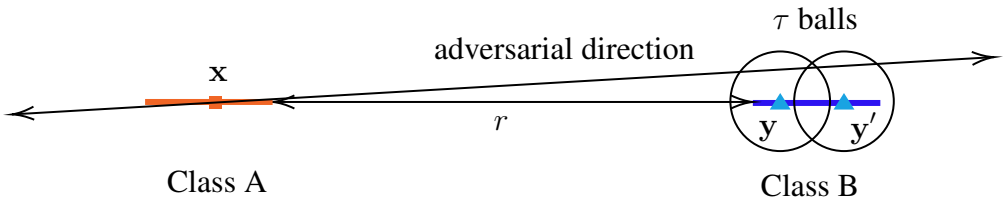


Figure 6.5: It suffices to consider the nearest point of the opposite class for adversarial perturbation.

Lemma 6.3.6. *In the setting of Theorem 6.3.3, the robust accuracy rate for $\tau \leq D$ is given by*

$$\mathcal{A}_{\text{adv}}(\tau) = 1 - \frac{\tau}{\pi r}\left(1 - \frac{m+3}{m+1} \cdot \Theta\left(\frac{D}{r}\right)\right) - \Theta\left(\left(\frac{\tau}{r}\right)^3\right).$$

Proof. Consider a test point $\mathbf{x} = (x, 0)$ from S_A. Let $\mathbf{y} = (y, 0)$ denote the nearest point in S_B. In the given geometry, it is easy to see that if $\mathbf{x}$ can be perturbed into the τ neighborhood of some point $\mathbf{y}' \in S_B$ when moved along a fixed direction, then it must be possible to perturb it into the τ neighborhood of $\mathbf{y}$ (Figure 6.5). Therefore it suffices to consider directions where perturbation to the τ-ball around $\mathbf{y}$ is possible.

Therefore the probability of adversary's success for $\mathbf{x}$, given $\mathbf{y}$ is the nearest point of the opposite class, is

$$\text{err}_{\mathbf{x}|\mathbf{y}}(\tau) = \frac{1}{\pi}\arcsin\left(\frac{\tau}{y-x}\right) = \frac{1}{\pi}\arcsin\left(\frac{\tau}{r+d}\right),$$

where $d = y - x - r \in [0, 2D]$. Now since $\tau \leq D = o(r)$, we have

$$\text{err}_{\mathbf{x}|\mathbf{y}}(\tau) = \frac{\tau}{\pi(r+d)} + \Theta\left(\left(\frac{\tau}{r}\right)^3\right) = \frac{\tau}{\pi r}\left(1 - \Theta\left(\frac{d}{r}\right)\right) + \Theta\left(\left(\frac{\tau}{r}\right)^3\right).$$

We can now compute the average error using the probability distributions for $\mathbf{x}$ and $\mathbf{y}$, $\mathbf{x}$ is a uniform distribution over S_A, while $\mathbf{y}$ is a nearest-neighbor distribution.

$$p(x) = \frac{1}{D}, \;\; p(y) = \frac{m}{D}\left(1 - \frac{y-r-D}{D}\right)^{m-1}.$$

The average value of d is

$$\bar{d} = \int_0^D \int_0^D (y' + x')\frac{m}{D}\left(1 - \frac{y'}{D}\right)^{m-1} dy' \frac{dx'}{D} = \frac{D(m+3)}{2(m+1)}.$$

Using this to compute the average of $\text{err}_{\mathbf{x}|\mathbf{y}}(\tau)$ gives the result. □

6.4 Contrastive learning experiments

We will present results for self-supervised [58] and supervised [102] contrastive learning experiments. We verify the robustness of Algorithm 10 when the representations are learned

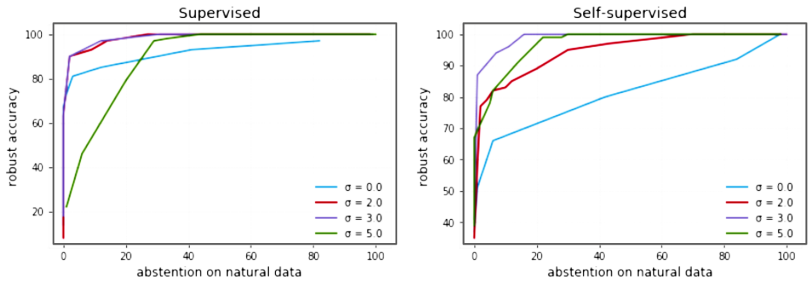


Figure 6.6: Adversarial accuracy (that is, rate of adversary failure) vs. abstention rate as threshold τ varies for $n_3 = 1$ and different outlier removal thresholds σ. Each colored line corresponds to a fixed σ, as τ is varied from 0 (always abstain) to infinity (vanilla nearest-neighbor).

by contrastive learning. Given embedding function F and a classifier f which outputs either a predicted class or abstains from predicting, recall that we define the natural and robust errors, respectively, as $\mathcal{E}_{\text{nat}}(f) := \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}}\mathbf{1}\{f(F(\mathbf{x})) \neq \mathbf{y}$ and $f(F(\mathbf{x}))$ does not abstain$\}$, and $\mathcal{E}_{\text{adv}}(f) := \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D},S\sim\mathcal{S}}\mathbf{1}\{\exists \mathbf{e} \in S + F(\mathbf{x}) \subseteq \mathbb{R}^{n_2}$ s.t. $f(\mathbf{e}) \neq \mathbf{y}$ and $f(\mathbf{e})$ does not abstain$\}$, where $S \sim \mathcal{S}$ is a random adversarial subspace of $\mathbb{R}^{n_2}$ with dimension n_3. We also define $\mathcal{D}_{\text{nat}}(f) := \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}}\mathbf{1}\{f(F(\mathbf{x}))$ abstains$\}$ as the abstention rate on the natural examples. Note that the robust error is always at least as large as the natural error. Consult [30] for additional experimental details. We summarize the main results here in Table 6.1 and Figure 6.6.

The threshold parameter τ captures the trade-off between the robust accuracy $\mathcal{A}_{\text{adv}} := 1 - \mathcal{E}_{\text{adv}}$ and the abstention rate $\mathcal{D}_{\text{nat}}$ on the natural data. We report both metrics for different values of τ for supervised and self-supervised contrastive learning. The supervised setting enjoys higher adversarial accuracy and a smaller abstention rate for fixed τ's due to the use of extra label information. We plot $\mathcal{A}_{\text{adv}}$ against $\mathcal{D}_{\text{nat}}$ for Algorithm 10 as hyperparameters are varied. For small τ, both accuracy and abstention rate approach 1.0. As the threshold increases, the abstention rate decreases rapidly and our algorithm enjoys good accuracy even with small abstention rates. For $\tau \to \infty$ (that is the nearest neighbor search), the abstention rate on the natural data $\mathcal{D}_{\text{nat}}$ is 0% but the robust accuracy is also roughly 0%. Increasing σ (for small σ) gives us higher robust accuracy for the same abstention rate. Too large σ may also lead to degraded performance (Figure 6.6).

The results in this chapter are joint work with Nina Balcan, Avrim Blum and Hongyang Zhang [30], and have appeared in JMLR 2023.

Chapter 7

Multiple similar tasks

Suppose we have multiple similar tasks involving hyperparameter tuning, in the sense that there is a shared good parameter value across all the tasks on average. The tasks appear online, but the learner has the opportunity to learn from previously encountered similar tasks. This setting seems more practical for modeling real-world repeated problem instances in typical applications, as the distributional setting may be too optimistic and the online (adversarial, possibly 'smoothed' in the sense of dispersion) setting may be too pessimistic.

In this chapter, we analyze the meta-learning of the initialization and step-size of learning algorithms for piecewise-Lipschitz functions. Starting from regret bounds for the exponential forecaster on losses with dispersed discontinuities due to Balcan et al. [15], we generalize them to be initialization-dependent and then use this result to propose a practical meta-learning procedure that learns both the initialization and the step-size of the algorithm from multiple online learning tasks. Asymptotically, we guarantee that the average regret across tasks scales with a natural notion of task-similarity that measures the amount of overlap between near-optimal regions of different tasks. Finally, we instantiate the method and its guarantee in two important settings: multi-task data-driven algorithm design and robust meta-learning.

Formally, for some $T, m > 0$ and all $t \in [T]$ and $i \in [m]$ we consider a meta-learner faced with a sequence of Tm loss functions $\ell_{t,i} : C \mapsto [0,1]$ over a compact subset $C \subset \mathbb{R}^d$ that lies within a ball $\mathbf{B}(\rho, R)$ of radius R around some point $\rho \in \mathbb{R}^d$. Before each loss function $\ell_{t,i}$ the meta-learner must pick an element $\rho_{t,i} \in C$ before then suffering a loss or cost $\ell_{t,i}(\rho_{t,i})$. For a fixed t, the subsequence $\ell_{t,1}, \ldots, \ell_{t,m}$ defines a *task* for which we expect a single element $\rho_t^* \in C$ to do well, and thus we will use the *within-task regret* on task t to describe the quantity

$$R_{t,m} = \sum_{i=1}^{m} \ell_{t,i}(\rho_{t,i}) - \ell_{t,i}(\rho_t^*) \quad \text{where} \quad \rho_t^* \in \operatorname{argmin}_{\rho \in C} \sum_{i=1}^{m} \ell_{t,i}(\rho)$$

Our goal will be to improve the guarantees for regret in the single-task case by using information obtained from solving multiple tasks. In particular, we expect average performance across tasks to improve as we see more tasks; formally we define the **task-averaged regret**

$$\bar{R}_{T,m} = \frac{1}{T} \sum_{t=1}^{T} R_{t,m} = \frac{1}{T} \sum_{t=1}^{T} \sum_{i=1}^{m} \ell_{t,i}(\rho_{t,i}) - \ell_{t,i}(\rho_t^*)$$

and claim improvement over single-task learning if in the limit of $T \to \infty$ it is smaller than $R_{t,m}$. Note that for simplicity in this work we assume all tasks have the same number of rounds within-task.

7.1 Meta-learning

We first generalize the regret bound for the exponential forecaster algorithm of Balcan et al. [15] to make it data-dependent and hyperparameter-dependent:

Theorem 7.1.1. *Let $\ell_1, \ldots, \ell_m : C \mapsto [0,1]$ be any sequence of piecewise L-Lipschitz functions that are β-dispersed. Suppose $C \subset \mathbb{R}^d$ is contained in a ball of radius R. The exponentially weighted forecaster [15] has expected regret $R_m \leq m\lambda + \frac{\log(1/Z)}{\lambda} + \tilde{O}((L+1)m^{1-\beta})$, where $Z = \frac{\int_{\mathbf{B}(\rho^*, m^{-\beta})} w(\rho)d\rho}{\int_C w(\rho)d\rho}$ for ρ^* the optimal action in hindsight.*

Proof. The proof adapts the analysis of the exponential forecaster in [15]. Let $W_t = \int_C w_t(\rho)d\rho$ be the normalizing constant and $P_t = \mathbb{E}_{\rho \sim p_t}[u_t(\rho)]$ be the expected payoff at round t. Also let $U_t(\rho) = \sum_{j=1}^t u_j(\rho)$. We seek to bound $R_T = OPT - P(T)$, where $OPT = U_T(\rho^*)$ for optimal parameter ρ^* and $P(T) = \sum_{t=1}^T P_t$ is the expected utility of Algorithm 11 in T rounds. We will do this by lower bounding $P(T)$ and upper bounding OPT by analyzing the normalizing constant W_t.

Lower bound for $P(T)$: This follows from standard arguments, included for completeness. Using the definitions in Algorithm 11, it follows that

$$\frac{W_{t+1}}{W_t} = \frac{\int_{\mathbf{C}} e^{\lambda u_t(\rho)} w_t(\rho)d\rho}{W_t} = \int_{\mathbf{C}} e^{\lambda u_t(\rho)} \frac{w_t(\rho)}{W_t} d\rho = \int_{\mathbf{C}} e^{\lambda u_t(\rho)} p_t(\rho)d\rho.$$

Use inequalities $e^{\lambda x} \leq 1 + (e^\lambda - 1)x$ for $x \in [0,1]$ and $1 + x \leq e^x$ to conclude

$$\frac{W_{t+1}}{W_t} \leq \int_{\mathbf{C}} p_t(\rho) \left(1 + (e^\lambda - 1)u_t(\rho)\right) d\rho = 1 + (e^{H\lambda} - 1)P_t \leq \exp\left((e^\lambda - 1)P_t\right).$$

Finally, we can write W_{T+1}/W_1 as a telescoping product to obtain

$$\frac{W_{T+1}}{W_1} = \prod_{t=1}^T \frac{W_{t+1}}{W_t} \leq \exp\left((e^\lambda - 1)\sum_t P_t\right) = \exp\left(P(T)(e^\lambda - 1)\right),$$

or, $W_{T+1} \leq \exp\left(P(T)(e^\lambda - 1)\right) \int_C w_1(\rho)d\rho$.

Upper bound for OPT: Let $\mathbf{B}^*(r)$ be the ball of radius r around ρ^*. If there are at most k discontinuities in any ball of radius r, we can conclude that for all $\rho \in \mathbf{B}^*(r)$, $U_T(\rho) \geq OPT - k - LTr$. Now, since $W_{T+1} = \int_C w_1(\rho) \exp(\lambda U_T(\rho))d\rho$, we have

$$
\begin{aligned}
W_{T+1} &\geq \int_{\mathbf{B}^*(r)} w_1(\rho) e^{\lambda U_T(\rho)} d\rho \\
&\geq \int_{\mathbf{B}^*(r)} w_1(\rho) e^{\lambda(OPT-k-LTr)} d\rho \\
&= e^{\lambda(OPT-k-LTr)} \int_{\mathbf{B}^*(r)} w_1(\rho) d\rho.
\end{aligned}
$$

Putting together with the lower bound, and rearranging, gives

$$
\begin{aligned}
OPT - P_T &\leq \frac{P(T)(e^\lambda - 1 - \lambda)}{\lambda} + \frac{\log(1/Z)}{\lambda} + k + LTr \\
&\leq T\lambda + \frac{\log(1/Z)}{\lambda} + k + LTr,
\end{aligned}
$$

where we use that $P(T) \leq T$ and for all $x \in [0,1], e^x \leq 1 + x + (e-2)x^2$. Take expectation over the sequence of utility functions and apply dispersion to conclude the result. □

The new bound is useful due to its explicit dependence on both the initialization w and the optimum in hindsight via the $\log(1/Z)$ term. Assuming w is a (normalized) distribution, this effectively measures the overlap between the chosen initialization and a small ball around the optimum; we thus call

$$
-\log Z = -\log \frac{\int_{\mathbf{B}(\rho^*, m^{-\beta})} w(\rho) d\rho}{\int_C w(\rho) d\rho}
$$

the *negative log-overlap* of initialization $w(.)$ with the optimum ρ^*. This motivates our notion of task similarity, namely the *average negative log-overlap*

$$
V^2 = -\min_{w: C \mapsto \mathbb{R}_{\geq 0}, \int_C w(\rho) d\rho = 1} \frac{1}{T} \sum_{t=1}^{T} \log \int_{\mathbf{B}(\rho_t^*, m^{-\beta})} w(\rho) d\rho
$$

which can be much smaller if the task optima ρ_t^* are close together; for example, if they are the same then $V = 0$, corresponding to assigning all the initial weight within the common ball $\mathbf{B}(\rho^*, m^{-\beta})$ around the shared optima. This is also true if $\mathsf{vol}(\cap_{t \in T} \mathbf{B}(\rho_t^*, m^{-\beta})) > 0$, as one can potentially initialize with all the weight in the intersection of the balls. On the other hand if $\mathsf{vol}(\cap_{t \in T} \mathbf{B}(\rho_t^*, m^{-\beta})) = 0$, $V > 0$. For example, if a p-fraction of tasks have optima ρ_0 and the remaining at ρ_1 with $||\rho_0 - \rho_1|| > 2m^{-\beta}$ the task similarity is given by the binary entropy function $V = H_b(p) = -p \log p - (1-p) \log(1-p)$.

We provide a bound on the task-averaged regret in terms of average negative log-overlap between the task-specific optima, by adapting and extending techniques developed for meta-learning with convex losses [101]. Notice that while initialization in the prior meta-learned algorithms would typically be a single point in the domain, here we learn a good initial distribution $w(\rho)$ over the parameter space. Our regret bound has an additional dispersion-based term, which we show is unavoidable without additional assumptions.

Theorem 7.1.2. *There is an algorithm which achieves task-averaged regret*

$$o_T(1) + \tilde{O}\left(V\sqrt{m} + m^{1-\beta}\right).$$

Here V is the task-similarity.

7.2 Applications

To demonstrate the usefulness of our method, we study the meta-learning algorithm in two settings of interest.

7.2.1 Multi-task data-driven hyperparameter selection

The online learning of hyperparameters under data-driven algorithm design [15, 20] can be improved by learning good initial distributions over the parameter space, i.e. the results in this chapter can be used on top of typical data-driven online tuning results to enable leveraging of good parameters shared across similar tasks. We will instantiate our results in a few classic problems to illustrate this.

k-center clustering. We consider a parameterized Llyod's algorithm family for initializing cluster centers [16]. In the seeding phase, each point x is sampled with probability proportional to $\min_{c\in C} d(v,c)^\alpha$, where $d(\cdot,\cdot)$ is the distance metric and C is the set of centers chosen so far. The family contains an algorithm for each $\alpha \in [0,\infty)\cup\infty$, and includes popular clustering heuristics like vanilla k-means (random initial centers, for $\alpha = 0$), k-means++ (corresponding to $\alpha = 2$) and farthest-first traversal ($\alpha = \infty$). The performance of the algorithm is measured using the Hamming distance to the optimal clustering, and is a piecewise constant function of α. Our meta-learning result can be instantiated for this problem even without smoothness assumptions (simply leveraging the smoothness induced by the internal randomness of the clustering algorithm).

Theorem 7.2.1. *Consider instances of the k-center clustering problem on n points, with Hamming loss $l_{i,j}$ for $i \in [m], j \in [T]$ w.r.t. some (unknown) ground truth clustering. Then the task-averaged regret for learning the algorithm parameter α for the α-Lloyd's clustering algorithm family of [16] is $o_T(1) + \tilde{O}((V+1)\sqrt{m})$.*

Proof. We start by applying Theorem 4 from [16] to an arbitrary α-interval $[\alpha_0, \alpha_0+\epsilon] \subseteq [0, D]$ of length ϵ. The expected number of discontinuities (expectation under the internal randomness of the algorithm when sampling successive centers), is at most

$$D(m,\epsilon) = O(nk\log(n)\log(\max\{(\alpha_0+\epsilon)/\alpha_0), (\alpha_0+\epsilon)\log R\}),$$

where R is an upper bound on the ratio between any pair of non-zero distances. Considering cases $\alpha_0 \lesseqgtr \frac{1}{\log R}$ and using the inequality $\log(1+x) \le x$ for $x \ge 0$ we get that there are, in expectation, at most $O(\epsilon nk \log n \log R)$ discontinuities in any interval of length ϵ. Theorem D.3.1 now implies $\frac{1}{2}$-dispersion using the recipe from [19]. The task-averaged regret bound follows from Theorem 7.1.2. □

Integer quadratic programming (IQP). The objective is to maximize a quadratic function $z^T Az$ for A with non-negative diagonal entries, subject to $z \in \{0,1\}^n$. In the classic Goemans-Williamson algorithm [82] one solves an SDP relaxation $U^T AU$ where columns u_i of U are unit vectors. u_i are then rounded to $\{\pm 1\}$ by projecting on a vector Z drawn according to the standard Gaussian, and using $\mathtt{sgn}(\langle u_i, Z\rangle)$. A simple parametric family is s-linear rounding where the rounding is as before if $|\langle u_i, Z\rangle| > s$ but uses probabilistic rounding to round u_i to 1 with probability $\frac{1+(\langle u_i,Z\rangle)/s}{2}$. Our results yield low task-averaged regret for learning the parameter of the s-linear rounding algorithms.

Theorem 7.2.2. *Consider instances of IQP given by matrices $A_{i,j}$ and rounding vectors $Z_{i,j} \sim \mathcal{N}_n$ for $i \in [m], j \in [T]$. Then the asymptotic task-averaged regret for learning the algorithm parameter s for s-linear rounding is $o_T(1) + 2V\sqrt{m} + O(\sqrt{m})$.*

Proof. As noted in [15], since $Z_{i,j}$ are normal, the local of discontinuities $s = |\langle u_i, Z\rangle|$ are distributed with a $\sqrt{\frac{2}{\pi}}$-bounded density. Thus in any interval of length ϵ, we have in expectation at most $\epsilon\sqrt{\frac{2}{\pi}}$ discontinuities. Theorem D.3.1 together with the general recipe from [19] implies $\frac{1}{2}$-dispersion. The task-averaged regret bound is now a simple application of Theorem 7.1.2. □

7.2.2 Robust meta-learning

In online learning, we seek to minimize a sequence of loss functions, and are required to perform well relative to the optimal choice in hindsight. It is possible for the observed loss functions to be noisy on some inputs, either naturally or due to adversarial intent. We will now explore the conditions under which learning robust to such an adversarial influence (i.e. outlier injection) is possible, which is particularly common in meta-learning with diverse sources.

Setup: At round i, we play x_i, observe perturbed loss $\tilde{l}_i : \mathbf{X} \to [0,1]$ which is set by the adversary by modifying the true loss $l_i : \mathbf{X} \to [0,1]$ using an *attack function* $a_i : \mathbf{X} \to [0,1]$ such that $\tilde{l}_i = l_i + a_i$ and may be non-Lipschitz, and suffer perturbed loss $\tilde{l}_i(x_i)$ and true loss $l_i(x_i)$. We seek to minimize regret relative to best fixed action in hindsight, i.e. $\tilde{R}_m = \sum_{i=1}^m \tilde{l}_i(x_i) - \min_{x\in\mathbf{X}} \sum_{i=1}^m \tilde{l}_i(x)$ for the perturbed loss and regret $R_m = \sum_{i=1}^m l_i(x_i) - \min_{x\in\mathbf{X}} \sum_{i=1}^m l_i(x)$ for the true loss.

No regret can be achieved provided the adversary distribution is sufficiently smooth, i.e. satisfies β-dispersion for some $\beta > 0$, as this corresponds to online optimization of the perturbed loss function. We can show this for both perturbed and true loss. The perturbed loss guarantee is immediate from standard results on online learning of piecewise Lipschitz functions [15, 18]. For the true loss, we can achieve no regret if the adversary perturbation a_i is limited to small balls and the centers of the balls are dispersed, which we capture using the following definition.

Definition 27 (δ-bounded, β_a-dispersed attack)**.** *An attack function a_i is δ-bounded if there exists a ball $\mathbf{B}(x_a, \delta)$ of radius δ such that $a_i(x) = 0$ for each $x \in \mathbf{X} \setminus \mathbf{B}(x_a, \delta)$. x_a is called a center c_{a_i} for attack a_i. A sequence of attack functions $a_1, \ldots, a_m$ is said to be β_a-dispersed, if the positions of attack centers x_a are dispersed i.e. for all m and for all $\epsilon \geq m^{-\beta_a}$,*

$$\mathbb{E}\left[\max_{x,x'\in\mathbf{X}, x\in\mathbf{B}(x',\epsilon)} \left|\{i \in [m] \mid x = c_{a_i}\}\right|\right] \leq \tilde{O}(\epsilon m).$$

Theorem 7.2.3. *Given a sequence of β-dispersed adversarially perturbed losses $\tilde{l}_i = l_i + a_i$, where $\tilde{l}_i, l_i, a_i$ are piecewise L-Lipschitz functions $\mathbf{X} \to [0,1]$ for $i = 1, \ldots, m$ and $\mathbf{X} \subset \mathbb{R}^d$, the exponential forecaster algorithm has*

$$\mathbb{E}[\tilde{R}_m] = \tilde{O}\left(m\lambda + \frac{\log(1/Z)}{\lambda} + (L+1)m^{1-\beta}\right)$$

(with Z as in Theorem 7.1.1). If in addition we have that a_i is a $m^{-\beta_a}$-bounded, β_a-dispersed attack, then

$$\mathbb{E}[R_m] = \tilde{O}\left(m\lambda + \frac{\log(1/Z)}{\lambda} + (L+1)m^{1-\min\{\beta,\beta_a\}}\right).$$

Proof. The bound on $\mathbb{E}[\tilde{R}_T]$ is immediate from Theorem 7.1.1. For $\mathbb{E}[R_T]$, we can upper bound the natural regret with the sum of robust regret, total adversarial perturbation at the optimum and a term corresponding to the difference between the loss of natural and robust optima.

$$\begin{aligned}
R_T &= \sum_{t=1}^{T} l_t(x_t) - \min_{x\in\mathbf{X}} \sum_{t=1}^{T} l_t(x) \\
&= \tilde{R}_T + \sum_{t=1}^{T} l_t(x_t) - \sum_{t=1}^{T} \tilde{l}_t(x_t) + \min_{x\in\mathbf{X}} \sum_{t=1}^{T} \tilde{l}_t(x) - \min_{x\in\mathbf{X}} \sum_{t=1}^{T} l_t(x) \\
&= \tilde{R}_T - \sum_{t=1}^{T} a_t(x_t) + \sum_{t=1}^{T} a_t(\tilde{x}^*) + \sum_{t=1}^{T} l_t(\tilde{x}^*) - \sum_{t=1}^{T} l_t(x^*) \\
&\leq \tilde{R}_T + \sum_{t=1}^{T} a_t(\tilde{x}^*) + \left|\sum_{t=1}^{T} l_t(\tilde{x}^*) - \sum_{t=1}^{T} l_t(x^*)\right|
\end{aligned}$$

where $\tilde{x}^* = \operatorname{argmin}_{x\in\mathbf{X}} \sum_{t=1}^{T} \tilde{l}_t(x)$ and $x^* = \operatorname{argmin}_{x\in\mathbf{X}} \sum_{t=1}^{T} l_t(x)$. We now use the β_a-dispersedness of the attack to show an excess expected regret of $\tilde{O}(T^{1-\beta_a})$. Using attack dispersion on a ball of radius $T^{-\beta_a}$ around $\tilde{x}^*$, the number of attacks that have non-zero $a_t(\tilde{x}^*)$ is at most $\tilde{O}(T^{1-\beta_a})$, and therefore $\sum_{t=1}^{T} a_t(\tilde{x}^*) \leq \tilde{O}(T^{1-\beta_a})$. Further, observe that the robust and natural optima coincide unless some attack occurs at the natural optimum x^*. We can use attack dispersion at x^*, and a union bound across rounds, to conclude $\mathbb{E}|\sum_{t=1}^{T} l_t(\tilde{x}^*) - \sum_{t=1}^{T} l_t(x^*)| \leq \tilde{O}(T^{1-\beta_a})$ which concludes the proof. □

Together with Theorem 7.1.2, this implies no regret meta-learning in the presence of dispersed adversaries, in particular the occurrence of unreliable data in small dispersed parts of the domain. We also show a lower bound below which establishes that our upper bounds are essentially optimal in the attack dispersion.

The results in this chapter are joint work with Nina Balcan, Misha Khodak and Ameet Talwalkar [24], and have appeared in NeurIPS 2021.

Chapter 8

Adaptivity

Optimization in the presence of sharp (non-Lipschitz), unpredictable (w.r.t. time and amount) changes is a challenging and largely unexplored problem of great significance. In this chapter we will approach this problem in the online learning setting, and in the process develop general tools for data-driven algorithm design in changing environments.

In online learning, the typical goal of the player is to minimize the regret, defined as the difference between the online cumulative payoff (i.e. $\sum_{t=1}^{T} u_t(\rho_t)$) and the cumulative payoff using an optimal offline choice in hindsight. In many real world problems, like online routing, detecting spam email/bots and ad/content ranking, it is often inadequate to assume a fixed point will yield good payoff at all times. It is more natural to compute regret against a stronger offline baseline, say one which is allowed to switch the point a few times (say s *shifts*), to accommodate events which significantly change the function values for certain time periods. The switching points are neither known in advance nor explicitly stated during the course of the game. This stronger baseline is known as *shifting regret* [94].

Definition 28. *The s-shifted regret (tracking regret in [94]) is given by*

$$\mathbb{E}\left[\max_{\substack{\rho_i^* \in \mathbf{C}, \\ t_0=1<t_1\cdots<t_s=T+1}} \sum_{i=1}^{s} \sum_{t=t_{i-1}}^{t_i-1} \left(u_t(\rho_i^*) - u_t(\rho_t)\right)\right]$$

Shifting regret is a particularly relevant metric for online learning problems in the context of data-driven algorithm design, 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. In the online setting, one has a configurable algorithm such as an algorithm for clustering data [13], and must solve a series of related problems, such as clustering news articles each day for a news reader or clustering drugstore sales information to detect disease outbreaks. For problems of this nature, significant events in the world or changing habits of buyers might require changes in algorithm parameters, and we would like the online algorithms to adapt well to sudden changes.

8.1 Online algorithms with low shifting regret

Algorithm 12 Fixed Share Exponential Forecaster (Fixed Share EF)

Input: step size parameter $\lambda \in (0, 1/H]$, exploration parameter $\alpha \in [0, 1]$
[1.] $w_1(\rho) = 1$ for all $\rho \in \mathbf{C}$
[2.] For each $t = 1, 2, \ldots, T$:
[i.] $W_t := \int_{\mathbf{C}} w_t(\rho) d\rho$
[ii.] Sample ρ with probability proportional to $w_t(\rho)$, i.e. with probability $p_t(\rho) = \frac{w_t(\rho)}{W_t}$
[iii.] Observe $u_t(\cdot)$
[iv.] Let $e_t(\rho) = e^{\lambda u_t(\rho)} w_t(\rho)$. For each $\rho \in \mathbf{C},$ set

$$w_{t+1}(\rho) = (1-\alpha) e_t(\rho) + \frac{\alpha}{\textsc{Vol}(\mathbf{C})} \int_{\mathbf{C}} e_t(\rho) d\rho \tag{8.1}$$

We present the first results for shifting regret for non-convex utility functions which potentially have sharp discontinuities. Algorithms with $O(\sqrt{sT \log NT})$ regret are known for the case of s *shifts* for prediction with N experts [94], using weight regularization (adding uniform exploration at each step, that increases with the fraction $\frac{s}{T}$ of shifts). We show how to extend the result to arbitrary compact sets of experts, and more general utility functions where convexity can no longer be exploited. Our key insight is to view the regularization as simultaneously inducing multiplicative weights update with restarts matching all possible shifted expert sequences, which allows us to use the *dispersion* condition introduced in [15].

Our shifting regret bounds are $O(\sqrt{sdT \log T} + sT^{1-\beta})$ which imply low regret for sufficiently dispersed (large enough β) functions. We provide and analyze an algorithm which achieves this, by a careful mix of exploration (mixing with uniform sampling) and exploitation (multiplicative weights).

Theorem 8.1.1. *Suppose the utility functions $u_t : \mathcal{C} \to [0, H], t \in [T]$ are piecewise L-Lipschitz and β-dispersed (Definition 2), where $\mathcal{C} \subset \mathbb{R}^d$ is contained in a ball of radius R. Then s-shifted regret $O(H\sqrt{sT(d \log(RT^\beta) + \log(T/s))} + (sH + L)T^{1-\beta})$ is achievable.*

Proof of Theorem 8.1.1. We first provide an upper and lower bound to $\frac{W_{T+1}}{W_1}$.
Upper bound: The proof is similar to the upper bound for exponential weighted forecaster in [15] and uses Lemma E.3.3 for W_t.

$$\frac{W_{t+1}}{W_t} = \frac{\int_{\mathbf{C}} e^{\lambda u_t(\rho)} w_t(\rho) d\rho}{W_t} = \int_{\mathbf{C}} e^{\lambda u_t(\rho)} \frac{w_t(\rho)}{W_t} d\rho = \int_{\mathbf{C}} e^{\lambda u_t(\rho)} p_t(\rho) d\rho.$$

Finally use inequalities $e^{\lambda z} \leq 1 + (e^\lambda - 1)z$ for $z \in [0, 1]$ and $1 + z \leq e^z$ to get

$$\frac{W_{t+1}}{W_t} \leq \int_{\mathbf{C}} p_t(\rho) \left(1 + (e^{H\lambda} - 1)\frac{u_t(\rho)}{H}\right) d\rho = 1 + (e^{H\lambda} - 1)\frac{P_t}{H} \leq \exp\left((e^{H\lambda} - 1)\frac{P_t}{H}\right).$$

where P_t denotes the expected payoff of the algorithm in round t. Let $P(\mathbf{A})$ be the expected total payoff. Then we can write $\frac{W_{T+1}}{W_1}$ as a telescoping product which gives

$$\frac{W_{T+1}}{W_1} = \prod_{t=1}^{T} \frac{W_{t+1}}{W_t} \leq \exp\left((e^{H\lambda} - 1)\frac{\sum_t P_t}{H}\right) = \exp\left(\frac{P(\mathcal{A})(e^{H\lambda} - 1)}{H}\right). \tag{8.2}$$

Lower bound: Again the proof is similar to [15] and the major difference is use of Lemma E.3.2. We first lower bound payoffs of points close to the optimal sequence of experts using dispersion. If the optimal sequence with s shifts has shifts at t_i^* ($1 \le i \le s-1$), by β-dispersion for any $\rho_i \in \mathbf{B}(\rho_i^*, w)$

$$\sum_{t=t_{i-1}^*}^{t_i^*-1} u_t(\rho_i) \ge \sum_{t=t_{i-1}^*}^{t_i^*-1} u_t(\rho_i^*) - kH - L(t_i^* - t_{i-1}^*)w, \tag{8.3}$$

where $w = T^{-\beta}$ and $k = O(T^{1-\beta})$. Summing both sides over $i \in [s-1]$ helps us relate the lower bound to the payoff OPT of the optimal sequence.

$$\sum_{i=1}^{s}\sum_{t=t_{i-1}^*}^{t_i^*-1} u_t(\rho_i) \ge \sum_{i=1}^{s}\sum_{t=t_{i-1}^*}^{t_i^*-1} u_t(\rho_i^*) - kH - L(t_i^* - t_{i-1}^*)w = OPT - ksH - LTw. \tag{8.4}$$

Now to lower bound $\frac{W_{T+1}}{W_1}$, we first lower bound W_{T+1}. We use Lemma E.3.2 and lower bound by picking the term corresponding to times of expert shifts in the optimal sequence with s-shifted expert.

$$W_{T+1} = \sum_{s\in[T]} \left[\sum_{t_0=1<t_1\cdots<t_s=T+1} \left(\frac{\alpha^{s-1}(1-\alpha)^{T-s}}{\textsc{Vol}(\mathbf{C})^{s-1}} \prod_{i=1}^{s} \tilde{W}(t_{i-1}, t_i) \right) \right] \quad \text{(Lemma E.3.2)} \tag{8.5}$$

$$\ge \frac{\alpha^{s-1}(1-\alpha)^{T-s}}{\textsc{Vol}(\mathbf{C})^{s-1}} \prod_{i=1}^{s} \tilde{W}(t_{i-1}^*, t_i^*). \tag{8.6}$$

The product of $\tilde{W}$'s can in turn be lower bounded by restricting attention to points close (i.e. within a ball of radius w centered at optimal expert ρ_i^*) to the optimal sequence. The payoffs of such points was lower-bounded in (8.3) and (8.4) in terms of the optimal payoff.

$$\begin{aligned}
\prod_{i=1}^{s} \tilde{W}(t_{i-1}^*, t_i^*) &= \prod_{i=1}^{s} \int_{\mathbf{C}} \exp\left(\lambda \sum_{t=t_{i-1}^*}^{t_i^*-1} u_t(\rho)\right) d\rho \\
&\ge \prod_{i=1}^{s} \int_{\mathbf{B}(\rho_i^*, w)} \exp\left(\lambda \sum_{t=t_{i-1}^*}^{t_i^*-1} u_t(\rho)\right) d\rho && \text{(Restrict domains)} \\
&\ge \prod_{i=1}^{s} \int_{\mathbf{B}(\rho_i^*, w)} \exp\left(\lambda\big(\sum_{t=t_{i-1}^*}^{t_i^*-1} u_t(\rho_i^*) - kH - L(t_i^* - t_{i-1}^*)w\big)\right) d\rho && \text{(Using equation 8.3)} \\
&= \textsc{Vol}(\mathbf{B}(w))^s \exp\left(\sum_{i=1}^{s} \lambda\Big(\sum_{t=t_{i-1}^*}^{t_i^*-1} u_t(\rho_i^*) - kH - L(t_i^* - t_{i-1}^*)w\Big)\right) && \text{(Independent of } \rho) \\
&= \textsc{Vol}(\mathbf{B}(w))^s \exp\Big(\lambda\big(OPT - ksH - LTw\big)\Big) && \text{(Using equation 8.4).}
\end{aligned}$$

Plugging into equation (8.6) we get

$$W_{T+1} \geq \frac{\alpha^{s-1}(1-\alpha)^{T-s}\textsc{Vol}(\mathbf{B}(w))^s}{\textsc{Vol}(\mathbf{C})^{s-1}} \exp\Big(\lambda\big(OPT - ksH - LTw\big)\Big).$$

Also, $W_1 = \int_{\mathbf{C}} w_1(\rho)d\rho = \textsc{Vol}(\mathbf{C})$. Thus, using the fact that ratio of volume of balls $\mathbf{B}(w)$ and $\mathbf{B}(R)$ in d-dimensions is $(w/R)^d$, and assuming $\mathbf{C}$ is bounded by some ball $\mathbf{B}(R)$.

$$\begin{aligned}\frac{W_{T+1}}{W_1} &\geq \frac{\alpha^{s-1}(1-\alpha)^{T-s}\textsc{Vol}(\mathbf{B}(w))^s}{\textsc{Vol}(\mathbf{C})^{s}} \exp\Big(\lambda\big(OPT - ksH - LTw\big)\Big) \\ &\geq \alpha^{s-1}(1-\alpha)^{T-s}\left(\frac{w}{R}\right)^{sd} \exp\Big(\lambda\big(OPT - ksH - LTw\big)\Big). \end{aligned} \tag{8.7}$$

Putting together: Combining upper and lower bounds from (E.4) and (8.7) respectively,

$$\log\left(\alpha^{s-1}(1-\alpha)^{T-s}\right) - sd\log\frac{R}{w} + \lambda(OPT - ksH - LTw) \leq \frac{P(\mathcal{A})(e^{H\lambda}-1)}{H},$$

which rearranges to

$$\begin{aligned} &OPT - P(\mathcal{A}) \\ &\quad\leq P(\mathcal{A})\frac{(e^{H\lambda}-1-H\lambda)}{H\lambda} + \frac{sd\log(R/w)}{\lambda} + ksH + LTw - \frac{\log(\alpha^{s-1}(1-\alpha)^{T-s})}{\lambda}.\end{aligned}$$

Using $P(\mathcal{A}) \leq HT$ and using $e^z \leq 1 + z + (e-2)z^2$ for $z \in [0,1]$ we have

$$\begin{aligned} &OPT - P(\mathcal{A}) \\ &\quad\leq HT\frac{(e^{H\lambda}-1-H\lambda)}{H\lambda} + \frac{sd\log(R/w)}{\lambda} + ksH + LTw - \frac{\log(\alpha^{s-1}(1-\alpha)^{T-s})}{\lambda} \\ &\quad< H^2T\lambda + \frac{sd\log(R/w)}{\lambda} + ksH + LTw - \frac{\log(\alpha^{s-1}(1-\alpha)^{T-s})}{\lambda}.\end{aligned}$$

Now we tighten the bound, first w.r.t. α then w.r.t. λ. Note $\min_\alpha -\log(\alpha^{s-1}(1-\alpha)^{T-s})$ occurs for $\alpha_0 = \frac{s-1}{T-1}$ and

$$\begin{aligned} -\log(\alpha_0^{s-1}(1-\alpha_0)^{T-s}) &= (T-1)\left[-\frac{s-1}{T-1}\log\frac{s-1}{T-1} - \frac{T-s}{T-1}\log\frac{T-s}{T-1}\right] \\ &\leq (s-1)\log e\frac{T-1}{s-1}\end{aligned}$$

(binary entropy function satisfies $h(x) \leq x\ln(e/x)$ for $x \in [0,1]$). Finally minimizing over λ gives

$$OPT - P(\mathcal{A}) \leq O(H\sqrt{sT(d\log(R/w) + \log(T/s))} + ksH + LTw),$$

for $\lambda = \sqrt{s(d\log(R/w) + \log(T/s))/T}/H$. Plugging back $w = T^{-\beta}$ and $k = O(T^{1-\beta})$ completes the proof. □

In a large range of applications, one is able to show $\beta \geq \frac{1}{2}$ [10, 15, 28]. We further show that our bounds on the expected regret are tight modulo sublogarithmic terms, providing a near-optimal characterization of the problem.

Theorem 8.1.2. *For each $\beta > \frac{\log 3s}{\log T}$, there exist utility functions $u_1, \ldots, u_T : [0,1] \to [0,1]$ which are β-dispersed, and the s-shifted regret of any online algorithm is $\Omega(\sqrt{sT} + sT^{1-\beta})$.*

Proof of Theorem 8.1.2. $I_1 = [0,1]$. In the first phase, for the first $\frac{T-3sT^{1-\beta}}{s}$ functions we have a single discontinuity in the interval $\left(\frac{1}{2}\left(1-\frac{1}{3s}\right), \frac{1}{2}\left(1+\frac{1}{3s}\right)\right) \subseteq (\frac{1}{3}, \frac{2}{3})$. The functions have payoff 1 before or after (with probability $1/2$ each) their discontinuity point, and zero elsewhere. We introduce $3T^{1-\beta}$ functions each for the same discontinuity point, and set the discontinuity points $T^{-\beta}$ apart for β-dispersion. This gives us $\frac{1/3s}{T^{-\beta}} - 1$ potential points inside $[\frac{1}{3}, \frac{2}{3}]$, so we can support $3T^{1-\beta}\left(\frac{1/3s}{T^{-\beta}} - 1\right) = \frac{T}{s} - 3T^{1-\beta}$ such functions ($\frac{T}{s} - 3T^{1-\beta} > 0$ since $\beta > \frac{\log 3s}{\log T}$). By Lemma E.6.1 we accumulate $\Omega(\sqrt{\frac{T-3sT^{1-\beta}}{s}}) = \Omega(\sqrt{T/s})$ regret for this part of the phase in expectation. Let I_1' be the interval from among $[0, \frac{1}{2}\left(1-\frac{1}{3s}\right)]$ and $[\frac{1}{2}\left(1+\frac{1}{3s}\right), 1]$ with more payoff in the phase so far. The next function has payoff 1 only at first or second half of I_1' (with probability $1/2$) and zero everywhere else. Any algorithm accumulates expected regret $1/2$ on this round. We repeat this in successively halved intervals. β-dispersion is satisfied since we use only $\Theta(T^{1-\beta})$ functions in the interval I' of size greater than $1/3$, and we accumulate an additional $\Omega(T^{1-\beta})$ regret. Notice there is a fixed point used by the optimal adversary for this phase.
Finally we repeat the construction inside the largest interval with no discontinuities at the end of the last phase for the next phase. Note that at the i-th phase the interval size will be $\Theta(\frac{1}{i})$. Indeed at the end of the first round we have *unused* intervals with sizes given by the sequence $\frac{1}{2}\left(1-\frac{1}{3s}\right), \frac{1}{4}\left(1-\frac{1}{3s}\right), \frac{1}{8}\left(1-\frac{1}{3s}\right), \ldots$ At the $i = 2^j$-th phase, we'll be repeating inside an interval of size $\frac{1}{2^{j+1}}\left(1-\frac{1}{3s}\right) = \Theta(\frac{1}{i})$. This allows us to run $\Theta(s)$ phases and get the desired lower bound (the intervals must be of size at least $\frac{1}{s}$ to support the construction). □

For $s = 1$, this improves over the lower bound construction of [15] where the lower bound is shown only for $\beta = 1/2$. In particular our results establish an almost tight characterization of static and dynamic regret under dispersion.

8.2 Efficient implementation

In this section we show that our algorithm can be implemented efficiently when u_t's are piecewise concave (dimishing returns). In particular we overcome the need to explicitly compute and update $w_t(\rho)$ (there are potentially uncountably infinite ρ in a compact domain $\mathcal{C}$) by showing that we can sample the points according to $p_t(\rho)$ directly.
The high-level strategy is to show that $p_t(\rho)$ is a mixture of t distributions which are Exponential Forecaster distributions from [15] i.e. $\tilde{p}_i(\rho) := \frac{\tilde{w}(\rho;i,t)}{\tilde{W}(i,t)}$ for each $1 \leq i \leq t$, with proportions $C_{i,t}$. As shown in [15] these distributions can be approximately sampled from (exactly in the one-dimensional case, $\mathcal{C} \subset \mathbb{R}$). We need to sample from one of these t distributions with probability $C_{t,i}$ to get the distribution p_t, and we can approximate these coefficients efficently (or compute exactly in one-dimensional case). We provide algorithms to do these approximations efficiently

Algorithm 13 Fixed Share Exponential Forecaster - efficient approximate implementation

Input: approximation parameter $\eta \in (0,1)$, confidence parameter $\zeta \in (0,1)$
[1.] $W_1 = \textsc{Vol}(\mathbf{C})$
[2.] For each $t = 1, 2, \ldots, T$:
[i.] Estimate $C_{t,j}$ using Lemma 8.2.3 for each $1 \le j \le t$.
[ii.] Sample i with probability $C_{t,i}$.
[iii.] Sample ρ with probability approximately proportional to $\tilde{w}(\rho; i, t)$ by running Algorithm BDV-18 with approximation-confidence parameters $(\eta/3, \zeta/2)$.
[iv.] Estimate W_{t+1} using Lemma 8.2.2. Algorithm BDV-18 to get $(\eta/6T, \eta/2T^2)$ estimates for all $\tilde{W}(\tau, \tau')$ and memoize values of $W_i, i \le t$.

(in poly(d, T) time), and we bound the extra expected regret. Asymptotically we get the same bound as the exact algorithm.

Theorem 8.2.1. *If utility functions are piecewise concave and L-Lipschitz, we can approximately sample a point ρ with probability $p_{t+1}(\rho)$ in time $\tilde{O}(Kd^4T^4)$ for approximation parameters $\eta = \zeta = 1/\sqrt{T}$ and $\lambda = \sqrt{s(d\ln(RT^\beta) + \ln(T/s))/T}/H$ and enjoy the same regret bound as the exact algorithm. (K is an upper bound on the number of discontinuities in the utility functions u_t; with probability at least $1-\zeta$, its expected payoff is within a factor of e^η of that of exact algorithm).*

Proof of Theorem 8.2.1. Based on Lemma 8.2.3, we can sample a uniformly random number r in $[0,1]$ and then sample a ρ from one of t distributions (selected based on r) that $p_t(\rho)$ is a mixture of with probability proportional to $C_{t,i}$. The sampling from the exponentials can be done in polynomial time for concave utility functions using sampling algorithm of [36]. At each round we sample from exactly one of t distributions in the sum for p_t in Lemma 8.2.3. We compute $(\eta/6T, \zeta/2T^2)$ approximations for $\tilde{W}(i,j)$, $1 \le i < j \le T$ in time $O(T^2K.T_\int)$ where $T_\int$ is the time to integrate a logconcave distribution (at most $\tilde{O}(d^4/\epsilon^2)$ from [113]). These give $(\eta/3, \zeta/2)$-approximation for $C_{t,i}$'s by corollary E.5.3. Finally we run Algorithm 2 from [15] with approximation-confidence parameters $(\eta/3, \zeta/2)$.
With probability at least $1-\zeta$, $C_{t,i}$ estimation and ρ sampling according to $\tilde{w}(\rho; i, t)$ succeeds. If $\hat{\mu}$ denotes output distribution of ρ with approximate sampling, and μ denotes the exact distribution per $p_t(\rho)$, then we show $D_\infty(\hat{\mu}, \mu) \le \eta$. Indeed, for any set of outcomes $E \subset \mathbf{C}$

$$\hat{\mu}(E) = \text{Pr}(\hat{\rho} \in E) = \sum_{i=1}^{t} \text{Pr}(\hat{\rho} \in E \mid E_{i,t})\text{Pr}(E_{i,t}) = \sum_{i=1}^{t} \hat{\mu}_i(E)\frac{\hat{C}_{t,i}}{\sum_j \hat{C}_{t,j}},$$

where $E_{i,t}$ denotes the event that $\tilde{w}(\rho; i, t)$ was used for sampling $p_t(\rho)$, and $\hat{\mu}_i$ corresponds to the distribution for approximate sampling of $\tilde{w}(\rho; i, t)$. Noting that we used $\eta/3$ approximation for $\hat{\mu}_i$ and each $\hat{C}_{t,i}$, we have

$$\hat{\mu}(E) \le \sum_{i=1}^{t} e^{\eta/3}\mu_i(E)e^{2\eta/3}\frac{C_{t,i}}{\sum_j C_{t,j}} = e^\eta \mu(E).$$

Similarly, $\hat{\mu}(E) \geq e^{-\eta}\mu(E)$ and hence $D_\infty(\hat{\mu}, \mu) \leq \eta$.
Finally we can show (cf. Theorem 12 of [15]) that with probability at least $1 - \zeta$ 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. Together with failure probability of ζ, this implies at most $(\eta + \zeta)HT$ additional regret which results in same asymptotic regret as the exact algorithm for $\eta = \zeta = 1/\sqrt{T}$.

To compute the time complexity, we note from [113] that logconcave functions can be integrated in $\tilde{O}(d^4/\epsilon^2)$ and sampled from in $\tilde{O}(d^3)$ time. The time to integrate dominates the complexity, and the overall complexity can be upper bounded by $O(T^2K \cdot d^4/(\eta/T)^2) = O(KT^4d^4)$. Note: The approximate integration and sampling are only needed for multi-dimensional case, for the one-dimensional case we can compute the weights and sample exactly in polynomial time. □

We use the following algorithm due to [15] as a sub-routine.
Algorithm BDV-18: Simply *integrate* pieces of the exponentiated utility function, pick a piece with probability proportional to its integral, and *sample from* that piece. [113] show how to efficient *sample from* and *integrate* logconcave distributions. See [15] for more details.
The coefficients have a simple form in terms of normalizing constants W_t's of the rounds so far, so we first express W_{t+1} in terms of W_t's from previous rounds and some $\tilde{W}(i, j)$'s.

Lemma 8.2.2. *In Algorithm 12, for $t \geq 1$,*

$$W_{t+1} = (1-\alpha)^{t-1}\tilde{W}(1, t+1) + \frac{\alpha}{\text{VOL}(\mathbf{C})}\sum_{i=2}^{t}\left[(1-\alpha)^{t-i}W_i\tilde{W}(i, t+1)\right].$$

Proof of Lemma 8.2.2. For $t = 1$, first term is $\tilde{W}(1, 2) = \int_{\mathbf{C}} e^{\lambda u_1(\rho)}d\rho = W_2$ and second term is zero. Also, by Lemma E.3.3, for $t > 1$,

$$\begin{aligned}
W_{t+1} &= \int_{\mathbf{C}} e^{\lambda u_t(\rho)} w_t(\rho) d\rho \\
&= \int_{\mathbf{C}} e^{\lambda u_t(\rho)} \left[(1-\alpha)e^{\lambda u_{t-1}(\rho)} w_{t-1}(\rho) + \frac{\alpha}{\text{VOL}(\mathbf{C})}\int_{\mathbf{C}} e^{\lambda u_{t-1}(\rho)} w_{t-1}(\rho) d\rho\right] d\rho \\
&= (1-\alpha)\int_{\mathbf{C}} e^{\lambda(u_t(\rho)+u_{t-1}(\rho))} w_{t-1}(\rho) d\rho + \frac{\alpha}{\text{VOL}(\mathbf{C})} W_t \int_{\mathbf{C}} e^{\lambda u_t(\rho)} d\rho.
\end{aligned}$$

Continue substituting $w_j(\rho) = (1-\alpha)e^{\lambda u_j(\rho)} w_{j-1}(\rho) + \frac{\alpha}{\text{VOL}(\mathbf{C})}\int_{\mathbf{C}} e^{\lambda u_j(\rho)} w_{j-1}(\rho) d\rho$ in the first summand until $w_1 = 1$ to get the desired expression. □

As indicated above, $p_t(\rho)$ is a mixture of t distributions.

Lemma 8.2.3. *In Algorithm 12, for $t \geq 1$, $p_t(\rho) = \sum_{i=1}^{t} C_{t,i}\frac{\tilde{w}(\rho; i, t)}{\tilde{W}(i,t)}$. The coefficients $C_{t,i}$ are given by*

$$C_{t,i} = \begin{cases} 1 & i = t = 1, \\ \alpha & i = t > 1, \\ (1-\alpha)\frac{W_{t-1}}{W_t}\frac{\tilde{W}(i,t)}{\tilde{W}(i,t-1)} C_{t-1,i} & i < t, \end{cases}$$

and $(C_{t,1}, \ldots, C_{t,t})$ *lies on the probability simplex* Δ^{t-1}.

Proof of Lemma 8.2.3. At each iteration, p_t is obtained by mixing $e^{u_t} p_{t-1}$ with the uniform distribution, i.e. we rescale distributions that p_{t-1} was a mixture of and add one more. Another way to view it is to consider a distribution over the sequences of exponentially updated or randomly chosen points. The final probability distribution is the mixture of a combinatorial number of distributions but a large number of them have a proportional density. $C_{t,i}$ are simply sums of mixture coefficients. This establishes the intuition for the expression for p_t and that the mixing coefficients should sum to 1, but we still need to convince ourselves that the coefficients can be computed efficiently.
We proceed by induction on t. For $t = 1$ (using definitions for $w_2(\rho)$ and $w_2(\rho)$)

$$p_1(\rho) = \frac{w_1(\rho)}{W_1} = \frac{1}{\textsc{Vol}(\mathbf{C})} = C_{1,1} \frac{\tilde{w}(\rho; 1, 1)}{\tilde{W}(1, 1)}$$

(recall $\tilde{w}(\rho; 1, 1) := 1$ and $\tilde{W}(1, 1) = \int_{\mathbf{C}} \tilde{w}(\rho; 1, 1) d\rho$). For the inductive step, we first express p_{t+1} in terms of p_t

$$\begin{aligned}
p_{t+1}(\rho) &= \frac{w_{t+1}(\rho)}{W_{t+1}} \\
&= (1 - \alpha) \frac{e^{\lambda u_t(\rho)} w_t(\rho)}{W_{t+1}} + \frac{\alpha}{\textsc{Vol}(\mathbf{C})} \\
&= (1 - \alpha) \frac{W_t}{W_{t+1}} \frac{e^{\lambda u_t(\rho)} w_t(\rho)}{W_t} + \frac{\alpha}{\textsc{Vol}(\mathbf{C})} \\
&= (1 - \alpha) \frac{W_t}{W_{t+1}} e^{\lambda u_t(\rho)} p_t(\rho) + \frac{\alpha}{\textsc{Vol}(\mathbf{C})}.
\end{aligned}$$

The lemma is now straightforward to see with induction hypothesis.

$$\begin{aligned}
p_{t+1}(\rho) &= (1 - \alpha) \frac{W_t}{W_{t+1}} e^{\lambda u_t(\rho)} \left[\sum_{i=1}^{t} C_{t,i} \frac{\tilde{w}(\rho; i, t)}{\tilde{W}(i, t)} \right] + \frac{\alpha}{\textsc{Vol}(\mathbf{C})} \\
&= \sum_{i=1}^{t} \left[(1 - \alpha) \frac{W_t}{W_{t+1}} C_{t,i} \frac{\tilde{w}(\rho; i, t+1)}{\tilde{W}(i, t)} \right] + \frac{\alpha}{\textsc{Vol}(\mathbf{C})} \\
&= \sum_{i=1}^{t} \left[\left((1 - \alpha) \frac{W_t}{W_{t+1}} \frac{\tilde{W}(i, t+1)}{\tilde{W}(i, t)} C_{t,i} \right) \frac{\tilde{w}(\rho; i, t+1)}{\tilde{W}(i, t+1)} \right] + \frac{C_{t+1,t+1}}{\textsc{Vol}(\mathbf{C})} \\
&= \sum_{i=1}^{t} C_{t+1,i} \frac{\tilde{w}(\rho; i, t+1)}{\tilde{W}(i, t+1)} + \frac{C_{t+1,t+1}}{\textsc{Vol}(\mathbf{C})}.
\end{aligned}$$

Finally noting

$$C_{t+1,t+1} \frac{\tilde{w}(\rho; t+1, t+1)}{\tilde{W}(t+1, t+1)} = C_{t+1,t+1} \frac{1}{\int_{\mathbf{C}} (1) d\rho} = \frac{C_{t+1,t+1}}{\textsc{Vol}(\mathbf{C})}.$$

completes the proof.

Thus W_t (by Lemma 8.2.2) and $C_{t,i}$ can be computed recursively for logconcave utility functions using integration algorithm from [113]. We can compute them efficiently using Dynamic Programming.
Finally it's straightforward to establish that the coefficients for p_t must lie on the probability simplex Δ^{t-1}. All coefficients are positive, which is easily seen from the recursive relation and noting all weights are positive. Also we know

$$p_t(\rho) = \sum_{i=1}^{t} C_{t,i} \frac{\tilde{w}(\rho; i, t)}{\tilde{W}(i, t)}.$$

Since $p_t(\rho)$ is a probability distribution by definition, integrating both sides over $\mathbf{C}$ gives

$$\int_{\mathbf{C}} p_t(\rho) d\rho = \sum_{i=1}^{t} C_{t,i} \frac{\int_{\mathbf{C}} \tilde{w}(\rho; i, t) d\rho}{\tilde{W}(i, t)} \qquad \text{, or,}$$

$$1 = \sum_{i=1}^{t} C_{t,i}.$$

□

8.3 Recurring environments

Algorithm 14 Generalized Share Exponential Forecaster (Generalized Share EF)

Input: step size parameter $\lambda \in (0, 1/H]$, exploration parameter $\alpha \in [0, 1]$, discount rate $\gamma \in [0, 1]$
[1.] $w_1(\rho) = 1$ for all $\rho \in \mathbf{C}$
[2.] For each $t = 1, 2, \ldots, T$:
[i.] $W_t := \int_{\mathbf{C}} w_t(\rho) d\rho$
[ii.] Sample ρ with probability proportional to $w_t(\rho)$, i.e. with probability $p_t(\rho) = \frac{w_t(\rho)}{W_t}$
[iii.] Let $e_t(\rho) = e^{\lambda u_t(\rho)} w_t(\rho)$ and $\beta_{i,t} = \frac{e^{-\gamma(t-i)}}{\sum_{j=1}^{t} e^{-\gamma(t-j)}}$. For each $\rho \in \mathbf{C}$, set

$$w_{t+1}(\rho) = (1 - \alpha) e_t(\rho) + \alpha \left(\int_{\mathbf{C}} e_t(\rho) d\rho \right) \sum_{i=1}^{t} \beta_{i,t} p_i(\rho)$$

An interesting special case of changing environments is where a small number of distinct experts are likely to be good baselines over the course of the online game.

Definition 29. *Extend Definition 28 with an additional constraint on the number of distinct experts used,* $\left| \{\rho_i^* \mid 1 \le i \le s\} \right| \le m$. *We call this ($m$-sparse, s-shifted) regret [43].*

This restriction makes sense if we think of the adversary as likely to reuse the same experts again, or the changing environment to experience recurring events with similar payoff distributions.

As it turns out adding equal exploration to all points does not allow us to exploit recurring environments of the (m-sparse, s-shifted) setting very well. To overcome this, we replace the uniform update with a prior consisting of a weighted mixture of all the previous probability distributions used for sampling. This includes uniformly random exploration as the first probability distribution $p_1(\cdot)$, but the weight on this distribution decreases exponentially with time according to *discount rate* γ (more precisely, it decays by a factor $e^{-\gamma}$ with each time step). While previously exploration was limited to starting afresh, here it includes partial resets to explore again from all past states, with an exponentially discounted rate (formally Algorithm 3 in [20]).

For this approach we can show low (m-sparse, s-shifted) regret as well.

Theorem 8.3.1. *There exists an algorithm with its (m-sparse, s-shifted) regret at most*

$$O(H\sqrt{T(md\log(RT^{\beta}) + s\log(mT/s))} + (mH + L)T^{1-\beta}).$$

Proof of Theorem 8.3.1. Like Theorem 8.1.1 we first provide an upper and lower bound to $\frac{W_{T+1}}{W_1}$. The upper bound proof is identical to that of Theorem 8.1.1 by replacing Lemma E.3.3 by Lemma E.3.8.

For the lower bound we use Corollaries E.3.10 and E.3.7. Applying corollary E.3.7 repeatedly to collect exponential updates for the times OPT played the same expert lets us use the arguments for Theorem 8.1.1. Indeed if $\{(s_i, f_i) \mid 1 \le i \le l\}$ are the start and finish times of a particular expert ρ in the OPT sequence, we can use Corollary E.3.10 to write

$$W_{f_l+1} \ge \alpha(1-\alpha)^{f_l+1-s_l} W_{s_l} \tilde{W}(\pi_{s_l}; s_l, f_l + 1).$$

Applying Corollary E.3.7 repeatedly now gets us

$$W_{f_l+1} \ge \frac{\alpha^l(1-\alpha)^{\sum_{j=1}^{l} f_j+1-s_j}(1-e^{-\gamma})^{l-1}}{(e^{-\gamma}+\alpha(1-e^{-\gamma}))^{\sum_{j=1}^{l-1} f_j+1-s_{j+1}}} \frac{\prod_{i=1}^{l} W_{s_i}}{\prod_{i=1}^{l-1} W_{f_i+1}} \int_{\mathbf{C}} \left(\pi_{s_1}(\rho) \prod_{j=1}^{l} \tilde{w}(\rho; s_j, f_j + 1)\right) d\rho,$$

or,

$$\frac{\prod_{i=1}^{l} W_{f_i+1}}{\prod_{i=1}^{l} W_{s_i}} \ge \frac{\alpha^l(1-\alpha)^{\sum_{j=1}^{l} f_j+1-s_j}(1-e^{-\gamma})^{l-1}}{(e^{-\gamma}+\alpha(1-e^{-\gamma}))^{\sum_{j=1}^{l-1} f_j+1-s_{j+1}}} \int_{\mathbf{C}} \left(\pi_{s_1}(\rho) \prod_{j=1}^{l} \tilde{w}(\rho; s_j, f_j + 1)\right) d\rho.$$

Multiplying these inequalities for each of m experts in the optimal sequence gives us $\frac{W_{T+1}}{W_1}$ on the left side. Also note

$$\int_{\mathbf{C}} \pi_t(\rho) f(\rho) d\rho \ge \frac{\alpha_{1,t}}{W_1} \int_{\mathbf{C}} f(\rho) d\rho,$$

and, using dispersion as in proof of Theorem 8.1.1,

$$\prod_{\text{experts in OPT}} \int_{\mathbf{C}} \left(\prod_{j=1}^{l} \tilde{w}(\rho; s_j, f_j + 1)\right) d\rho \ge \textsc{Vol}(\mathbf{B}(T^{-\beta}))^m \exp\left(\lambda\left(OPT - (mH + L)O(T^{1-\beta})\right)\right).$$

Putting it all together, and combining the lower and upper bounds on $\frac{W_{T+1}}{W_1}$ gives us a bound on $OPT - P(\mathbf{A})$.

$$OPT - P(\mathcal{A}) < H^2 T\lambda + \frac{md\log(RT^\beta)}{\lambda} + (mH + L)O(T^{1-\beta}) - \log\left(\frac{\alpha^s(1-\alpha)^T(1-e^{-\gamma})^s}{(e^{-\gamma} + \alpha(1-e^{-\gamma}))^{-mT}}\right)/\lambda.$$

We now chose parameters γ, α, λ to get the tightest regret bound. Note that $-\log(\alpha^s(1-\alpha)^T)$ is minimized for $\alpha = \frac{s}{T+s} = \Theta(\frac{s}{T})$ and $-\log((1-e^{-\gamma})^s(e^{-\gamma} + \alpha(1-e^{-\gamma}))^{mT})$ is minimized for $\gamma = \log\left(\frac{1+s/mT}{1-s\alpha/mT(1-\alpha)}\right) = \Theta(\frac{s}{mT})$. The corresponding minimum values can be bounded as

$$-\log(\alpha^s(1-\alpha)^T) = s\log\frac{T+s}{s} + T\log\left(1+\frac{s}{T}\right) \leq s\log\frac{T+s}{s} + s = O\left(s\log\frac{T}{s}\right).$$

Using $\log(1+x) \leq x$, and substituting $e^{-\gamma} = \frac{1-s\alpha/mT(1-\alpha)}{1+s/mT}$,

$$\begin{aligned}
&-\log((1-e^{-\gamma})^s(e^{-\gamma} + \alpha(1-e^{-\gamma}))^{mT}) \\
&\qquad = -s\log\frac{\frac{s}{mT}\cdot\frac{1}{(1-\alpha)}}{1+\frac{s}{mT}} - mT\log\frac{1}{1+\frac{s}{mT}} \\
&\qquad = s\log\left((1-\alpha)\left(\frac{mT}{s}+1\right)\right) + mT\log\left(1+\frac{s}{mT}\right) \\
&\qquad \leq s\log\left((1-\alpha)\left(\frac{mT}{s}+1\right)\right) + 1 \\
&\qquad = O\left(s\log\frac{mT}{s}\right).
\end{aligned}$$

Finally we minimize w.r.t. λ, to obtain the desired regret bound. □

The results in this chapter are joint work with Nina Balcan and Travis Dick [20], and have appeared in AISTATS 2020.

Chapter 9

Output-sensitivity

An important open problem is the design of computationally efficient data-driven algorithms for combinatorial algorithm families with multiple parameters. As one fixes the problem instance and varies the parameters, the "dual" loss function typically has a piecewise-decomposable structure, i.e. is well-behaved except at certain sharp transition boundaries. In this work we initiate the study of techniques to develop efficient ERM learning algorithms for data-driven algorithm design by enumerating the pieces of the sum dual loss functions for a collection of problem instances. The running time of our approach scales with the actual number of pieces that appear as opposed to worst case upper bounds on the number of pieces. Our approach involves two novel ingredients – an output-sensitive algorithm for enumerating polytopes induced by a set of hyperplanes using tools from computational geometry, and an *execution graph* which compactly represents all the states the algorithm could attain for all possible parameter values. We illustrate our techniques by giving algorithms for pricing problems, linkage-based clustering and dynamic-programming based sequence alignment.

9.1 Output-sensitive Parameterized Complexity

Output-sensitive algorithms have a running time that depends on the size of the output for any input problem instance. Output-sensitive analysis is frequently employed in computational geometry, for example Chan's algorithm [52] computes the convex hull of a set of 2-dimensional points in time $O(n \log R)$, where R is the size of the (output) convex hull. Output-sensitive algorithms are useful if the output size is variable, and 'typical' output instances are much smaller than worst case instances. Parameterized complexity extends classical complexity theory by taking into account not only the total input length n, but also other aspects of the problem encoded in a *parameter* k[1]. The motivation is to confine the super-polynomial runtime needed for solving many natural problems strictly to the parameter. Formally, a parameterized decision problem (or language) is a subset $L \subseteq \Sigma^* \times \mathcal{N}$, where Σ is a fixed alphabet, i.e. an input (x, k) to a parameterized problem consists of two parts, where the second part k is the parameter. A parameterized problem L is *fixed-parameter tractable* if there exists an algorithm which on a given input $(x, k) \in \Sigma^* \times \mathcal{N}$,

[1]N.B. The term "parameter" is overloaded. It is used to refer to the real-valued parameters in the algorithm family, as well as the parameter to the optimization problem over the algorithm family.

decides whether $(x, k) \in L$ in $f(k) \cdot \text{poly}(|x|)$ time, where f is an arbitrary computable function in k [67]. FPT is the class of all parameterized problems which are fixed-parameter tractable. In contrast, the class XP (aka *slicewise-polynomial*) consists of problems for which there is an algorithm with running time $|x|^{f(k)}$. It is known that FPT $\subsetneq$ XP. We consider an extension of the above to search problems and incorporate output-sensitivity in the following definition.

Definition 30 (Output-polynomial Fixed Parameter Tractable)**.** *A parameterized search problem $P : \Sigma^* \times \mathcal{N} \to \tilde{\Sigma}^*$ is said to be output-polynomial fixed-parameter tractable if there exists an algorithm which on a given input $(x, k) \in \Sigma^* \times \mathcal{N}$, computes the output $P(x, k) \in \tilde{\Sigma}^*$ in time $f(k) \cdot poly(|x|, R)$, where $R = |P(x, k)|$ is the output size and f is an arbitrary computable function in k.*

As discussed above, output-sensitvity and fixed-parameter tractability both offer more fine-grained complexity analysis than traditional (input) polynomial time complexity. Both techniques have been employed in efficient algorithmic enumeration [77, 129] and have gathered recent interest [78]. In this work, we consider the optimization problem of selecting tunable parameters over a continuous domain $\mathcal{C} \subset \mathbb{R}^d$ with the fixed-parameter $k = d$, the number of tunable parameters. We will design OFPT enumeration algorithms which, roughly speaking, output a finite "search space" which can be used to easily find the best parameter for the problem instance.

9.2 Output-sensitive cell enumeration

Let H be a collection of t hyperplanes in $\mathbb{R}^d$. We consider the problem of enumerating the d-faces (henceforth *cells*) of the convex polyhedral regions induced by the hyperplanes. The cells will be represented as sign-patterns[2] of facet-inducing hyperplanes. We will present an approach for enumerating these cells in OFPT time (Definition 30), which involves two key ingredients: (a) locality-sensitivity, and (b) output-sensitivity. By locality sensitivity, we mean that our algorithm exploits problem-specific local structure in the neighborhood of each cell to work with a smaller candidate set of hyperplanes which can potentially constitute the cell facets. This is abstracted out as a sub-routine COMPUTELOCALLYRELEVANTSEPARATORS which we will instantiate and analyse for each individual problem. In this section we will focus more on the output-sensitivity aspect.

To provide an output-sensitive guarantee for this enumeration problem, we compute only the *non-redundant* hyperplanes which provide the boundary of each cell c in the partition induced by the hyperplanes. We denote the closed polytope bounding cell c by P_c. A crucial ingredient for ensuring good output-sensitive runtime of our algorithm is Clarkson's algorithm for computing non-redundant constraints in a system of linear inequalities [62]. A constraint is *redundant* in a system if removing it does not change the set of solutions. The key idea is to maintain a set I of non-redundant constraints detected so far, and solve LPs that detect the redundancy of a remaining constraint (not in I) when added to I. If the constraint is redundant relative to I, it must also be redundant in the full system, otherwise we can add a (potentially different) non-redundant constraint to I. The following runtime guarantee is known for the algorithm.

[2]For simplicity we will denote these by vectors of the form $\{0, 1, -1\}^t$ where non-zero co-ordinates correspond to signs of facet-inducing hyperplanes. Hash tables would be a practical data structure for implementation.

Theorem 9.2.1 (Clarkson's algorithm)**.** *Given a list L of k half-space constraints in d dimensions, Clarkson's algorithm outputs the set $I \subseteq L$ of non-redundant constraints in L in time $O(k \cdot \mathrm{LP}(d, |I|+1))$, where $\mathrm{LP}(v, c)$ is the time for solving an LP with v variables and c constraints.*

Algorithm 15 uses AUGMENTEDCLARKSON, which modifies Clarkson's algorithm with some additional bookkeeping to facilitate a search for neighboring regions in our algorithm, while retaining the same asymptotic runtime complexity. Effectively, our algorithm can be seen as a breadth-first search over an implicit underlying graph (Definition 31), where the neighbors (and some auxiliary useful information) are computed dynamically by AUGMENTEDCLARKSON.

Algorithm 15 OUTPUTSENSITIVEPARTITIONSEARCH

Input: Set $H = \{\mathbf{a_i} \cdot \mathbf{x} = b_i\}_{i \in [t]}$ of t hyperplanes in $\mathbb{R}^d$, convex polytopic domain $\mathcal{P} \subseteq \mathbb{R}^d$ bounded by hyperplane set P.
Output: Partition cells $\tilde{C} = \{\tilde{c}^{(j)}\}$, $c^{(j)} \in \{0, 1, -1\}^t$ with $|\tilde{c}_i^{(j)}| = 1$ iff $\mathbf{a_i} \cdot \mathbf{x} = b_i$ is a bounding hyperplane for cell j, and $\mathsf{sgn}(\tilde{c}_i^{(j)}) = \mathsf{sgn}(\mathbf{a_i} \cdot \mathbf{x_j} - b_i)$ for interior point $\mathbf{x_j}$.
$\mathbf{x}_1 \leftarrow$ an arbitrary point in $\mathbb{R}^d$ (assumed general position w.r.t. H)
Cell $c^{(1)}$ with $\mathsf{sgn}(c_i^{(1)}) = \mathsf{sgn}(\mathbf{a_i} \cdot \mathbf{x_1} - b_i)$
$\mathtt{q} \leftarrow$ empty queue; $\mathtt{q.enqueue}([c^{(1)}, \mathbf{x}_1])$
$C \leftarrow \{\}; \tilde{C} \leftarrow \{\}$
while $\mathtt{q.non\_empty}()$ **do**
  $[c, \mathbf{x}] \leftarrow \mathtt{q.dequeue}()$
  Continue to next iteration if $c \in C$
  $C \leftarrow C \cup \{c\}$
  $\tilde{H} \leftarrow$ COMPUTELOCALLYRELEVANTSEPARATORS$(\mathbf{x}, H)$ subset of hyperplanes in H that can be facets for cell containing $\mathbf{x}$
  $H' \leftarrow \{(-\mathsf{sign}(c_i)\mathbf{a_i} \cdot \mathbf{x}, -\mathsf{sign}(c_i)b_i) \mid \mathbf{a_i} \cdot \mathbf{x} = b_i \in \tilde{H}\} \cup P$
  $(\tilde{c}, \mathtt{neighbors}) \leftarrow$ AUGMENTEDCLARKSON$(\mathbf{x}, H', c)$
  $\tilde{C} \leftarrow \tilde{C} \cup \{\tilde{c}\}$
  $\mathtt{q.enqueue}([c', \mathbf{x}'])$ for each $[c', \mathbf{x}'] \in \mathtt{neighbors}$
return $\tilde{C}$

Definition 31 (Cell adjacency graph)**.** *Define the* cell adjacency graph *for a set H of hyperplanes in $\mathbb{R}^d$, written $G_H = (V_H, E_H)$, as follows. There is a vertex $v \in V_H$ corresponding to each cell in the partition $\tilde{C}$ of $\mathbb{R}^d$ induced by the hyperplanes; for $v, v' \in V_H$, add the edge $\{v, v'\}$ to E_H if the corresponding polytopes intersect, i.e. $P_v \cap P_{v'} \neq \emptyset$. This generalizes to a subdivision of a polytope in $\mathbb{R}^d$.*

This allows us to state the following guarantee about the runtime of Algorithm 15.

Theorem 9.2.2. *Let H be a set of t hyperplanes in $\mathbb{R}^d$. Suppose $|E_H| = E$ and $|V_H| = V$ in the cell adjacency graph $G_H = (V_H, E_H)$ of H; then if the domain $\mathcal{P}$ is bounded by $|P| \leq t$ hyperplanes, Algorithm 15 computes the set V_H in time $\tilde{O}(dE + VT_{\mathit{CLRS}} + t_{\mathit{LRS}} \cdot \sum_{c \in V_H} \mathrm{LP}(d, |I_c|+1))$, where $\mathrm{LP}(\mathrm{r}, \mathrm{s})$ denotes the time to solve an LP in r variables and s constraints, I_c denotes the*

number of facets for cell $c \in V_H$, T_{CLRS} denotes the running time of COMPUTELOCALLYRELEVANTSEPARATORS *and t_{LRS} denotes an upper bound on the number of locally relevant hyperplanes in Line 10.*

Proof of Theorem 9.2.2. Algorithm 15 maintains a set of *visited* or *explored* cells in C, and their bounding hyperplanes (corresponding to cell facets) in $\tilde{C}$. It also maintains a queue of cells, such that each cell in the queue has been discovered in Line 12 as a neighbor of some visited cell in C, but is yet to be explored itself. The algorithm detects if a cell has been visited before by using its sign pattern on the hyperplanes in H. This can be done in $O(d \log t)$ time, since $|C| = O(t^d)$ using a well-known combinatorial fact (e.g. [50]). For a new cell c, we run AUGMENTEDCLARKSON to compute its bounding hyperplanes I_c as well as sign patterns for neighbors in time $O(t_{\text{LRS}} \cdot \text{LP}(d, |I_c|+1))$ by Theorem 9.2.1. The computed neighbors are added to the queue in Line 14. A cell c gets added to the queue at this step up to $|I_c|$ times, but is not explored if already visited. Thus we run up to V iterations of AUGMENTEDCLARKSON and up to $1+\sum_{c \in V_H} |I_c| = 2E+1$ queue insertions/removals. Using efficient union-find data-structures, the set union and membership queries (for C and $\tilde{C}$) can be done in $\tilde{O}(1)$ time per iteration of the loop. So total time over the V cell explorations and no more than $2E + 1$ iterations of the **while** loop is $\tilde{O}(dE) + O(t_{\text{LRS}} \cdot \sum_{c \in V_H} \text{LP}(d, |I_c| + 1)) + O(VT_{\text{CLRS}})$. □

We will demonstrate several data-driven parameter selection problems which satisfy Definition 14 and for which the COMPUTELOCALLYRELEVANTSEPARATORS can be implemented in FPT (often poly(d, t)) time for fixed-parameter d. The above result then implies that the problem of enumerating the induced polytopic cells is in OFPT (Definition 30).

Corollary 9.2.3. *If* COMPUTELOCALLYRELEVANTSEPARATORS *runs in FPT time for fixed-parameter d, Algorithm 15 enumerates the cells induced by a set H of t hyperplanes in $\mathbb{R}^d$ in OFPT time.*

Proof of Corollary 9.2.3. Several approaches for solving a (low-dimensional) LP in r variables and s constraints in deterministic $r^{O(r)}s$ time are known [53, 57] (the well-known randomized algorithm of [150] runs in expected $r^{O(r)}s$ time). Noting $t_{\text{LRS}} \leq T_{\text{CLRS}}$, the runtime bound in Theorem 9.2.2 simplifies to $\tilde{O}(d^{O(d)}(E+V)T_{\text{CLRS}})$, and the result follows by the assumption on T_{CLRS} and since $E \leq V^2$. □

We illustrate the significance of output-sensitivity and locality-sensitivity with simple examples.

Example 2. *The worst-case size of V_H is $O(t^d)$ and standard (output-insensitive) enumeration algorithms for computing V_H (e.g. [70, 170]) take $O(t^d)$ time even when output size may be much smaller. For example, if H is a collection of parallel planes in $\mathbb{R}^3$, running time of these approaches is $O(t^3)$. Even a naive implementation of* COMPUTELOCALLYRELEVANTSEPARATORS *which always outputs the complete set H gives a better runtime of $O(t^2)$. By employing a straightforward algorithm which binary searches the closest hyperplanes to $\mathbf{x}$ (in a pre-processed H) as* COMPUTELOCALLYRELEVANTSEPARATORS *we have $t_{LRS} = O(1)$ and $T_{CLRS} = \tilde{O}(1)$, and Algorithm 15 attains a running time of $\tilde{O}(t)$. Analogously, if H is a collection of t hyperplanes in $\mathbb{R}^d$ with $1 \leq k < d$ distinct unit normal vectors, then output-sensitivity improves runtime from $O(t^d)$ to $O(t^{k+1})$, and locality-sensitivity further improves it to $\tilde{O}(t^k)$.*

Example 3. *If H is a collection of t hyperplanes in $\mathbb{R}^d, d \geq 2$ which all intersect in a $d-2$ hyperplane (e.g. collinear planes in $\mathbb{R}^3$), we can again obtain worst-case $O(t^2)$ runtime due to output-sensitivity of Algorithm 15 and further improvement to $\tilde{O}(t)$ runtime by exploiting locality sensitivity similar to the above example.*

9.2.1 ERM in the statistical learning setting

We now use Algorithm 15 to compute the sample mimimum (aka ERM, Empirical Risk Minimization) for the $(\mathcal{F}, t)$ piecewise-structured dual losses with linear boundaries (Definition 14) over a problem sample $S \in \Pi^m$, provided piece functions in $\mathcal{F}$ can be efficiently optimized over a polytope (typically the piece functions are constant or linear functions in our examples). Formally, we define search-ERM for a given parameterized algorithm family $\mathcal{A}$ with parameter space $\mathcal{P}$ and the dual class utility function being $(\mathcal{F}, t)$-piecewise decomposable (Definition 14) as follows: given a set of m problem instances $S \in \Pi^m$, compute the pieces, i.e. a partition of the parameter space into connected subsets such that the utility function is a fixed piece function in $\mathcal{F}$ over each subset for each of the m problem instances. The following result gives a recipe for efficiently solving the search-ERM problem provided we can efficiently compute the dual function pieces in individual problem instances, and the the number of pieces in the sum dual class function over the sample S is not too large. The key idea is to apply Algorithm 15 for each problem instance, and once again for the search-ERM problem.

Theorem 9.2.4. *Let $\tilde{C}_i$ denote the cells partitioning the polytopic parameter space $\mathcal{P} \subset \mathbb{R}^d$ corresponding to pieces of the dual class utility function u_i on a single problem instance $x_i \in \Pi$, from a collection $s = \{x_1, \ldots, x_m\}$ of m problem instances. Let (V_S, E_S) be the cell adjacency graph corresponding to the polytopic pieces in the sum utility function $\sum_{i=1}^m u_i$. Then there is an algorithm for computing V_S given the cells $\tilde{C}_i$ in time $\tilde{O}((d+m)|E_S| + m t_{LRS} \cdot \sum_{c \in V_S} \mathrm{LP}(d, |I_c|+1))$, where I_c denotes the number of facets for cell $c \in V_S$, and t_{LRS} is the number of locally relevant hyperplanes in a single instance.*

Proof of Theorem 9.2.4. We will apply Algorithm 15 and compute the locally relevant hyperplanes by simply taking a union of the facet-inducing hyperplanes at any point $\mathbf{x}$, across the problem instances.

Let $G^{(i)} = (V^{(i)}, E^{(i)})$ denote the cell adjacency graph for the cells in $\tilde{C}_i$. We apply Algorithm 15 implicitly over $H = \cup_i \overline{H}^{(i)}$ where $\overline{H}^{(i)}$ is the collection of facet-inducing hyperplanes in $\tilde{C}_i$. To implement ComputeLocallyRelevantSeparators$(\mathbf{x}, H)$ we simply search the computed partition cell $\tilde{C}_i$ for the cell containing $\mathbf{x}$ in each problem instance x_i in time $O(\sum_i |E^{(i)}|) = O(m|E_S|)$, obtain the set of corresponding facet-inducing hyperplanes $H_{\mathbf{x}}^{(i)}$, and output $\tilde{H}_{\mathbf{x}} = \cup_i H_{\mathbf{x}}^{(i)}$ in time $O(m t_{\text{LRS}})$. The former step only needs to be computed once, as the partition cells for subsequent points can be tracked in $O(m)$ time. Theorem 9.2.2 now gives a running time of $\tilde{O}(d|E_S| + m|E_S| + m t_{\text{LRS}}|V_S| + m t_{\text{LRS}} \cdot \sum_{c \in V_S} \mathrm{LP}(d, |I_c|+1))$. □

An important consequence of the above result is an efficient output-sensitive algorithm for data-driven algorithm design when the dual class utility function is $(\mathcal{F}, t)$-piecewise decomposable. In the following sections, we will instantiate the above results for various data-driven parameter selection problems where the dual class functions are piecewise-structured with linear boundaries

(Definition 14). Prior work [17, 18, 23] has shown polynomial bounds on the sample complexity of learning near-optimal parameters via the ERM algorithm for these problems in the statistical learning setting, i.e. the problem instances are drawn from a fixed unknown distribution. In other words, ERM over polynomially large sample size m is sufficient for learning good parameters. In particular, we will design and analyze running time for problem-specific algorithms for computing locally relevant hyperplanes. Given Theorem 9.2.4, it will be sufficient to give an algorithm for computing the pieces of the dual class function for a single problem instance.

9.2.2 Online learning

In the online learning setting, one receives a sequence of problem instances $x_1, \ldots, x_T \in \Pi$, and needs to select an algorithm in each round. Recent research has shown that no-regret learning in online algorithm design is possible under additional smoothness assumptions on the online adversary [15, 69], but has noted computationally efficient implementation of the online learner as an open direction. Efficient algorithms are known for the case where the utility is a piecewise constant function of a single parameter, our tools allow output-sensitive enumeration for the $(\mathcal{F}, t)$ piecewise-structured dual functions with linear boundaries (Definition 14) even in the multiparameter setting.

Suppose the dual class functions for the online sequence of problem instances are given by $u_1(\rho), \ldots, u_T(\rho)$. A key ingredient in implementing an online learner is to compute the partition of the parameter space where the partial sum function $\sum_{i=1}^{t} u_i(\rho)$ is well-behaved. This allows us to use the Exponential Forecaster algorithm for the $(\mathcal{F}, t)$ piecewise-structured dual functions with linear boundaries (Algorithm 2 of [15]).

Theorem 9.2.5. *Let $\tilde{C}_u$ denote the cells partitioning the polytopic parameter space $\mathcal{P} \subset \mathbb{R}^d$ corresponding to pieces of the dual class utility function u, and $u_1, \ldots, u_T$ denote an online sequence of dual class functions corresponding to problem instances $\{x_1, \ldots, x_m\}$. Let (V_t, E_t) be the cell adjacency graph corresponding to the polytopic pieces in the partial sum utility function $U_t := \sum_{i=1}^{t} u_i$ for any $t \in [T]$. Then there is an algorithm for computing V_t given the cells $\tilde{C}_{U_{t-1}}$ in time $\tilde{O}((d+|V_t|)|E_t|+T_{CLRS}|V_t|+tt_{LRS}\cdot\sum_{c\in V_t}\mathrm{LP}(d,|I_c|+1))$, where I_c denotes the number of facets for cell $c \in V_t$, and t_{LRS} is the number of locally relevant hyperplanes in a single instance.*

Proof. We will apply Algorithm 15 and compute the locally relevant hyperplanes by simply taking a union of the facet-inducing hyperplanes of the cell containing $\mathbf{x}$ in $\tilde{C}_{U_{t-1}}$, and the locally relevant hyperplanes for the new problem instance x_t.

Let $G^{(i)} = (V^{(i)}, E^{(i)})$ denote the cell adjacency graph for the cells in $\tilde{C}_i$. We apply Algorithm 15 implicitly over $H = \cup_i \overline{H}^{(i)}$ where $\overline{H}^{(i)}$ is the collection of facet-inducing hyperplanes in $\tilde{C}_i$. To implement ComputeLocallyRelevantSeparators$(\mathbf{x}, H)$ we simply search the computed partition cell $\tilde{C}_{U_{t-1}}$ for the cell containing $\mathbf{x}$ in time $O(|E_{t-1}|)$, obtain the set of corresponding facet-inducing hyperplanes $H_{\mathbf{x}}^{(t-1)}$, compute the locally relevant separators $H_{\mathbf{x}}^{t}$ for the new problem instance x_t and output $\tilde{H}_{\mathbf{x}} = H_{\mathbf{x}}^{(t-1)} \cup H_{\mathbf{x}}^{t}$ in time $O(|E_{t-1}| + T_{\text{CLRS}} + tt_{\text{LRS}})$. Theorem 9.2.2 now gives a running time of $\tilde{O}(d|E_t| + (|E_{t-1}| + T_{\text{CLRS}} + tt_{\text{LRS}})|V_t| + tt_{\text{LRS}} \cdot \sum_{c\in V_t}\mathrm{LP}(d,|I_c|+1))$. □

Above result gives a way to bound the per-iteration complexity of the online learning algorithm of [15] in an output-sensitive way. In particular, if the utility functions $u_1, \ldots, u_T$ are concave in ρ, the remaining per-iteration runtime overhead of sampling from $\tilde{C}_{U_t}$ is $O(|\tilde{C}_{U_t}|\text{poly}(d,T))$ using approximate logconcave sampling and integration algorithms of [113]. This online learner enjoys no-regret guarantees under a mild assumption on the sequence $u_1, \ldots, u_T$, called *dispersion*. Roughly speaking, a sequence of $(\mathcal{F}, t)$ piecewise-structured utility functions $u_1, \ldots, u_T$ is said to be dispersed if the number of functions for which the piece boundaries intersect any small ball in the parameter space is bounded (see [15, 19] for formal definition).

Remark 6. *A related recent work [8] studies an alternative discretization-based algorithm for online learning of pricing problems, including the two-part tariff pricing studied here. The proposed approach is however not output-sensitive and only known to work for pricing problems, but it obtains no-regret even without the dispersion assumption.*

9.2.3 Augmented Clarkson's Algorithm

We describe here the details of the Augmented Clarkson's algorithm, which modifies the algorithm of Clarkson [62] with additional bookkeeping needed for tracking the partition cells in Algorithm 15. The underlying problem solved by Clarkson's algorithm may be stated as follows.

Problem Setup. Given a linear system of inequalities $Ax \leq b$, an inequality $A_i x \leq b_i$ is said to be *redundant* in the system if the set of solutions is unchanged when the inequality is removed from the system. Given a system $(A \in \mathbb{R}^{m \times d}, b \in \mathbb{R}^m)$, find an equivalent system with no redundant inequalities.

Note that to test if a single inequality $A_i x \leq b_i$ is redundant, it is sufficient to solve the following LP in d variables and m constraints.

$$\begin{aligned} \textbf{maximize} \quad & A_i x \\ \textbf{subject to} \quad & A_j x \leq b_j, \quad \forall j \in [m] \setminus \{i\} \\ & A_i x \leq b_i + 1 \end{aligned} \tag{9.1}$$

Using this directly to solve the redundancy removal problem gives an algorithm with running time $m \cdot \text{LP}(\text{d}, \text{m})$, where $\text{LP}(\text{d}, \text{m})$ denotes the time to solve an LP in d variables and m constraints. This can be improved using Clarkson's algorithm if the number of non-redundant constraints s is much less than the total number of constraints m (Theorem 9.2.1).

We assume that an interior point $z \in \mathbb{R}^d$ satisfying $Ax < b$ is given. At a high level, one maintains the set of non-redundant constraints I discovered so far i.e. $A_i x \leq b_i$ is not redundant for each $i \in I$. When testing a new index k, the algorithm solves an LP of the form 9.1 and either detects that $A_k x \leq b_k$ is redundant, or finds index $j \in [m] \setminus I$ such that $A_j x \leq b_j$ is non-redundant. The latter involves the use of a procedure RayShoot(A, b, z, x) which finds the non-redundant hyperplane hit by a ray originating from z in the direction $x - z$ ($x \in \mathbb{R}^d$) in the system A, b. The size of the LP needed for this test is $\text{LP}(\text{d}, |\text{I}| + 1)$ from which the complexity follows.

To implement the RayShoot procedure, we can simply find the intersections of the ray $x^* - z$ with the hyperplanes $A_j x \leq b_j$ and output the one closest to z (defining the cell facet in that direction). We also output an interior point from the adjacent cell during this computation, which saves us time relative to [154] where the interior point is computed for each cell (our Clarkson

Algorithm 16 AUGMENTEDCLARKSON($z, H = (A, b), c$)

Input: $A \in \mathbb{R}^{m \times d}, b \in \mathbb{R}^m, z \in \mathbb{R}^d$, sign-pattern $c \in \{0, 1, -1\}^m$.
Output: list of non-redundant hyperplanes $I \subseteq [m]$, points in neighboring cells $Z \subset \mathbb{R}^d$.
$I \leftarrow \emptyset, J \leftarrow [m], Z \leftarrow \emptyset$
while $J \neq \emptyset$ **do**
  Select $k \in J$
  Detect if $A_k x \leq b_k$ is redundant in $A_{I \cup \{k\}} x \leq b_{I \cup \{k\}}$ by solving LP (9.1).
  $x^* \leftarrow$ optimal solution of the above LP
  if redundant **then**
    $J \leftarrow J \setminus \{k\}$
    $j, z^* \leftarrow$ RayShoot(A, b, z, x^*)
    $J \leftarrow J \setminus \{j\}$
    $c'_j \leftarrow -c_j, c'_i \leftarrow c_i \; \forall \; i \in [m] \setminus \{j\}$
    $I \leftarrow I \cup \{j\}, Z \leftarrow Z \cup \{[c', z^*]\}$
return I, Z

based approach also circumvents their need for the Raindrop procedure [46]). Finally we state the running time guarantee of Algorithm 16, which follows from the original result of Clarkson [62].

Algorithm 17 RayShoot

Input: $A \in \mathbb{R}^{m \times d}, b \in \mathbb{R}^m, z \in \mathbb{R}^d, x \in \mathbb{R}^d$.
if $A_i \cdot (x - z) = 0$ for some i **then**
  $z \leftarrow z + (\epsilon, \epsilon^2, \ldots, \epsilon^d)$ for sufficiently small ϵ
$t_i \leftarrow \frac{b_i - A_i \cdot z}{A_i \cdot (x - z)}$
$j = \text{argmin}_i \{t_i \mid t_i > 0\}, t' = \max\{\min_{i \neq j}\{t_i \mid t_i > 0\}, 0\}$
return $j, z + \frac{t_j + t'}{2}(x - z)$

Theorem 9.2.6 (Augmented Clarkson's algorithm)**.** *Given a list L of k half-space constraints in d dimensions, Algorithm 16 outputs the set $I \subseteq [L]$ of non-redundant constraints in L, as well as auxiliary neighbor information, in time $O(k \cdot \text{LP}(d, |I| + 1))$, where $\text{LP}(v, c)$ is the time for solving an LP with v variables and c constraints.*

9.3 Profit maximization in pricing problems

Prior work [17, 126] on data-driven mechanism design has shown that the profit as a function of the prices (parameters) is $(\mathcal{F}, t)$-decomposable with $\mathcal{F}$ the set of linear functions on $\mathbb{R}^d$ for a large number of mechanism classes. We will instantiate our approach for multi-item pricing problems which are $(\mathcal{F}, t)$-decomposable and analyse the running times. In contrast, recent work [8] employs data-independent discretization for computationally efficient data-driven algorithm design for mechanism design problems even in the worst-case. This discretization based approach is not output-sensitive and is known to not work for other applications like data-driven clustering.

In the *two-part tariff* problem [108, 131], the seller with multiple identical items charges a fixed price, as well as a price per item purchased. For example, cab meters often charge a base cost for any trip and an additional cost per mile traveled. Subscription or membership programs often require an upfront joining fee plus a membership fee per renewal period, or per service usage. Often there is a menu or tier of prices, i.e. a company may design multiple subscription levels (say basic, silver, gold, platinum), each more expensive than the previous but providing a cheaper per-unit price. Given access to market data (i.e. profits for different pricing schemes for typical buyers) we would like to learn how to set the base and per-item prices to maximize the profit. We define these settings formally as follows.

Two-part tariffs. The seller has K identical units of an item. Suppose the buyers have valuation functions $v_i : \{1, \dots, K\} \to \mathbb{R}_{\geq 0}$ where $i \in \{1, \dots, m\}$ denotes the buyer, and the value is assumed to be zero if no items are bought. Buyer i will purchase q quantities of the item that maximizes their utility $u_i(q) = v_i(q) - (p_1 + p_2 q)$, buying zero units if the utility is negative for each $q > 0$. The revenue, which we want to maximize as the seller, is zero if no item is bought, and $p_1 + p_2 q$ if $q > 0$ items are bought. The algorithmic parameter we want to select is the price $\rho = \langle p_1, p_2 \rangle$, and the problem instances are specified by the valuations v_i. We also consider a generalization of the above scheme: instead of just a single two-part tariff (TPT), suppose the seller provides a menu of TPTs $(p_1^1, p_2^1), \dots, (p_1^\ell, p_2^\ell)$ of length ℓ. Buyer i selects a tariff (p_1^j, p_2^j) from the menu as well as the item quantity q to maximize their utility $u_i^j(q) = v_i(q) - (p_1^j + p_2^j q)$. This problem has 2ℓ parameters, $\rho = (p_1^1, p_2^1, \dots, p_1^\ell, p_2^\ell)$, and the single two-part tariff setting corresponds to $\ell = 1$.

The dual class functions in this case are known to be piecewise linear with linear boundaries [17]. We will now implement Algorithm 15 for this problem by specifying how to compute the locally relevant hyperplanes. For any price vector $\mathbf{x} = \rho$, say the buyers buy quantities $(q_1, \dots, q_m) \in \{0, \dots, K\}^m$ according to tariffs $(j_1, \dots, j_m) \in [\ell]^m$. For a fixed price vector this can be done in time $O(mK\ell)$ by computing $\operatorname{argmax}_{q,j} u_i^j(q)$ for each buyer at that price for each two-part tariff in the menu. Then for each buyer we have $K(\ell - 1)$ potential alternative quantities and tariffs given by hyperplanes $u_i^{j_i}(q_i) \geq u_i^{j'}(q'), q' \neq q_i, j' \neq j_i$, for a total of $t_{\text{LRS}} = mK(\ell - 1)$ locally relevant hyperplanes. Thus $T_{\text{CLRS}} = O(mK\ell)$ for the above approach, and Theorem 9.2.2 implies the following runtime bound.

Theorem 9.3.1. *There exists an implementation of* ComputeLocallyRelevantSeparators *in Algorithm 15, which given valuation function $v(\cdot)$ for a single problem instance, computes all the R pieces of the dual class function $u_v(\cdot)$ in time $\tilde{O}(R^2(2\ell)^{\ell+2}K)$, where the menu length is ℓ, and there are K units of the good.*

Proof of Theorem 9.3.1. In the terminology of Theorem 9.2.2, we have $d = 2\ell$, $E \leq R^2$, $V = R$, $T_{\text{CLRS}} = O(K\ell)$, $t_{\text{LRS}} \leq K\ell$. By [53], we have $\sum_{c \in V_H} \text{LP}(d, |I_c| + 1) \leq O(Ed^{d/2+1}) \leq O(R^2(2\ell)^{\ell+1})$. Thus, Theorem 9.2.2 implies a runtime bound on Algorithm 15 of $\tilde{O}(dE + VT_{\text{CLRS}} + t_{\text{LRS}} \cdot \sum_{c \in V_H} \text{LP}(d, |I_c| + 1)) = \tilde{O}(R^2(2\ell)^{\ell+2}K)$. □

Theorem 9.3.1 together with Theorem 9.2.4 implies an implementation of the search-ERM problem over m buyers (with valuation functions $v_i(\cdot)$ for $i \in [m]$) in time $O(R_\Sigma^2(2\ell)^{\ell+2}mK)$, where R_Σ denotes the number of pieces in the total dual class function $U_{\langle v_1, \dots, v_m \rangle}(\cdot) = \sum_i u_{v_i}(\cdot)$.

Corollary 9.3.2. *There exists an implementation of* COMPUTELOCALLYRELEVANTSEPARATORS *in Algorithm 15, which given valuation functions $v_i(\cdot)$ for $i \in [m]$, computes all the R_Σ pieces of the total dual class function $U_{\langle v_1,\ldots,v_m\rangle}(\cdot) = \sum_i u_{v_i}(\cdot)$ in time $\tilde{O}(R_\Sigma^2(2\ell)^{\ell+2}mK)$, where the menu length is ℓ, there are K units of the good and m is the number of buyers.*

Proof. We first compute the pieces for each of the problem instances (single buyers) and then the pieces in the sum dual class function using Theorem 9.2.4. By Theorem 9.3.1, the former takes time $\tilde{O}(mR^2(2\ell)^{\ell+2}K)$ and the latter can be implemented in time $O((m+2\ell)R_\Sigma^2 + mK\ell\sum_{c\in V_S}\mathrm{LP}(d,|I_c|+1)) = \tilde{O}(R_\Sigma^2(2\ell)^{\ell+2}mK)$, which dominates the overall running time. □

In contrast, prior work for this problem has only obtained an XP runtime of $(mK)^{O(\ell)}$ [21]. For the special case $\ell = 1$, we also provide an algorithm that uses additional structure of the polytopes and employs a computational geometry algorithm due to [56] to compute the pieces in optimal $O(mK\log(mK) + R_\Sigma)$ time, improving over the previously best known runtime of $O(m^3K^3)$ due to [21] even for worst-case R_Σ. The worst-case improvement follows from a bound of $R_\Sigma = O(m^2K)$ on the number of pieces. We further show that our running time for $\ell = 1$ is asymptotically optimal under the algebraic decision-tree model of computation, by a linear time reduction from the element uniqueness problem.

9.3.1 Piecewise structure of the dual class function

The following lemma restates the result from [17] in terms of Definition 14. Note that u_ρ in the following denotes the revenue function (or seller's utility) and should not be confused with the buyer utility function u_i.

Lemma 9.3.3. *Let $\mathcal{U}$ be the set of functions $\{u_\rho : v(\cdot) \mapsto p_1^{j^*} + p_2^{j^*}q^* \mid q^*, j^* = \mathrm{argmax}_{q,j}\, v(q) - \rho^j\cdot\langle 1, q\rangle, \rho^j = \langle p_1^j, p_2^j\rangle\}$ that map valuations $v(\cdot)$ to $\mathbb{R}$. The dual class $\mathcal{U}^*$ is $(\mathcal{F}, (K\ell)^2)$-piecewise decomposable, where $\mathcal{F} = \{f_c : \mathcal{U} \to \mathbb{R} \mid c \in \mathbb{R}^{2\ell}\}$ consists of linear functions $f_c : u_\rho \mapsto \rho \cdot c$.*

We also bound the number of pieces R in the worst-case for $\ell = 1$. The following bound implies that our algorithm is better than prior best algorithm which achieves an $O(m^3K^3)$ runtime bound, even for worst case outputs.

Theorem 9.3.4. *Let menu length $\ell = 1$. The number of pieces R_Σ in the total dual class function $U_{\langle v_1,\ldots,v_m\rangle}(\cdot) = \sum_i u_{v_i}(\cdot)$ is at most $O(m^2K)$.*

Proof. By Lemma 9.3.3, the dual class is $(\mathcal{F}, K^2)$-piecewise decomposable, where $\mathcal{F}$ is the class of linear functions. That is, the two-dimensional parameter space (p_1, p_2) can be partitioned into polygons by at most K^2 straight lines such that any dual class function u_{v_i} is a linear function inside any polygon.

We first tighten the above result to show that the dual class is in fact $(\mathcal{F}, 2K+2)$-piecewise decomposable, that is the number of bounding lines for the pieces is $O(K)$. This seems counterintuitive since for any buyer i, we have $\Theta(K^2)$ lines $u_i(q) \geq u_i(q')$ for $q < q' \in \{0,\ldots,K\}$. If $q > 0$, $u_i(q) = u_i(q')$ are axis-parallel lines with intercepts $\frac{v_i(q')-v_i(q)}{q'-q}$. Since for any pair q, q' the buyer has a fixed (but opposite) preference on either side of the axis-parallel line, we have at

most K distinct horizontal 'slabs' corresponding to buyer's preference of quantities, i.e. regions between lines $p_2 = a$ and $p_2 = b$ for some $a, b > 0$. Thus we have at most K non-redundant lines. Together with another K lines $u_i(0) = u_i(q')$ and the axes, we have $2K+2$ bounding lines in all as claimed.

We will next use this tighter result to bound the number of points of intersection of non-collinear non-axial bounding lines, let's call them *crossing points*, across all instances $\langle v_1, \ldots, v_m \rangle$. Consider a pair of instances given by buyer valuation functions v_i, v_j. We will establish that the number of crossing points are at most $4K$ for the pair of instances. Let l_i and l_j be bounding lines for the pieces of u_{v_i} and u_{v_j} respectively. If they are both axis-parallel, then they cannot result in a crossing point. For pairs of bounding lines l_i and l_j such that l_i is axis-parallel and l_j is not, we can have at most K crossing points in all. This is because any fixed l_i can intersect at most one such l_j since buyer j's preferred quantity q_j is fixed along any horizontal line, unless q_j changes across l_i in which case the crossing points for the consecutive l_j's coincide. Thus, there is at most one such crossing point for each of at most K axis-parallel l_i's. By symmetry, there are at most K crossing points between l_i, l_j where l_j is axis parallel and l_i is not. Finally, if neither l_i, l_j is axis parallel, we claim there can be no more than $2K$ crossing points. Indeed, if we arrange these points in the order of increasing p_2, then the preferred quantity of at least one of the buyers i or j strictly decreases between consecutive crossing points. Thus, across all instances, there are at most $2m^2K$ crossing points.

Finally, observe that the cell adjacency graph G_U for the pieces of the total dual class function $U_{\langle v_1,\ldots,v_m\rangle}(\cdot)$ is planar in this case. The vertices of this graph correspond to crossing points, or intersections of bounding lines with the axes. The latter is clearly $O(mK)$ since there are $O(K)$ bounding lines in any problem instance. Using the above bound on crossing points, the number of vertices in G_U is $O(m^2K)$. Since G_U is a simple, connected, planar graph, the number of faces is no more than twice the number of vertices and therefore the number of pieces R_Σ is also $O(m^2K)$. □

9.3.2 Optimal algorithm for Single TPT pricing

Consider the setting with menu-length $\ell = 1$. The key insight is to characterize the polytopic structure of the pieces of the dual class function for a single buyer. We do this in Lemma 9.3.5.

Lemma 9.3.5. *Consider a single buyer with valuation function $v(\cdot)$. The buyer buys zero units of the item except for a set $\varrho_v \subset \mathbb{R}^2$, where ϱ_v is a convex polygon with at most $K+2$ sides. Moreover, ϱ_v can be subdivided into $K' \leq K$ polygons $\varrho_v^{(i)}$, each a triangle or a trapezoid with bases parallel to the ρ_1-axis, such that for each $i \in [K']$ the buyer buys the same quantity $q^{(i)}$ of the item for all prices in $\varrho_v^{(i)}$.*

Proof. We proceed by an induction on K, the number of items. For $K = 1$, it is straightforward to verify that ϱ_v is the triangle $p_1 \geq 0, p_2 \geq 0, p_1 + p_2 \leq v(1)$.

Let $K > 1$. If we consider the restriction of the valuation function $v(\cdot)$ to $K-1$ items, we have a convex polygon ϱ'_v satisfying the induction hypothesis. To account for the K-th item we only need to consider the region $p_1 \geq 0, p_2 \geq 0, p_2 \leq \frac{v(K)-v(q)}{K-q}$ for $0 < q < K$, and $p_1 + p_2 K \leq v(K)$. If this region is empty, $\varrho_v = \varrho'_v$, and we are done. Otherwise, denoted by

Algorithm 18 ComputeFixedAllocationRegions

Input: $v_i(\cdot)$, valuation functions
1. For $i = 1 \dots m$ do
2. $Q \leftarrow \emptyset$ (stack), $q' = 1$, $h = v_i(1)$.
3. For $q = 2 \dots K$ do
3.1 $h' \leftarrow (v_i(q) - v_i(q'))/(q - q')$
3.2 if $0 < h' < h$
$h \leftarrow h'$
Push (q, h') onto Q
$q' \leftarrow q$
3.3 else if $0 < h'$
while $h' \geq h$ do
Pop (q', h) from Q
$(q_1, h_1) \leftarrow \text{Top}(Q)$
$h' \leftarrow (v_i(q) - v_i(q_1))/(q - q_1)$
$h \leftarrow h_1$
$q' \leftarrow q$
Push (q, h') onto Q
4. For $(q, h) \in Q$ do
Obtain $\varrho_{v_i}^{(j)}$ for $q^{(j)} = q$ using lines $p_2 = h$ and $p_1 = v(q) - p_2 q$.
5. Compute intersection of segments in $\varrho_{v_i}^{(j)}$ for $i \in [m]$ using [56].
6. Use the intersection points to compute boundaries of all polygons (pieces) formed by the intersections.

$\varrho_v^{(K')}$ with $q^{(K')} = K$, the region where the buyer would buy K units of the item is a trapezoid with bases parallel to the ρ_1-axis. We claim that $\varrho_v = \left(\varrho'_v \cap p_2 \leq \frac{v(K)-v(q)}{K-q}\right) \cup \varrho_v^{(K')}$ and it satisfies the properties in the lemma.

We have $q^{(K'-1)} = \text{argmin}_q \frac{v(K)-v(q)}{K-q}$ such that the buyer's preference changes from $q' = q^{(K'-1)}$ to $q^{(K')} = K$ units across the line $p_2 = \frac{v(K)-v(q')}{K-q'}$. To prove ϱ_v is convex, we use the inductive hypothesis on ϱ'_v and observe that $\rho_1 = v(K) - p_2 K$ coincides with $\rho'_1 = v(q') - p_2 q'$ for $p_2 = \frac{v(K)-v(q')}{K-q'}$. Also the only side of ϱ_v that is not present in ϱ'_v lies along the line $p_1 + p_2 K = v(K)$, thus ϱ_v has at most $K + 2$ sides. The subdivison property is also readily verified given the construction of ϱ_v from ϱ'_v and $\varrho_v^{(K')}$. □

Based on this structure, we propose Algorithm 18 which runs in in $O(mK \log(mK) + R_\Sigma)$ time.

Theorem 9.3.6. *There is an algorithm (Algorithm 18) that, given valuation functions $v_i(\cdot)$ for $i \in [m]$, computes all the R pieces of the total dual class function $U_{\langle v_1,\dots,v_m\rangle}(\cdot)$ for K units of the good from the m samples in $O(mK \log(mK) + R_\Sigma)$ time.*

Proof. Note that if $0 < q < q'$ and $v_i(q) > v_i(q')$, then for any $\rho_1 \geq 0, \rho_2 \geq 0$ we have that $u_i(q) = v_i(q) - (p_1 + p_2 q) > v_i(q') - (p_1 + p_2 q)$, or the buyer will never prefer quantity q' of the item over the entire tariff domain. Thus, we will assume the valuations $v_i(q)$ are monotonic in q

(we can simply ignore valuations at the larger value for any violation). Algorithm 18 exploits the structure in Lemma 9.3.5 and computes the $O(K)$ line segments bounding the dual class pieces for a single buyer i in $O(K)$ time. Across m buyers, we have $O(mK)$ line segments (computed in $O(mK)$ time). The topological plane-sweep based algorithm of [56] now computes all the intersection points in $O(mK(\log mK) + R_\Sigma)$ time. Here we have used that the number of polytopic vertices is $O(R_\Sigma)$ using standard result for planar graphs. □

We further show that this bound is essentially optimal. A runtime lower bound of $\Omega(mK + R_\Sigma)$ follows simply due to the amount of time needed for reading the complete input and producing the complete output. We prove a stronger lower bound which matches the above upper bound by reduction to the *element uniqueness problem* (given a list of n numbers, are there any duplicates?) for which an $\Omega(n \log n)$ lower bound is known in the algebraic decision-tree model of computation.

Theorem 9.3.7. *Given a list of n numbers $\mathcal{L} = \langle x_1, \ldots, x_n \rangle \in \mathcal{N}^n$, there is a linear time reduction to a m-buyer, K-item TPT pricing given by $v_i(\cdot), i \in [m]$, with $mK = \Theta(n)$, such that the pieces of the total dual class function $U_{\langle v_1, \ldots, v_m \rangle}(\cdot)$ can be used to solve the element uniqueness problem for $\mathcal{L}$ in $O(n)$ time.*

Proof. Let $mK = n$ be any factorization of n into two factors. We construct a m-buyer, $K + 1$ item single TPT pricing scheme as follows. Define $y_j = x_j + \max_k x_k + 1$ for each x_j in the list $\mathcal{L}$. For every buyer $i \in [m]$, we set $v_i(1) = \max_k x_k + 1$ and $v_i(q+1) = \sum_{j=1}^{q} y_{j+(i-1)K}$ for each $q \in [K]$. Buyer i's pieces include the segments $p_2 = (v_i(q+1) - v_i(q))/(q+1-1) = x_{q+(i-1)K}$ for $q \in [K]$ (Lemma 9.3.5). Thus, across all buyers $i \in [m]$, we have $mK = n$ segments along the lines $p_2 = x_j$ for $j \in [n]$. We say a duplicate is present if there are fewer than mK segments parallel to the p_1-axis in the pieces of the total dual class function, otherwise we say 'No' (i.e. all elements are distinct). This completes the linear-time reduction. □

9.3.3 Item-Pricing with anonymous prices

We will consider a market with a single seller, interested in designing a mechanism to sell m distinct items to n buyers. We represent a *bundle* of items by a quantity vector $q \in \mathbb{Z}^m_{\geq 0}$, such that the number of units of the ith item in the bundle denoted by q is given by its ith component $q[i]$. In particular, the bundle consisting of a single copy of item i is denoted by the standard basis vector e_i, where $e_i[j] = \mathbb{I}\{i = j\}$, where $\mathbb{I}\{\cdot\}$ is the 0-1 valued indicator function. Each buyer $j \in [n]$ has a valuation function $v_j : \mathbb{Z}^m_{\geq 0} \to \mathbb{R}_{\geq 0}$ over bundles of items. We denote an allocation as $Q = (q_1, \ldots, q_n)$ where q_j is the bundle of items that buyer j receives under allocation Q. Under *anonymous* prices, the seller sets a price p_i per item i. There is some fixed but arbitrary ordering on the buyers such that the first buyer in the ordering arrives first and buys the bundle of items that maximizes their utility, then the next buyer in the ordering arrives, and so on. For a given buyer j and bundle pair $q_1, q_2 \in \{0, 1\}^m$, buyer j will prefer bundle q_1 over bundle q_2 so long as $u_j(q_1) > u_j(q_2)$ where $u_j(q) = v_j(q) - \sum_{i:q[i]=1} p_i$. Therefore, for a fixed set of buyer values and for each buyer j, their preference ordering over the bundles is completely determined by the $\binom{2^m}{2}$ hyperplanes. The dual class functions are known to be piecewise linear with linear boundaries [17].

To implement Algorithm 15 for this problem we specify how to compute the locally relevant hyperplanes. For any price vector $\mathbf{x} = (p_1, \ldots, p_m)$, say the buyers buy bundles $(q_1, \ldots, q_n)$. For a fixed price vector this can be done in time $O(n2^m)$ by computing $\operatorname{argmax}_{q \subseteq I_j} u_j(q)$ for each buyer at that price vector, where I_j denotes the remaining items after allocations to buyers $1, \ldots, j-1$ at that price. Then for each buyer we have at most $2^m - 1$ potential alternative bundles given by hyperplanes $u_j(q) \geq u_j(q'), q' \neq q_i$, for a total of $t_{\text{LRS}} \leq n(2^m - 1)$ locally relevant hyperplanes. Thus $T_{\text{CLRS}} = O(n2^m)$ for the above approach, and Theorem 9.2.2 implies the following runtime bound.

Theorem 9.3.8. *There exists an implementation of* COMPUTELOCALLYRELEVANTSEPARATORS *in Algorithm 15, which given valuation functions $v_i(\cdot)$ for $i \in [n]$, computes all the R pieces of the dual class function in time $\tilde{O}(R^2(2m)^m n)$, where there are m items and n buyers.*

Proof. In the terminology of Theorem 9.2.2, we have $d = m$, $E \leq R^2$, $V = R$, $T_{\text{CLRS}} = O(n2^m)$, $t_{\text{LRS}} = n(2^m - 1)$. By [53], we have $\sum_{c \in V_H} \text{LP}(d, |I_c| + 1) \leq O(Ed^d) \leq O(R^2 m^m)$. Thus, Theorem 9.2.2 implies a runtime bound on Algorithm 15 of $\tilde{O}(dE + VT_{\text{CLRS}} + t_{\text{LRS}} \cdot \sum_{c \in V_H} \text{LP}(d, |I_c| + 1)) = \tilde{O}(R^2(2m)^m n)$. □

Our approach yields an efficient algorithm when the number of items m and the number of dual class function pieces R are small. Prior work on item-pricing with anonymous prices has only focussed on sample complexity of the data-driven design problem [17].

9.4 Linkage-based clustering

Clustering data into groups of similar points is a fundamental tool in data analysis and unsupervised machine learning. A variety of clustering algorithms have been introduced and studied but it is not clear which algorithms will work best on specific tasks. Also the quality of clustering is heavily dependent on the distance metric used to compare data points. Interpolating multiple metrics and clustering heuristics can result in significantly better clustering [18].

Problem setup. Let $\mathcal{X}$ be the data domain. A clustering instance from the domain consists of a point set $S = \{x_1, \ldots, x_n\} \subseteq \mathcal{X}$ and an (unknown) target clustering $\mathcal{C} = (C_1, \ldots, C_k)$, where the sets $C_1, \ldots, C_k$ partition S into k clusters. Linkage-based clustering algorithms output a hierarchical clustering of the input data, represented by a cluster tree. We measure the agreement of a cluster tree T with the target clustering $\mathcal{C}$ in terms of the Hamming distance between $\mathcal{C}$ and the closest pruning of T that partitions it into k clusters (i.e., k disjoint subtrees that contain all the leaves of T). More formally, the loss $\ell(T, \mathcal{C}) = \min_{P_1, \ldots, P_k} \min_{\sigma \in S_n} \frac{1}{|S|} \sum_{i=1}^{k} |C_i \setminus P_{\sigma_i}|$, where the first minimum is over all prunings $P_1, \ldots, P_k$ of the cluster tree T into k subtrees, and the second minimum is over all permutations of the k cluster indices.

A merge function D defines the distance between a pair of clusters $C_i, C_j \subseteq \mathcal{X}$ in terms of the pairwise point distances given by a metric d. Cluster pairs with smallest values of the merge function are merged first. For example, single linkage uses the merge function $D_{\text{sgl}}(C_i, C_j; d) = \min_{a \in C_i, b \in C_j} d(a, b)$ and complete linkage uses $D_{\text{cmpl}}(C_i, C_j; d) = \max_{a \in C_i, b \in C_j} d(a, b)$. Instead of using extreme points to measure the distance between pairs of clusters, one may also use more central points, e.g. we define *median linkage* as $D_{\text{med}}(C_i, C_j; d) = \texttt{median}(\{d(a, b) \mid$

$a \in C_i, b \in C_j\}$), where $\texttt{median}(\cdot)$ is the usual statistical median of an ordered set $S \subset \mathbb{R}^3$. Single, median and complete linkage are *2-point-based*, i.e. the merge function $D(A, B; d)$ only depends on the distance $d(a, b)$ for two points $(a, b) \in A \times B$.

Parameterized algorithm families. Let $\Delta = \{D_1, \ldots, D_l\}$ denote a finite family of merge functions (measure distances between clusters) and $\delta = \{d_1, \ldots, d_m\}$ be a finite collection of distance metrics (measure distances between points). We define a parameterized family of linkage-based clustering algorithms that allows us to learn both the merge function and the distance metric. It is given by the interpolated merge function $D^{\Delta}_{\alpha}(A, B; \delta) = \sum_{D_i \in \Delta, d_j \in \delta} \alpha_{i,j} D_i(A, B; d_j)$, where $\alpha = \{\alpha_{i,j} \mid i \in [l], j \in [m], \alpha_{i,j} \geq 0\}$. In order to ensure linear boundary functions for the dual class function, our interpolated merge function $D^{\Delta}_{\alpha}(A, B; \delta)$ takes all pairs of distance metrics and linkage procedures. Due to invariance under constant multiplicative factors, we can set $\sum_{i,j} \alpha_{i,j} = 1$ and obtain a set of parameters which allows α to be parameterized by $d = lm - 1$ values[4]. Define the parameter space $\mathcal{P} = \blacktriangle^d = \left\{\rho \in \left(\mathbb{R}^{\geq 0}\right)^d \mid \sum_i \rho_i \leq 1\right\}$; for any $\rho \in \mathcal{P}$ we get $\alpha^{(\rho)} \in \mathbb{R}^{d+1}$ as $\alpha^{(\rho)}_i = \rho_i$ for $i \in [d]$, $\alpha^{(\rho)}_{d+1} = 1 - \sum_{i \in [d]} \rho_i$. We focus on learning the optimal $\rho \in \mathcal{P}$ for a single instance $(S, \mathcal{Y})$. With slight abuse of notation we will sometimes use $D^{\Delta}_{\rho}(A, B; \delta)$ to denote the interpolated merge function $D^{\Delta}_{\alpha^{(\rho)}}(A, B; \delta)$. As a special case we have the family $D^{\Delta}_{\rho}(A, B; d_0)$ that interpolates merge functions (from set Δ) for different linkage procedures but the same distance metric d_0. Another interesting family only interpolates the distance metrics, i.e. use a distance metric $d_\rho(a, b) = \sum_{d_j \in \delta} \alpha^{(\rho)}_j d_j(a, b)$ and use a fixed linkage procedure. We denote this by $D^{1}_{\rho}(A, B; \delta)$.

Overview of techniques. First, we show that for a fixed clustering instance $S \in \Pi$, the dual class function $u_S(\cdot)$ is piecewise constant with a bounded number of pieces and linear boundaries. We only state the result for the interpolation of merge functions $D^{\Delta}_{\rho}(A, B; d_0)$ below for Δ a set of 2-point-based merge functions.

Lemma 9.4.1. *Let Δ be a finite set of 2-point-based merge functions. Let T^S_ρ denote the cluster tree computed using the parameterized merge function $D^{\Delta}_{\rho}(A, B; d_0)$ on sample S. Let $\mathcal{U}$ be the set of functions $\{u_\rho : S \mapsto \ell(T^S_\rho, \mathcal{C}) \mid \rho \in \mathbb{R}^d\}$ that map a clustering instance S to $\mathbb{R}$. The dual class $\mathcal{U}^*$ is $(\mathcal{F}, |S|^{4d+4})$-piecewise decomposable, where $\mathcal{F}$ consists of constant functions $f_c : u_\rho \mapsto c$.*

We will extend the *execution tree* approach introduced by [18] which computes the pieces (intervals) of *single-parameter* linkage-based clustering. Informally, for a single parameter, the execution tree is defined as the partition tree where each node represents an interval where the first t merges are identical, and edges correspond to the subset relationship between intervals obtained by refinement from a single merge. The execution, i.e. the sequence of merges, is unique along any path of this tree. The same properties, i.e. refinement of the partition with each merge and correspondence to the algorithm's execution, continue to hold in the multidimensional case, but

[3] $\texttt{median}(S)$ is the smallest element of S such that S has at most half its elements less than $\texttt{median}(S)$ and at most half its elements more than $\texttt{median}(S)$. For comparison, the more well-known average linkage is $D_{\text{avg}}(C_i, C_j; d) = \texttt{mean}(\{d(a, b) \mid a \in C_i, b \in C_j\})$. We may also use geometric medians of clusters. For example, we can define *mediod linkage* as $D_{\text{geomed}}(C_i, C_j; d) = d(\text{argmin}_{a \in C_i} \sum_{a' \in C_i} d(a, a'), \text{argmin}_{b \in C_j} \sum_{b' \in C_j} d(b, b'))$.

[4] In contrast, the parametric family in [18] has $l + m - 2$ parameters but it does not satisfy Definition 14.

with convex polytopes instead of intervals. Computing the children of any node of the execution tree corresponds to computing the subdivision of a convex polytope into polytopic cells where the next merge step is fixed. The children of any node of the execution tree can be computed using Algorithm 15. We compute the cells by following the neighbors, keeping track of the cluster pairs merged for the computed cells to avoid recomputation. For any single cell, we find the bounding polytope along with cluster pairs corresponding to neighboring cells by computing the tight constraints in a system of linear inequalities. Theorem 9.2.2 gives the runtime complexity of the proposed algorithm for computing the children of any node of the execution tree. It only remains to specify COMPUTELOCALLYRELEVANTSEPARATORS. For a given $\mathbf{x} = \rho$ we find the next merge candidate in time $O(dn^2)$ by computing the merge function $D_\rho^\Delta(A, B; \delta)$ for all pairs of candidate (unmerged) clusters A, B. If (A^*, B^*) minimizes the merge function, the locally relevant hyperplanes are given by $D_\rho^\Delta(A^*, B^*; \delta) \leq D_\rho^\Delta(A', B'; \delta)$ for $(A', B') \neq (A^*, B^*)$ i.e. $t_{\text{LRS}} \leq n^2$. Using Theorem 9.2.2, we give the following bound for the overall runtime of the algorithm.

Theorem 9.4.2. *Let S be a clustering instance with $|S| = n$, and let $R_i = |\mathcal{P}_i|$ and $R = R_n$. Furthermore, let $H_t = |\{(\mathcal{Q}_1, \mathcal{Q}_2) \in \mathcal{P}_t^2 \mid \mathcal{Q}_1 \cap \mathcal{Q}_2 \neq \emptyset\}|$ denote the total number of adjacencies between any two pieces of $\mathcal{P}_i$ and $H = H_n$. Then, the leaves of the execution tree on S can be computed in time $\widetilde{O}\left(\sum_{i=1}^n (H_i + R_i T_M)(n - i + 1)^2\right)$, where T_M is the time to compute the merge function.*

Proof. Let T be the execution tree with respect to S, and let T_t denote the vertices of T at depth t. From Theorem 9.2.2, for each node $v = (\mathcal{M}, \mathcal{Q}) \in T$ with depth t, we can compute the children of v in time $O(n_t^2 \cdot \text{LP}(d, E_v) + V_v \cdot n_t^2 K)$, where V_v is the number of children of v, and

$$E_v = \left| \left\{ \begin{array}{c} (\mathcal{Q}_1, \mathcal{Q}_2) \in \mathcal{P}_{t+1}^2 \mid \mathcal{Q}_1 \cap \mathcal{Q}_2 \neq \emptyset \text{ and} \\ u_1 = (\mathcal{M}_1, \mathcal{Q}_1), u_2 = (\mathcal{M}_2, \mathcal{Q}_2) \\ \text{for some children } u_1, u_2 \text{ of } v \end{array} \right\} \right|.$$

Now, observe

$$\sum_{v \in T_{t+1}} E_v \leq H_{t+1}$$

since H_{t+1} counts all adjacent pieces $\mathcal{Q}_1, \mathcal{Q}_2$ in $\mathcal{P}_{t+1}$; each pair is counted at most once by some E_v. Similarly, we have $\sum_{v \in T_{t+1}} V_v \leq R_{t+1}$, since R_{t+1} counts the total size of $\mathcal{P}_{t+1}$. Note that $n_{t+1} = (n - t)$, since t merges have been executed by time $t + 1$, so $n_i = n - i + 1$. Seidel's algorithm is a randomized algorithm that may be used for efficiently solving linear programs in low dimensions, the expected running time for solving an LP in d variables and m constraints is $O(d! \cdot E_v)$ (also holds with high probability, e.g. Corollary 2.1 of [150]). There are also deterministic algorithms with the same (in fact slightly better) worst-case runtime bounds [53]. Therefore, we can set $\text{LP}(d, E_v) = O(d! \cdot E_v)$. So that the total cost of computing $\mathcal{P}_i$ is

$$O\left(\sum_{i=1}^n \sum_{v \in T_i} d! \cdot E_v (n - i + 1)^2 + V_v K (n - i + 1)^2\right), \text{ or,}$$
$$O\left(\sum_{i=1}^n (d! \cdot H_i + R_i K)(n - i + 1)^2\right),$$

as desired. □

In the case of single, median, and complete linkage, we may assume $T_M = O(d)$ by carefully maintaining a hashtable containing distances between every pair of clusters. Each merge requires overhead at most $O(n_t^2)$, $n_t = n - t$ being the number of unmerged clusters at the node at depth t, which is absorbed by the cost of solving the LP corresponding to the cell of the merge. We have the following corollary which states that our algorithm is output-linear for $d = 2$.

Corollary 9.4.3. *For $d = 2$ the leaves of the execution tree of any clustering instance S with $|S| = n$ can be computed in time $O(RT_M n^3)$.*

Proof of Corollary 9.4.3. The key observation is that on any iteration i, the number of adjacencies $H_i = O(R_i)$. This is because for any region $P \in \mathcal{P}_i$, P is a polygon divided into convex subpolygons, and the graph G_P has vertices which are faces and edges which cross between faces. Since the subdivision of P can be embedded in the plane, so can the graph G_P. Thus G_P is planar, meaning $H_i = O(R_i)$. Plugging into Theorem 9.4.2, noting that $(n-i+1)^2 \leq n^2$, $H_i \leq H$, and $R_i \leq R$, we obtain the desired runtime bound of $O\left(\sum_{i=1}^{n}(R + RT_M)n^2\right) = O(RT_M n^3)$. □

Above results yield bounds on T_S, the enumeration time for dual function of the pieces in a single problem instance. Theorem 9.2.4 further implies bounds on the runtime of ERM.

9.4.1 Comparing the quality of single, complete and median linkage procedures on different data distributions

We will construct clutering instances where each of two-point based linkage procedures, i.e. single, complete and median linkage, dominates the other two procedures. Let $T^S_{\text{sgl}}, T^S_{\text{cmpl}}$ and T^S_{med} denote the cluster tree on clustering instance S using $D_{\text{sgl}}, D_{\text{cmpl}}$ and D_{med} as the merge function (defined in Section 9.4) respectively for some distance metric d which will be evident from context. We have the following theorem.

Theorem 9.4.4. *For any $n \geq 10$, for $i \in \{1, 2, 3\}$, there exist clustering instances S_i with $|S_i| = n$ and target clusterings $\mathcal{C}_i$ such that the hamming loss of the optimal pruning of the cluster trees constructed using single, complete and median linkage procedures (using the same distance metric d) satisfy*

(i) $\ell(T^{S_1}_{sgl}, \mathcal{C}_1) = O(\frac{1}{n})$, $\ell(T^{S_1}_{cmpl}, \mathcal{C}_1) = \Omega(1)$ *and* $\ell(T^{S_1}_{med}, \mathcal{C}_1) = \Omega(1)$,
(ii) $\ell(T^{S_2}_{cmpl}, \mathcal{C}_2) = O(\frac{1}{n})$, $\ell(T^{S_2}_{sgl}, \mathcal{C}_2) = \Omega(1)$ *and* $\ell(T^{S_2}_{med}, \mathcal{C}_2) = \Omega(1)$,
(iii) $\ell(T^{S_3}_{med}, \mathcal{C}_3) = O(\frac{1}{n})$, $\ell(T^{S_3}_{cmpl}, \mathcal{C}_3) = \Omega(1)$ *and* $\ell(T^{S_3}_{sgl}, \mathcal{C}_3) = \Omega(1)$.

Proof. In the following constructions we will have $S_i \subset \mathbb{R}^2$ and the distance metric d will be the Euclidean metric. Also we will have number of target clusters $k = 2$.

Construction of $S_1, \mathcal{C}_1$. For S_1, we will specify the points using their polar coordinates. We place a single point x at the origin $(0, \phi)$ and $\frac{n-1}{8}$ points each along the unit circle at locations $y_1 = (1, 0), y_2 = (1, \frac{\pi}{4} - \epsilon), y_3 = (1, \frac{\pi}{2}), y_4 = (1, \frac{3\pi}{4} - \epsilon), y_5 = (1, \pi), y_6 = (1, \frac{5\pi}{4} - \epsilon), y_7 = (1, \frac{3\pi}{2})$ and $y_8 = (1, \frac{7\pi}{4} - \epsilon)$, where $\epsilon = 0.001$. Also set $\mathcal{C}_1 = \{\{x\}, S_1 \setminus \{x\}\}$.

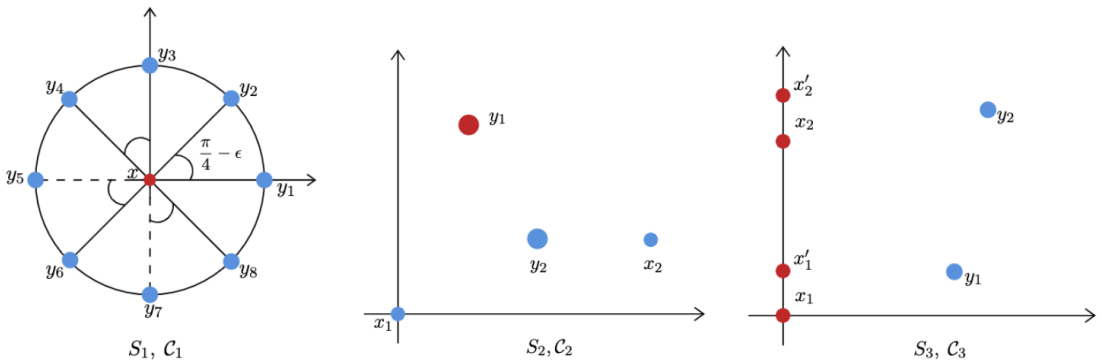


Figure 9.1: Construction of clustering instances showing the need for interpolating linkage heuristics. We give concrete instances and target clusterings where each of two-point based linkage procedures, i.e. single, complete and median linkage, dominates the other two.

In each linkage procedure, the first $n - 9$ merges will join coincident points at locations $y_j, j \in [8]$, let $\tilde{y}_j$ denote the corresponding sets of merged points. The next four merges will be $z_j := \{\tilde{y}_j, \tilde{y}_{j+1}\}$ for $j \in \{1, 3, 5, 7\}$ for each procedure since $d(y_j, y_{j+1}) = \sqrt{2 - 2\cos(\frac{\pi}{4} - \epsilon)} < \min\{\sqrt{2 - 2\cos(\frac{\pi}{4} + \epsilon)}, 1\}$, again common across all procedures. Now single linkage will continue to merge clusters on the unit circle since $\sqrt{2 - 2\cos(\frac{\pi}{4} + \epsilon)} < 1$, however both complete and median linkage will join each of $z_j, j \in \{1, 3, 5, 7\}$ to the singleton cluster $\{x\}$ since the median (and therefore also the maximum distance between points in $z_j, z_{j'}, j \neq j'$ is at least $\sqrt{2} > 1$. Therefore a 2-pruning[5] of $T^{S_1}_{\text{sgl}}$ yields $\mathcal{C}_1$ i.e $\ell(T^{S_1}_{\text{sgl}}, \mathcal{C}_1) = 0$, while a 2-pruning of $T^{S_1}_{\text{cmpl}}$ or $T^{S_1}_{\text{med}}$ would yield $\{z_j, S_1 \setminus z_j\}$ for some $j \in \{1, 3, 5, 7\}$, corresponding to a hamming loss of $\frac{1}{4} + \Omega(\frac{1}{n}) = \Omega(1)$.

Construction of $S_2, \mathcal{C}_2$. For S_2, we will specify the points using their Cartesian coordinates. We place single points x_1, x_2 at $(0, 0)$ and $(3.2, 0.5)$ and $\frac{n-2}{2}$ points each at $y_1 = (1.1, 1.8)$ and $y_2 = (1.8, 0.5)$. We set $\mathcal{C}_2 = \{\{(x, y) \in S_2 \mid y > 1\}, \{(x, y) \in S_2 \mid y \leq 1\}\}$. The distances between pairs of points may be ordered as

$$d(x_2, y_2) = 1.4 < d(y_1, y_2) \approx 1.5 < d(x_1, y_2) \approx 1.9 < d(x_1, y_1) \approx 2.1 < d(x_2, y_1) \approx 2.5 < d(x_1, x_2)$$

All linkage procedures will merge the coincident points at y_1 and y_2 (respectively) for the first $n - 4$ merges. Denote the corresponding clusters by $\tilde{y}_1$ and $\tilde{y}_2$ respectively. The next merge will be $z_2 := \{x_2, \tilde{y}_2\}$ in all cases. Now single linkage will join z_2 with $\tilde{y}_1$. Further, since $n \geq 10$, $\frac{n-2}{2} \geq 4$ and therefore the median distance between z_2 and $\tilde{y}_1$ is also $d(y_1, y_2)$. However, since $d(x_1, y_1) < d(x_2, y_1)$, the complete linkage procedure will merge $\{x_1, z_2\}$. Finally, the two remaining clusters are merged in each of the two procedures. Clearly, 2-pruning of $T^{S_2}_{\text{cmpl}}$ yields $\mathcal{C}_2$ or $\ell(T^{S_2}_{\text{cmpl}}, \mathcal{C}_2) = 0$. However, $\ell(T^{S_2}_{\text{sgl}}, \mathcal{C}_2) = \ell(T^{S_2}_{\text{med}}, \mathcal{C}_2) = \frac{1}{2} - O(\frac{1}{n}) = \Omega(1)$.

[5]A k_0-pruning for a tree T is a partition of the points contained in T's root into k_0 clusters such that each cluster is an internal node of T.

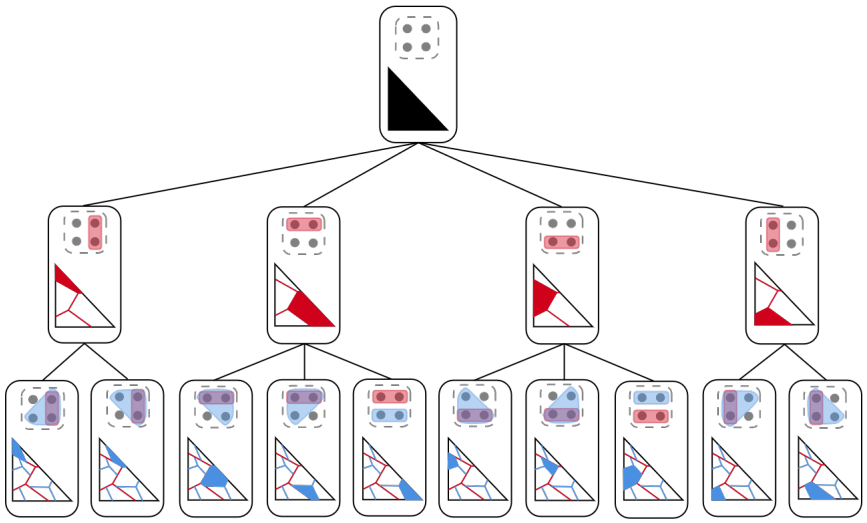


Figure 9.2: The first three levels of an example execution tree of a clustering instance on four points, with a two-parameter algorithm ($\mathcal{P} = \blacktriangle^2$). Successive partitions $\mathcal{P}_0$, $\mathcal{P}_1$, $\mathcal{P}_2$ are shown at merge levels 0, 1, and 2, respectively, and the nested shapes show cluster merges.

Construction of $S_3, \mathcal{C}_3$. We specify the points in S_3 using their Cartesian coordinates. We place $\frac{n-1}{6}$ points each at $x_1 = (0,0), x_2' = (0, 1+2\epsilon)$, $\frac{n-1}{12}$ points each at $x_1' = (0,\epsilon), x_2 = (0, 1+\epsilon)$, $\frac{n-1}{4}$ points each at $y_1 = (1+0.9\epsilon, \epsilon), y_2 = (1+\epsilon, 1+1.9\epsilon)$, and one point $z_1 = (0,2)$ with $\epsilon = 0.3$. With some abuse of notation we will use the coordinate variables defined above to also denote the collection of points at their respective locations. We set $\mathcal{C}_3 = \{\{(x,y) \in S_2 \mid x \leq 0\}, \{(x,y) \in S_2 \mid x > 0\}\}$.

After merging the coincident points, all procedures will merge clusters $\tilde{x}_1 := \{x_1, x_1'\}$ and $\tilde{x}_2 := \{x_2, x_2', z_1\}$. Let us now consider the single linkage merge function. We have that $D_{\text{sgl}}(\tilde{x}_1, \tilde{x}_2; d) = 1$ and all other cluster pairs are further apart. The next merge is therefore $\tilde{x} := \{\tilde{x}_1, \tilde{x}_2\}$. Also, $D_{\text{sgl}}(\tilde{x}, y_1; d) = 1 + 0.9\epsilon < \min\{D_{\text{sgl}}(\tilde{x}, y_2; d), D_{\text{sgl}}(y_1, y_2; d)\}$ leading to the merge $\{\tilde{x}, y_1\}$, and finally y_2 is merged in. A 2-pruning therefore has loss $\ell(T^{S_3}_{\text{sgl}}, \mathcal{C}_3) = \Omega(1)$. On the other hand, $D_{\text{med}}(\tilde{x}_1, \tilde{x}_2; d) = 1 + \epsilon > D_{\text{med}}(y_1, y_2; d) = \sqrt{(1+0.9\epsilon)^2 + 0.01\epsilon^2}$ and $D_{\text{med}}(\tilde{x}_1, y_1; d) = \sqrt{(1+0.9\epsilon)^2 + \epsilon^2} > \{D_{\text{med}}(y_1, y_2; d), D_{\text{med}}(\tilde{x}_1, \tilde{x}_2; d)\}$. As a result, median linkage would first merge $\{y_1, y_2\}$, followed by $\{\tilde{x}_1, \tilde{x}_2\}$, and 2-pruning yields $\mathcal{C}_3$. Complete linkage also merges $\{y_1, y_2\}$ first. But $D_{\text{cmpl}}(\tilde{x}_1, \tilde{x}_2; d) = 2 > D_{\text{cmpl}}(\tilde{x}_1, \{y_1, y_2\}; d)$. Thus, $\ell(T^{S_3}_{\text{cmpl}}, \mathcal{C}_3) = \Omega(1)$.

□

9.4.2 Piecewise dual structure

Definition 32 (2-point-based merge function [18])**.** *A merge function D is* 2-point-based *if for any pair of clusters $A, B \subseteq \mathcal{X}$ and any metric d, there exists a set of points $(a, b) \in A \times B$ such that $D(A, B; d) = d(a, b)$. Furthermore, the selection of a and b only depend on the relative*

ordering of the distances between points in A and B. More formally, for any metrics d and d' such that $d(a,b) \le d(a',b')$ if and only if $d'(a,b) \le d'(a',b')$, then $D(A,B;d) = d(a,b)$ implies $D(A,B;d') = d'(a,b)$.

For instance, single, median and complete linkage are 2-point-based, since the merge function $D(A,B;d)$ only depends on the distance $d(a,b)$ for some $a \in A, b \in B$. We have the following observation about our parameterized algorithm families $D_\rho^\Delta(A,B;\delta)$ when Δ consists of 2-point-based merge functions which essentially establishes piecewise structure with linear boundaries (in the sense of Definition 14).

Lemma 9.4.5. *Suppose $S \in \Pi$ is a clustering instance, Δ is a set of 2-point-based merge functions with $|\Delta| = l$, and δ is a set of distance metrics with $|\delta| = m$. Consider the family of clustering algorithms with the parameterized merge function $D_\rho^\Delta(A,B;\delta)$. The corresponding dual class function $u_S(\cdot)$ is piecewise constant with $O(|S|^{4lm})$ linear boundaries.*

Proof. Let $(a_{ij}, b_{ij}, a'_{ij}, b'_{ij})_{1\le i\le l, 1\le j\le m} \subseteq S$ be sequences of lm points each; for each such a, let $g_a : \mathcal{P} \to \mathbb{R}$ denote the function

$$g_a(\rho) = \sum_{i\in[l], d_j\in\delta} \alpha_{i,j}(\rho)(d_j(a_{ij}, b_{ij}) - d_j(a'_{ij}, b'_{ij}))$$

and let $\mathcal{G} = \{g_a \mid (a_{ij}, b_{ij}, a'_{ij}, b'_{ij})_{1\le i\le l, 1\le j\le m} \subseteq S\}$ be the collection of all such linear functions; notice that $|\mathcal{G}| = O(|S|^{4lm})$. Fix $\rho, \rho' \in \mathcal{P}$ with $g(\rho)$ and $g(\rho')$ having the same sign patterns for all such g. For each $A, B, A', B' \subseteq S$, $D_i \in \Delta$, and $d_j \in \delta$, we have $D_i(A,B;d_j) = d_j(a,b)$ and $D_i(A',B';d_j) = d_j(a',b')$ for some $a, b, a', b' \in S$ (since D_i is 2-point-based). Thus we can write $D_\rho(A,B;\delta) = \sum_{i\in[m],d_j\in\delta} \alpha_{i,j}(\rho) d_j(a_{ij}, b_{ij})$ for some $a_{ij}, b_{ij} \in S$; similarly, $D_\rho(A',B';\delta) = \sum_{i\in[m],d_j\in\delta} \alpha_{i,j}(\rho) d_j(a'_{ij}, b'_{ij})$ for some $a'_{ij}, b'_{ij} \in S$. As a result, $D_\rho(A,B;\delta) \le D_\rho(A',B';\delta)$ if and only if

$$\sum_{i\in[l], d_j\in\delta} \alpha_{i,j}(\rho) \left(d_j(a_{ij}, b_{ij}) - d_j(a'_{ij}, b'_{ij})\right) \le 0$$

which is exactly when $g_a(\rho) \le 0$ for some sequence a. Since $g_a(\rho)$ and $g_a(\rho')$ have the same sign pattern, we have $D_\rho(A,B;\delta) \le D_\rho(A',B';\delta)$ if and only if $D_{\rho'}(A,B;\delta) \le D_{\rho'}(A',B';\delta)$. So ρ and ρ' induce the same sequence of merges, meaning the algorithm's output is constant on each piece induced by g, as desired. □

From Lemma 9.4.5, we obtain a bound on the number of hyperplanes needed to divide $\mathcal{P}$ into output-constant pieces. Let H be a set of hyperplanes which splits $\mathcal{P}$ into output-constant pieces; then, a naive approach to finding a dual-minimizing $\rho \in \mathcal{P}$ is to enumerate all pieces generated by H, requiring $O(|H|^d)$ runtime. However, by constructing regions merge-by-merge and successively refining the parameter space, we can obtain a better runtime bound which is output-sensitive in the total number of pieces.

Proof of Lemma 9.4.1. This is a simple corollary of Lemma 9.4.5 for $m = 1$. In this case, we have $l = d + 1$. □

Lemma 9.4.6. *Consider the family of clustering algorithms with the parameterized merge function $D^1_\rho(A, B; \delta)$. Let T^S_ρ denote the cluster tree computed using the parameterized merge function $D^\Delta_\rho(A, B; d_0)$ on sample S. Let $\mathcal{U}$ be the set of functions $\{u_\rho : S \mapsto \ell(T^S_\rho, \mathcal{C}) \mid \rho \in \mathbb{R}^d\}$ that map a clustering instance S to $\mathbb{R}$. The dual class $\mathcal{U}^*$ is $(\mathcal{F}, |S|^4)$-piecewise decomposable, where $\mathcal{F} = \{f_c : \mathcal{U} \to \mathbb{R} \mid c \in \mathbb{R}\}$ consists of constant functions $f_c : u_\rho \mapsto c$.*

The key observation for the proof comes from [18] where it is observed that two parameterized distance metrics d_{ρ_1}, d_{ρ_2} behave identically (yield the same cluster tree) on a given dataset S if the relative distance for all pairs of two points $(a, b), (a', b') \in S^2 \times S^2$, $d_\rho(a, b) - d_\rho(a', b')$, has the same sign for ρ_1, ρ_2. This corresponds to a partition of the parameter space with $|S|^4$ hyperplanes, with all distance metrics behaving identically in each piece of the partition. More formally, we have

Proof of Lemma 9.4.6. Let S be any clustering instance. Fix points $a, b, a', b' \in S$. Define the linear function $g_{a,b,a',b'}(\rho) = \sum_i \rho_i(d_i(a, b) - d_i(a', b'))$. If $d_\rho(\cdot, \cdot)$ denotes the interpolated distance metric, we have that $d_\rho(a, b) \leq d_\rho(a', b')$ if and only if $g_{a,b,a',b'}(\rho) \leq 0$. Therefore we have a set $H = \{g_{a,b,a',b'}(\rho) \leq 0 \mid a, b, a', b' \in S\}$ of $|S|^4$ hyperplanes such that in any piece of the sign-pattern partition of the parameter space by the hyperplanes, the interpolated distance metric behaves identically, i.e. for any ρ, ρ' in the same piece $d_\rho(a, b) \leq d_\rho(a', b')$ iff $d_{\rho'}(a, b) \leq d_{\rho'}(a', b')$. The resulting clustering is therefore identical in these pieces. This means that for any connected component R of $\mathbb{R}^d \setminus H$, there exists a real value c_R such that $u_\rho(s_1, s_2) = c_R$ for all $\rho \in \mathbb{R}^d$. By definition of the dual, $u^*_{s_1,s_2}(u_\rho) = u_\rho(s_1, s_2) = c_R$. For each hyperplane $h \in H$, let $g^{(h)} \in \mathcal{G}$ denote the corresponding halfspace. Order these $k = |S|^4$ functions arbitrarily as $g_1, \ldots, g_k$. For a given connected component R of $\mathbb{R}^d \setminus H$, let $\mathbf{b}_R \in \{0, 1\}^k$ be the corresponding sign pattern. Define the function $f^{(\mathbf{b}_R)} = f_{c_R}$ and for $\mathbf{b}$ not corresponding to any R, $f^{(\mathbf{b})} = f_0$. Thus, for each $\rho \in \mathbb{R}^d$,

$$u^*_{s_1,s_2}(u_\rho) = \sum_{\mathbf{b} \in \{0,1\}^k} \mathbb{I}\{g_i(u_\rho) = b_i \forall i \in [k]\} f^{(\mathbf{b})}(u_\rho).$$

□

Corollary 9.4.7. *For any clustering instance $S \in \Pi$, the dual class function $u_S(\cdot)$ for the family in Lemma 9.4.6 is piecewise constant with $O\left(|S|^{4d}\right)$ pieces.*

Lemma 9.4.8. *Let $S \in \Pi$ be a clustering instance, Δ be a set of merge functions, and δ be a set of distance metrics. Then, the corresponding dual class function $u_S(\cdot)$ is piecewise constant with $O(16^{|S|})$ linear boundaries of pieces.*

Proof. For each subset of points $A, B, A', B' \subseteq S$, let $g_{A,B,A',B'} : \mathcal{P} \to \mathbb{R}$ denote the function

$$g_{A,B,A',B'}(\rho) = D_\rho(A, B; \delta) - D_\rho(A', B'; \delta)$$

and let $\mathcal{G}$ be the collection of all such functions for distinct subsets A, B, A', B'. Observe that $\mathcal{G}$ is a class of linear functions with $|\mathcal{G}| \leq \left(2^{|S|}\right)^4 = 16^{|S|}$. Suppose that for $\rho, \rho' \in \mathcal{P}$, $g(\rho)$ and $g(\rho')$ have the same sign for all $g \in G$; then, the ordering over all cluster pairs A, B of $D_\rho(A, B; \delta)$ is the same as that of $D_{\rho'}(A, B; \delta)$. At each stage of the algorithm, the cluster pair $A, B \subseteq S$

minimizing $D_\rho(A, B; \delta)$ is the same as that which minimizes $D_{\rho'}(A, B; \delta)$, so the sequences of merges produced by ρ and ρ' are the same. Thus the algorithm's output is constant on the region induced by $g_{A,B,A',B'}$, meaning $u_S(\cdot)$ is piecewise constant on the regions induced by $\mathcal{G}$, which have linear boundaries. □

9.4.3 Execution Tree

Formally we define an execution tree (Figure 9.2) as follows.

Definition 33 (Execution tree)**.** *Let S be a clustering instance with $|S| = n$, and $\emptyset \neq \mathcal{P} \subseteq [0, 1]^d$. The* execution tree *on S with respect to $\mathcal{P}$ is a depth-n rooted tree T, whose nodes are defined recursively as follows: $r = ([], \mathcal{P})$ is the root, where $[]$ denotes the empty sequence; then, for any node $v = ([u_1, u_2, \ldots, u_t], \mathcal{Q}) \in T$ with $t < n - 1$, the children of v are defined as*

$$\textit{children}(v) = \left\{ \left([u_1, \ldots, u_t, (A, B)], \mathcal{Q}_{A,B} \right) : \begin{array}{c} A, B \subseteq S \textit{ is the } (t+1)^{st} \textit{ merge by} \\ \mathcal{A}_\rho \textit{ for exactly the } \rho \in \mathcal{Q}_{A,B} \subseteq \mathcal{P}, \\ \textit{with } \emptyset \neq \mathcal{Q}_{A,B} \subseteq \mathcal{Q} \end{array} \right\}.$$

For an execution tree T with $v \in T$ and each i with $0 \leq i \leq n$, we let $\mathcal{P}_i$ denote the set of $\mathcal{Q}$ such that there exists a depth-i node $v \in T$ and a sequence of merges $\mathcal{M}$ with $v = (\mathcal{M}, \mathcal{Q})$. Intuitively, the execution tree represents all possible execution paths (i.e. sequences for the merges) for the algorithm family when run on the instance S as we vary the algorithm parameter $\rho \in \mathcal{P}$. Furthermore, each $\mathcal{P}_i$ is a subdivision of the parameter space into pieces where each piece has the first i merges constant. We establish the execution tree captures all possible sequences of merges by some algorithm $\mathcal{A}_\rho$ in the parameterized family via its nodes, and each node corresponds to a convex polytope if the parameter space $\mathcal{P}$ is a convex polytope (Lemmata 9.4.9 and 9.4.10).

Our cell enumeration algorithm for computing all the pieces of the dual class function now simply computes the execution tree, using Algorithm 15 to compute the children nodes for any given node, starting with the root.

Lemma 9.4.9. *Let S be a clustering instance and T be its execution tree with respect to $\mathcal{P}$. Then, if a sequence of merges $\mathcal{M} = [u_1, u_2, \ldots, u_t]$ is attained by $\mathcal{A}_\rho$ for some $\rho \in \mathcal{P}$, then there exists some $v \in T$ at depth t with $v = (\mathcal{M}, \mathcal{Q})$ and with $\mathcal{Q} \subseteq \mathcal{P}$ being the exact set of values of ρ for which $\mathcal{A}_\rho$ may attain $\mathcal{M}$. Conversely, for every node $v = (\mathcal{M}, \mathcal{Q}) \in T$, $\mathcal{M}$ is a valid sequence of merges attainable by $\mathcal{A}_\rho$ for some $\rho \in \mathcal{P}$.*

Proof. We proceed by induction on t. For $t = 0$, the only possible sequence of merges is the empty sequence, which is obtained for all $\rho \in \mathcal{P}$. Furthermore, the only node in T at depth 0 is the root $([], \mathcal{P})$, and the set $\mathcal{P}$ is exactly where an empty sequence of merges occurs.

Now, suppose the claim holds for some $t \geq 0$. We show both directions in the induction step.

For the forward direction, let $\mathcal{M}_{t+1} = [u_1, u_2, \ldots, u_t, u_{t+1}]$, and suppose $\mathcal{M}_{t+1}$ is attained by $\mathcal{A}_\rho$ for some $\rho \in \mathcal{P}$. This means that $\mathcal{M}_t = [u_1, u_2, \ldots, u_t]$ is attained by $\mathcal{A}_\rho$ as well; by the induction hypothesis, there exists some node $v_t = (\mathcal{M}_t, \mathcal{Q}_t) \in T$ at depth t, where $\rho \in \mathcal{Q}_t$ and $\mathcal{Q}_t$ is exactly the set of values for which $\mathcal{A}$ may attain $\mathcal{M}_t$. Now, u_{t+1} is a possible next merge by $\mathcal{A}_\rho$ for some $\rho \in \mathcal{Q}_t$; by definition of the execution tree, this means v_t has some child $v_{t+1} = (\mathcal{M}_{t+1}, \mathcal{Q}_{t+1})$ in T such that $\mathcal{Q}_{t+1}$ is the set of values where u_{t+1} is the next merge in $\mathcal{Q}_t$.

Moreover, $\mathcal{Q}_{t+1}$ is exactly the set of values $\rho \in \mathcal{P}$ for which A_ρ can attain the merge sequence $\mathcal{M}_{t+1}$. In other words for any $\rho' \in \mathcal{P} \setminus \mathcal{Q}_{t+1}$, $A_{\rho'}$ cannot attain the merge sequence $\mathcal{M}_{t+1}$. Otherwise, either some $\rho' \in \mathcal{P} \setminus \mathcal{Q}_t$ attains $\mathcal{M}_{t+1}$, meaning $A_{\rho'}$ attains $\mathcal{M}_t$ (contradicting the induction hypothesis), or $A_{\rho'}$ attains $\mathcal{M}_{t+1}$ for some $\rho' \in \mathcal{Q}_{t+1} \setminus \mathcal{Q}_t$, contradicting the definition of $\mathcal{Q}_{t+1}$.

For the backward direction, let $v_{t+1} = (\mathcal{M}_{t+1}, \mathcal{Q}_{t+1}) \in T$ at depth $t+1$. Since v_{t+1} is not the root, v_{t+1} must be the child of some node v_t, which has depth t. By the induction hypothesis, $v_t = (\mathcal{M}_t, \mathcal{Q}_t)$, where $\mathcal{M}_t = [u_1, u_2, \ldots, u_t]$ is attained by $\mathcal{A}_\rho$ for some $\rho \in \mathcal{P}$. Thus by definition of the execution tree, $\mathcal{M}_{t+1}$ has the form $[u_1, u_2, \ldots, u_t, (A, B)]$, for some merging of cluster pairs (A, B) which is realizable for $\rho \in \mathcal{Q}_t$. Thus $\mathcal{M}_{t+1}$ is a valid sequence of merges attainable by $\mathcal{A}_\rho$ for some $\rho \in \mathcal{P}$. □

Lemma 9.4.10. *Let S be a clustering instance and T be its execution tree with respect to $\mathcal{P}$. Suppose $\mathcal{P}$ is a convex polytope; then, for each $v = (\mathcal{M}, \mathcal{Q}) \in T$, $\mathcal{Q}$ is a convex polytope.*

Proof. We proceed by induction on the tree depth t. For $t = 0$, the only node is $([], \mathcal{P})$, and $\mathcal{P}$ is a convex polytope. Now, consider a node $v \in T$ at depth $t+1$; by definition of the execution tree, $v = (\mathcal{M}_v, \mathcal{Q}_v)$ is the child of some node $u \in T$, where the depth of u is t. Inductively, we know that $w = (\mathcal{M}_w, \mathcal{Q}_w)$, for some convex polytope $\mathcal{Q}_w$. We also know $\mathcal{M}_w$ has the form $\mathcal{M}_w = [u_1, u_2, \ldots, u_t]$, and thus $\mathcal{Q}_v$ is defined to be the set of points $\rho \in \mathcal{Q}_w$ where the merge sequence $\mathcal{M}_v = [u_1, u_2, \ldots, u_t, (A, B)]$ is attainable for some fixed $A, B \subseteq S$. Notice that the definition of being attainable by the algorithm $\mathcal{A}_\rho$ is that $D_\rho(A, B; \delta)$ is minimized over all choices of next cluster pairs A', B' to merge. That is, $\mathcal{Q}_v$ is the set of points

$$\{\rho \in \mathcal{Q}_w \mid D_\rho(A, B; \delta) \leq D_\rho(A', B'; \delta) \text{ for all available cluster pairs } A', B' \text{ after } \mathcal{M}_w\}$$

Since $D_\rho(A, B; \delta)$ is an affine function of ρ, the constraint $D_\rho(A, B; \delta) \leq D_\rho(A', B'; \delta)$ is a half-space. In other words, $\mathcal{Q}_v$ is the intersection of a convex polytope $\mathcal{Q}_w$ with finitely many half space constraints, meaning $\mathcal{Q}_v$ is itself a convex polytope. □

It follows from Lemma 9.4.10 that $\mathcal{P}_i$ forms a convex subdivision of $\mathcal{P}$, where each $\mathcal{P}_{i+1}$ is a refinement of $\mathcal{P}_i$; Figure 9.2 (in the appendix) shows an example execution tree corresponding to a partition of a 2-dimensional parameter space. From Lemma 9.4.9, the sequence of the first i merges stays constant on each region $P \in \mathcal{P}_i$. Our algorithm computes a representation of the execution tree of an instance S with respect to $\mathcal{P}$; to do so, it suffices to provide a procedure to list the children of a node in the execution tree. Then, a simple breadth-first search from the root will enumerate all the leaves in the execution tree.

Now, our goal is to subdivide P into regions in which the $(j+1)^{\text{st}}$ merge is constant. Each region corresponds to a cluster pair being merged at step $j+1$. Since we know these regions are always convex polytopes (Lemma 9.4.10), we can provide an efficient algorithm for enumerating these regions.

Our algorithm provides an output-sensitive guarantee by ignoring the cluster pairs which are never merged. Supposing there are n_t unmerged clusters, we start with some point $x \in P$ and determine which piece W it is in. Then, we search for more non-empty pieces contained in P by listing the "neighbors" of W. The neighbors of W are pieces inside P that are adjacent to W; to this end, we will more formally define a graph G_P associated with P where each vertex is a

piece and two vertices have an edge when the pieces are adjacent in space. Then we show that we can enumerate neighbors of a vertex efficiently and establish that G_P is connected. It follows that listing the pieces is simply a matter of running a graph search algorithm from one vertex of G_P, thus only incurring a cost for each *non-empty piece* rather than enumerating through all n_t^4 pairs of pieces.

9.4.4 Auxiliary lemmas and proofs of runtime bounds

Lemma 9.4.11. *Fix an affine function $f : \mathbb{R} \to \mathbb{R}^d$ via $f(x) = xa + b$, for $a, b \in \mathbb{R}^d$ and $a \neq 0^d$. For a subset $S \subseteq \mathbb{R}$, if $f(S)$ is convex and closed, then S is also convex and closed.*

Proof. First note that f is injective, since $a \neq 0^d$. To show convexity of S, take arbitrary $x, y \in S$ and $\lambda \in [0, 1]$; we show that $\lambda x + (1 - \lambda)y \in S$. Consider $f(\lambda x + (1 - \lambda)y)$:

$$\begin{aligned} f(\lambda x + (1 - \lambda)y) &= (\lambda x + (1 - \lambda)y)a + b \\ &= \lambda(xa + b) + (1 - \lambda)(ya + b) \end{aligned}$$

By definition, $ya + b, xa + b \in f(S)$, so it follows that $f(\lambda x + (1 - \lambda)y) \in f(S)$ by convexity of $f(S)$. So there exists some $z \in S$ with $f(z) = f(\lambda x + (1 - \lambda)y)$, but since f is injective, $\lambda x + (1 - \lambda)y = z \in S$. Thus S is convex.

To show closedness of S, we show $\mathbb{R} \setminus S$ is open. Let $N(x, r)$ denote the open ball of radius r around x, in either one-dimensional or d-dimensional space. Let $x \in \mathbb{R} \setminus S$; we know $f(x) \notin f(S)$ since f is injective. Since $\mathbb{R}^d \setminus f(S)$ is open, there exists some $r > 0$ with $N(f(x), r) \subseteq \mathbb{R}^d \setminus f(S)$. Then, take $\mathbf{e} = \frac{r}{\|a\|_2} > 0$; for every $y \in N(x, \mathbf{e})$, we have

$$\|f(x) - f(y)\|_2 = \|xa + b - ya - b\|_2 < |x - y|\|a\|_2 \leq r$$

and so $f(y) \in N(f(x), r) \subseteq \mathbb{R}^d \setminus f(S)$, meaning $y \notin S$ since f is injective. Thus $N(x, \mathbf{e}) \subseteq R \setminus S$, meaning S is closed as desired. □

This allows us to prove the following key lemma. We describe a proof sketch first. For arbitrary $(i, j), (i', j') \in V_P^*$, we show that there exists a path from (i, j) to (i', j') in G_P. We pick arbitrary points $w \in Q_{i,j}, x \in Q_{i',j'}$; we can do this because by definition, V_P^* only has elements corresponding to non-empty cluster pairs. Then, we draw a straight line segment in P from w to x. When we do so, we may pass through other sets on the way; each time we pass into a new region, we traverse an edge in G_P, so the sequence of regions we pass through on this line determines a G_P-path from w to x.

Lemma 9.4.12. *The vertices V_P^* of the region adjacency graph G_P form a connected component; all other connected components of G_P are isolated vertices.*

Proof. It suffices to show that for arbitrary vertices $(i_1, j_1), (i_2, j_2) \in V_P$, there exists a path from (i_1, j_1) to (i_2, j_2) in G_P. For ease of notation, define $Q_1 = Q_{i_1,j_1}$ and $Q_2 = Q_{i_2,j_2}$.

Fix arbitrary points $u \in Q_1$ and $w \in Q_2$. If $u = w$ then we're done, since the edge from (i_1, j_1) to (i_2, j_2) exists, so suppose $u \neq w$. Consider the line segment L defined as

$$L = \{\lambda u + (1 - \lambda)w : \lambda \in [0, 1]\}$$

Since $Q_1, Q_2 \subseteq P$, we have $u, w \in P$. Furthermore, by convexity of P, it follows that $L \subseteq P$.

Define the sets $R_{i,j}$ as

$$R_{i,j} = Q_{i,j} \cap L$$

Since each $Q_{i,j}$ and L are convex and closed, so is each $R_{i,j}$. Furthermore, since $\bigcup_{i,j} Q_{i,j} = P$, we must have $\bigcup_{i,j} R_{i,j} = L$. Finally, define the sets $S_{i,j}$ as

$$S_{i,j} = \{t \in [0,1] : tu + (1-t)w \in R_{i,j}\} \subseteq [0,1]$$

Note that $S_{i,j}$ is convex and closed; the affine map $f : S_{i,j} \to R_{i,j}$ given by $f(x) = xu + (1-x)w = x(u-w) + w$ has $R_{i,j}$ as an image. Furthermore, $u - w \neq 0^d$; by Lemma 9.4.11, the preimage $S_{i,j}$ must be convex and closed. Furthermore, $\bigcup_{i,j} S_{i,j} = [0,1]$.

The only convex, closed subsets of $[0,1]$ are closed intervals. We sort the intervals in increasing order based on their lower endpoint, giving us intervals $I_1, I_2, \ldots, I_\ell$. We also assume all intervals are non-empty (we throw out empty intervals). Let $\sigma(p)$ denote the corresponding cluster pair associated with interval I_p; that is, if the interval I_p is formed from the set $S_{i,j}$, then $\sigma(p) = (i,j)$.

Define a_i, b_i to be the lower and upper endpoints, respectively, of I_i. We want to show that for all $1 \leq i \leq \ell - 1$, the edge $\{\sigma(i), \sigma(i+1)\} \in E_P$; this would show that $\sigma(1)$ is connected to $\sigma(\ell)$ in the V_P. But $\sigma(1) = (i_1, j_1)$ and $\sigma(\ell) = (i_2, j_2)$, so this suffices for our claim.

Now consider intervals $I_i = [a_i, b_i]$ and $I_{i+1} = [a_{i+1}, b_{i+1}]$. It must be the case that $b_i = a_{i+1}$; otherwise, some smaller interval would fit in the range $[b_i, a_{i+1}]$, and it would be placed before I_{i+1} in the interval ordering.

Since $b_i \in I_i \cap I_{i+1}$, by definition, $ub_i + (1-b_i)w \in R_{\sigma(i)} \cap R_{\sigma(i+1)}$. In particular, $ub_i + (1-b_i)w \in Q_{\sigma(i)} \cap Q_{\sigma(i+1)}$; by definition of E_P, this means $\{\sigma(i), \sigma(i+1)\} \in E_P$, as desired. □

9.5 Global sequence alignment

Sequence alignment is a fundamental combinatorial problem with applications to computational biology. For example, to compare two DNA, RNA or amino acid sequences the standard approach is to align two sequences to detect similar regions and compute the optimal alignment. However, the optimal alignment depends on the relative costs or weights used for specific substitutions, insertions/deletions, or *gaps* (consecutive deletions) in the sequences. Given a set of weights, the optimal alignment computation is typically a simple dynamic program. Our goal is to learn the weights, such that the alignment produced by the dynamic program has application-specific desirable properties.

Problem setup. Given a pair of sequences s_1, s_2 over some alphabet Σ of lengths $m = |s_1|$ and $n = |s_2|$, and a 'space' character $- \notin \Sigma$, a space-extension t of a sequence s over Σ is a sequence over $\Sigma \cup \{-\}$ such that removing all occurrences of $-$ in t gives s. A global alignment (or simply alignment) of s_1, s_2 is a pair of sequences t_1, t_2 such that $|t_1| = |t_2|$, t_1, t_2 are space-extensions of s_1, s_2 respectively, and for no $1 \leq i \leq |t_1|$ we have $t_1[i] = t_2[i] = -$. Let $s[i]$ denote the i-th character of a sequence s and $s[:i]$ denote the first i characters of sequence s. For $1 \leq i \leq |t_1|$, if $t_1[i] = t_2[i]$ we call it a *match*. If $t_1[i] \neq t_2[i]$, and one of $t_1[i]$ or $t_2[i]$

is the character $-$ we call it a *space*, else it is a *mismatch*. A sequence of $-$ characters (in t_1 or t_2) is called a *gap*. Matches, mismatches, gaps and spaces are commonly used *features* of an alignment, i.e. functions that map sequences and their alignments (s_1, s_2, t_1, t_2) to $\mathbf{Z}_{\geq 0}$ (for example, the number of spaces). A common measure of *cost* of an alignment is some linear combination of features. For example if there are d features given by $l_k(\cdot)$, $k \in [d]$, the cost may be given by $c(s_1, s_2, t_1, t_2, \rho) = \sum_{k=1}^{d} \rho_k l_k(s_1, s_2, t_1, t_2)$ where $\rho = (\rho_1, \ldots, \rho_d)$ are the parameters that govern the relative weight of the features. Let $\tau(s, s', \rho) = \operatorname{argmin}_{t_1,t_2} c(s, s', t_1, t_2, \rho)$ and $C(s, s', \rho) = \min_{t_1,t_2} c(s, s', t_1, t_2, \rho)$ denote the optimal alignment and its cost respectively.

A general DP update rule. For a fixed ρ, suppose the sequence alignment problem can be solved, i.e. we can find the alignment with the smallest cost, using a dynamic program A_ρ with linear parameter dependence (described below). Our main application will be to the family of dynamic programs A_ρ which compute the optimal alignment $\tau(s_1, s_2, \rho)$ given any pair of sequences $(s_1, s_2) \in \Sigma^m \times \Sigma^n = \Pi$ for any $\rho \in \mathbb{R}^d$, but we will proceed to provide a more general abstraction. See Section 9.5.1 for example DPs using well-known features in computational biology, expressed using the abstraction below. For any problem $(s_1, s_2) \in \Pi$, the dynamic program A_ρ ($\rho \in \mathcal{P} \subseteq \mathbb{R}^d$, the set of parameters) solves a set $\pi(s_1, s_2) = \{P_i \mid i \in [k], P_k = (s_1, s_2)\}$ of k subproblems (typically, $\pi(s_1, s_2) \subseteq \Pi_{s_1,s_2} = \{(s_1[:i'], s_2[:j']) \mid i' \in \{0, \ldots, m\}, j' \in \{0, \ldots, n\}\} \subseteq \Pi$) in some fixed order $P_1, \ldots, P_k = (s_1, s_2)$. Crucially, the subproblems sequence $P_1, \ldots, P_k$ do not depend on ρ[6]. In particular, a problem P_j can be efficiently solved given optimal alignments and their costs $(\tau_i(\rho), C_i(\rho))$ for problems P_i for each $i \in [j-1]$. Some initial problems in the sequence $P_1, \ldots, P_k$ of subproblems are *base case* subproblems where the optimal alignment and its cost can be directly computed without referring to a previous subproblem. To solve a (non base case) subproblem P_j, we consider V alternative *cases* $q : \Pi \to [V]$, i.e. P_j belongs to exactly one of the V cases (e.g. if $P_j = (s_1[:i'], s_2[:j'])$, we could have two cases corresponding to $s_1[i'] = s_2[j']$ and $s_1[i'] \neq s_2[j']$). Typically, V will be a small constant. For any case $v = q(P_j) \in [V]$ that P_j may belong to, the cost of the optimal alignment of P_j is given by a minimum over L_v terms of the form $c_{v,l}(\rho, P_j) = \rho \cdot w_{v,l} + \sigma_{v,l}(\rho, P_j)$, where $l \in [L_v]$, $w_{v,l} \in \mathbb{R}^d$, $\sigma_{v,l}(\rho, P_j) = C_t(\rho) \in \{C_1(\rho), \ldots, C_{j-1}(\rho)\}$ is the cost of some previously solved subproblem $P_t = (s_1[:i'_t], s_2[:j'_t]) = (s_1[:i'_{v,l,j}], s_2[:j'_{v,l,j}])$ (i.e. t depends on v, l, j but not on ρ), and $c_{v,l}(\rho, P_j)$ is the cost of alignment $\tau_{v,l}(\rho, P_j) = T_{v,l}(\tau_t(\rho))$ which extends the optimal alignment for subproblem P_t by a ρ-independent transformation $T_{v,l}(\cdot)$. That is, the DP update for computing the cost of the optimal alignment takes the form

$$DP(\rho, P_j) = \min_l \{\rho \cdot w_{q(P_j),l} + \sigma_{q(P_j),l}(\rho, P_j)\}, \tag{9.2}$$

and the optimal alignment is given by $DP'(\rho, P_j) = \tau_{q(P_j),l^*}(\rho, P_j)$, where $l^* = \operatorname{argmin}_l \{\rho \cdot w_{q(P_j),l} + \sigma_{q(P_j),l}(\rho, P_j)\}$. The DP specification is completed by including base cases $\{C(s, s', \rho) = \rho \cdot w_{s,s'} \mid (s, s') \in \mathcal{B}(s_1, s_2)\}$ (or $\{\tau(s, s', \rho) = \tau_{s,s'} \mid (s, s') \in \mathcal{B}(s_1, s_2)\}$ for the optimal alignment DP) corresponding to a set of base case subproblems $\mathcal{B}(s_1, s_2) \subseteq \Pi_{s_1,s_2}$. Let $L = \max_{v \in [V]} L_v$ denote the maximum number of subproblems needed to compute a single DP

[6]For the sequence alignment DP in Appendix 9.5.1, we have $\pi(s_1, s_2) = \Pi_{s_1,s_2}$ and we first solve the base case subproblems (which have a fixed optimal alignment for all ρ) followed by problems $(s_1[:i'], s_2[:j'])$ in a non-decreasing order of $i' + j'$ for any value of ρ.

Algorithm 19 COMPUTECOMPACTEXECUTIONDAG

Input: Execution DAG $G_e = (V_e, E_e)$, problem instance (s_1, s_2)
$\mathcal{P}_0 \leftarrow \mathcal{P}$
$v_1, \ldots, v_n \leftarrow$ topological ordering of vertices V_e
for $i = 1$ to n **do**
  Let S_i be the set of nodes with incoming edges to v_i
  For $v_s \in S_i$, let $\mathcal{P}_s$ denote the partition corresponding to v_s
  $\mathcal{P}_i \leftarrow$ COMPUTEOVERLAYDP$(\{\mathcal{P}_s \mid s \in S_i\})$
  for each $p \in \mathcal{P}_i$ **do**
    $p' \leftarrow$ COMPUTESUBDIVISIONDP$(p, (s_1, s_2))$
    $\mathcal{P}_i \leftarrow \mathcal{P}_i \setminus \{p\} \cup p'$
  $\mathcal{P}_i \leftarrow$ RESOLVEDEGENERACIESDP$(\mathcal{P}_i)$
return Partition $\mathcal{P}_n$

update in any of the cases. L is often small, typically 2 or 3. Our main result is to provide an algorithm for computing the polytopic pieces of the dual class functions efficiently for small constants d and L. We provide a summary of the key steps and our main result. The main idea is to use an Execution DAG which compactly represents the partition of the parameter space corresponding to the pieces of the dual class loss (Algorithm 19), and involves three subroutines. We give intuitive overviews of the main subroutines in Algorithm 19.

- COMPUTEOVERLAYDP computes an overlay $\mathcal{P}_i$ of the input polytopic subdivisions $\{\mathcal{P}_s \mid s \in S_i\}$ and uses Clarkson's algorithm for intersecting polytopes with output-sensitive efficiency. The overlay can be computed by solving at most $\tilde{R}^L$ linear programs.
- COMPUTESUBDIVISIONDP applies Algorithm 15, in each piece of the overlay we need to find the polytopic subdivision induced by $O(L^2)$ hyperplanes (specific to the piece). This works because all relevant subproblems have the same fixed solution within any piece of the overlay.
- Finally RESOLVEDEGENERACIESDP merges pieces where the optimal alignment is identical using a simple search over the resulting subdivision.

For our implementation of the subroutines, we have the following guarantee for Algorithm 19.

Theorem 9.5.1. *Let $R_{i,j}$ denote the number of pieces in $P[i][j]$, and $\tilde{R} = \max_{i \le m, j \le n} R_{i,j}$. If the time complexity for computing the optimal alignment is $O(T_{\text{DP}})$, then Algorithm 19 can be used to compute the pieces for the dual class function for any problem instance (s_1, s_2), in time $O(d! L^{4d} \tilde{R}^{2L+1} T_{\text{DP}})$.*

For the special case of $d = 2$, we show that the pieces may be computed in $O(RT_{\text{DP}})$ time using the ray search technique of [118]. We conclude with a remark on the size of R.

Remark 7. *For the sequence alignment problem, the upper bounds on R depend on the nature of cost functions involved. In practice, R is usually much smaller than the worst case bounds, making our results even stronger. In prior work on computational biology data (sequence alignment with $d = 2$, sequence length ~100), [87] observe that of a possible more than 2^{200} align-*

ments, only $R = 120$ appear as possible optimal sequence alignments (over $\mathbb{R}^2$) for a pair of immunoglobin sequences.

9.5.1 Example dynamic programs for sequence alignment

We exhibit how two well-known sequence alignment formulations can be solved using dynamic programs which fit our model in Section 9.5. In Section 9.5.1 we show a DP with two free parameters ($d = 2$), and in Section 9.5.1 we show another DP which has three free parameters ($d = 3$).

Mismatches and spaces

Suppose we only have two features, *mismatches* and *spaces*. The alignment that minimizes the cost c may be obtained using a dynamic program in $O(mn)$ time. The dynamic program is given by the following recurrence relation for the cost function which holds for any $i, j > 0$, and for any $\rho = (\rho_1, \rho_2)$,

$$C(s_1[:i], s_2[:j], \rho) = \begin{cases} C(s_1[:i-1], s_2[:j-1], \rho) & \text{if } s_1[i] = s_2[j], \\ \min \big\{ \rho_1 + C(s_1[:i-1], s_2[:j-1], \rho), & \\ \quad \rho_2 + C(s_1[:i-1], s_2[:j], \rho), & \text{if } s_1[i] \neq s_2[j]. \\ \quad \rho_2 + C(s_1[:i], s_2[:j-1], \rho) \quad \big\} & \end{cases}$$

The base cases are $C(\phi, \phi, \rho) = 0, C(\phi, s_2[:j], \rho) = j\rho_2, = C(s_1[:i], \phi, \rho) = i\rho_2$ for $i, j \in [m] \times [n]$. Here ϕ denotes the empty sequence. One can write down a similar recurrence for computing the optimal alignment $\tau(s_1, s_2, \rho)$.

We can solve the non base-case subproblems $(s_1[:i], s_2[:j])$ in any non-decreasing order of $i + j$. Note that the number of cases $V = 2$, and the maximum number of subproblems needed to compute a single DP update $L = 3$ ($L_1 = 1, L_2 = 3$). For a non base-case problem (i.e. $i, j > 0$) the cases are given by $q(s_1[:i], s_2[:j]) = 1$ if $s_1[i] = s_2[j]$, and $q(s_1[:i], s_2[:j]) = 2$ otherwise. The DP update in each case is a minimum of terms of the form $c_{v,l}(\rho, (s_1[:i], s_2[:j])) = \rho \cdot w_{v,l} + \sigma_{v,l}(\rho, (s_1[:i], s_2[:j]))$. For example if $q(s_1[:i], s_2[:j]) = 2$, we have $w_{2,1} = \langle 1, 0 \rangle$ and $\sigma_{2,1}(\rho, (s_1[:i], s_2[:j]))$ equals $C(s_1[:i-1], s_2[:j-1], \rho)$, i.e. the solution of previously solved subproblem $(s_1[:i-1], s_2[:j-1])$, the index of this subproblem depends on l, v and index of $(s_1[:i], s_2[:j])$ but not on ρ itself.

Mismatches, spaces and gaps

Suppose we have three features, *mismatches*, *spaces* and *gaps*. Typically gaps (consecutive spaces) are penalized in addition to spaces in this model, i.e. the cost of a sequence of three consecutive gaps in an alignment $(\ldots a - - - b \ldots, \ldots a' \; p \; q \; r \; b' \ldots)$ would be $3\rho_2 + \rho_3$ where ρ_2, ρ_3 are costs for *spaces* and *gaps* respectively [100]. The alignment that minimizes the cost c may again be obtained using a dynamic program in $O(mn)$ time. We will need a slight extension of our DP model from Section 9.5 to capture this. We have three subproblems corresponding to any problem in Π_{s_1,s_2} (as opposed to exactly one subproblem, which was sufficient for the

example in 9.5.1). We have a set of subproblems $\pi(s_1, s_2)$ with $|\pi(s_1, s_2)| \leq 3|\Pi_{s_1,s_2}|$ for which our model is applicable. For each $(s_1[:i], s_2[:j])$ we can compute the three costs (for any fixed ρ)

- $C_s(s_1[:i], s_2[:j], \rho)$ is the cost of optimal alignment that ends with substitution of $s_1[i]$ with $s_2[j]$.
- $C_i(s_1[:i], s_2[:j], \rho)$ is the cost of optimal alignment that ends with insertion of $s_2[j]$.
- $C_d(s_1[:i], s_2[:j], \rho)$ is the cost of optimal alignment that ends with deletion of $s_1[i]$.

The cost of the overall optimal alignment is simply $C(s_1[:i], s_2[:j], \rho) = \min\{C_s(s_1[:i], s_2[:j], \rho), C_i(s_1[:i], s_2[:j], \rho), C_d(s_1[:i], s_2[:j], \rho)\}$.
The dynamic program is given by the following recurrence relation for the cost function which holds for any $i, j > 0$, and for any $\rho = (\rho_1, \rho_2, \rho_3)$,

$$C_s(s_1[:i], s_2[:j], \rho) = \min \begin{cases} \rho_1 + C_s(s_1[:i-1], s_2[:j-1], \rho), \\ \rho_1 + C_i(s_1[:i-1], s_2[:j-1], \rho), \\ \rho_1 + C_d(s_1[:i-1], s_2[:j-1], \rho) \end{cases}$$

$$C_i(s_1[:i], s_2[:j], \rho) = \min \begin{cases} \rho_2 + \rho_3 + C_s(s_1[:i], s_2[:j-1], \rho), \\ \rho_2 + C_i(s_1[:i], s_2[:j-1], \rho), \\ \rho_2 + \rho_3 + C_d(s_1[:i], s_2[:j-1], \rho) \end{cases}$$

$$C_d(s_1[:i], s_2[:j], \rho) = \min \begin{cases} \rho_2 + \rho_3 + C_s(s_1[:i-1], s_2[:j], \rho), \\ \rho_2 + \rho_3 + C_i(s_1[:i-1], s_2[:j], \rho), \\ \rho_2 + C_d(s_1[:i-1], s_2[:j], \rho) \end{cases}$$

By having three subproblems for each $(s_1[:i], s_2[:j])$ and ordering the non base-case problems again in non-decreasing order of $i + j$, the DP updates again fit our model (9.2).

9.5.2 Piecewise dual structure

The following results closely follow and extend the corresponding results from [23]. Specifically, we generalize to the case of two sequences of unequal length, and provide sharper bounds on the number of distinct alignments and boundary functions in the piecewise decomposition (even in the special case of equal lengths). We first have a bound on the total number of distinct alignments.

Lemma 9.5.2. *For a fixed pair of sequences $s_1, s_2 \in \Sigma^m \times \Sigma^n$, with $m \leq n$, there are at most $m(m+n)^m$ distinct alignments.*

Proof. For any alignment (t_1, t_2), by definition, we have $|t_1| = |t_2|$ and for all $i \in [|t_1|]$, if $t_1[i] = -$, then $t_2[i] \neq -$ and vice versa. This implies that t_1 has exactly $n - m$ more gaps than t_2. To prove the upper bound, we count the number of alignments (t_1, t_2) where t_2 has exactly i gaps for $i \in [m]$. There are $\binom{n+i}{i}$ choices for placing the gap in t_2. Given a fixed

t_2 with i gaps, there are $\binom{n}{n-m+i}$ choices for placing the gap in t_1. Thus, there are at most $\binom{n+i}{i}\binom{n}{n-m+i} = \frac{(n+i)!}{i!(m-i)!(n-m+i)!} \leq (m+n)^m$ possibilities since $i \leq m$. Summing over all i, we have at most $m(m+n)^m$ alignments of s_1, s_2. □

This implies that the dual class functions are piecewise-structured in the following sense.

Lemma 9.5.3. *Let $\mathcal{U}$ be the set of functions $\{u_\rho : (s_1, s_2) \mapsto u(s_1, s_2, \rho) \mid \rho \in \mathbb{R}^d\}$ that map sequence pairs $s_1, s_2 \in \Sigma^m \times \Sigma^n$ to $\mathbb{R}$ by computing the optimal alignment cost $C(s_1, s_2, \rho)$ for a set of features $(l_i(\cdot))_{i\in[d]}$. The dual class $\mathcal{U}^*$ is $(\mathcal{F}, \mathcal{G}, m^2(m+n)^{2m})$-piecewise decomposable, where $\mathcal{F} = \{f_c : \mathcal{U} \to \mathbb{R} \mid c \in \mathbb{R}\}$ consists of constant functions $f_c : u_\rho \mapsto c$ and $\mathcal{G} = \{g_w : \mathcal{U} \to \{0, 1\} \mid w \in \mathbb{R}^d\}$ consists of halfspace indicator functions $g_w : u_\rho \mapsto \mathbb{I}\{w \cdot \rho < 0\}$.*

Proof. Fix a pair of sequences s_1 and s_2. Let τ be the set of optimal alignments as we range over all parameter vectors $\rho \in \mathbb{R}^d$. By Lemma 9.5.2, we have $|\tau| \leq m(m+n)^m$. For any alignment $(t_1, t_2) \in \tau$, the algorithm A_ρ will return (t_1, t_2) if and only if

$$\sum_{i=1}^{d} \rho_i l_i(s_1, s_2, t_1, t_2) > \sum_{i=1}^{d} \rho_i l_i(s_1, s_2, t_1', t_2')$$

for all $(t_1', t_2') \in \tau \setminus \{(t_1, t_2)\}$. Therefore, there is a set H of at most $\binom{|\tau|}{2} \leq m^2(m+n)^{2m}$ hyperplanes such that across all parameter vectors ρ in a single connected component of $\mathbb{R}^d \setminus H$, the output of the algorithm A_ρ on (s_1, s_2) is fixed. This means that for any connected component R of $\mathbb{R}^d \setminus H$, there exists a real value c_R such that $u_\rho(s_1, s_2) = c_R$ for all $\rho \in \mathbb{R}^d$. By definition of the dual, $u^*_{s_1,s_2}(u_\rho) = u_\rho(s_1, s_2) = c_R$. For each hyperplane $h \in H$, let $g^{(h)} \in \mathcal{G}$ denote the corresponding halfspace. Order these $k = \binom{|\tau|}{2}$ functions arbitrarily as $g_1, \ldots, g_k$. For a given connected component R of $\mathbb{R}^d \setminus H$, let $\mathbf{b}_R \in \{0, 1\}^k$ be the corresponding sign pattern. Define the function $f^{(\mathbf{b}_R)} = f_{c_R}$ and for $\mathbf{b}$ not corresponding to any R, $f^{(\mathbf{b})} = f_0$. Thus, for each $\rho \in \mathbb{R}^d$,

$$u^*_{s_1,s_2}(u_\rho) = \sum_{\mathbf{b}\in\{0,1\}^k} \mathbb{I}\{g_i(u_\rho) = b_i \forall i \in [k]\} f^{(\mathbf{b})}(u_\rho).$$

□

For the special case of $d = 2$, we have an algorithm which runs in time $O(RT_{\text{DP}})$, where R is the number of pieces in $P[m][n]$ which improves on the prior result $O(R^2 + RT_{\text{DP}})$ for two-parameter sequence alignment problems. The algorithm employs the ray search technique of [118] (also employed by [88] but for more general sequence alignment problems) and enjoys the following runtime guarantee.

Theorem 9.5.4. *For the global sequence alignment problem with $d = 2$, for any problem instance (s_1, s_2), there is an algorithm to compute the pieces for the dual class function in $O(RT_{\text{DP}})$ time, where T_{DP} is the time complexity of computing the optimal alignment for a fixed parameter $\rho \in \mathbb{R}^2$, and R is the number of pieces of $u_{(s_1,s_2)}(\cdot)$.*

Proof. We note that for any alignment (t_1, t_2), the boundary functions for the piece where (t_1, t_2) is an optimal alignment are straight lines through the origin of the form

$$\rho_1 l_1(s_1, s_2, t_1,t_2) + \rho_2 l_2(s_1, s_2, t_1, t_2) > \rho_1 l_1(s_1, s_2, t'_1, t'_2) + \rho_2 l_2(s_1, s_2, t'_1, t'_2)$$

for some alignment (t'_1, t'_2) different from (t_1, t_2). The intersection of these halfplanes is either the empty set or the region between two straight lines through the origin. The output subdivision therefore only consists of the axes and straight lines through the origin in the positive orthant.

We will present an algorithm using the ray search technique of [118]. The algorithm computes the optimal alignment (t_1, t_2), (t'_1, t'_2) at points $\rho = (0, 1)$ and $\rho = (1, 0)$. If the alignments are identical, we conclude that (t_1, t_2) is the optimal alignment everywhere. Otherwise, we find the optimal alignment (t''_1, t''_2) for the intersection of line L joining $\rho = (0, 1)$ and $\rho = (1, 0)$, with the line L' given by

$$\rho_1 l_1(s_1, s_2, t_1,t_2) + \rho_2 l_2(s_1, s_2, t_1, t_2) > \rho_1 l_1(s_1, s_2, t'_1, t'_2) + \rho_2 l_2(s_1, s_2, t'_1, t'_2)$$

If $(t''_1, t''_2) = (t_1, t_2)$ or $(t''_1, t''_2) = (t'_1, t'_2)$, we have exactly 2 optimal alignments and the piece boundaries are given by L' and the axes. Otherwise we repeat the above process for alignment pairs $(t''_1, t''_2), (t'_1, t'_2)$ and $(t''_1, t''_2), (t_1, t_2)$. Notice we need to compute at most $R + 1$ dynamic programs to compute all the pieces, giving the desired time bound. □

9.5.3 Details and Analysis of the Algorithm

We start with an overview of the techniques.

Overview of techniques. [88] provide a solution for computing the pieces for sequence alignment with only two free parameters, i.e. $d = 2$, using a ray-search based approach in $O(R^2 + RT_{\text{DP}})$ time, where R is the number of pieces in $u_{(s_1,s_2)}(\cdot)$. [99] solve the simpler problem of finding a point in the polytope (if one exists) in the parameter space where a given alignment (t_1, t_2) is optimal by designing an efficient separation oracle for a linear program with one constraint for every alignment $(t'_1, t'_2) \neq (t_1, t_2)$, $c(s_1, s_2, t_1, t_2, \rho) \leq c(s_1, s_2, t'_1, t'_2, \rho)$, where $\rho \in \mathbb{R}^d$ for general d. We provide a first (to our knowledge) algorithm for the global pairwise sequence alignment problem with general d for computing the full polytopic decomposition of the parameter space with fixed optimal alignments in each polytope. We adapt the idea of execution tree to the sequence alignment problem by defining an execution DAG of decompositions for prefix sequences $(s[: i], s[: j])$. We overlap the decompositions corresponding to the subproblems in the DP update and sub-divide each piece in the overlap using fixed costs of all the subproblems within each piece. The number of pieces in the overlap are output polynomial (for constant d and L) and the sub-division of each piece involves at most L^2 hyperplanes. By memoizing decompositions and optimal alignments for the subproblems, we obtain our runtime guarantee for constant d, L. For $d = 2$, we get a running time of $O(RT_{\text{DP}})$.

As indicated above, we consider the family of dynamic programs A_ρ which compute the optimal alignment $\tau(s_1, s_2, \rho)$ given any pair of sequences $(s_1, s_2) \in \Sigma^m \times \Sigma^n = \Pi$ for any $\rho \in \mathbb{R}^d$. For any alignment (t_1, t_2), the algorithm has a fixed real-valued utility (different from the cost function above) which captures the quality of the alignment, i.e. the utility function $u((s_1, s_2), \rho)$ only depends on the alignment $\tau(s_1, s_2, \rho)$. The dual class function is piecewise constant with

convex polytopic pieces (Lemma 9.5.3). For any fixed problem (s_1, s_2), the space of parameters ρ can be partitioned into R convex polytopic regions where the optimal alignment is fixed. The optimal parameter can then be found by simply comparing the costs of the alignments in each of these pieces. For the rest of this section we consider the algorithmic problem of computing these R pieces efficiently.

For the clustering algorithm family, as we have seen in Section 9.4, we get a refinement of the parameter space with each new step (merge) performed by the algorithm. This does not hold for the sequence alignment problem. Instead we obtain the following DAG, from which the desired pieces can be obtained by looking at nodes with no out-edges (call these *terminal* nodes). Intuitively, the DAG is built by iteratively adding nodes corresponding to subproblems $P_1, \ldots, P_k$ and adding edges directed towards P_j from all subproblems that appear in the DP update for it. That is, for *base case* subproblems, we have singleton nodes with no incoming edges. Using the recurrence relation (9.2), we note that the optimal alignment for the pair of sequences $(s_1[:i], s_2[:j])$ can be obtained from the optimal alignments for subproblems $\{(s_1[:i_{v',l}], s_2[:j_{v',l}])\}_{l\in[L_{v'}]}$ where $v' = q(s_1[:i], s_2[:j])$. The DAG for $(s_1[:i], s_2[:j])$ is therefore simply obtained by using the DAGs $G_{v',l}$ for the subproblems and adding directed edges from the terminal nodes of $G_{v',l}$ to new nodes $v_{p,i,j}$ corresponding to each piece p of the partition $P[i][j]$ of $\mathcal{P}$ given by the set of pieces of $u_{(s_1[i],s_2[j])}(\rho)$. A more compact representation of the execution graph would have only a single node $v_{i,j}$ for each subproblem $(s_1[:i], s_2[:j])$ (the node stores the corresponding partition $P[i][j]$) and edges directed towards $v_{i,j}$ from nodes of subproblems used to solve $(s_1[:i], s_2[:j])$. Note that the graph depends on the problem instance (s_1, s_2) as the relevant DP cases $v' = q(s_1[:i], s_2[:j])$ depend on the sequences s_1, s_2. A naive way to encode the execution graph would be an exponentially large tree corresponding to the recursion tree of the recurrence relation (9.2).

Execution DAG. Formally we define a *compact execution graph* $G_e = (V_e, E_e)$ as follows. For the base cases, we have nodes labeled by $(s, s') \in \mathcal{B}(s_1, s_2)$ storing the base case solutions $(w_{s,s'}, \tau_{s,s'})$ over the unpartitioned parameter space $\mathcal{P} = \mathbb{R}^d$. For $i, j > 0$, we have a node $v_{i,j}$ labeled by $(s_1[:i], s_2[:j])$ and the corresponding partition $P[i][j]$ of the parameter space, with incoming edges from nodes of the relevant subproblems $\{(s_1[:i_{v',l}], s_2[:j_{v',l}])\}_{l\in[L_{v'}]}$ where $v' = q(s_1[:i], s_2[:j])$. This graph is a DAG since every directed edge is from some node $v_{i,j}$ to a node $v_{i',j'}$ with $i' + j' > i + j$ in typical sequence alignment dynamic programs (Appendix 9.5.1); an example showing a few nodes of the DAG is depicted in Figure 9.3. Algorithm 19 gives a procedure to compute the partition of the parameter space for any given problem instance (s_1, s_2) using the compact execution DAG.

We now present some well-known terminology from computational geometry.

Definition 34. *A (convex) subdivision S of $P \subseteq \mathbb{R}^d$ is a finite set of disjoint d-dimensional (convex) sets (called cells) whose union is P. The overlay S of subdivisions $S_1, \ldots, S_n$ is defined as all nonempty sets of the form $\bigcap_{i\in[n]} s_i$ with $s_i \in S_i$. With slight abuse of terminology, we will refer to closures of cells also as cells.*

The COMPUTEOVERLAY procedure takes a set of partitions, which are convex polytopic subdivisions of $\mathbb{R}^d$, and computes their overlay. We will represent a convex polytopic subdivision as a list of cells, each represented as a list of bounding hyperplanes. Now to compute the overlay of subdivisions $P_1, \ldots, P_L$, with lists of cells $\mathcal{C}_1, \ldots, \mathcal{C}_L$ respectively, we define $|\mathcal{C}_1| \times \cdots \times |\mathcal{C}_L|$ sets of hyperplanes $H_{j_1,\ldots,j_L} = \{\bigcup_{l\in[L]} \mathcal{H}(c_{j_l}^{(l)})\}$, where $c_{j_l}^{(l)}$ is the j_l-th cell of P_l and $\mathcal{H}(c)$ denotes the hy-

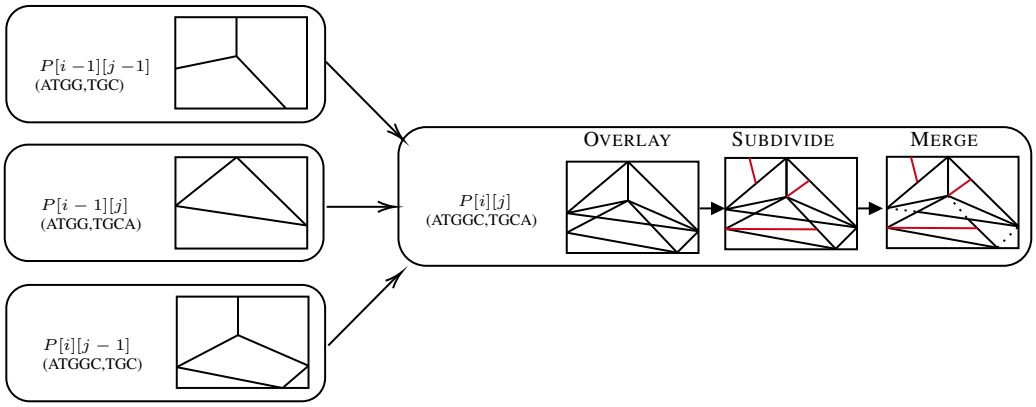


Figure 9.3: Incoming nodes used for computing the pieces at the node $P[i][j]$ in the execution DAG.

perplanes bounding cell c. We compute the cells of the overlay by applying Clarkson's algorithm [62] to each $H_{j_1,\ldots,j_L}$. We have the following guarantee about the running time of Algorithm 20.

Algorithm 20 COMPUTEOVERLAYDP

Input: Convex polytopic subdivisions $P_1, \ldots, P_L$ of $\mathbb{R}^d$, represented as lists $\mathcal{C}_j$ of hyperplanes for each cell in the subdivision
$\mathcal{H}(c_{j_l}^{(l)}) \leftarrow$ hyperplanes bounding j_l-th cell of P_l for $l \in [L], j_l \in \mathcal{C}_l$
for each $j_1, \ldots, j_L \in |\mathcal{C}_1|, \ldots, |\mathcal{C}_L|$ **do**
  $H_{j_1,\ldots,j_L} \leftarrow \{\bigcup_{l\in[L]} \mathcal{H}(c_{j_l}^{(l)})\}$
  $H'_{j_1,\ldots,j_L} \leftarrow$ CLARKSON$(H_{j_1,\ldots,j_L})$
$\mathcal{C} \leftarrow$ non-empty lists of hyperplanes in $H'_{j_1,\ldots,j_L}$ for $j_l \in \mathcal{C}_l$
return Partition represented by $\mathcal{C}$

Lemma 9.5.5. *Let $R_{i,j}$ denote the number of pieces in $P[i][j]$, and $\tilde{R} = \max_{i\le m, j\le n} P[i][j]$. There is an implementation of the* COMPUTEOVERLAYDP *routine in Algorithm 19 which computes the overlay of L convex polytopic subdivisions in time $O(L\tilde{R}^{L+1} \cdot \mathrm{LP}(d, \tilde{R}^L + 1))$, which is $O(d!L\tilde{R}^{2L+1})$ using algorithms for solving low-dimensional LPs [53].*

Proof. Consider Algorithm 20. We apply the Clarkson's algorithm at most $\tilde{R}^L$ times, once corresponding to each L-tuple of cells from the L subdivisions. Each iteration corresponding to cell c in the output overlay $\mathcal{O}$ (corresponding to $\mathcal{C}$) has a set of at most $L\tilde{R}$ hyperplanes and yields at most R_c non-redundant hyperplanes. By Theorem 9.2.1, each iteration takes time $O(L\tilde{R} \cdot \mathrm{LP}(d, R_c + 1))$, where $\mathrm{LP}(d, R_c + 1)$ is bounded by $O(d!R_c)$ for the algorithm of [53]. Note that $\sum_c R_c$ corresponds to the total number of edges in the cell adjacency graph of $\mathcal{O}$, which is bounded by $\tilde{R}^{2L}$. Further note that $R_c \le \tilde{R}^L$ for each $c \in \mathcal{C}$ and $|\mathcal{C}| \le \tilde{R}^L$ to get a runtime bound of $O(L\tilde{R}^{L+1} \cdot \mathrm{LP}(d, \tilde{R}^L + 1))$. □

We now consider an implementation for the COMPUTESUBDIVISIONDP subroutine. The algorithm computes the hyperplanes across which the subproblem used for computing the optimal

Algorithm 21 COMPUTESUBDIVISIONDP

Input: Convex Polytope P, problem instance (s_1, s_2)
$v \leftarrow$ the DP *case* $q((s_1, s_2))$ for the problem instance
$\rho_0 \leftarrow$ an arbitrary point in P
$(t_1^0, t_2^0) \leftarrow$ optimal alignment of (s_1, s_2) for parameter ρ_0, using subproblem $(s_1[: i_{v,l_0}], s_2[: j_{v,l_0}])$ for some $l_0 \in [L_v]$
`mark` $\leftarrow \emptyset$, `polys` $\leftarrow$ new hashtable, `poly_queue` $\leftarrow$ new queue
`poly_queue.enqueue`(l_0)
while `poly_queue.non_empty()` **do**
  $l \leftarrow$ `poly_queue.dequeue()` Continue to next iteration if $l \in$ `mark`
  `mark` $\leftarrow$ `mark` $\cup \{l\}$
  $\mathcal{L} \leftarrow P$
  for all subproblems $(s_1[: i_{v,l_1}], s_2[: j_{v,l_1}])$ for $l_1 \in [L_v], l_1 \neq l$ **do**
    Add the half-space inequality $b^T\rho \leq c$ corresponding to $c_{v,l}(\rho, (s_1, s_2)) \leq c_{v,l_1}(\rho, (s_1, s_2))$ to $\mathcal{L}$
  $I \leftarrow$ CLARKSON$(\mathcal{L})$
  `poly_queue.enqueue`(l') for each l' such that the constraint labeled by it is in I
  `polys`$[l] \leftarrow \{\mathcal{L}[\ell] : \ell \in I\}$
return `polys`

alignment changes in the recurrence relation (9.2) by adapting Algorithm 15. We restate the algorithm in the context of sequence alignment as Algorithm 21.

Lemma 9.5.6. *Let $R_{i,j}$ denote the number of pieces in $P[i][j]$, and $\tilde{R} = \max_{i \leq m, j \leq n} R_{i,j}$. There is an implementation of* COMPUTESUBDIVISIONDP *routine in Algorithm 19 with running time at most $O((L^{2d+2} + L^{2d}\tilde{R}^L) \cdot \mathrm{LP}(d, L^2 + \tilde{R}^L))$ for each outer loop of Algorithm 19. If the algorithm of [53] is used to solve the LP, this is at most $O(d! L^{2d+2} \tilde{R}^{2L})$.*

Proof. Consider Algorithm 21. For any piece p in the overlay, all the required subproblems have a fixed optimal alignment, and we can find the subdivision of the piece by adapting Algorithm 15 (using $O(L^2 + \tilde{R}^L)$ hyperplanes corresponding to subproblems and piece boundaries). The number of pieces in the subdivision is at most L^{2d} since we have at most L^2 hyperplanes intersecting the piece, so we need $O(L^{2d+2} + L^{2d}\tilde{R}^L)$ time to list all the pieces $\mathcal{C}_p$. The time needed to run Clarkson's algorithm is upper bounded by $O(\sum_{c \in \mathcal{C}_p}(L^2 + \tilde{R}^L) \cdot \mathrm{LP}(d, R_c + 1)) = O(\sum_{c \in \mathcal{C}_p}(L^2 + \tilde{R}^L) \cdot \mathrm{LP}(d, L^2 + \tilde{R}^L)) = O((L^{2d+2} + L^{2d}\tilde{R}^L) \cdot \mathrm{LP}(d, L^2 + \tilde{R}^L))$. Using [53] to solve the LP, this is at most $O(d! \tilde{R}^{2L} L^{2d+4})$. □

Lemma 9.5.7. *Let $R_{i,j}$ denote the number of pieces in $P[i][j]$, and $\tilde{R} = \max_{i \leq m, j \leq n} R_{i,j}$. There is an implementation of* RESOLVEDEGENERACIESDP *routine in Algorithm 19 with running time at most $O(\tilde{R}^{2L} L^{4d})$ for each outer loop of Algorithm 19.*

Proof. The RESOLVEDEGENERACIESDP is computed by a simple BFS over the cell adjacency graph $G_c = (V_c, E_c)$ (i.e. the graph with polytopic cells as nodes and edges between polytopes sharing facets). We need to find (maximal) components of a subgraph of the cell adjacency graph

where each node in the same component has the same optimal alignment. This is achieved by a simple BFS in $O(|V_c|+|E_c|)$ time. Indeed, by labeling each polytope with the corresponding optimal alignment, we can compute the components of the subgraph of G_c with edges restricted to nodes joining the same optimal alignment. Note that the resulting polytopic subdivision after the merge is still a convex subdivision using arguments in Lemma 9.5.3, but applied to appropriate sequence alignment subproblem. As noted in the proof of Lemma 9.5.6, we have $|V_c| \leq L^{2d}\tilde{R}^L$ since the number of cells within each piece p is at most L^{2d} and there are at most $\tilde{R}^L$ pieces in the overlay. Since $|E_c| \leq |V_c|^2$, we have an implementation of RESOLVEDEGENERACIESDP in time $O((L^{2d}\tilde{R}^L)^2) = O(\tilde{R}^{2L}L^{4d})$. □

Finally we can put all the above together to give a proof of Theorem 9.5.1.

Proof of Theorem 9.5.1. The proof follows by combining Lemma 9.5.5, Lemma 9.5.6 and Lemma 9.5.7. Note that in the execution DAG, we have $|V_e| \leq |E_e| = O(T_{\text{DP}})$. Further, we invoke COMPUTEOVERLAYDP and RESOLVEDEGENERACIESDP $|V_e|$ times across all iterations and COMPUTESUBDIVISIONDP across the $|V_e|$ outer loops. □

The results in this chapter are joint work with Nina Balcan and Chris Seiler [33], and have appeared in AAAI 2024 Learnable Optimization (LEANOPT) workshop.

Chapter 10

Ongoing and future work

The results in this thesis suggest several exciting directions for further research, a couple are noted below.

10.1 Computational efficiency

Most of the prior work on data-driven algorithm design has focused on *sample efficiency* of learning good algorithms from the parameterized families, and a major open question for this line of research is to design *computationally efficient* learning algorithms [41, 85]. Computational efficiency of typical approaches is particularly a bottleneck when learning multiple real parameters. A concrete example follows.

The empirical (sample) loss function for the typical problems studied in data-driven design is discontinuous (even on a single problem instance). One approach to find good parameters on a sample is to find a partition of the parameter space into regions or *pieces* where the loss (as a function of the parameters, on fixed problem instance, also known as *dual class* loss function [23]) behaves well.

In Chapter 2, we use approximation as a key tool to speed up computation. This includes approximating the graph by sparsification, and approximately computing the boundary functions and piece functions for the dual loss function. Future work can explore this direction in the context of other problems. In Chapter 9, we use classic techniques from computational geometry for designing output-sensitive enumeration algorithms which are applicable to several data-driven design problems. A future direction is to extend our approaches beyond piecewise-structure loss functions with linear piece boundaries.

10.2 Robustness

Chapter 4 in the proposal discusses how a small amount of abstention can help improve robustness of some learning algorithms. While relatively novel in the recently popular area of adversarial robustness, the signficance of abstention in learning has been long known [61]. In particular, Rivest and Sloan [144] formulate a model of reliable learning where one must correctly predict or abstain, and provide an algorithm based on Mitchell's candidate elimination approach [124]

with the goal being to maximize usefulness (i.e. how often we predict). This model has been recently extended to a novel notion of robustly-reliable learning under data poisoning attacks [27]. This connection raises potentially interesting questions, besides broader incorporation of reliability in robustness studies.

If we are willing to relax the reliability criterion slightly and allow a trade-off between reliability and usefulness, we could formulate the optimization over the trade-off as a data-driven design problem similar to the one considered in Chapter 4. The formulation is interesting even without the robustness or data-driven lens, and the objective has been recently studied for general hypothesis classes [44, 138]. Can we provide interesting guarantees in the repeated problem setting, i.e. we receive a sequence/collection of related data sets (examples within each data set might not even be i.i.d.)? For linear separators, one natural choice is to abstain within a margin of the classifier and the margin-width can be the tunable parameter. It could be particularly interesting to further understand reliability for widely used concept classes like neural networks. One concrete direction is to try to bound the Hanneke's disagreement coefficient [90] (and apply fundamental results connecting disagreement coefficient with reliability [27, 71, 72]). This would have further implications for deep active learning, and is noted as an open question by Zhu and Nowak [176].

Appendix A

Background from prior work

We include background on learning theory, data-driven algorithm design and other prior work needed for Chapters 2, 3 and 4 here.

A.1 A general tool for analyzing dispersion

If the weights of the graph are given by a polynomial kernel $w(u,v) = (\tilde{d}(u,v) + \tilde{\alpha})^d$, we can apply the general tool developed by Balcan et al. [19] to learn $\tilde{\alpha}$, which we summarize below.

1. Bound the probability density of the random set of discontinuities of the loss functions.
2. Use a VC-dimension based uniform convergence argument to transform this into a bound on the dispersion of the loss functions.

Formally, we have the following theorems from [19], which show how to use this technique when the discontinuities are roots of a random polynomial.

Theorem A.1.1 ([19]). *Consider a random degree d polynomial ϕ with leading coefficient 1 and subsequent coefficients which are real of absolute value at most R, whose joint density is at most κ. There is an absolute constant K depending only on d and R such that every interval I of length $\leq \epsilon$ satisfies Pr(ϕ has a root in I) $\leq \kappa\epsilon/K$.*

Theorem A.1.2 ([19]). *Let $l_1, \ldots, l_T : \mathbb{R} \to \mathbb{R}$ be independent piecewise L-Lipschitz functions, each having at most K discontinuities. $D(T, \epsilon, \rho) = |\{1 \leq t \leq T \mid l_t$ is not L-Lipschitz on $[\rho - \epsilon, \rho + \epsilon]\}|$ is the number of functions that are not L-Lipschitz on the ball $[\rho - \epsilon, \rho + \epsilon]$. Then we have $E[\max_{\rho \in \mathbb{R}} D(T, \epsilon, \rho)] \leq \max_{\rho \in \mathbb{R}} E[D(T, \epsilon, \rho)] + O(\sqrt{T \log(TK)})$.*

A.2 Background for regularized regression

We include additional details from Chapter 3 here.

A.2.1 Background for the structural result

We first present some useful terminology from algebraic geometry.

Definition 35 (Semialgebraic sets, Algebraic curves.)**.** *A semialgebraic subset of $\mathbb{R}^n$ is a finite union of sets of the form $\{x \in \mathbb{R}^n \mid p_i(x) \geq 0$ for each $i \in [m]\}$, where $p_1, \ldots, p_m$ are polynomials. An algebraic curve is the zero set of a polynomial in two dimensions.*

We will next review some properties of LASSO solutions from prior work that are useful in proving our results. Let (X, y) with $X = [\mathbf{x}_1, \ldots, \mathbf{x}_p] \in \mathbb{R}^{m \times p}$ and $y \in \mathbb{R}^m$ denote a (training) dataset consisting of m labeled examples with p features. LASSO regularization may be formulated as the following optimization problem.

$$\min_{\beta \in \mathbb{R}^p} \|y - X\beta\|_2^2 + \lambda_1 ||\beta||_1,$$

where $\lambda_1 \in \mathbb{R}^+$ is the regularization parameter. Dealing with the case $\lambda_1 = 0$ (i.e. Ordinary Least Squares) is not difficult, but is omitted here to keep the statements of the definitions and results simple. We will use the following well-known facts about the solution of the LASSO optimization problem [80, 158]. Applying the Karush-Kuhn-Tucker (KKT) optimality conditions to the problem gives,

Lemma A.2.1 (KKT Optimality Conditions for LASSO)**.** *The optimal solution for LASSO, $\beta^* \in \operatorname{argmin}_{\beta \in \mathbb{R}^p} \|y - X\beta\|_2^2 + \lambda_1 ||\beta||_1$ iff for all $j \in [p]$,*

$$\begin{aligned} \mathbf{x}_j^T(y - X\beta^*) &= \lambda_1 \mathsf{sign}(\beta^*), \text{ if } \beta_j^* \neq 0, \\ |\mathbf{x}_j^T(y - X\beta^*)| &\leq \lambda_1, \text{ otherwise.} \end{aligned}$$

Here $\mathbf{x}_j^T(y - X\beta^*)$ is simply the correlation of the the j-th covariate with the residual $y - X\beta^*$ (when y, X have been standardized). This motivates the definition of *equicorrelation sets* of covariates (Definition 36).

Definition 36 (Equicorrelation sets, [158])**.** *Let $\beta^* \in \operatorname{argmin}_{\beta \in \mathbb{R}^p} \|y - X\beta\|_2^2 + \lambda_1 ||\beta||_1$. The equicorrelation set corresponding to β^*, $\mathcal{E} = \{j \in [p] \mid |\mathbf{x}_j^T(y - X\beta^*)| = \lambda_1\}$, is simply the set of covariates with maximum absolute correlation. We also define the equicorrelation sign vector for β^* as $s = \mathsf{sign}(X_{\mathcal{E}}^T(y - X\beta^*)) \in \{\pm 1\}$.*

In terms of the equicorrelation set and the equicorrelation sign vector, the characterization of the LASSO solution in Lemma A.2.1 implies

$$X_{\mathcal{E}}^T(y - X_{\mathcal{E}}\beta_{\mathcal{E}}^*) = \lambda_1 s.$$

This implies a necessary and sufficient condition for the uniqueness of the LASSO solution, namely that $X_{\mathcal{E}}$ is full rank for all equicorrelation sets $\mathcal{E}$ [158]. Our results will hold if the dataset X satisfies this condition, but for simplicity we will use the a simpler (and possibly more natural) sufficient condition involving the *general position*.

Definition 37. *A matrix $X \in \mathbb{R}^{m\times p}$ is said to have its columns in the general position if the affine span of any $k \leq m$ points $(\sigma_i \mathbf{x}_{j_i})_{i\in[k],\{j_i\}_i=J\subseteq[p]}$ for arbitrary signs $\sigma_{[k]} \in \{-1,1\}^k$ and subset J of the columns of size k, does not contain any element of $\{\mathbf{x}_i \mid i \notin J\}$.*

Finally, we state the following useful characterization of the LASSO solutions in terms of the equicorrelation sets and sign vectors.

Lemma A.2.2 ([158], Lemma 3)**.** *If the columns of X are in general position, then for any y and $\lambda_1 > 0$, the LASSO solution is unique and is given by*

$$\beta^*_{\mathcal{E}} = (X^T_{\mathcal{E}} X_{\mathcal{E}})^{-1}(X^T_{\mathcal{E}} y - \lambda_1 s), \beta^*_{[p]\setminus\mathcal{E}} = 0.$$

We remark that Lemma A.2.2 does not give a way to compute β^* for a given value of λ_1, since $\mathcal{E}$ and s depend on β^*, but still gives a property of β^* that is convenient to use. In particular, since we have at most 3^p possible choices for $(\mathcal{E}, s)$, this implies that the LASSO solution $\beta^*(\lambda_1)$ is a piecewise linear function of λ_1, with at most 3^p pieces (for $\lambda_1 > 0$). Following popular terminology, we will refer to this function as a *solution path* of LASSO for the given dataset (X, y). LARS-LASSO of [177] is an efficient algorithm for computing the *solution path* of LASSO.

Corollary A.2.3. *Let X be a matrix with columns in the general position. If the unique LASSO solution for the dataset (X, y) is given by the function $\beta^* : \mathbb{R}^+ \to \mathbb{R}^p$, then β^* is piecewise linear with at most 3^p pieces given by Lemma A.2.2.*

A.2.2 Background for online learning

At a high level, the plan is to show dispersion using the general recipe developed in [19]. The recipe may be summarized at a high level as follows.

S1. Bound the probability density of the random set of discontinuities of the loss functions. Intuitively this corresponds to computing the average number of loss functions that may be discontinuous along a path connecting any two points within distance ϵ in the domain.

S2. Use a VC-dimension based uniform convergence argument to transform this into a bound on the dispersion of the loss functions.

Formally, we have the following theorems from [19], which show how to use this technique when the discontinuities are roots of a random polynomial with bounded coefficients. The theorems implement steps S1 and S2 of the above recipe respectively.

Theorem A.2.4 ([19])**.** *Consider a random degree d polynomial ϕ with leading coefficient 1 and subsequent coefficients which are real of absolute value at most R, whose joint density is at most κ. There is an absolute constant K_0 depending only on d and R such that every interval I of length $\leq \epsilon$ satisfies Pr(ϕ has a root in I) $\leq \kappa\epsilon/K_0$.*

Theorem A.2.5 ([19])**.** *Let $l_1, \ldots, l_T : \mathbb{R} \to \mathbb{R}$ be independent piecewise L-Lipschitz functions, each having at most K discontinuities. Let $D(T, \epsilon, \rho) = |\{1 \leq t \leq T \mid l_t$ is not L-Lipschitz on $[\rho - \epsilon, \rho + \epsilon]\}|$ be the number of functions that are not L-Lipschitz on the ball $[\rho - \epsilon, \rho + \epsilon]$. Then we have $E[\max_{\rho\in\mathbb{R}} D(T, \epsilon, \rho)] \leq \max_{\rho\in\mathbb{R}} E[D(T, \epsilon, \rho)] + O(\sqrt{T \log(TK)})$.*

Lemma A.2.6. *Let A be an $r \times s$ matrix with R-bounded max-norm, i.e. $||A||_{\infty,\infty} = \max_{i,j} |A_{ij}| \leq R$. Then each entry of the matrix $(A^T A + \lambda I_s)^{-1}$ is a rational polynomial $P_{ij}(\lambda)/Q(\lambda)$ for $i, j \in [s]$ with each P_{ij} of degree at most $s-1$, Q of degree s, and all the coefficients have absolute value at most $r^s(Rs)^{2s}$.*

Proof. Let $G = A^T A$ be the Gram matrix. $|G_{ij}| = |\sum_k A_{ki}A_{kj}| \leq \sum_k |A_{ki}A_{kj}| \leq rR^2$, by the triangle inequality and using $\max_{i,j} |A_{ij}| \leq R$. The determinant $\text{DET}(A^T A + \lambda I_s)$ is a sum of $s! \leq s^s$ signed terms, each a product of s elements of the form G_{ij} or $G_{ii} + \lambda$. Thus, in each of the $s!$ terms, the coefficient of λ^k is a sum of at most $\binom{s}{s-k} \leq s^k \leq s^s$ expressions of the form $\Pi_{(i,j)\in S} G_{ij}$ with $|S| \leq s - k$. Now $|\Pi_{(i,j)\in S} G_{ij}| \leq (rR^2)^{|S|} \leq (rR^2)^s$, and by triangle inequality the coefficient of λ^k is upper bounded by $(rR^2)^s \cdot s^k \cdot s^s$ for any k. This establishes the bound on the coefficients of $Q(\lambda)$. A similar argument implies the upper bound for each $P_{ij}(\lambda)$. □

We will also need the following result, which is a simple extension of Lemma 24 from [10].

Lemma A.2.7. *Suppose X and Y are real-valued random variables taking values in $[m, m+M]$ for some $m, M \in \mathbb{R}^+$ and suppose that their joint distribution is κ-bounded. Let c be an absolute constant. Then,*

(i) $Z = X + Y$ is drawn from a $K_1\kappa$-bounded distribution, where $K_1 \leq M$.
(ii) $Z = XY$ is drawn from a $K_2\kappa$-bounded distribution, where $K_2 \leq M/m$.
(iii) $Z = X - Y$ is drawn from a $K_1\kappa$-bounded distribution, where $K_1 \leq M$.
(iv) $Z = X + c$ has a κ-bounded distribution, and $Z = cX$ has a $\frac{\kappa}{|c|}$-bounded distribution.

Proof. Let $f_{X,Y}(x, y)$ denote the joint density of X, Y. (i) and (ii) are immediate from Lemma 24 from [10], (iii) is a simple extension. Indeed, the cumulative density function for Z is given by

$$\begin{aligned} F_Z(z) &= \Pr(Z \leq z) = \Pr(X - Y \leq z) = \Pr(X \leq z + Y) \\ &= \int_m^{m+M} \int_m^{z+y} f_{X,Y}(x, y) dx dy. \end{aligned}$$

The density function for Z can be obtained using Leibniz's rule as

$$\begin{aligned} f_Z(z) = \frac{d}{dz} F_Z(z) &= \frac{d}{dz} \int_m^{m+M} \int_m^{z+y} f_{X,Y}(x, y) dx dy \\ &= \int_m^{m+M} \left(\frac{d}{dz} \int_m^{y} f_{X,Y}(x, y) dx + \frac{d}{dz} \int_0^{z} f_{X,Y}(t + y, y) dt \right) dy \\ &= \int_m^{m+M} f_{X,Y}(z + y, y) dy \\ &\leq \int_m^{m+M} \kappa dy \\ &= M\kappa. \end{aligned}$$

Finally, (iv) follows from simple change of variable manipulations (e.g. Theorem 22 of [19]). □

A.3 Standard results from learning theory and data-driven algorithm design

The pseudo-dimension is frequently used to analyze the learning theoretic complexity of real-valued function classes. The formal definition is stated here for convenience.

Definition 38 (Shattering and Pseudo-dimension, [4])**.** *Let $\mathcal{F}$ be a set of functions mapping from $\mathcal{X}$ to $\mathbb{R}$, and suppose that $S = \{x_1, \ldots, x_m\} \subseteq \mathcal{X}$. Then S is pseudo-shattered by $\mathcal{F}$ if there are real numbers $r_1, \ldots, r_m$ such that for each $b \in \{0,1\}^m$ there is a function f_b in $\mathcal{F}$ with $\operatorname{sign}(f_b(x_i) - r_i) = b_i$ for $i \in [m]$. We say that $r = (r_1, \ldots, r_m)$ witnesses the shattering. We say that $\mathcal{F}$ has pseudo-dimension d if d is the maximum cardinality of a subset S of $\mathcal{X}$ that is pseudo-shattered by $\mathcal{F}$, denoted $\operatorname{Pdim}(\mathcal{F}) = d$. If no such maximum exists, we say that $\mathcal{F}$ has infinite pseudo-dimension.*

Pseudo-dimension is a real-valued analogue of VC-dimension, and is a classic complexity notion in learning theory due to the following theorem which implies the uniform convergence sample complexity for any function in class $\mathcal{F}$ when $\operatorname{Pdim}(\mathcal{F})$ is finite.

Theorem A.3.1 (Uniform convergence sample complexity via pseudo-dimension, [4])**.** *Suppose $\mathcal{H}$ is a class of real-valued functions with range in $[0, H]$ and finite $\operatorname{Pdim}(\mathcal{F})$. For every $\epsilon > 0$ and $\delta \in (0,1)$, the sample complexity of (ϵ, δ)-uniformly learning the class $\mathcal{H}$ is $O\left(\left(\frac{H}{\epsilon}\right)^2 \left(\operatorname{Pdim}(\mathcal{F}) \log\left(\frac{H}{\epsilon}\right) + \log\left(\frac{1}{\delta}\right)\right)\right)$.*

Uniform learning is closely related to the notion of PAC (probably approximately correct) learning, indeed (ϵ, δ)-uniform learning corresponds to $(\epsilon/2, \delta)$-PAC learning [125].

We also need the following lemma from data-driven algorithm design.

Lemma A.3.2. *(Lemma 2.3, [7]) Suppose that for every problem instance $D \in \mathbf{D}$, the function $L_D(\rho) : \mathbb{R} \to \mathbb{R}$ is piecewise constant with at most N pieces. Then the family $\{L_\rho(\cdot)\}$ over instances in $\mathbf{D}$ has pseudo-dimension $O(\log N)$.*

The follwing theorem is due to [35] and is useful in obtaining some of our pseudodimension bounds.

Theorem A.3.3 ([35])**.** *Suppose that each function $f \in \mathcal{F}$ is specified by n real parameters. Suppose that for every $x \in \mathcal{X}$ and $r \in \mathbb{R}$, there is a GJ algorithm $\Gamma_{x,r}$ that given $f \in \mathcal{F}$, returns "true" if $f(x) \geq r$ and "false" otherwise. Assume that $\Gamma_{x,r}$ has degree Δ and predicate complexity Λ. Then, $\operatorname{Pdim}(\mathcal{F}) = O(n \log(\Delta\Lambda))$.*

Appendix B

Subsidy design in games

B.1 Proofs from Section 5.3.1

Proposition 5.3.1 (restated). *In the two-agent series prior game (defined above and cost matrix noted in the first row of Table 5.2), the Price of Anarchy in the absence of subsidy is at least PoA* $\geq \frac{2}{p_1+p_2}$*, for some repair costs* C_1, C_2*. More generally, for n agents, PoA* $\geq \tilde{H}/\tilde{G}^n$ *for some repair costs* $C_1, \ldots, C_n$*, where* $\tilde{H}$ *and* $\tilde{G}$ *are the harmonic and geometric means, respectively, of the prior probabilities* $p_1, \ldots, p_n$*.*

Proof. We set $C_1 = \overline{p_1}p_2$ and $C_2 = \overline{p_2}p_1$. Observe that DN-DN is an equilibrium since[1] $\overline{p_1p_2} = \overline{p_1} + \overline{p_2}p_1 \leq \overline{p_1} + C_2$, and similarly $\overline{p_1p_2} \leq \overline{p_2} + C_1$. Also, RE-RE is an equilibrium since $C_1 = \overline{p_1}p_2 \leq \overline{p_1}$ and $C_2 = \overline{p_2}p_1 \leq \overline{p_2}$. Clearly,

$$\text{PoA} \geq \frac{\text{cost(DN,DN)}}{\text{cost(RE,RE)}} = \frac{2\overline{p_1p_2}}{C_1 + C_2} \geq \frac{2}{p_1 + p_2},$$

where the last inequality follows from the observations

$$\overline{p_1p_2}(p_1 + p_2) = p_1 + p_2 - (p_1 + p_2)p_1p_2 \geq p_1 + p_2 - 2p_1p_2 = C_1 + C_2.$$

For the n agent series game, we set the costs $C_i = \overline{p_i}\Pi_{j\neq i}p_j$. DN^n (i.e., $s = 0^n$) is a Nash equilibrium, since

$$C_i + \overline{\Pi_{j\neq i}p_j} = (1 - p_i)\Pi_{j\neq i}p_j + 1 - \Pi_{j\neq i}p_j \geq \overline{\Pi_j p_j}.$$

[1] In our notation, $\overline{p_1p_2} := 1 - p_1p_2,$ while $\overline{p_1} \cdot \overline{p_2} := (1 - p_1) \cdot (1 - p_2)$.

Moreover, RE^n is also an equilibrium as $C_i \leq \overline{p_i}$ for all $i \in [n]$. Therefore,

$$\begin{aligned}
\text{PoA} &\geq \frac{\text{cost}(\text{DN}^n)}{\text{cost}(\text{RE}^n)} = \frac{n\overline{\Pi_i p_i}}{\sum_i C_i} \\
&= \frac{n\left(1 - \Pi_i p_i\right)}{\sum_i \Pi_{j\neq i} p_j - n\Pi_i p_i} \\
&\geq \frac{n\left(1 - \Pi_i p_i\right)}{\sum_i \Pi_{j\neq i} p_j - (\sum_i \Pi_{j\neq i} p_j)\Pi_i p_i} \\
&= \frac{n}{\sum_i \Pi_{j\neq i} p_j} \\
&= \frac{n}{\Pi_i p_i \cdot \sum_i \frac{1}{p_i}},
\end{aligned}$$

and the claim follows by noting $\tilde{H} = \frac{n}{\sum_i \frac{1}{p_i}}$ and $\tilde{G} = (\Pi_i p_i)^{1/n}$.

□

Theorem 5.3.2 (restated). *Consider the two-agent series component maintenance game with $C_1, C_2 > 0$ and $0 < p_1, p_2 < 1$. Let $s^* = \mathbf{I}\{(C_1, C_2) \in [\overline{p_1}p_2, \overline{p_1}] \times [\overline{p_2}p_1, \overline{p_2}]\} \cdot \min\{C_1 - \overline{p_1}p_2, C_2 - \overline{p_2}p_1\}$, where $\mathbf{I}\{\cdot\}$ denotes the 0-1 valued indicator function. Then there exists a subsidy scheme $\mathbb{S}$ with total subsidy s for any $s > s^*$ such that $\widetilde{PoA}(\mathbb{S}) = 1$. Moreover, a total subsidy of at least s^* is necessary for any subsidy scheme $\mathbb{S}$ that guarantees $\widetilde{PoA}(\mathbb{S}) = 1$.*

Proof. We will characterize the set of values of C_1, C_2 for which there are multiple Nash equilibria and design subsidy schemes that achieve $\widetilde{\text{PoA}}(\mathbb{S}) = 1$. We consider the following cases.

- C0: $C_1 < \overline{p_1}p_2, C_2 > \overline{p_2}$. In this case the only NE is RE-DN (Table 5.2). Thus, PoA $= 1$ even in the absense of subsidy and $s^* = 0$ in this case.
- C1: $C_1 = \overline{p_1}p_2, C_2 > \overline{p_2}$. Both RE-DN and DN-DN are Nash equilibria, and

  $$\text{cost(RE,DN)} = C_1 + 2\overline{p_2} = \overline{p_1}p_2 + 2\overline{p_2} = 2 - p_1p_2 - p_2 < 2\overline{p_1p_2} = \text{cost(DN,DN)}.$$

  An arbitrarily small subsidy to agent 1 is sufficient to guarantee $\widetilde{\text{PoA}}(\mathbb{S}) = 1$ (therefore $s^* = 0$ works) as DN-DN would no longer be a NE.
- C2: $C_1 > \overline{p_1}p_2, C_2 > \overline{p_2}$. In this case the only NE is DN-DN. Thus, PoA $= 1$ even in the absense of subsidy.
- C3: $C_1 < \overline{p_1}p_2, C_2 = \overline{p_2}$. Both RE-DN and RE-RE are Nash equilibria, and

  $$\text{cost(RE,DN)} = C_1 + 2\overline{p_2} > C_1 + \overline{p_2} = C_1 + C_2 = \text{cost(RE,RE)}.$$

  An arbitrarily small subsidy to agent 2 is sufficient to guarantee $\widetilde{\text{PoA}}(\mathbb{S}) = 1$ (therefore $s^* = 0$ works) as RE-DN would no longer be a NE.
- C4: $C_1 < \overline{p_1}p_2, C_2 < \overline{p_2}$. In this case the only NE is RE-RE. Thus, PoA $= 1$ even in the absense of subsidy.

C5: $(C_1, C_2) \in [\overline{p_1}p_2, \overline{p_1}] \times [\overline{p_2}p_1, \overline{p_2}]$. Both RE-RE and DN-DN are Nash equilibria, and OPT corresponds to RE-RE. A subsidy greater than $C_1 - \overline{p_1}p_2$ to agent 1, or a subsidy greater than $C_2 - \overline{p_2}p_1$ to agent 2 guarantees that the only NE is RE-RE. Further, in either case $\widetilde{\text{PoA}}(\mathbb{S}) = 1$ as the subsidy equals the reduction in the repair cost of the respective agent. Further suppose a subsidy of $s^* = s_1^* + s_2^*$ is sufficient to ensure $\widetilde{\text{PoA}}(\mathbb{S}) = 1$ in this case. Now if subsidy to agent 1 $s_1^* \leq C_1 - \overline{p_1}p_2$ and subsidy to agent 2 $s_2^* \leq C_2 - \overline{p_2}p_1$. Then both DN-DN and RE-RE are NE and $\widetilde{\text{PoA}}(\mathbb{S}) > 1$ since the worst-case equilibrium (i.e. DN-DN) cost does not depend on the subsidy. Therefore either $s_1^* > C_1 - \overline{p_1}p_2$ or $s_2^* > C_2 - \overline{p_2}p_1$, establishing that a subsidy of at least s^* is necessary in this case to ensure $\widetilde{\text{PoA}}(\mathbb{S}) = 1$.

C6: Otherwise. By symmetry, the case is similar to one of C0 through C4 with agents 1 and 2 switched. $s^* = 0$ and price of anarchy of 1 is achieved by no or arbitrarily small subsidy as above.

Note that s^* is non-zero only in case C5, in which case we have established both sufficiency and necessity of a total subsidy of s^* to ensure $\widetilde{\text{PoA}}(\mathbb{S}) = 1$.

□

Theorem 5.3.3 (restated). *Consider the two-agent series component maintenance game with $C_1, C_2 > 0$ and $0 < p_1, p_2 < 1$. Define $R \subset \mathbb{R}^+ \times \mathbb{R}^+$ as the set of cost pairs that satisfy $C_1 \leq \overline{p_1} \wedge C_2 \leq \overline{p_2} \wedge (C_1 \leq \overline{p_1}p_2 \vee C_2 \leq \overline{p_2}p_1)$, depicted in Figure 5.2. Let $s^* = \min_{(x,y)\in R} ||(C_1, C_2) - (x, y)||_1$, where $|| \cdot ||_1$ denotes the L_1-norm. Then there exists a subsidy scheme $\mathbb{S}$ with total subsidy s for any $s > s^*$ such that the system functions in any NE. Moreover, a total subsidy of at least s^* is necessary for any subsidy scheme $\mathbb{S}$ that guarantees that the system functions in any NE.*

Proof. (Sufficiency of s^*). We do this by cases on the cost vector (C_1, C_2), as follows.

C0: (C_1, C_2) lies in the interior of R. In this case it is easy to see that PoA $= 1$ and $s^* = 0$. In particular, the conditions $C_1 < \overline{p_1}$, $C_2 < \overline{p_2}$ rule out DN-RE, RE-DN as candidate equilibria respectively, and since $C_1 < \overline{p_1}p_2$ or $C_2 < \overline{p_2}p_1$, DN-DN cannot be a NE either (agent 1 or agent 2 will prefer to repair).

C1: $C_1 \leq \overline{p_1}p_2, C_2 \geq \overline{p_2}$. In this case, a subsidy more than $s^* = C_2 - \overline{p_2}$ to agent 2 is sufficient to bring the cost vector to the interior of R.

C2: $C_2 \leq \overline{p_2}p_1, C_1 \geq \overline{p_1}$. Symmetric to C1.

C3: $\overline{p_1}p_2 < C_1 \leq \overline{p_1}, \overline{p_2}p_1 < C_2 \leq \overline{p_2}$. A subsidy more than $\min_i\{C_i - \overline{p_i}p_{3-i}\}$ to agent $\text{argmin}_i\{C_i - \overline{p_i}p_{3-i}\}$ is sufficient to bring the cost vector to the interior of R.

C4: Otherwise. It is straighforward to verify by direct calculation that a subsidy of $s_1^* = \max\{C_1 - \overline{p_1}, 0\} + \overline{p_1} \cdot \overline{p_2}$ to agent 1 and a subsidy of $s_2^* = \max\{C_2 - \overline{p_2}, 0\} + \overline{p_1} \cdot \overline{p_2}$ to agent 2 is sufficient.

The necessity argument essentially follows by updating the cost matrix with conditional subsidies (s_1, s_2) and noting that the system is guaranteed to function in any NE if the costs (C_1, C_2) are in the interior of R. □

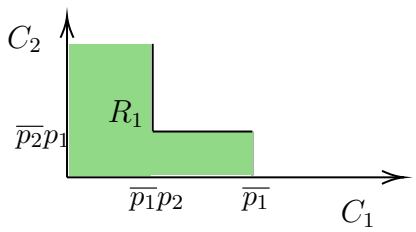


Figure B.1: Cost region R_1 with non-negative VoI for each agent when component 1 is inspected in a two-series game (Theorem B.1.1).

In addition to expected VoI, we can also ensure posterior conditioned $\text{VoI}_{i,j}$ when component c_j is inspected is non-negative for each agent i and each posterior y_j via subsidy. The following result gives the optimal value of subsidy to ensure this.

Theorem B.1.1. *Consider the two-agent series component inspection game with $C_1, C_2 > 0$ and $0 < p_1, p_2 < 1$. Then*

(a) $\text{VoI}_{i,j}(s_i, \tilde{s}^{j,1}) \geq 0$ for any agents i, j, any prior NE s_i and any posterior NE $\tilde{s}^{j,1}$, when the inspected component j is working, except when $C_{3-j} = \overline{p_{3-j}}$ and an arbitrarily small subsidy is sufficient to ensure VoI is non-negative in this case.

(b) Define $R_1 \subset \mathbb{R}^+ \times \mathbb{R}^+$ as the set of cost pairs that satisfy $(C_j \leq \overline{p_j} \wedge C_{3-j} \leq \overline{p_{3-j}}p_j) \vee (C_j \leq \overline{p_j}p_{3-j})$. Let $s^* = \min_{(x,y)\in R_1} ||(C_1, C_2) - (x, y)||_1$, where $||\cdot||_1$ denotes the L_1-norm. Then there exists a subsidy scheme $\mathbb{S}$ with total (unconditional) subsidy s for any $s > s^*$ such that $\text{VoI}_{i,j}(s_i, \tilde{s}^{j,0}) \geq 0$ for any agents i, j, any prior NE s_i and any posterior NE $\tilde{s}^{j,0}$, when the inspected component j is broken. Moreover, a total (unconditional) subsidy of at least s^* is necessary for any subsidy scheme $\mathbb{S}$ that guarantees that the system functions in any NE.

Proof. WLOG $j = 1$.

(a) If $y_1 = 1$, the only candidate posterior equilibria are DN-DN if $C_2 \geq \overline{p_2}$ and DN-RE if $C_2 \leq \overline{p_2}$.
If $C_2 > \overline{p_2}$, then the prior NE is either DN-DN or RE-DN. In either case both agents have a non-negative Value of Information (Table 5.2). If $C_2 < \overline{p_2}$, then the prior NE is either DN-DN, RE-RE or DN-RE. In each case both agents can be readily verified to have a non-negative Value of Information. If $C_2 = \overline{p_2}$ and $C_1 \leq \overline{p_2}$, then VoI can be negative for agent 1 if the prior equilibrium is RE-RE and the posterior equilibrium is DN-DN. But in this case a (unconditional, or conditional on inspection) subsidy of $s_2^* > 0$ to agent 2 ensures that DN-DN is not a posterior equilibrium and both agents have non-negative VoI.

(b) For $y_1 = 0$, we consider the following cases.

C0: $C_1 \leq \overline{p_1}p_2, C_2 > \overline{p_2}$. If $C_1 < \overline{p_1}p_2$, the only prior NE is RE-DN and the only posterior NE is also RE-DN. Thus, VoI $= 0$ for each agent even in the absense of subsidy and $s^* = 0$ in this case. If $C_1 = \overline{p_1}p_2$, DN-DN is also a prior NE but VoI is still non-negative for each agent, since the posterior costs

$$\text{cost}_1(\text{RE,DN}) = C_1 + \overline{p_2} \leq \overline{p_1}p_2 + \overline{p_2} = \overline{p_1 p_2},$$

and

$$\text{cost}_2(\text{RE,DN}) = \overline{p_2} < \overline{p_1 p_2}.$$

C1: $\overline{p_1} p_2 < C_1 \leq p_2, C_2 > \overline{p_2}$. DN-DN is the only prior NE and the only posterior equilibrium is RE-DN (except if $C_1 = p_2$, when DN-DN is also a posterior NE), and

$$\text{cost}_1(\text{RE,DN}) = C_1 + \overline{p_2} > \overline{p_1 p_2}.$$

A subsidy of $C_1 - \overline{p_1} p_2$ to agent 1 is sufficient to ensure VoI is non-negative for agent 1 as noted in case C0. Alternatively, a subsidy of $C_2 - p_1 \overline{p_2}$ to agent 2 ensures that RE-RE is the only prior and posterior NE, and VoI is non-negative for each agent. The smaller of the two subsidies works and is necessary in this case.

C2: $C_1 > p_2, C_2 > \overline{p_2}$. In this case the only NE is DN-DN in both prior and posterior games. The value of information is negative for both agents in the absense of subsidy as the posterior costs are 1. Subsidy schemes described in case C1 above can be verified to be optimal in this case as well.

C3: $C_1 < \overline{p_1} p_2, C_2 = \overline{p_2}$. Both RE-DN and RE-RE are prior and posterior Nash equilibria, and

$$\text{cost(RE,DN)} = C_1 + 2\overline{p_2} > C_1 + \overline{p_2} = C_1 + C_2 = \text{cost(RE,RE)}.$$

An arbitrarily small subsidy to agent 2 is sufficient to guarantee non-negative VoI as RE-DN would no longer be a NE.

C5: Otherwise. Value of Information for agent 1 is negative since either DN-DN or DN-RE is a posterior NE with cost 1, and there is a prior NE with smaller cost to agent 1. It may be verified that the subsidy scheme from Theorem 5.3.3 is optimal in ensuring non-negative VoI in this case.

□

B.2 Optimal subsidy in two agent parallel game

The cost matrix for a two-agent parallel game is summarized in Table B.1. p_R denotes the probability that remaining components work. These additional components are not controlled by any agent, and connected in parallel to the components controlled by the two agents.

B.2.1 Guaranteeing system functions in any NE

Theorem B.2.1. *Consider the two-agent parallel component maintenance game such that $C_1, C_2 > 0$ and $0 < p_1, p_2 < 1$. Let $s^* = \mathbf{I}\{(C_1, C_2) \in [\overline{p_R} \cdot \overline{p_1} \cdot \overline{p_2}, \infty)^2\} \cdot (\min\{C_1, C_2\} - \overline{p_R} \cdot \overline{p_1} \cdot \overline{p_2})$, where $\mathbf{I}\{\cdot\}$ denotes the 0-1 valued indicator function. Then there exists a subsidy scheme $\mathbb{S}$ with total subsidy s for any $s > s^*$ such that the system functions in any NE. Moreover, a total subsidy of at least s^* is necessary for any subsidy scheme $\mathbb{S}$ that guarantees that the system functions in any NE.*

Conditions	DN-DN	DN-RE	RE-DN	RE-RE
Prior	$\overline{p_R}\cdot\overline{p_1}\cdot\overline{p_2}, \overline{p_R}\cdot\overline{p_1}\cdot\overline{p_2}$	$0, C_2$	$C_1, 0$	C_1, C_2
$y_1 = 1$	$0, 0$	$0, C_2$	$C_1, 0$	C_1, C_2
$y_1 = 0$	$\overline{p_R}\cdot\overline{p_2}, \overline{p_R}\cdot\overline{p_2}$	$0, C_2$	$C_1, 0$	C_1, C_2

Table B.1: Matrix for cost-pairs (agent 1, agent 2) when component c_1 is inspected for the two-agent parallel system.

Proof. Note that the system does not function only in DN-DN, for which to be a NE we must have $(C_1, C_2) \in [\overline{p_R}\cdot\overline{p_1}\cdot\overline{p_2}, \infty)^2$. Now a subsidy $s_i^* > C_i - \overline{p_R}\cdot\overline{p_1}\cdot\overline{p_2}$ to agent i guarantees that agent i chooses repair, establishing the first part of the theorem.

Conversely, suppose $(C_1, C_2) \in [\overline{p_R}\cdot\overline{p_1}\cdot\overline{p_2}, \infty)^2$ and a subsidy scheme with total subsidy s^* guarantees that the system functions in any NE. But if $s_i^* \leq C_i - \overline{p_R}\cdot\overline{p_1}\cdot\overline{p_2}$ for $i \in \{1, 2\}$, then DN-DN is still an NE, and the system does not function for this NE. By contradiction, the total subsidy must be at least s^* as claimed. □ □

B.2.2 Value of Information

For this case as well, the Value of Information may be negative if a local equilibrium is selected for some parameter settings. In the following we will show a dichotomy—if the repair costs are small then a central agent using subsidy must subsidize the full costs of repair to avoid negative Value of Information of component inspection for the agents. Otherwise, the central agent can partially subsidize to avoid negative VoI.

Theorem B.2.2. *Suppose* $C_1, C_2 \notin \{0, \overline{p_R}\cdot\overline{p_2}, \overline{p_R}\cdot\overline{p_1}\cdot\overline{p_2}\}$. *A subsidy scheme with* $s_1^* > \max\{C_1 - \overline{p_R}\cdot\overline{p_1}\cdot\overline{p_2}, \min\{C_1, \overline{p_R}\cdot\overline{p_2}\}\}$ *and* $s_2^* > \max\{C_2 - \overline{p_R}\cdot\overline{p_1}\cdot\overline{p_2}, \min\{C_2, \overline{p_R}\cdot\overline{p_2}\}\}$, *conditional on inspection, is sufficient to avoid negative VoI for both agents when component 1 is inspected.*

Proof. Note that RE-RE cannot be an equilibrium since $C_1, C_2 > 0$. Also negative VoI is not possible when the posterior is $y_1 = 1$ as the only Nash equilibrium is DN-DN with zero cost for each agent.

First suppose $\min\{C_1, C_2\} \geq \overline{p_R}\cdot\overline{p_1}\cdot\overline{p_2}$. This implies DN-DN is the prior equilibrium. In this case, the conditional subsidies of $s_i^* = C_i - \overline{p_R}\cdot\overline{p_1}\cdot\overline{p_2}$ are sufficient to ensure that posterior repair costs for each agent is less than $\overline{p_R}\cdot\overline{p_1}\cdot\overline{p_2} \leq \overline{p_R}\cdot\overline{p_2}$. Thus, DN-DN cannot be a posterior NE for $y_1 = 0$. For DN-RE and RE-DN, the subsidy ensures that VoI is non-negative for both agents.

Otherwise, we have three cases to consider w.r.t. relative choice values of repair costs and failure probabilities,

C1: $C_1 < \overline{p_R}\cdot\overline{p_1}\cdot\overline{p_2} \leq C_2$. In this case, RE-DN is the prior equilibrium. Moreover, since $C_1 < \overline{p_R}\cdot\overline{p_1}\cdot\overline{p_2} \leq \overline{p_R}\cdot\overline{p_2}$, agent 1 would prefer action RE over DN and so DN-DN cannot be a posterior NE for $y_1 = 0$. If RE-DN is the posterior NE, then by Table B.1 clearly VoI is non-negative for both agents. So it only remains to consider the posterior equilibrium DN-RE. If $C_2 > \overline{p_R}\cdot\overline{p_2}$, DN-RE cannot be an equilibrium, and we are done.

If $C_2 \leq \overline{p_R} \cdot \overline{p_2}$, then the subsidy $s_2^* > \min\{C_2, \overline{p_R} \cdot \overline{p_2}\}$ ensures that VoI of agent 2 is non-negative even if DN-RE is the posterior equilibrium.

C2: $C_2 < \overline{p_R} \cdot \overline{p_1} \cdot \overline{p_2} \leq C_1$. The argument for this case is symmetric to the previous case, with DN-RE as the only possible prior equilibrium.

C3: $\max\{C_1, C_2\} < \overline{p_R} \cdot \overline{p_1} \cdot \overline{p_2}$. In this case DN-RE as well as RE-DN can be prior Nash equilibria. Since $\max\{C_1, C_2\} < \overline{p_R} \cdot \overline{p_1} \cdot \overline{p_2} \leq \overline{p_R} \cdot \overline{p_2}$, DN-RE and RE-DN are the only candidate posterior NE. The setting of subsidies $s_i^* > \overline{p_R} \cdot \overline{p_2} \geq \overline{p_R} \cdot \overline{p_1} \cdot \overline{p_2} > \max\{C_1, C_2\}$ ensures that VoI is non-negative for both agents in this case.

□

B.3 Additional proofs from Section 5.3.2

Theorem 5.3.6 (restated). CMG-SYSTEM *is NP-Hard.*

Proof. We will reduce the VERTEX-COVER problem to CMG-SYSTEM. Given an instance $\mathcal{G}, k$ of the VERTEX COVER problem, we create a corresponding CMG-SYSTEM problem as follows. Introduce an agent i for every vertex $i \in V$ and consider the (2-CNF) formula $\phi(\mathbf{x}) = \bigwedge_{(i,j)\in E}(x_i \vee x_j)$, where the clauses consist of states x_i, x_j for all pairs i, j of agents/components corresponding to edges in E. Set the probability distribution θ to be the constant distribution with the entire probability mass on 0^n (i.e. all the components are guaranteed to fail without repair). Set repair cost $C_i = 1 - \epsilon$ for $0 < \epsilon < 1/n$ for all components i.

Observe that,

$$\begin{aligned} l_i(s_i, s_{-i}, \theta) &= \mathbb{E}_{\mathbf{x}\sim\theta}[\text{cost}_i] \\ &= C_i s_i + P_\phi(\theta) \\ &= (1-\epsilon)s_i + 1 - \phi(\mathbf{x}') \\ &= (1-\epsilon)s_i + 1 - \phi(s), \end{aligned}$$

where $x_i' = \max\{0, s_i\} = s_i$ denotes the state of component i after agent i takes action s_i. Note that WLOG $s = 0^n$ is a NE for this game, since any repair action by any agent increases the agent's cost by $(1-\epsilon)$ if the repair does not change the state ϕ of the system, and 0 otherwise (since ϕ is monotonic, it can only change from 0 to 1). We will now show that the remaining NEs for the game correspond to vertex covers of $\mathcal{G}$.

Suppose $K \subseteq [n]$ be a set of agents for which the corresponding nodes in $\mathcal{G}$ constitute a minimal vertex cover. Let $s_{K'} = (s_1, \ldots, s_n)$ where $s_i = \mathbf{I}[i \in K']$, where $\mathbf{I}[\cdot]$ is the 0-1 valued indicator function and $K' \subset K$ with $|K'| = |K| - 1$. Similarly, $s_K := (s_1, \ldots, s_n)$ where $s_i = \mathbf{I}[i \in K]$. Clearly, $\phi(s_{K'}) = 0$ and $\phi(s_K) = 1$. If $i \in K$, agent i does not reduce cost by switching from RE to DN since K is a minimal cover therefore not repairing component i causes the system to fail. If $j \notin K$, agent j does not reduce cost by switching from DN to RE as the system was already functioning. Further, if K is the set of agents corresponding to a non-minimal vertex cover, then there must be some agent that can reduce its cost by switching from RE to DN. Finally, if $K' \neq \emptyset$ is the set of agents with one or more agents short of a vertex

cover, then any agent in K' can reduce its cost by switching from RE to DN. This establishes that besides 0^n, only possible NE must correspond to a minimal vertex cover.

To complete the reduction, we consider the game defined above and subsidy budget $s^* = k-1$. If there exists a vertex cover of size k, then a minimal cover K' has size $k' \leq k$. We design a subsidy scheme with total subsidy $s' = k' - 1 \leq s^*$, allocating subsidy of 1 for repair to all but one agent in the minimal cover K' and subsidy of 0 otherwise. Clearly the only agent not given subsidy will choose repair since cost of repair $1-\epsilon$ is more than compensated by the change due to system state. As argued above, the only candidate NE are 0^n and s_K corresponding to some minimal vertex cover K. Without the subsidy, the social cost for 0^n is n (except the trivial case $k = 0$) and that for $s_{K'}$ is $k'(1-\epsilon)$ which is smaller. If we provide subsidy in our scheme $\mathbb{S}$ to the agents in K' except one then 0^n is no longer an NE. In particular, every subsidized agent in K' would now choose repair at cost $1-\epsilon$ over doing nothing (even when the system stays broken after the repair) and the remaining agent in K' will choose repair if the system is broken. Thus, the system functions in all NEs.

Conversely, suppose there exists a subsidy scheme $\mathbb{S}$ with total subsidy at most $s^* = k-1$, such that the system functions in any NE. Then either the system functions in 0^n and there is a 0-cover for graph $\mathcal{G}$, or the NE corresponds to a minimal vertex-cover K' of size k' as the repaired components (in the subsidized game). In the latter case, we seek to show $k' \leq k$ to complete the proof. Since the system is not assumed to function for $s = s_\kappa$ for repair actions by agents in any $\kappa \subset K'$ with $\kappa = k' - 1$, we need to provide subsidy at least $1-\epsilon$ to all but one agent in K'. That is, $k - 1 = s^* \geq (1-\epsilon)(k'-1) > k' - 1 - (k'-1)/n$ (since $\epsilon < 1/n$), or $k \geq k'$ since both k, k' are integers. □

Theorem 5.3.7 (restated). CIG-VOI *is NP-Hard.*

Proof. We will reduce VERTEX-COVER to CIG-VOI. Recall that VERTEX-COVER is the following decision problem—given a graph $\mathcal{G} = (V, E)$ and integer k, does there exist a vertex cover of size k? In contrast to proof of Theorem 5.3.6, we will need to set a slightly higher subsidy and carefully adapt the argument to the value of information computation.

We will create an instance of the CIG-VOI problem with $n = |V| + 1$ agents, an agent each for vertices in $\mathcal{G}$ and an additional agent $j = |V| + 1$. The construction of the instance and several arguments are similar to the proof of Theorem 5.3.6. The key difference is that we have an additional agent j that does not correspond to a vertex in $\mathcal{G}$. We will consider the inspection of the component c_j corresponding to this agent.

Consider the (2-CNF) formula $\phi(\mathbf{x}) = \bigwedge_{(u,v)\in E}(x_u \vee x_v)$, where the clauses consist of states x_u, x_v for all pairs u, v of agents/components corresponding to edges in E. Set the probability distribution θ to be the constant distribution with the entire probability mass on 0^n (i.e. all the components are guaranteed to fail without repair). Set repair cost $C_i = 1-\epsilon$ for $0 < \epsilon < 1/n$ for all components $i \in [|V|]$ and $C_j = 1$. Therefore, $l_i(s_i, s_{-i}, \theta) = (1-\epsilon)s_i + 1 - \phi(s)$ for $i \in [|V|]$ and $l_j(s_j, s_{-j}, \theta) = 2 - \phi(s)$. Note that WLOG $s = 0^n$ is a NE for this game, since any repair action by any agent increases the agent's cost by $(1-\epsilon)$ if the repair does not change the state ϕ of the system, and 0 otherwise (since ϕ is monotonic, it can only change from 0 to 1). As shown in the proof of Theorem 5.3.6, the remaining NEs for the game correspond to minimal vertex covers of $\mathcal{G}$. Moreover, since $\phi(s)$ does not depend on s_j by definition, agent j will always prefer action DN for any s_{-j}. Let $s_K := (s_1, \ldots, s_n)$ where $s_i = \mathbf{I}[i \in K]$ for any $K \subseteq V$.

Notice that the prior and posterior games (for inspection of c_j) have identical cost matrices and equilibria for this component inspection game. To complete the reduction, we consider the game defined above and subsidy budget $s^* = k$. Suppose there exists a vertex cover of $\mathcal{G}$ of size k, then there exists a minimal vertex cover, say K' of size $k' \leq k$. We design a subsidy scheme with total subsidy $s' = k' \leq s^*$, allocating subsidy of 1 for repair to exactly the agents in K'. Clearly, all subsidized agents will always choose repair. We claim that the only NE after subsidy is $s_{K'}$. Indeed, by the above observation, any NE must be s_K for some $K \supseteq K'$. But if $K \neq K'$, then any agent in $K \setminus K'$ will choose to do nothing as the system would function without their repair action. Since there is exactly one NE in prior and posterior games, Value of Information is exactly zero for all agents.

Conversely, if there is no vertex cover of size k, then we show that no subsidy scheme with $s^* \leq k$ may guarantee that no agent has negative value of information when a single component j is inspected. In this case the any vertex cover K' has $|K'| > k$. We consider two cases:

C0: $|K'| > k + 1$. Observe that if the subsidy provided to an agent is less than the repair cost $1 - \epsilon$, then the agent will prefer to do nothing, except when repairing their component (given other players actions) changes the system state from 0 to 1. However, with a budget of $s^* = k$, the maximum number of agents that can receive a subsidy of at least $1 - \epsilon$ is at most $\frac{k}{1-\epsilon} < k + 1$, since $\epsilon < 1/n$ and $k < n$ WLOG. Thus, at least two agents are without subsidy at least $1 - \epsilon$ in K', and these agents will prefer to do nothing if only the agents $K^* = \{i \in [|V|] \mid s_i^* > 1 - \epsilon\}$ with sufficient subsidy choose repair. Observe that both $s_{K'}$ and s_{K^*} are Nash equilibria in the subsidized game. If $s_{K'}$ is chosen as the prior equilibrium and s_{K^*} a posterior equilibrium, then the value of information for agents in $K' \setminus K^*$ is $(1 - \epsilon) - 1 < 0$ since the system does not work in s_{K^*}.

C1: $|K'| = k + 1$. In this case, the only new possibility is if at least $1 - \epsilon$ subsidy is provided to all but one agent (say k') in K', then the remaining agent will choose repair. Without loss of generality, we assume $k + 1 < n$, and that K' is a minimal vertex cover. Now if $v_{k'}$ denotes the vertex corresponding to agent k' in $\mathcal{G}$, and let E' denote the set of edges incident on vertices $V' \subseteq V \setminus K'$ with one end at $v_{k'}$. E' is non-empty, as otherwise $K' \setminus \{v_{K'}\}$ would constitute a vertex cover for $\mathcal{G}$ contradicting minimality of K'. Observe that $K_1 = K' \setminus \{v_{K'}\} \cup V'$ is a vertex cover. Let K_2 denote a minimal vertex cover which is a subset of K_1. Now both s_{K_2} and $s_{K'}$ are NEs in the subsidized game. If the former is set as the prior equilibrium, and the latter a posterior equilibrium then, the value of information is negative (equals $0 - (1 - \epsilon) = \epsilon - 1$) for agent k'.

Thus in either case, some agent has a negative value of information when the subsidy budget is k. This completes the reduction. □

B.4 Additional proofs from Section 5.4

Theorem 5.4.3 (restated). *For any $\epsilon, \delta > 0$ and any distribution $\mathcal{D}$ over component maintenance games with n agents, $O(\frac{n^2H^2}{\epsilon^2}(n^2 + \log\frac{1}{\delta}))$ samples of the component maintenance game drawn from $\mathcal{D}$ are sufficient to ensure that with probability at least $1 - \delta$ over the draw of the*

samples, the best vector of subsidies over the sample $\hat{s^*}$ has expected loss L_{prior} that is at most ϵ larger than the expected loss of the best vector of subsidies over $\mathcal{D}$.

Proof. Consider any fixed game G. Given any joint action $s = (s_i, s_{-i})$, an agent i's decision for switching their action from s_i to $\overline{s_i} := 1 - s_i$ is determined by the inequality $(C_i - s_i^*)s_i + 1 - \Phi(s_i, s'_{-i}) \leq (C_i - s_i^*)(\overline{s_i}) + 1 - \Phi(\overline{s_i}, s'_{-i})$, where $\Phi(s) := \mathbb{E}_\theta[\phi(\mathbf{x}'(s))]$, which is linear in s_i^*, the subsidy provided to agent i. Thus, for each agent i, we have at most 2^{n-1} axis-parallel hyperplanes in the parameter space in $\mathbb{R}^n$, or a total of $n2^{n-1}$ hyperplanes overall. Moreover, the loss function as a function of the parameters is piecewise constant in any fixed piece. Therefore the loss function class is $(n, n2^{n-1})$-delineable in the sense of [17], that is the subsidy parameter space is Euclidean in n dimensions and is partitioned by at most $n2^{n-1}$ hyperplanes into regions where the loss is linear (in this case constant) in the parameters.
By using a general result from [17] which states that a (d, t)-delineable function class has pseudo-dimension $O(d \log(dt))$, the above structural argument implies that the pseudo-dimension of the loss function class parameterized by the subsidy value is at most $O(n \log(n^2 2^{n-1})) = O(n^2)$ and the sample complexity result follows [4, 7]. □ □

Learning conditional subsidies. We will now obtain a sample complexity bound for non-uniform subsidy schemes in component inspection games, where the central agent provides subsidy only in posterior games. Let $\mathbb{S}$ denote the subsidy scheme. Let $\mathcal{S}^0_{NE}(\mathbb{S})$ (resp. $\mathcal{S}^1_{NE}(\mathbb{S})$) denote the subset of states in S corresponding to Nash equilibria when the cost for agent i is the subsidized cost $\text{cost}_i^{\mathbb{S},0}$ (resp. $\text{cost}_i^{\mathbb{S},1}$) for posterior $y_j = 0$ (resp. $y_j = 1$). For component inspection game of component c_1 (wlog), define

$$L_{\text{posterior}}(\mathbb{S}) := p_1 L^1_{\text{posterior}}(\mathbb{S}) + \overline{p_1} L^0_{\text{posterior}}(\mathbb{S}),$$

where $L^i_{\text{posterior}}(\mathbb{S}) := \max_{s \in \mathcal{S}^i_{NE}(\mathbb{S})} \text{cost}^{\mathbb{S},i}(s) + \text{subs}^i(s)$. We assume that $\text{subs}^j_i(s) \leq H, C_i \leq H$ for each $i \in [n]$, $j \in \{0, 1\}$, thus $L_{\text{posterior}}(\mathbb{S}) \leq (2H + 1)n$. In this case too, we are able to give a polynomial sample complexity for the number of games needed to learn a good value of subsidy with high probability over the draw of game samples coming from some fixed but unknown distribution.

Theorem B.4.1. *For any $\epsilon, \delta > 0$ and any distribution $\mathcal{D}$ over component inspection games with n agents, $O(\frac{n^2H^2}{\epsilon^2}(n^2 + \log \frac{1}{\delta}))$ samples of the component inspection game drawn from $\mathcal{D}$ are sufficient to ensure that with probability at least $1 - \delta$ over the draw of the samples, the best vector of subsidies over the sample $\hat{s^*}$ has expected loss $L_{posterior}$ that is at most ϵ larger than the expected loss of the best vector of subsidies over $\mathcal{D}$.*

Proof. Consider any fixed game G. Given any joint action $s = (s_i, s_{-i})$, an agent i's decision for switching their action from s_i to $\overline{s_i} := 1 - s_i$ in posterior game $y_1 = y$ is determined by the inequality $(C_i - s_i^y)s_i + 1 - \Phi(s_i, s'_{-i}) \leq (C_i - s_i^y)(\overline{s_i}) + 1 - \Phi(\overline{s_i}, s'_{-i})$ (with $\Phi(s) := \mathbb{E}_{\tilde{\theta}^{1,y}}[\phi(\mathbf{x}'(s))]$), which is linear in s_i^y, the subsidy provided to agent i conditional on $y_1 = y$. Thus, for each agent i, we have at most $2 \cdot 2^{n-1}$ axis-parallel hyperplanes in the parameter space in $\mathbb{R}^{2n}$, or a total of $n2^n$ hyperplanes overall. Moreover, the loss function as a function of the parameters is piecewise constant in any fixed piece. Therefore the loss function class is $(2n, n2^n)$-delineable in the sense

of [17]. The rest of the argument is similar to the proof of Theorem 5.4.3, differing only in some multiplicative constants. □

Theorem 5.4.4 (restated). *Suppose Assumption 5 holds. Let $L_1, \ldots, L_T : [0, H] \to [0, (2H + 1)N]$ denote an independent sequence of losses as a function of the subsidy value σ, in an online sequence of T component maintenance games. Then sequence of functions is $\frac{1}{2}$-dispersed and there is an online algorithm with $\tilde{O}(\sqrt{nT})$ expected regret.*

Proof. The key idea is to observe that each loss function L_t has at most $K = n2^{n-1}$ discontinuities (as in proof of Theorem 5.4.2 above). Further, any interval of length ϵ has at most $O(\kappa\epsilon T)$ functions that are non-Lipschitz in that interval, in expectation. This uses Assumption 5, and the observation that critical values of s^* are linear in some cost C_i. Indeed, as shown in the proof of Theorem 5.4.2, the critical values of subsidy are given by $s^* = C_i + \mathbb{E}_\theta\phi(0, s'_{-i}) - \mathbb{E}_\theta\phi(1, s'_{-i})$ for some agent i and joint action s'_{-i}. By [19] then the expected number of non-Lipschitz losses on the worst interval of length ϵ is at most $\tilde{O}(T\epsilon + \sqrt{T\log(TK)}) = \tilde{O}(\sqrt{(n + \log T)T})$ for $\epsilon \geq \frac{1}{\sqrt{T}}$.

This implies $1/2$-dispersion of the sequence of loss functions in the sense of Definition 2. Then Theorem 5 from [19] with $M = 1$ implies the desired regret bound. □

Appendix C

Robustness

We include proofs and additional details from Chapter 6 here.

C.1 Technical lemmas for robustness of Algorithm 10

The following lemma gives a bound on the fraction of the surface of the sphere at some fixed small distance from the subspace in Theorem 6.1.3. The bound involves a geometric calculation of a surface element of a sphere in $\mathbb{R}^n$.

Lemma C.1.1. *The fraction of the surface of the unit $(n-1)$-sphere at a distance at most small $\varepsilon = o(1)$ from a fixed $(n-k)$-hyperplane through its center is at most $\frac{2\varepsilon^k}{k} \cdot \frac{A(k-1)A(n-k-1)}{A(n-1)}$, where $A(m)$ is the surface-area of the unit m-sphere embedded in $m+1$ dimensions.*

Proof. Let the fixed hyperplane be $x_1 = x_2 = \cdots = x_k = 0$. We change the coordinates to a product of spherical coordinates (ρ is the distance from the hyperplane, r is the orthogonal component of the radius vector).

$$x_j = \begin{cases} \rho S_{j-1} \cos\phi_j, & \text{if } j < k; \\ \rho S_{j-1}, & \text{if } j = k; \\ r T_{j-k-1} \cos\alpha_{j-k}, & \text{if } k < j < n; \\ r T_{j-k-1}, & \text{if } j = n. \end{cases}$$

where $S_l = \prod_{i=1}^{l} \sin\phi_i$ and $T_l = \prod_{i=1}^{l} \sin\alpha_i$. The desired surface area is easier to compute in the new coordinate system.

The new coordinates are $(y_1, \ldots, y_n) = (\rho, \phi_1, \phi_2, \ldots, \phi_{k-1}, r, \alpha_1, \ldots, \alpha_{n-k-1})$. Let $z = \sqrt{r^2+\rho^2} = \sqrt{\sum_{i=1}^{n} x_i^2}$ denote the usual radial spherical coordinate. Volume element in this new coordinate system is given by

$$dV = |\det(J)|\, d\rho\, d\phi_1 \ldots d\phi_{k-1} dr\, d\alpha_1 \ldots d\alpha_{n-k-1},$$

where J is the Jacobian matrix, $J_{ij} = \frac{\partial x_i}{\partial y_j}$. We can write

$$J = \begin{bmatrix} A & 0 \\ 0 & B \end{bmatrix},$$

where $A_{ij} = \frac{\partial x_i}{\partial y_j}$ for $1 \le i, j \le k$ and $B_{ij} = \frac{\partial x_{i+k}}{\partial y_{j+k}}$ for $1 \le i, j \le n-k$.

By Leibniz formula for determinants, it is easy to see

$$\begin{aligned}\det(J) &= \det(A) \cdot \det(B) \\ &= \rho^{k-1} \left(\prod_{i=1}^{k-2} \sin^{k-i-1} \phi_i \right) \cdot r^{n-k-1} \left(\prod_{i=1}^{n-k-2} \sin^{n-k-i-1} \alpha_i \right) \\ &= \rho^{k-1} r^{n-k-1} \left(\prod_{i=1}^{k-2} \sin^{k-i-1} \phi_i \right) \left(\prod_{i=1}^{n-k-2} \sin^{n-k-i-1} \alpha_i \right).\end{aligned}$$

Now the surface element is given by

$$dS = \frac{1}{z^{n-1}} \frac{dV}{dz} = \frac{1}{z^{n-1}} \left(\frac{dV}{dr} \frac{\partial r}{\partial z} + \frac{dV}{d\rho} \frac{\partial \rho}{\partial z} \right) = \frac{1}{r z^{n-2}} \frac{dV}{dr} + \frac{1}{\rho z^{n-2}} \frac{dV}{d\rho}.$$

Plugging in our computation for dV,

$$dS = \left(\frac{\rho^{k-1} r^{n-k-2}}{z^{n-2}} \, d\rho + \frac{\rho^{k-2} r^{n-k-1}}{z^{n-2}} \, dr \right) \left(\prod_{i=1}^{k-2} \sin^{k-i-1} \phi_i d\phi_i \right) \left(\prod_{i=1}^{n-k-2} \sin^{n-k-i-1} \alpha_i d\alpha_1 \right).$$

We care about $z = 1$ and $\rho \le \varepsilon$ (or $r \ge \sqrt{1-\varepsilon^2}$). Notice

$$\int_{\sqrt{1-\varepsilon^2}}^{1} \frac{\rho^{k-2} r^{n-k-1}}{z^{n-2}} \, dr = \int_{\varepsilon}^{0} \rho^{k-2} r^{n-k-1} \frac{-\rho d\rho}{r} = \int_0^{\varepsilon} \rho^{k-1} r^{n-k-2} d\rho.$$

Thus, using the surface element in the new coordinates and integrating, we get

$$\text{Area of } \varepsilon\text{-close points} = A(k-1)A(n-k-1) \cdot 2 \int_0^{\varepsilon} \rho^{k-1} r^{n-k-2} d\rho \le A(k-1)A(n-k-1) \cdot \frac{2\varepsilon^k}{k}$$

which gives the desired fraction. □

C.2 Technical lemmas for online threshold parameter tuning

We begin with an observation, which allows us to focus on small τ. In particular we note that the nearest-neighbor distance for most points is $O(m^{-1/n_2})$, and therefore searching for threshold in the range $[0, \tau_{\max}]$ with $\tau_{\max} = O(m^{-1/n_2})$ is sufficient for almost no abstention. This can provide a useful guide in setting $\tau_{\max}$ in Theorem 6.3.2. To simplify our results, we will treat n_2, n_3 as constants in the following.

Lemma C.2.1. *Let Φ be a distribution defined on a compact convex subset C of $\mathbb{R}^n$ whose density function ϕ is continuous and strictly positive on C (that is $\phi(\mathbf{x}) > 0$ for $\mathbf{x} \in C$), and has bounded partial derivatives throughout C. If m samples $B = \{\beta_1, \ldots, \beta_m\}$ are drawn from Φ, for any β_i the probability that the distance d_i to its nearest neighbor in B is not $O(m^{-1/n})$ is $o(1)$.*

Proof. We use the asymptotic moments of nearest neighbor distance distribution due to [76] together with a concentration inequality to complete the proof. Indeed, the asymptotic mean nearest neighbor distance is shown to be $O(m^{-1/n})$, and the variance is $O(m^{-2/n})$. By Chebyshev's inequality, the probability that d_i is outside $\omega(1)$ standard deviations is $o(1)$. □

We will need the following lemma about Lipschitzness of $\mathcal{E}_{\text{adv}}(\tau)$. The argument can also be adapted to bounded density adversary (Corollary C.2.3), and to show a bound on the breakpoints in $\hat{\mathcal{E}}_{\text{adv}}(\tau)$ (Corollary C.2.4).

Lemma C.2.2. *If $\tau \leq \tau_{\max} = o(r)$, $\mathcal{E}_{\text{adv}}(\tau)$ is $O\left(m\tau_{\max}^{n_2-n_3-1}/r^{n_2-n_3}\right)$-Lipschitz.*

Proof. Consider the probability that the adversary is able to succeed in misclassifying a test point x as a fixed training point x' (of different label) only when the threshold increases from τ to $\tau + d\tau$. Scale all distances by a factor of $\frac{1}{\mathsf{dist}(x,x')} =: \frac{1}{r'}$. WLOG, let x be the origin and the adversarial subspace S be given by $x_{n_3+1} = x_{n_3+2} = \cdots = x_{n_2} = 0$, and x' is the uniformly random unit vector $(z_1, \ldots, z_{n_2})$. The adversary can win only if the distance Δ of x' from S is at most $\frac{\tau}{r'}$. Therefore a threshold change of τ to $\tau + d\tau$ corresponds to $\Delta \in \left(\frac{\tau}{r'}, \frac{\tau+d\tau}{r'}\right)$. We observe from the proof of Lemma C.1.1 that

$$\begin{aligned}\Pr\left[\Delta \in \left(\frac{\tau}{r'}, \frac{\tau + d\tau}{r'}\right)\right] &= C(n_2, n_3) \cdot \int_{\tau/r'}^{(\tau+d\tau)/r'} \rho^{n_2-n_3-1}\left(\sqrt{1-\rho^2}\right)^{n_3-2} d\rho \\ &\leq C(n_2, n_3) \cdot \frac{\tau^{n_2-n_3-1} d\tau}{r'^{n_2-n_3}},\end{aligned}$$

where $C(n_2, n_3) = 2A(n_3 - 1)A(n_2 - n_3 - 1)$ is a constant for fixed dimensions n_2, n_3. This holds for any test point $\mathbf{x} \in \mathcal{T}$, and in particular, in average over the test points. Using a union bound over training points we conclude,

$$\mathcal{E}_{\text{adv}}(\tau + d\tau) - \mathcal{E}_{\text{adv}}(\tau) \leq mC(n_2, n_3)\frac{\tau^{n_2-n_3-1} d\tau}{r'^{n_2-n_3}}.$$

The slope bound increases with τ, substituting $\tau \leq \tau_{\max}$ and $r' \geq r$ gives the desired bound on Lipschitzness. □

Corollary C.2.3. *For a $\tilde{\kappa}$-bounded adversary distribution $\mathcal{S}$ in Lemma C.2.2, we have that $\mathcal{E}_{\text{adv}}(\tau)$ is $O\left(\tilde{\kappa}m\tau_{\max}^{n_2-n_3-1}/r^{n_2-n_3}\right)$-Lipshcitz.*

Proof. The proof follows using the same arguments in the proof of Theorem 6.1.5 applied to Lemma C.2.2 (instead of our upper bounds on the robust error). □

Corollary C.2.4. *For S drawn from a $\tilde{\kappa}$-bounded adversary distribution $\mathcal{S}$, the expected number of discontinuities of $\mathcal{E}_{\text{adv}}(\tau, S)$ in any τ-interval of length w is $O\left(\tilde{\kappa}bmw\tau_{\max}^{n_2-n_3-1}/r^{n_2-n_3}\right)$.*

Proof. Consider the interval $[\tau, \tau + w]$. We are interested in bounding the probability that for a given test point $\mathbf{x}$, the smallest threshold τ' for which the adversary succeeds when perturbing along S (over the draw $S \sim \mathcal{S}$) lies in the interval $[\tau, \tau + w]$.

For a fixed training point $\mathbf{x}_i$, the probability of adversarial success on any $\mathbf{x} \in \mathcal{T}$ by perturbing to a point at distance $\tau' \in [\tau, \tau + w]$ from $\mathbf{x}_i$ is bounded by $O\left(\tilde{\kappa}w\tau_{\max}^{n_2-n_3-1}/r^{n_2-n_3}\right)$ as

argued above (Lemma C.2.2). Taking a union bound over training points $\mathbf{x}_i$ implies the adversary succeeds with probability at most $O\left(\tilde{\kappa} m w \tau_{\max}^{n_2-n_3-1}/r^{n_2-n_3}\right)$ by perturbing to within $[\tau, \tau+w]$ of some training point. Thus, for b test points the expected number of breakpoints is at most $O\left(\tilde{\kappa} b m w \tau_{\max}^{n_2-n_3-1}/r^{n_2-n_3}\right)$.

□

The following lemma gives a bound on the expected number of breakpoints in $\mathcal{D}_{\text{nat}}(\tau)$, a piecewise constant function in τ, in a small interval of width w.

Lemma C.2.5. *Suppose that the data distribution satisfies the assumptions in Lemma C.2.1, and further is κ-bounded. The expected number of discontinuties in $\mathcal{D}_{\text{nat}}(\tau)$ in any interval of width w for $\tau \leq \tau_{\max}$ is $O(\kappa b m w \tau_{\max}^{n_2-1})$.*

Proof. Note that the discontinuities of $\mathcal{D}_{\text{nat}}(\tau)$ in an interval $(\tau, \tau+w)$ corresponds to points $(\mathbf{x}, \mathbf{y}) \in T$ such that nearest neighbor distance of $\mathbf{x}$ is in that interval.

$$\begin{aligned}
&E[\text{number of discontinuities in } (\tau, \tau+w)] \\
&\qquad = b \Pr[\text{nearest neighbor of a test point } \in (\tau, \tau+w)] \\
&\qquad \leq b \Pr[\text{some neighbor of a test point } \in (\tau, \tau+w)] \\
&\qquad \leq \kappa b m \mathsf{vol}(\text{spherical shell of radius } \tau \text{ and width } w) \\
&\qquad = \kappa b m O(\tau_{\max}^{n_2-1} w) \\
&\qquad = O(\kappa b m w \tau_{\max}^{n_2-1}).
\end{aligned}$$

□

For the full proof of Theorem 6.3.2, we will need a low-regret bound for dispersed functions due to [15]. If the sequence of functions is dispersed, we can bound the regret of a simple exponential forecaster algorithm (Algorithm 11) by the following theorem.

Theorem C.2.6 ([15])**.** *Let $u_1, \ldots, u_T : C \to [0,1]$ be any sequence of piecewise L-Lipschitz functions that are (w,k)-dispersed. Suppose $C \subset \mathbb{R}^d$ is contained in a ball of radius R and $B(\rho^*, w) \subset C$, where $\rho^* = \operatorname{argmax}_{\rho \in C} \sum_{i=1}^{T} u_i(\rho)$. The exponentially weighted forecaster with $\lambda = \sqrt{d \ln(R/w)/T}$ has expected regret bounded by $O\left(\sqrt{Td \log(R/w)} + k + TLw\right)$.*

Appendix D

Multiple similar tasks

We include proofs and additional details from Chapter 7 here.

D.1 Lower Bound

We extend the construction in [18] to the multi-task setting. The main difference is that we generalize the construction for any task similarity, and show that we get the same lower bound asymptotically.

Theorem D.1.1. *There is a sequence of piecewise L-Lipschitz β-dispersed functions $\ell_{i,j} : [0,1] \mapsto [0,1]$, whose optimal actions in hindsight $\operatorname{argmin}_\rho \sum_{i=1}^m l_{t,i}(\rho)$ are contained in some fixed ball of diameter D^*, for which any algorithm has expected regret $R_m \geq \tilde{\Omega}(m^{1-\beta})$.*

Proof. Define $u^{(b,x)}(\rho) = \mathbb{I}[b=0] * \mathbb{I}[\rho > x] + \mathbb{I}[b=1] * \mathbb{I}[\rho \leq x]$, where $b \in \{0,1\}$, $x, \rho \in [0,1]$ and $I[\cdot]$ is the indicator function. For each iteration the adversary picks $u^{(0,x)}$ or $u^{(1,x)}$ with equal probability for some $x \in [a, a+D^*]$, the ball of diameter D^* containing all the optima.

For each task t, $m - \frac{3}{D^*} m^{1-\beta}$ functions are presented with the discontinuity $x \in [a + D^*/3, a + 2D^*/3]$ while ensuring β-dispersion. The remaining $\frac{3}{D^*} m^{1-\beta}$ are presented with discontinuities located in successively halved intervals (the 'halving adversary') containing the optima in hindsight, any algorithm gets half of these wrong in expectation. It is readily verified that the functions are β-dispersed. The construction works provided m is sufficiently large ($m > \left(\frac{3}{D^*}\right)^{1/\beta}$). The task averaged regret is therefore also $\tilde{\Omega}(m^{1-\beta})$. □

D.2 Robustness lower bound

Theorem D.2.1. *There exist sequences of piecewise L-Lipschitz functions $\tilde{l}_i, l_i, a_i\ [0,1] \to [0,1]$ for $i = 1, \ldots, m$ such that for any online algorithm*

1. *$\tilde{l}_i$ is β-dispersed and $\mathbb{E}[\tilde{R}_m] = \Omega(m^{1-\beta})$,*
2. *$\tilde{l}_i$ is β-dispersed, a_i is $m^{-\beta}$-bounded, β_a-dispersed and $\mathbb{E}[R_m] = \Omega(m^{1-\min\{\beta,\beta_a\}})$.*

Proof. Part 1 follows from the lower bound in Theorem D.1.1, by setting $\tilde{l}_i = l_i$ as the loss sequence used in the proof.

To establish Part 2, we extend the construction as follows. $\tilde{l}_i = l_i$ are both equal and correspond to the 'halving adversary' from the proof of Theorem D.1.1 for the first $\Theta(m^{1-\beta})$ rounds. If $\beta \leq \beta_a$ we are done, so assume otherwise. Let I denote the interval containing the optima over the rounds so far. Notice that the length of I is at most $|I| \leq (\frac{1}{2})^{\Theta(m^{1-\beta})} \leq (\frac{1}{2})^{\beta \log m} = m^{-\beta}$ for $\beta > 0$. For further rounds l_i continues to be the halving adversary for $\Theta(m^{1-\beta_a})$ rounds, which implies any algorithm suffers $\Omega(m^{1-\beta_a})$ regret. We set attack a_i on interval I such that $\tilde{l}_i = 0$ on I on these rounds. This ensures that a_i is β_a-dispersed and $\tilde{l}_i$ is β-dispersed. Putting together with the case $\beta \leq \beta_a$, we obtain $\Omega(m^{1-\min\{\beta,\beta_a\}})$ bound on the regret of any algorithm. □

D.3 Learning algorithmic parameters for combinatorial problems

We discuss implications of our results for several combinatorial problems of widespread interest including integer quadratic programming and auction mechanism design. We will need the following theorem from [10], which generalizes the recipe for establishing dispersion given by [19] for $d = 1, 2$ dimensions to arbitrary constant d dimendions. It is straightforward to apply the recipe to establish dispersion for these problems, which in turn implies that our meta-learning results are applicable. We demonstrate this for a few important problems below for completeness.

Theorem D.3.1 ([10]). *Let $l_1, \ldots, l_m : \mathbb{R}^d \to \mathbb{R}$ be independent piecewise L-Lipschitz functions, each having discontinuities specified by a collection of at most K algebraic hypersurfaces of bounded degree. Let $\mathcal{L}$ denote the set of axis-aligned paths between pairs of points in $\mathbb{R}^d$, and for each $s \in \mathcal{L}$ define $D(m, s) = |\{1 \leq t \leq m \mid l_t$ has a discontinuity along $s\}|$. Then we have $\mathbb{E}[\sup_{s\in\mathcal{L}} D(m, s)] \leq \sup_{s\in\mathcal{L}} \mathbb{E}[D(m, s)] + O(\sqrt{m \log(mK)})$.*

D.3.1 Greedy knapsack

We are given a knapsack with capacity $\texttt{cap}$ and items $i \in [m]$ with sizes w_i and values v_i. The goal is to select a subset S of items to add to the knapsack such that $\sum_{i\in S} w_i \leq \texttt{cap}$ while maximizing the total value $\sum_{i\in S} v_i$ of selected items. We consider a general greedy heuristic to insert items with largest v_i/w_i^ρ first (due to [84]) for $\rho \in [0, 10]$.

The classic greedy heuristic sets $\rho = 1$ and can be used to provide a 2-approximation for the problem. However other values of ρ can improve the knapsack objective on certain problem instances. For example, for the value-weight pairs $\{(0.99, 1), (0.99, 1), (1.01, 1.01)\}$ and capacity $\texttt{cap} = 2$ the classic heuristic $\rho = 1$ gives value 1.01 as the greedy heuristic is maximized for the third item. However, using $\rho = 3$ (or any $\rho > 1 + \log(1/0.99)/\log(1.01) > 2.01$) allows us to pack the two smaller items giving the optimal value 1.98.

Our result (Theorem 7.1.2) when applied to this problem shows that it is possible to learn the optimal parameter values for the greedy heuristic algorithm family for knapsack from similar tasks.

Theorem D.3.2. *Consider instances of the knapsack problem given by bounded weights $w_{i,j} \in [1, C]$ and κ-bounded independent values $v_{i,j} \in [0, 1]$ for $i \in [m], j \in [T]$. Then the asymptotic task-averaged regret for learning the algorithm parameter ρ for the greedy heuristic family described above is $o_T(1) + 2V\sqrt{m} + O(\sqrt{m})$.*

Proof. Lemma 11 of [19] shows that the loss functions form a $\frac{1}{2}$-dispersed sequence. The result follows by applying Theorem 7.1.2 with $\beta = \frac{1}{2}$. □

D.3.2 k-center clustering

We consider the α-Lloyd's clustering algorithm family from [16], where the initial k centers in the procedure are set by sampling points with probability proportional to d^α where d is the distance from the centers selected so far for some $\alpha \in [0, D], D \in \mathbb{R}_{\geq 0}$. For example, $\alpha = 0$ corresponds to the vanilla k-means with random initial centers, and $\alpha = 2$ setting is the k-means++ procedure. For this algorithm family, we are able to show the following guarantee. Interestingly, for this family it is sufficient to rely on the internal randomness of the algorithmic procedure and we do not need assumptions on data smoothness.

Theorem D.3.3. *Consider instances of the k-center clustering problem on n points, with Hamming loss $l_{i,j}$ for $i \in [m], j \in [T]$ against some (unknown) ground truth clustering. Then the asymptotic task-averaged regret for learning the algorithm parameter α for the α-Lloyd's clustering algorithm family of [16] is $o_T(1) + 2V\sqrt{m} + O(\sqrt{m})$.*

D.3.3 Integer quadratic programming (IQP)

The objective is to maximize a quadratic function $z^T Az$ for A with non-negative diagonal entries, subject to $z \in \{0, 1\}^n$. In the classic Goemans-Williamson algorithm [82] one solves an SDP relaxation $U^T AU$ where columns u_i of U are unit vectors. u_i are then rounded to $\{\pm 1\}$ by projecting on a vector Z drawn according to the standard Gaussian, and using $\mathtt{sgn}(\langle u_i, Z\rangle)$. A simple parametric family is s-linear rounding where the rounding is as before if $|\langle u_i, Z\rangle| > s$ but uses probabilistic rounding to round u_i to 1 with probability $\frac{1+(\langle u_i, Z\rangle)/s}{2}$. The dispersion analysis of the problem from [15] and the general recipe from [19] imply that our results yield low task-averaged regret for learning the parameter of the s-linear rounding algorithms.

Theorem D.3.4. *Consider instances of IQP given by matrices $A_{i,j}$ and rounding vectors $Z_{i,j} \sim \mathcal{N}_n$ for $i \in [m], j \in [T]$. Then the asymptotic task-averaged regret for learning the algorithm parameter s for s-linear rounding is $o_T(1) + 2V\sqrt{m} + O(\sqrt{m})$.*

Our results are an improvement over prior work which have only considered iid and (single-task) online learning settings. Similar improvements can be obtained for auction design, as described below. We illustrate this using a relatively simple auction, but the same idea applies for an extensive classes of auctions as studied in [17].

D.3.4 Posted price mechanisms with additive valuations

There are m items and n bidders with valuations $v_j(b_i), j \in [n], i \in [2^m]$ for all 2^m bundles of items. We consider additive valuations which satisfy $v_j(b) = \sum_{i \in b} v_j(\{i\})$. The objective is to maximize the social welfare (sum of buyer valuations). If the item values for each buyer have κ-bounded distributions, then the corresponding social welfare is dispersed and our results apply.

Theorem D.3.5. *Consider instances of posted price mechanism design problems with additive buyers and κ-bounded marginals of item valuations. Then the asymptotic task-averaged regret for learning the price which maximizes the social welfare is $o_T(1) + 2V\sqrt{m} + O(\sqrt{m})$.*

Proof. As noted in [15], the locations of discontinuities are along axis-parallel hyperplanes (buyer j will be willing to buy item i at a price p_i if and only if $v_j(\{i\}) \geq p_i$, each buyer-item pair in each instance corresponds to a hyperplane). Thus in any pair of points p, p' (corresponding to pricing) at distance ϵ, we have in expectation at most $\epsilon\kappa mn$ discontinuities along any axis-aligned path joining p, p', since discontinuities for an item can only occur along axis-aligned segment for the axis corresponding to the item. Theorem D.3.1 now implies $\frac{1}{2}$-dispersion. The task-averaged regret bound is now a simple application of Theorem 7.1.2. □

Appendix E

Adaptivity

We include proofs and additional details from Chapter 8 here.

E.1 Discretization based algorithm

Algorithm 22 Discrete Fixed Share Forecaster

Input: β, the dispersion parameter
[1.] Obtain a $T^{-\beta}$-discretization $\mathbf{D}$ of $\mathbf{C}$ (i.e. any $c \in \mathbf{C}$ is within $T^{-\beta}$ of some $d \in \mathbf{D}$)
[2.] Apply an optimal algorithm for finite experts with points in $\mathbf{D}$ as the experts (e.g. fixed share [94])

Recall that $\mathbf{C} \subset \mathbb{R}^d$ is contained in a ball of radius R. A standard greedy construction gives an r-discretization of size at most $(3R/r)^d$ [15]. Given the dispersion parameter β, a natural choice is to use a $T^{-\beta}$-discretization as in Algorithm 22.

Theorem E.1.1. *Let $R^{\textit{finite}}(T, s, N)$ denote the s-shifted regret for the finite experts problem on N experts, for the algorithm used in step 2 of Algorithm 22. Then Algorithm 22 enjoys s-shifted regret $R^{\mathbf{C}}(T, s)$ which satisfies*

$$R^{\mathbf{C}}(T, s) \leq R^{\textit{finite}}\left(T, s, \left(3RT^{\beta}\right)^{d}\right) + (sH + L)O(T^{1-\beta}).$$

Proof of Theorem E.1.1. We show we can round the optimal points in $\mathbf{C}$ to points in the $(T^{-\beta})$-discretization $\mathcal{D}$ with a payoff loss at most $(sH + L)T^{1-\beta}$ in expectation. But in $\mathcal{D}$ we know a way to bound regret by $R^{finite}(T, s, N)$, where N, the number of points in $\mathcal{D}$, is at most $\left(\frac{3R}{T^{-\beta}}\right)^{d} = \left(3RT^{\beta}\right)^{d}$.
Let $t_{0:s}$ denote the expert switching times in the optimal offline payoff, and ρ_i^* be the point picked by the optimal offline algorithm in $[t_{i-1}, t_i - 1]$. Consider a ball of radius $T^{-\beta}$ around ρ_i^*. It must have some point $\hat{\rho}_i^* \in \mathcal{D}$. We then must have that $\{u_t \mid t \in [t_{i-1}, t_i - 1]\}$ has at most $O(T^{-\beta}T) = O(T^{1-\beta})$ discontinuities due to β-dispersion, which implies

$$\sum_{t=t_{i-1}}^{t_i-1} u_t(\hat{\rho}_i^*) \geq \sum_{t=t_{i-1}}^{t_i-1} u_t(\rho_i^*) - O(T^{1-\beta})H - L(t_i - t_{i-1})T^{-\beta}.$$

Let $\hat{\rho}_t = \hat{\rho}_i^*$ for each $t_{i-1} \le t \le t_i - 1$. Summing over i gives

$$\sum_{t=1}^{T} u_t(\hat{\rho}_t) \ge OPT - O(T^{1-\beta})sH - LT^{1-\beta} = OPT - (sH + L)O(T^{1-\beta}).$$

Now payoff of this algorithm is bounded above by the payoff of the optimal sequence of experts with s shifts

$$\sum_{t=1}^{T} u_t(\hat{\rho}_t) \le OPT^{finite}.$$

Let the finite experts algorithm with shifted regret bounded by $R^{finite}(T, s, N)$ choose ρ_t at round t. Then, using the above inequalities,

$$\sum_{t=1}^{T} u_t(\rho_t) \ge OPT^{finite} - R^{finite}(T, s, N) \ge OPT - (sH + L)O(T^{1-\beta}) - R^{finite}(T, s, N).$$

We use this to bound the regret for the continuous case

$$\begin{aligned} R^{\mathbf{C}}(T, s) &= OPT - \sum_{t=1}^{T} u_t(\rho_t) \\ &\le OPT - (OPT - (sH + L)O(T^{1-\beta}) - R^{finite}(T, s, N)) \\ &= R^{finite}(T, s, N) + (sH + L)O(T^{1-\beta}). \end{aligned}$$

□

E.2 Counterexamples

We will construct problem instances where some sub-optimal algorithms mentioned in the paper suffer high regret.

We first show that the Exponential Forecaster algorithm of [15] suffers linear s-shifted regret even for $s = 2$. This happens because pure exponential updates may accumulate high weights on well-performing experts and may take a while to adjust weights when these experts suddenly start performing poorly.

Lemma E.2.1. *There exists an instance where Exponential Forecaster algorithm of [15] suffers linear s-shifted regret.*

Proof. Let $\mathbf{C} = [0, 1]$. Define utility functions

$$u^{(0)}(\rho) = \begin{cases} 1 & \text{if } \rho < \frac{1}{2} \\ 0 & \text{if } \rho \ge \frac{1}{2} \end{cases} \text{ and } u^{(1)}(\rho) = \begin{cases} 0 & \text{if } \rho < \frac{1}{2} \\ 1 & \text{if } \rho \ge \frac{1}{2} \end{cases}$$

Now consider the instance where $u^{(0)}(\rho)$ is presented for the first $T/2$ rounds and $u^{(1)}(\rho)$ is presented for the remaining rounds. In the second half, with probability at least $\frac{1}{2}$, the Exponential

Algorithm 23 Random Restarts Exponential Forecaster (Random Restarts EF)

Input: step size parameter $\lambda \in (0, 1/H]$, exploration parameter $\alpha \in [0, 1]$
[1.] $\hat{w}_1(\rho) = 1$ for all $\rho \in \mathbf{C}$
[2.] For each $t = 1, 2, \ldots, T$:
[i.] $\hat{W}_t := \int_{\mathbf{C}} \hat{w}_t(\rho) d\rho$
[ii.] Sample ρ with probability proportional to $\hat{w}_t(\rho)$, i.e. with probability $p_t(\rho) = \frac{\hat{w}_t(\rho)}{\hat{W}_t}$
[iii.] Sample z_t uniformly in $[0, 1]$ and set

$$\hat{w}_{t+1}(\rho) = \begin{cases} e^{\lambda u_t(\rho)} \hat{w}_t(\rho) & \text{if } z_t < 1 - \alpha \\ \frac{\int_{\mathbf{C}} e^{\lambda u_t(\rho)} \hat{w}_t(\rho) d\rho}{\text{VOL}(\mathbf{C})} & \text{otherwise} \end{cases}$$

Forecaster algorithm will select a point from $[0, \frac{1}{2}]$ and accumulate a regret of 1. Thus the expected 2-shifted regret of the algorithm is at least $\frac{T}{2} \cdot \frac{1}{2} = \Omega(T)$. Notice that the construction does not depend on the *step size parameter* λ. □

We further look at the performance of Random Restarts EF (Algorithm 23), an easy-to-implement algorithm which looks deceptively similar to Algorithm 12, against this adversary. Turns out Random Restarts EF may not restart frequently enough for the optimal value of the exploration parameter, and have sufficiently long chains of pure exponential updates in expectation to suffer high regret.

Theorem E.2.2. *There exists an instance where Random Restarts EF (Algorithm 23) with parameters λ and α as in Theorem 8.1.1 suffers linear s-shifted regret.*

Proof. The probability of pure exponential updates from $t = T/4$ through $t = 3T/4$ is at least

$$(1 - \alpha)^{T/2} = \left(1 - \frac{1}{T - 1}\right)^{T/2} > \frac{1}{2}$$

for $T > 5$. By Lemma E.2.1, this implies at least $\frac{T}{8}$ regret in this case, and so the expected regret of the algorithm is at least $\frac{T}{16} = \Omega(T)$. □

E.3 Analysis of algorithms

Lemma E.3.1 (Algorithm 12). *For each $t \in [T]$, $w_t(\rho) = \mathbb{E}[\hat{w}_t(\rho)]$ and $W_t = \mathbb{E}[\hat{W}_t]$, where the expectations are over random restarts $\mathbf{z}_t = \{z_1, \ldots, z_{t-1}\}$.*

Proof of Lemma E.3.1. $w_t(\rho) = \mathbb{E}[\hat{w}_t(\rho)]$ implies $W_t = \mathbb{E}[\hat{W}_t]$ by Fubini's theorem (recall $\mathbf{C}$ is closed and bounded). $w_t(\rho) = \mathbb{E}[\hat{w}_t(\rho)]$ follows by simple induction on t. In the base case, $\mathbf{z}_1$ is

the empty set and $w_1(\rho) = 1 = \hat{w}_t(\rho) = \mathbb{E}[\hat{w}_t(\rho)]$. For $t > 1$,

$$\begin{aligned}
&\mathbb{E}[\hat{w}_t(\rho)] \\
&= (1-\alpha)\mathbb{E}[e^{\lambda u_t(\rho)}\hat{w}_{t-1}(\rho)] + \frac{\alpha}{\text{VOL}(\mathbf{C})}\mathbb{E}\left[\int_{\mathbf{C}} e^{\lambda u_t(\rho)}\hat{w}_{t-1}(\rho)d\rho\right] && \text{(definition of } \hat{w}_t\text{)} \\
&= (1-\alpha)e^{\lambda u_t(\rho)}\mathbb{E}[\hat{w}_{t-1}(\rho)] + \frac{\alpha}{\text{VOL}(\mathbf{C})}\int_{\mathbf{C}} e^{\lambda u_t(\rho)}\mathbb{E}[\hat{w}_{t-1}(\rho)]d\rho && \text{(expectation is over } \mathbf{z}_t\text{)} \\
&= (1-\alpha)e^{\lambda u_t(\rho)}w_{t-1}(\rho) + \frac{\alpha}{\text{VOL}(\mathbf{C})}\int_{\mathbf{C}} e^{\lambda u_t(\rho)}w_{t-1}(\rho)d\rho && \text{(inductive hypothesis)} \\
&= w_t(\rho) && \text{(definition of } w_t\text{)}
\end{aligned}$$

□

Lemma E.3.2 (Algorithm 12). *W_{T+1} equals the sum*

$$\sum_{s\in[T]}\sum_{t_0=1<t_1\cdots<t_s=T+1}\frac{\alpha^{s-1}(1-\alpha)^{T-s}}{\text{VOL}(\mathbf{C})^{s-1}}\prod_{i=1}^{s}\tilde{W}(t_{i-1},t_i).$$

Proof of Lemma E.3.2. Recall that we wish to show that $\hat{w}_{T+1}(\rho) \mid s, \mathbf{t}_s$ (weights of Algorithm 23 at time $T+1$ given restarts occur exactly at $\mathbf{t}_s$) can be expressed as the product of weight $\tilde{w}(\rho; t_{s-1}, t_s)$ at ρ of regular Exponential Forecaster since the last restart times the normalized total weights accumulated over previous runs, i.e.

$$\hat{w}_{T+1}(\rho) \mid s, \mathbf{t}_s = \tilde{w}(\rho; t_{s-1}, t_s)\prod_{i=1}^{s-1}\frac{\tilde{W}(t_{i-1},t_i)}{\text{VOL}(\mathbf{C})}.$$

We show this by induction on s. For $s = 1$, we have no restarts and

$$\begin{aligned}
\tilde{w}(\rho; t_{s-1}, t_s)\prod_{i=1}^{s-1}\frac{\tilde{W}(t_{i-1},t_i)}{\text{VOL}(\mathbf{C})} &= \tilde{w}(\rho; t_0, t_1)\prod_{i=1}^{0}\frac{\tilde{W}(t_{i-1},t_i)}{\text{VOL}(\mathbf{C})} \\
&= \tilde{w}(\rho; 1, T+1) \\
&= \hat{w}_{T+1}(\rho) \mid 1, \mathbf{t}_1.
\end{aligned}$$

For $s > 1$, the last restart occurs at $t_{s-1} > 1$. By inductive hypothesis for time $t_{s-1} - 1$ until which we've had $s-2$ restarts,

$$\hat{w}_{t_{s-1}-1}(\rho) \mid s, \mathbf{t}_s = \hat{w}_{t_{s-1}-1}(\rho) \mid s-1, \mathbf{t}_{s-1} = \tilde{w}(\rho; t_{s-2}, t_{s-1}-1)\prod_{i=1}^{s-2}\frac{\tilde{W}(t_{i-1},t_i)}{\text{VOL}(\mathbf{C})}.$$

Due to restart at t_{s-1},

$$\hat{w}_{t_{s-1}}(\rho) \mid s, \mathbf{t}_s = \frac{\int_{\mathbf{C}} e^{\lambda u_t(\rho)}\hat{w}_{t_{s-1}-1}(\rho)d\rho}{\text{VOL}(\mathbf{C})} = \prod_{i=1}^{s-1}\frac{\tilde{W}(t_{i-1},t_i)}{\text{VOL}(\mathbf{C})}.$$

It's regular exponential updates from this point to t_s, which gives the result.

□

Lemma E.3.3 (Algorithm 12)**.** $W_{t+1} = \int_{\mathbf{C}} e^{\lambda u_t(\rho)} w_t(\rho) d\rho$.

The rest of this section is concerned with the analysis of Algorithm 14 for the sparse experts setting.

Lemma E.3.4. *For any $t < T$,*

$$w_T(\rho) \geq \alpha(1-\alpha)^{T-t} \pi_t(\rho) \tilde{w}(\rho; t, T) W_t$$

Proof. Follows using the restart algorithm technique used in Lemma E.3.2. Consider the probability of last *restart* being at time t. □

Lemma E.3.5. *Let $\pi_t(\rho) = \sum_{i=1}^{t} \beta_{i,t} p_i(\rho)$ in Algorithm 14. Then*

$$\pi_t(\rho) = \frac{\alpha_{1,t}}{W_1} + \sum_{i=1}^{t-1} \alpha_{i+1,t} \frac{e^{\lambda u_i(\rho)} w_i(\rho)}{W_{i+1}},$$

where

$$\alpha_{i,t} \geq \frac{1-\alpha}{e_t} \left(e^{-\gamma} + \frac{\alpha}{e_t} \right)^{t-i},$$

and $e_t := \sum_{i=1}^{t} e^{-\gamma(i-1)}$.

Proof. Notice, by definition of weight update in Algorithm 14,

$$\begin{aligned} p_t(\rho) &= (1-\alpha) \frac{e^{\lambda u_{t-1}(\rho)} w_{t-1}(\rho)}{W_t} + \alpha \sum_{i=1}^{t-1} \beta_{i,t-1} p_i(\rho) \\ &= (1-\alpha) \frac{e^{\lambda u_{t-1}(\rho)} w_{t-1}(\rho)}{W_t} + \alpha \pi_{t-1}(\rho). \end{aligned}$$

This gives us a recursive relation for $\alpha_{i,t}$.

$$\alpha_{i,t} = \begin{cases} \beta_{i,t}(1-\alpha) + \alpha \sum_{j=i+1}^{t} \beta_{j,t} \alpha_{i,j-1} & \text{, if } i > 1, \\ \beta_{i,t} + \alpha \sum_{j=i+1}^{t} \beta_{j,t} \alpha_{i,j-1} & \text{, if } i = 1. \end{cases}$$

Thus for each $1 \leq i \leq t$

$$\alpha_{i,t} \geq \beta_{i,t}(1-\alpha) + \alpha \sum_{j=i+1}^{t} \beta_{j,t} \alpha_{i,j-1}.$$

We proceed by induction on $t - i$. For $i = t$,

$$\alpha_{t,t} \geq \beta_{t,t}(1-\alpha) = \frac{1-\alpha}{e_t} \left(e^{-\gamma} + \frac{\alpha}{e_t} \right)^{t-t}.$$

For $i < t$, by inductive hypothesis

$$\begin{aligned}
\alpha_{i,t} &\geq \beta_{i,t}(1-\alpha) + \sum_{j=i+1}^{t} \beta_{j,t}\alpha\alpha_{i,j-1} \\
&\geq (1-\alpha)\frac{e^{-\gamma(t-i)}}{e_t} + \alpha\frac{1-\alpha}{e_t}\sum_{j=i+1}^{t}\frac{e^{-\gamma(t-j)}}{e_t}\left(e^{-\gamma}+\frac{\alpha}{e_t}\right)^{j-1-i} \\
&= \frac{1-\alpha}{e_t}\left(e^{-\gamma(t-i)} + \frac{\alpha e^{-\gamma t}}{e_t}\sum_{j=i+1}^{t} e^{\gamma j}\left(e^{-\gamma}+\frac{\alpha}{e_t}\right)^{j-1-i}\right) \\
&= \frac{1-\alpha}{e_t}\left(e^{-\gamma(t-i)} + \frac{\alpha e^{-\gamma t}}{e_t}\frac{e^{\gamma(t+1)}\left(e^{-\gamma}+\frac{\alpha}{e_t}\right)^{t-i} - e^{\gamma(i+1)}}{e^{\gamma}\left(e^{-\gamma}+\frac{\alpha}{e_t}\right)-1}\right) \\
&= \frac{1-\alpha}{e_t}\left(e^{-\gamma}+\frac{\alpha}{e_t}\right)^{t-i},
\end{aligned}$$

which completes the induction step. □

Lemma E.3.6. *Let $\pi_t(\rho) = \sum_{i=1}^{t-1}\beta_{i,t}p_i(\rho)$. For Algorithm 14, W_{T+1} can be shown to be equal to the sum*

$$\sum_{s\in[T]}\sum_{t_0=1<\ldots t_s=T+1}\alpha^{s-1}(1-\alpha)^{T-s}\prod_{i=1}^{s}\tilde{W}(\pi_{t_{i-1}};t_{i-1},t_i),$$

where $\tilde{W}(p;\tau,\tau') := \int_{\mathbf{C}} p(\rho)\tilde{w}(\rho;\tau,\tau')d\rho$.

Corollary E.3.7. *Let $w_t(\rho), W_t$ be as in Algorithm 14 and π_t as in Lemma E.3.6. For each $\tau < \tau' < t$ and any bounded f defined on $\mathbf{C}$.*

$$\int_{\mathbf{C}}\pi_t(\rho)f(\rho)d\rho \geq \frac{\alpha(1-\alpha)^{\tau'-\tau}(1-e^{-\gamma})}{(e^{-\gamma}+\alpha(1-e^{-\gamma}))^{\tau'-t}}\frac{W_\tau}{W_{\tau'}}\int_{\mathbf{C}}\pi_\tau(\rho)\tilde{w}(\rho;\tau,\tau')f(\rho)d\rho.$$

Proof. By Lemma E.3.5,

$$\begin{aligned}
\int_{\mathbf{C}}\pi_t(\rho)f(\rho)d\rho &= \int_{\mathbf{C}}\pi_t(\rho)f(\rho)d\rho \\
&\geq \int_{\mathbf{C}}\alpha_{\tau',t}\frac{e^{\lambda u_{\tau'-1}(\rho)}w_{\tau'-1}(\rho)}{W_{\tau'}}f(\rho)d\rho \\
&\geq \frac{1-\alpha}{e_t}\left(e^{-\gamma}+\frac{\alpha}{e_t}\right)^{t-\tau'}\frac{1}{W_{\tau'}}\int_{\mathbf{C}}e^{\lambda u_{\tau'-1}(\rho)}w_{\tau'-1}(\rho)f(\rho)d\rho \\
&\geq \frac{1-\alpha}{e_t}\left(e^{-\gamma}+\frac{\alpha}{e_t}\right)^{t-\tau'}\frac{\alpha(1-\alpha)^{\tau'-1-\tau}W_\tau}{W_{\tau'}}\int_{\mathbf{C}}\pi_\tau(\rho)\tilde{w}(\rho;\tau,\tau')f(\rho)d\rho,
\end{aligned}$$

where for the last inequality we have used Lemma E.3.4. The lemma then follows by noting

$$\frac{1}{e_t} = \frac{1-e^{-\gamma}}{1-e^{-\gamma t}} \geq 1 - e^{-\gamma},$$

where $e_t = \sum_{i=1}^{t} e^{-\gamma(i-1)}$ as defined in Lemma E.3.5. □

Lemma E.3.8 (Algorithm 14)**.** $W_{t+1} = \int_{\mathbf{C}} e^{\lambda u_t(\rho)} w_t(\rho) d\rho$.

Lemma E.3.9. *Let $\pi_t(\rho) = \sum_{i=1}^{t-1} \beta_{i,t} p_i(\rho)$. For Algorithm 14, W_{T+1} can be shown to be equal to the sum*

$$\sum_{s \in [T]} \sum_{t_0 = 1 < \ldots t_s = T+1} \alpha^{s-1} (1-\alpha)^{T-s} \prod_{i=1}^{s} \tilde{W}(\pi_{t_{i-1}}; t_{i-1}, t_i)$$

where $\tilde{W}(p; \tau, \tau') := \int_{\mathbf{C}} p(\rho) \tilde{w}(\rho; \tau, \tau') d\rho$.

Corollary E.3.10. $W_T \geq \alpha(1-\alpha)^{T-t} \tilde{W}(\pi_t; t, T) W_t$*, for all $t < T$.*

Proof. Consider the probability of last *reset* (setting $w_t(\rho) = W_t \pi_t(\rho)$) at time t when computing W_{T+1} as the expected weight of a random restart version which matches Algorithm 14 till time t. □

E.4 Adaptive Regret

It is known that the fixed share algorithm obtains good adaptive regret for finite experts and OCO. We show that it is the case here as well.

Definition 39. *The τ-adaptive regret (due to [93]) is given by*

$$\mathbb{E}\left[\max_{\substack{\rho^* \in \mathbf{C}, \\ 1 \leq r < s \leq T, s-r \leq \tau}} \sum_{t=r}^{s} (u_t(\rho^*) - u_t(\rho_t)) \right].$$

The goal here is to ensure small regret on all intervals of size up to τ simultaneously. Adaptive regret measures how well the algorithm approximates the best expert locally, and it is therefore somewhere between the static regret (measured on all outcomes) and the shifted regret, where the algorithm is compared to a good sequence of experts.

Theorem E.4.1. *Algorithm 12 enjoys $O(H\sqrt{\tau(d\log(R/w) + \log\tau)} + (H+L)\tau^{1-\beta})$ τ-adaptive regret for $\lambda = \sqrt{(d\log(R\tau^{\beta}) + \log(\tau))/\tau}/H$ and $\alpha = 1/\tau$.*

Proof sketch of Theorem E.4.1. Apply arguments of Theorem 8.1.1 to upper and lower bound W_{s+1}/W_r for any interval $[r, s] \subseteq [1, T]$ of size τ. We get

$$\frac{W_{s+1}}{W_r} \leq \exp\left(\frac{P(\mathcal{A})(e^{H\lambda} - 1),}{H} \right),$$

where $P(\mathbf{A})$ is the expected payoff of the algorithm in $[r, s]$, Also, by Corollary E.3.10 (equivalent for Algorithm 12),

$$W_{s+1} \geq \frac{\alpha(1-\alpha)^{s+1-r}}{\text{VOL}(\mathbf{C})}\tilde{W}(r,s)W_r = \frac{\alpha(1-\alpha)^{\tau}}{\text{VOL}(\mathbf{C})}\tilde{W}(r,s)W_r.$$

By dispersion, as in proof of Theorem 8.1.1,

$$\tilde{W}(r,s) \geq \text{VOL}(\mathbf{B}(\tau^{-\beta}))\exp\left(\lambda\left(OPT - (H+L)O(\tau^{1-\beta})\right)\right).$$

Putting the upper and lower bounds together gives us a bound on $OPT - P(\mathbf{A})$, which gives the desired regret bound for $\alpha = \frac{1}{\tau}$. □

E.5 Efficient Sampling

In Section 8.2 we introduced Algorithm 13 for efficient implementation of Algorithm 12 in $\mathbb{R}^d$. We present proofs of the results in that section, and an exact algorithm for the case $d = 1$.

Algorithm 24 Fixed Share Exponential Forecaster - exact algorithm for one dimension

Input: $\lambda \in (0, 1/H]$.
[1.] $W_1 = \text{VOL}(\mathbf{C})$.
[2.] For each $t = 1, 2, \ldots, T$:
[i.] Estimate $C_{t,j}$ using Lemma 8.2.3 for each $1 \leq j \leq t$ using memoized values for weights.
[ii.] Sample i with probability $C_{t,i}$.
[iii.] Sample ρ with probability proportional to $\tilde{w}(\rho; i, t)$.
[iv.] Estimate W_{t+1} using Lemma 8.2.2.

Definition 40. *For $\alpha \geq 0$ we say $\hat{A}$ is an (α, ζ)-approximation of A if*

$$Pr\left(e^{-\alpha}A \leq \hat{A} \leq e^{\alpha}A\right) \geq 1 - \zeta.$$

Lemma E.5.1. *If $\hat{A}$ is an (α, ζ)-approximation of A and $\hat{B}$ is a (β, ζ')-approximation of B, such that $A, B, \hat{A}, \hat{B}$ are all positive reals*

1. *$\hat{A}\hat{B}$ is an $(\alpha + \beta, \zeta + \zeta')$-approximation of AB.*
2. *$p\hat{A} + q\hat{B}$ is a $(\max\{\alpha, \beta\}, \zeta + \zeta')$-approximation of $pA + qB$ for $p, q \geq 0$.*

Proof. The results follow from a union bound on failure probabilities. □

Corollary E.5.2. *For one-dimensional case, we can exactly compute $\tilde{W}(i,j),\ 1 \leq i < j \leq t$, hence W_t at each iteration can be computed in $O(t)$ time using Lemma 8.2.2. More generally, if we have a (β, ζ) approximation for each $\tilde{W}(i,j),\ 1 \leq i < j \leq t$, then by Lemma 8.2.2 we can compute a $(t\beta, t^2\zeta)$-approximation for W_{t+1}.*

Proof. Union bound on failure probabilities of all $\tilde{W}(i,j),\ 1 \le i < j \le t$ gives we have a β approximation for each with probability at least $1 - t^2\zeta$. This covers failure for all terms in $W_i, 2 \le i \le t$. Further, by induction, the error for estimates for W_i is at most $(i-1)\beta$. By Lemma E.5.1, the error for W_{t+1} estimates is at most $t\beta$. □

Corollary E.5.3. *If we have a (β, ζ) approximation for each $\tilde{W}(i,j),\ 1 \le i < j \le t$, then by Corollary E.5.2 and Lemma 8.2.3 we can compute $\hat{C}_{t+1,i}$ which are $(2t\beta, t^2\zeta)$-approximation for each $C_{t+1,i}$.*

Proof. For $i = t$, we know $C_{t,i}$ exactly by Lemma 8.2.3. For $i < t$,

$$C_{t,i} = (1-\alpha)^{t-i} \frac{W_i}{W_t} \frac{\tilde{W}(i,t)}{\textsc{Vol}(\mathbf{C})} C_{i,i} \tag{E.1}$$

In Corollary E.5.2, we show how to compute $((i-1)\beta, (i-1)^2\zeta)$-approximation for W_i and $((t-1)\beta, (t-1)^2\zeta)$-approximation for W_t given (β, ζ) approximations for each $\tilde{W}(i,j),\ 1 \le i < j \le t$. A similar argument using Lemma E.5.1 shows with failure probability at most $t^2\zeta$, plugging in the approximations in equation E.1 has at most $(t+i)\beta$ error. □

E.6 Lower bounds

We start with a simple lower bound argument for s-shifted regret for prediction with two experts based on a well-known $\Omega(\sqrt{T})$ lower bound argument for static regret. We will then extend it to the continuous setting and use it for the $\Omega(\sqrt{sT})$ part of the lower bound.

Lemma E.6.1. *For prediction with two experts, there exists a stochastic sequence of losses for which the s-shifted regret of any online learning algorithm satisfies*

$$\mathbb{E}[R_T] \ge \sqrt{sT/8}.$$

Proof. Let the two experts predict 0 and 1 respectively at each time $t \in [T]$. The utility at each time t is computed by flipping a coin - with probability $1/2$ we have $u(0) = 1, u(1) = 0$ and with probability $1/2$ it's $u(0) = 0, u(1) = 1$. Expected payoff for any algorithm $\mathbf{A}$ is

$$P(\mathbf{A}, T) = \mathbb{E}\Big[\sum_{t=1}^{T} u_t(\rho_t)\Big] = \sum_{t=1}^{T} \mathbb{E}[u_t(\rho_t)] = \frac{T}{2},$$

since expected payoff is $1/2$ at each t no matter which expert is picked.
To compute shifted regret we need to compare this payoff with the best sequence of experts with $s-1$ switches. We compare with a weaker adversary $\mathbf{A}'$ which is only allowed to switch *up to*

$s-1$ times, and switches at only a subset of fixed times $t_i = iT/s$ to lower bound the regret.

$$\begin{aligned}\mathbb{E}[R_T] &= OPT - P(\mathbf{A}, T)\\ &\geq P(\mathbf{A}', T) - P(\mathbf{A}, T)\\ &= \sum_{t=1}^{T} \mathbb{E}[u_t(\rho_t')] - \sum_{t=1}^{T} \mathbb{E}[u_t(\rho_t)]\\ &= \sum_{i=0}^{s-1} \sum_{t=t_i+1}^{t_{i+1}} \mathbb{E}[u_t(\rho_t')] - \mathbb{E}[u_t(\rho_t)].\end{aligned}$$

Now let $P_{i,j} = \sum_{t=t_i+1}^{t_{i+1}} \mathbb{E}[u_t(j)]$ for $i+1 \in [s]$ and $j \in \{0,1\}$.

$$\begin{aligned}\sum_{t=t_i+1}^{t_{i+1}} \mathbb{E}[u_t(\rho_t')] &= \max_{\rho\in\{0,1\}} \sum_{t=t_i+1}^{t_{i+1}} \mathbb{E}[u_t(\rho)]\\ &= \frac{1}{2}\Big[P_{i,0} + P_{i,1} + |P_{i,0} - P_{i,1}|\Big]\\ &= \frac{T}{2s} + |P_{i,0} - T/2s|,\end{aligned}$$

using $P_{i,0} + P_{i,1} = T/s$. Thus,

$$\mathbb{E}[R_T] \geq \sum_{i=0}^{s-1} \Big[\Big(\frac{T}{2s} + |P_{i,0} - T/2s|\Big) - \frac{T}{2s}\Big] = \sum_{i=0}^{s-1} |P_{i,0} - T/2s|.$$

Noting $P_{i,0} = \sum_{t=t_i+1}^{t_{i+1}} \mathbb{E}[u_t(0)] = \sum_{t=t_i+1}^{t_{i+1}} \left(\frac{1+\sigma_t}{2}\right)$ where σ_t are Rademacher variables over $\{-1,1\}$ and applying Khintchine's inequality (see for example [37]) we get

$$\mathbb{E}[R_T] \geq \sum_{i=0}^{s-1} \left| \sum_{t=t_i+1}^{t_{i+1}} \sigma_t/2 \right| \geq \sum_{i=0}^{s-1} \sqrt{T/8s} = \sqrt{sT/8}.$$

□

Corollary E.6.2. *We can embed the two-expert setting to get a lower bound for the continuous case.*

Proof. Indeed in Lemma E.6.1 let $\mathbf{C} = [0,1]$, expert 0 correspond to $\rho_1 = 1/4$, expert 1 corresponds to $\rho_2 = 3/4$ and replace the loss functions by

$$u^{(0)}(\rho) = \begin{cases} 1 & \text{if } \rho < \frac{1}{2},\\ 0 & \text{if } \rho \geq \frac{1}{2},\end{cases} \text{ and } u^{(1)}(\rho) = \begin{cases} 0 & \text{if } \rho < \frac{1}{2},\\ 1 & \text{if } \rho \geq \frac{1}{2}.\end{cases}$$

We can further generalize this while dispersing the discontinuities somewhat. Instead of having all the discontinuties at $\rho = \frac{1}{2}$, we can have discontinuities dispersed say within an interval $[\frac{1}{3}, \frac{2}{3}]$ and still have $\Omega(\sqrt{sT})$ regret. □

Bibliography

[1] Hirotugu Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, 1974. 3.1

[2] Noga Alon, Nicolo Cesa-Bianchi, Claudio Gentile, Shie Mannor, Yishay Mansour, and Ohad Shamir. Nonstochastic multi-armed bandits with graph-structured feedback. *SIAM Journal on Computing*, 46(6):1785–1826, 2017. 2.5

[3] Doug Altner and Ozlem Ergun. Rapidly solving an online sequence of maximum flow problems. *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pages 283–287, 2008. 2.3.3

[4] Martin Anthony and Peter Bartlett. *Neural network learning: Theoretical foundations*. Cambridge University Press, 1999. 3.3, 5.4.1, 5.4.1, 5.4.2, 38, A.3.1, B.4

[5] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009. 5.3.2

[6] O. Axelsson. A class of iterative methods for finite element equations. *Computer Methods in Applied Mechanics and Engineering*, 9(2):123–137, 1976. ISSN 0045-7825. doi: https://doi.org/10.1016/0045-7825(76)90056-6. URL `https://www.sciencedirect.com/science/article/pii/0045782576900566`. 2.6.3, 2.6.3

[7] Maria-Florina Balcan. Data-Driven Algorithm Design (book chapter). In *Beyond Worst Case Analysis of Algorithms, Tim Roughgarden (Ed)*. Cambridge University Press, 2020. 1, 1.3, 1.4, 2, 2.5, 2.6.1, 2.6.1, 3, 3.3, 4.3, 5.4.1, 5.4.1, 5.4.2, 5.4.2, A.3.2, B.4

[8] Maria-Florina Balcan and Hedyeh Beyhaghi. Learning revenue maximizing menus of lotteries and two-part tariffs. *Transactions on Machine Learning Research (TMLR)*, 2024. 6, 9.3

[9] Maria-Florina Balcan and Avrim Blum. A discriminative model for semi-supervised learning. *Journal of the ACM (JACM)*, 57(3):1–46, 2010. 2

[10] Maria-Florina Balcan and Dravyansh Sharma. Data driven semi-supervised learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 34:14782–14794, 2021. 1, 1.3, 1.4, 1.4, 1.4, 2.5, 2.6.1, 2.6.1, 2.6.1, 2.6.2, 2.6.2, 2.6.3, 2.7, 8.1, A.2.2, A.2.2, D.3, D.3.1

[11] Maria-Florina Balcan and Dravyansh Sharma. Learning accurate and interpretable decision trees. *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2024. 1.4, 1.4

[12] Maria-Florina Balcan, Avrim Blum, Patrick Pakyan Choi, John Lafferty, Brian Pantano,

Mugizi Robert Rwebangira, and Xiaojin Zhu. Person identification in webcam images: An application of semi-supervised learning. In *ICML 2005 Workshop on Learning with Partially Classified Training Data*, volume 2, page 6, 2005. 2.3.1

[13] Maria-Florina Balcan, Vaishnavh Nagarajan, Ellen Vitercik, and Colin White. Learning-theoretic foundations of algorithm configuration for combinatorial partitioning problems. In *Conference on Learning Theory (COLT)*, pages 213–274, 2017. 1, 1.3, 2, 5.4, 8

[14] Maria-Florina Balcan, Travis Dick, Tuomas Sandholm, and Ellen Vitercik. Learning to branch. In *International Conference on Machine Learning (ICML)*, pages 344–353. PMLR, 2018. 1.3

[15] Maria-Florina Balcan, Travis Dick, and Ellen Vitercik. Dispersion for data-driven algorithm design, online learning, and private optimization. In *Annual Symposium on Foundations of Computer Science (FOCS)*, pages 603–614. IEEE, 2018. 1, 1.2, 2, 1.3, 2.3, 2.3.1, 2.3.1, 2.4.3, 3.2.3, 3.2.3, 5.4, 5.4.3, 21, 11, 6.3, 6.3, 7, 7.1, 7.1.1, 7.1, 7.2.1, 7.2.1, 7.2.2, 8.1, 8.1, 8.1, 8.1, 8.1, 8.2, 8.2, 9.2.2, 9.2.2, C.2, C.2.6, D.3.3, D.3.4, E.1, E.2, E.2.1

[16] Maria-Florina Balcan, Travis Dick, and Colin White. Data-driven clustering via parameterized Lloyd's families. *Advances in Neural Information Processing Systems (NeurIPS)*, 31, 2018. 1, 1.3, 7.2.1, 7.2.1, D.3.2, D.3.3

[17] Maria-Florina Balcan, Tuomas Sandholm, and Ellen Vitercik. A general theory of sample complexity for multi-item profit maximization. In *Proceedings of the 2018 ACM Conference on Economics and Computation (EC)*, pages 173–174, 2018. 1.3, 9.2.1, 9.3, 9.3.1, 9.3.3, 9.3.3, B.4, B.4, D.3.3

[18] Maria-Florina Balcan, Travis Dick, and Manuel Lang. Learning to link. In *International Conference on Learning Representations (ICLR)*, 2020. 1.3, 2.3.1, 7.2.2, 9.2.1, 9.4, 9.4, 4, 32, 9.4.2, D.1

[19] Maria-Florina Balcan, Travis Dick, and Wesley Pegden. Semi-bandit optimization in the dispersed setting. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 909–918. PMLR, 2020. 1.3, 1.4, 2.2, 2.2, 2.2.1, 2.2.1, 2.2.2, 2.2.2, 2.3.1, 2.3.1, 2.3.2, 2.5, 2.5, 3.2.3, 5.4.3, 21, 5.4.3, 6.3, 7.2.1, 7.2.1, 9.2.2, A.1, A.1, A.1.1, A.1.2, A.2.2, A.2.4, A.2.5, A.2.2, B.4, D.3, D.3.1, D.3.3

[20] Maria-Florina Balcan, Travis Dick, and Dravyansh Sharma. Learning piecewise Lipschitz functions in changing environments. In *Artificial Intelligence and Statistics (AISTATS)*, pages 3567–3577, 2020. 1.2, 1.3, 1.4, 2.3, 2.3.1, 5.4.3, 7.2.1, 8.3, 8.3

[21] Maria-Florina Balcan, Siddharth Prasad, and Tuomas Sandholm. Efficient algorithms for learning revenue-maximizing two-part tariffs. In *International Joint Conferences on Artificial Intelligence (IJCAI)*, pages 332–338, 2020. 9.3

[22] Maria-Florina Balcan, Tuomas Sandholm, and Ellen Vitercik. Refined bounds for algorithm configuration: The knife-edge of dual class approximability. In *International Conference on Machine Learning (ICML)*, pages 580–590. PMLR, 2020. 2.5

[23] Maria-Florina Balcan, Dan DeBlasio, Travis Dick, Carl Kingsford, Tuomas Sandholm, and Ellen Vitercik. How much data is sufficient to learn high-performing algorithms?

generalization guarantees for data-driven algorithm design. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 919–932, 2021. 1.3, 14, 3.1.1, 3.1.1, 9.2.1, 9.5.2, 10.1

[24] Maria-Florina Balcan, Mikhail Khodak, Dravyansh Sharma, and Ameet Talwalkar. Learning-to-learn non-convex piecewise-Lipschitz functions. *Advances in Neural Information Processing Systems (NeurIPS)*, pages 15056–15069, 2021. 1.3, 1.4, 1.4, 6.3, 7.2.2

[25] Maria-Florina Balcan, Siddharth Prasad, Tuomas Sandholm, and Ellen Vitercik. Sample complexity of tree search configuration: Cutting planes and beyond. *Advances in Neural Information Processing Systems*, 34:4015–4027, 2021. 1.3

[26] Maria-Florina Balcan, Tuomas Sandholm, and Ellen Vitercik. Generalization in portfolio-based algorithm selection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 12225–12232, 2021. 1.3

[27] Maria-Florina Balcan, Avrim Blum, Steve Hanneke, and Dravyansh Sharma. Robustly-reliable learners under poisoning attacks. In *Conference on Learning Theory (COLT)*, pages 4498–4534. PMLR, 2022. 10.2

[28] Maria-Florina Balcan, Mikhail Khodak, Dravyansh Sharma, and Ameet Talwalkar. Provably tuning the ElasticNet across instances. *Advances in Neural Information Processing Systems (NeurIPS)*, 35:27769–27782, 2022. 1.3, 1.4, 3.1.1, 3.3, 4.3, 8.1

[29] Maria-Florina Balcan, Siddharth Prasad, Tuomas Sandholm, and Ellen Vitercik. Structural analysis of branch-and-cut and the learnability of Gomory mixed integer cuts. In *Advances in Neural Information Processing Systems*, 2022. 1.3

[30] Maria-Florina Balcan, Avrim Blum, Dravyansh Sharma, and Hongyang Zhang. An analysis of robustness of non-lipschitz networks. *Journal of Machine Learning Research (JMLR)*, 24, 2023. 1.3, 1.4, 1.4, 6.1, 6.2, 6.4, 6.4

[31] Maria-Florina Balcan, Anh Nguyen, and Dravyansh Sharma. New bounds for hyperparameter tuning of regression problems across instances. *Advances in Neural Information Processing Systems (NeurIPS)*, 36, 2023. 1.4, 3.3, 4.3

[32] Maria-Florina Balcan, Matteo Pozzi, and Dravyansh Sharma. Subsidy for repair in component maintenance games. *Engineering Mechanics Institute Conference and Probabilistic Mechanics and Reliability Conference (EMI/PMC)*, 2024. 1.4

[33] Maria-Florina Balcan, Christopher Seiler, and Dravyansh Sharma. Output-sensitive erm-based techniques for data-driven algorithm design. *Association for the Advancement of Artificial Intelligence (AAAI) Workshop on Learnable Optimization (LEANOPT)*, 36, 2024. 1.4, 9.5.3

[34] Peter Bartlett and Shahar Mendelson. Rademacher and Gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research (JMLR)*, 3(Nov): 463–482, 2002. 11

[35] Peter Bartlett, Piotr Indyk, and Tal Wagner. Generalization bounds for data-driven numerical linear algebra. In *Conference on Learning Theory (COLT)*, pages 2013–2040. PMLR, 2022. 1.3, 3.1.1, 3.1.1, 4.4, A.3, A.3.3

[36] Raef Bassily, Adam Smith, and Abhradeep Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *Foundations of Computer Science (FOCS)*, pages 464–473. IEEE, 2014. 8.2

[37] Shai Ben-David, Dávid Pál, and Shai Shalev-Shwartz. Agnostic online learning. In *Conference on Learning Theory (COLT)*, 2009. E.6

[38] John Bennett and Sarah Jayes. *Trusting the team: the best practice guide to partnering in construction*. Thomas Telford, 1995. 5.1

[39] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research (JMLR)*, 13(2), 2012. 1.3

[40] Avrim Blum and Shuchi Chawla. Learning from labeled and unlabeled data using graph mincuts. In *International Conference on Machine Learning (ICML)*, 2001. **??**, 2.1, 2.4.2, 2.6.1

[41] Avrim Blum, Chen Dan, and Saeed Seddighin. Learning complexity of simulated annealing. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1540–1548. PMLR, 2021. 1.3, 1.4, 10.1

[42] Marko Bohanec and Ivan Bratko. Trading accuracy for simplicity in decision trees. *Machine Learning*, 15:223–250, 1994. 4.4

[43] Olivier Bousquet and Manfred Warmuth. Tracking a small set of experts by mixing past posteriors. *Journal of Machine Learning Research (JMLR)*, 3(Nov):363–396, 2002. 29

[44] Olivier Bousquet and Nikita Zhivotovskiy. Fast classification rates without standard margin assumptions. *Information and Inference: A Journal of the IMA*, 10(4):1389–1421, 2021. 10.2

[45] Leo Breiman, Jerome Friedman, Richard Olshen, and Charles Stone. *Classification and Regression Trees*. Wadsworth International Group, 1984. 4.1, 4.3.1, 4.4, 4.4

[46] David Bremner, Komei Fukuda, and Ambros Marzetta. Primal-dual methods for vertex and facet enumeration (preliminary version). In *Symposium on Computational Geometry (SoCG)*, pages 49–56, 1997. 9.2.3

[47] Mike Bresnen and Nick Marshall. Partnering in construction: a critical review of issues, problems and dilemmas. *Construction Management and Economics*, 18(2):229–237, 2000. 5.1

[48] Nader Bshouty, Yi Li, and Philip Long. Using the doubling dimension to analyze the generalization of learning algorithms. *Journal of Computer and System Sciences*, 75(6): 323–335, 2009. 6.2

[49] Niv Buchbinder, Liane Lewin-Eytan, Joseph Seffi Naor, and Ariel Orda. Non-cooperative cost sharing games via subsidies. In *International Symposium on Algorithmic Game Theory (SAGT)*, pages 337–349. Springer, 2008. 5.3, 5.5

[50] Robert Creighton Buck. Partition of space. *The American Mathematical Monthly*, 50(9): 541–544, 1943. 9.2

[51] Nicolo Cesa-Bianchi and Gábor Lugosi. *Prediction, learning, and games*. Cambridge

University Press, 2006. 3.2.3

[52] Timothy M Chan. Optimal output-sensitive convex hull algorithms in two and three dimensions. *Discrete & Computational Geometry*, 16(4):361–368, 1996. 9.1

[53] Timothy M Chan. Improved deterministic algorithms for linear programming in low dimensions. *ACM Transactions on Algorithms (TALG)*, 14(3):1–10, 2018. 9.2, 9.3, 9.3.3, 9.4, 9.5.5, 9.5.3, 9.5.6, 9.5.3

[54] Olivier Chapelle, Bernhard Schlkopf, and Alexander Zien. *Semi-Supervised Learning*. The MIT Press, 1st edition, 2010. ISBN 0262514125. 2

[55] Magorzata Charytanowicz, Jerzy Niewczas, Piotr Kulczycki, Piotr Kowalski, and Szymon Lukasik. Seeds. UCI Machine Learning Repository, 2012. 4.6.2

[56] Bernard Chazelle and Herbert Edelsbrunner. An optimal algorithm for intersecting line segments in the plane. *Journal of the ACM (JACM)*, 39(1):1–54, 1992. 9.3, 11, 9.3.2

[57] Bernard Chazelle and Jiri Matousek. On linear-time deterministic algorithms for optimization problems in fixed dimension. *Journal of Algorithms*, 21(3):579–597, 1996. 9.2

[58] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning (ICML)*, 2020. (document), 6.1, 6.4

[59] Hugh A Chipman, Edward I George, and Robert E McCulloch. Bayesian CART model search. *Journal of the American Statistical Association (JASA)*, 93(443):935–948, 1998. 4.3.1

[60] Hugh A Chipman, Edward I George, and Robert E McCulloch. Bayesian treed models. *Machine Learning*, 48:299–320, 2002. 4.3.1

[61] C Chow. On optimum recognition error and reject tradeoff. *IEEE Transactions on Information Theory*, 16(1):41–46, 1970. 6.3, 10.2

[62] Kenneth Clarkson. More output-sensitive geometric algorithms. In *Proceedings 35th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 695–702. IEEE, 1994. 9.2, 9.2.3, 9.2.3, 9.5.3

[63] Sanjoy Dasgupta and Anupam Gupta. An elementary proof of a theorem of Johnson and Lindenstrauss. *Random Structures & Algorithms*, 22(1):60–65, 2003. 6.1

[64] Alexandre d'Aspremont. Smooth optimization with approximate gradient. *SIAM Journal on Optimization*, 19(3):1171–1183, 2008. 2.6.4, 2.6.4

[65] Rocco De Rosa and Nicolo Cesa-Bianchi. Splitting with confidence in decision trees with application to stream mining. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2015. 4.3

[66] Olivier Delalleau, Yoshua Bengio, and Nicolas Le Roux. Efficient non-parametric function induction in semi-supervised learning. In *Artificial Intelligence and Statistics (AISTATS)*, pages 96–103. PMLR, 2005. 2.6.2, 2.6.2, 2.6.2, 2.6.4, 2.6.3, 2.6.3, 2

[67] Rodney G Downey and Michael Ralph Fellows. *Parameterized complexity*. Springer Science & Business Media, 2012. 9.1

[68] Richard Dudley. The sizes of compact subsets of Hilbert space and continuity of Gaussian processes. *Journal of Functional Analysis*, 1(3):290–330, 1967. 1.2

[69] Naveen Durvasula, Nika Haghtalab, and Manolis Zampetakis. Smoothed analysis of online non-parametric auctions. In *Proceedings of the 24th ACM Conference on Economics and Computation*, pages 540–560, 2023. 9.2.2

[70] Herbert Edelsbrunner and Leonidas J Guibas. Topologically sweeping an arrangement. In *Symposium on Theory of Computing (STOC)*, pages 389–403, 1986. 2

[71] Ran El-Yaniv and Yair Wiener. On the foundations of noise-free selective classification. *Journal of Machine Learning Research (JMLR)*, 11(5), 2010. 10.2

[72] Ran El-Yaniv and Yair Wiener. Active learning via perfect selective classification. *Journal of Machine Learning Research (JMLR)*, 13(2), 2012. 10.2

[73] Matthew England and James H Davenport. The complexity of cylindrical algebraic decomposition with respect to polynomial degree. In *International Workshop on Computer Algebra in Scientific Computing*, pages 172–192. Springer, 2016. 2.2.2

[74] Floriana Esposito, Donato Malerba, Giovanni Semeraro, and J Kay. A comparative analysis of methods for pruning decision trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 19(5):476–491, 1997. 4.1, 4.4, 4.6.3

[75] Floriana Esposito, Donato Malerba, Giovanni Semeraro, and Valentina Tamma. The effects of pruning methods on the predictive accuracy of induced decision trees. *Applied Stochastic Models in Business and Industry (ASMBI)*, 15(4):277–299, 1999. 4.4

[76] Dafydd Evans, Antonia J Jones, and Wolfgang M Schmidt. Asymptotic moments of near–neighbour distance distributions. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 458(2028):2839–2849, 2002. C.2

[77] Henning Fernau. On parameterized enumeration. In *International Computing and Combinatorics Conference*, pages 564–573. Springer, 2002. 9.1

[78] Henning Fernau, Petr Golovach, Marie-France Sagot, et al. Algorithmic enumeration: Output-sensitive, input-sensitive, parameterized, approximative (Dagstuhl Seminar 18421). In *Dagstuhl Reports*, volume 8. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019. 9.1

[79] Ronald A Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179–188, 1936. 4.6

[80] Jean-Jacques Fuchs. Recovery of exact sparse representations in the presence of bounded noise. *IEEE Transactions on Information Theory*, 51(10):3601–3608, 2005. A.2.1

[81] B. German. Glass Identification. UCI Machine Learning Repository, 1987. 4.6.2

[82] Michel Goemans and David Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 42(6):1115–1145, 1995. 7.2.1, D.3.3

[83] Rishi Gupta and Tim Roughgarden. A PAC approach to application-specific algorithm selection. In *Innovations in Theoretical Computer Science (ITCS)*, pages 123–134, 2016. 5.4

[84] Rishi Gupta and Tim Roughgarden. A PAC approach to application-specific algorithm selection. *SIAM Journal on Computing*, pages 992–1017, 2017. 1, 1.3, 2, 3, D.3.1

[85] Rishi Gupta and Tim Roughgarden. Data-driven algorithm design. *Communications of the ACM*, 63(6):87–94, 2020. 1.4, 10.1

[86] Sandeep Kumar Gupta, Angappa Gunasekaran, Jiju Antony, Shivam Gupta, Surajit Bag, and David Roubaud. Systematic literature review of project failures: Current trends and scope for future research. *Computers & Industrial Engineering*, 127:274–285, 2019. 5.1

[87] Dan Gusfield and Paul Stelling. Parametric and inverse-parametric sequence alignment with XPARAL. *Methods in Enzymology*, 266:481–494, 1996. 7

[88] Dan Gusfield, Krishnan Balasubramanian, and Dalit Naor. Parametric optimization of sequence alignment. *Algorithmica*, 12(4):312–326, 1994. 9.5.2, 9.5.3

[89] Nika Haghtalab, Tim Roughgarden, and Abhishek Shetty. Smoothed analysis of online and differentially private learning. *Advances in Neural Information Processing Systems*, 33:9203–9215, 2020. 1.3

[90] Steve Hanneke. Theory of disagreement-based active learning. *Foundations and Trends in Machine Learning*, 7(2-3):131–309, 2014. 10.2

[91] John C Harsanyi. Games with incomplete information played by "Bayesian" players part II. Bayesian equilibrium points. *Management Science*, 14(5):320–334, 1968. 5.3.3

[92] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009. 3, 3, 3.3

[93] Elad Hazan and Comandur Seshadhri. Adaptive algorithms for online decision problems. In *Electronic colloquium on computational complexity (ECCC)*, volume 14, 2007. 39

[94] Mark Herbster and Manfred Warmuth. Tracking the best expert. *Machine Learning*, pages 151–178, 1998. 8, 28, 8.1, 3

[95] Arthur Hoerl and Robert Kennard. Ridge regression: applications to nonorthogonal problems. *Technometrics*, 12(1):69–82, 1970. 3

[96] Xiyang Hu, Cynthia Rudin, and Margo Seltzer. Optimal sparse decision trees. *Advances in Neural Information Processing Systems (NeurIPS)*, 32, 2019. 4.5

[97] Richard M Karp. Reducibility among combinatorial problems (book chapter). In *Complexity of Computer Computations, R. E. Miller and J. W. Thatcher (Eds)*, 1972. 5.3.2

[98] Michael Kearns and Yishay Mansour. On the boosting ability of top-down decision tree learning algorithms. In *Symposium on Theory of Computing (STOC)*, pages 459–468, 1996. 4.1, 4.3, 4.3.1, 4.3, 4.3, 4.6, 4.6.2

[99] John Kececioglu and Eagu Kim. Simple and fast inverse alignment. In *Annual International Conference on Research in Computational Molecular Biology*, pages 441–455. Springer, 2006. 9.5.3

[100] John Kececioglu, Eagu Kim, and Travis Wheeler. Aligning protein sequences with predicted secondary structure. *Journal of Computational Biology*, 17(3):561–580, 2010. 9.5.1

[101] Mikhail Khodak, Maria-Florina Balcan, and Ameet Talwalkar. Adaptive gradient-based

meta-learning methods. *Advances in Neural Information Processing Systems*, 2019. 7.1

[102] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *Advances in Neural Information Processing Systems*, 33:18661–18673, 2020. (document), 6.1, 6.4

[103] Fahime Khozeimeh, Roohallah Alizadehsani, Mohamad Roshanzamir, and Pouran Layegh. Cryotherapy Dataset . UCI Machine Learning Repository, 2018. 4.6.2

[104] Robert Kleinberg, Kevin Leyton-Brown, Brendan Lucier, and Devon Graham. Procrastinating with confidence: Near-optimal, anytime, adaptive algorithm configuration. *Advances in Neural Information Processing Systems*, 32, 2019. 1.3

[105] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015. 2.7

[106] Thomas Lavastida, Benjamin Moseley, R Ravi, and Chenyang Xu. Learnable and instance-robust predictions for online matching, flows and load balancing. In *29th Annual European Symposium on Algorithms (ESA 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021. 1.3

[107] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 2.7

[108] W Arthur Lewis. The two-part tariff. *Economica*, 8(31):249–270, 1941. 9.3

[109] Moshe Lichman et al. UCI machine learning repository, 2013. 4.6

[110] Chaochao Lin, Maria-Florina Balcan, Avrim Blum, and Matteo Pozzi. Multi-agent value of information for components' inspections. *The 13th International Conference on Structural Safety and Reliability (ICOSSAR)*, 2021. 5.2, 5.3.1, 5.3.4

[111] Jimmy Lin, Chudi Zhong, Diane Hu, Cynthia Rudin, and Margo Seltzer. Generalized and scalable optimal sparse decision trees. In *International Conference on Machine Learning (ICML)*, pages 6150–6160. PMLR, 2020. 4.5

[112] Volker Lohweg. Banknote authentication. UCI Machine Learning Repository, 2013. 4.6

[113] László Lovász and Santosh Vempala. Fast algorithms for logconcave functions: Sampling, rounding, integration and optimization. In *Foundations of Computer Science (FOCS)*, pages 57–68. IEEE, 2006. 8.2, 8.2, 9.2.2

[114] Oded Z Maimon and Lior Rokach. *Data mining with decision trees: theory and applications*, volume 81. World scientific, 2014. 4.1

[115] Julien Mairal and Bin Yu. Complexity analysis of the lasso regularization path. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, pages 1835–1842, 2012. 3.2.2, 3.2.2, 3.2.5

[116] Yishay Mansour. Pessimistic decision tree pruning based on tree size. In *International Conference on Machine Learning (ICML)*, pages 195–201, 1997. 4.1, 4.4, 4.4, 4.6.3

[117] Pascal Massart. Some applications of concentration inequalities to statistics. In *Annales de la Faculté des sciences de Toulouse: Mathématiques*, volume 9, pages 245–303, 2000.

4.3.1

[118] Nimrod Megiddo. Combinatorial optimization with rational objective functions. In *Symposium on Theory of Computing (STOC)*, pages 1–12, 1978. 9.5, 9.5.2, 9.5.2

[119] Reshef Meir, Yoram Bachrach, and Jeffrey S Rosenschein. Minimal subsidies in expense sharing games. In *International Symposium on Algorithmic Game Theory (SAGT)*, pages 347–358. Springer, 2010. 5.5

[120] Ulla Miekkala. Graph properties for splitting with grounded Laplacian matrices. *BIT Numerical Mathematics*, 33(3):485–495, 1993. 2.6.3

[121] John Mingers. Expert systems—rule induction with statistical data. *Journal of the Operational Research Society (JORS)*, 38:39–47, 1987. 4.1, 4.4

[122] John Mingers. An empirical comparison of pruning methods for decision tree induction. *Machine Learning*, 4:227–243, 1989. 4.1, 4.4, 4.6.3

[123] John Mingers. An empirical comparison of selection measures for decision-tree induction. *Machine learning*, 3:319–342, 1989. 4.1

[124] Tom Mitchell. Version spaces: A candidate elimination approach to rule learning. In *Proceedings of the 5th international joint conference on Artificial intelligence-Volume 1*, pages 305–310, 1977. 10.2

[125] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. MIT press, 2018. 3, 5.4.1, A.3

[126] Jamie Morgenstern and Tim Roughgarden. On the pseudo-dimension of nearly optimal auctions. *Advances in Neural Information Processing Systems (NeurIPS)*, 28, 2015. 9.3

[127] Kevin Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012. 3

[128] Sreerama K Murthy. Automatic construction of decision trees from data: A multi-disciplinary survey. *Data mining and knowledge discovery*, 2:345–389, 1998. 4.1

[129] Shin-ichi Nakano. Efficient generation of triconnected plane triangulations. *Computational Geometry*, 27(2):109–122, 2004. 9.1

[130] Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V Vazirani. *Algorithmic game theory*. Cambridge University Press, 2007. 5.3, 5.3

[131] Walter Y Oi. A disneyland dilemma: Two-part tariffs for a mickey mouse monopoly. *The Quarterly Journal of Economics*, 85(1):77–96, 1971. 9.3

[132] Martin J Osborne and Ariel Rubinstein. *A course in game theory*. MIT press, 1994. 5.3.3

[133] Kohei Ozaki, Masashi Shimbo, Mamoru Komachi, and Yuji Matsumoto. Using the mutual k-nearest neighbor graphs for semi-supervised classification on natural language data. In *Computational Natural Language Learning (CoNLL)*, pages 154–162, 2011. 2.6.1

[134] Vardan Papyan, XY Han, and David L Donoho. Prevalence of neural collapse during the terminal phase of deep learning training. *Proceedings of the National Academy of Sciences*, 117(40):24652–24663, 2020. 6.1.1

[135] Fabian Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau,

M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research (JMLR)*, 12:2825–2830, 2011. 3.2.3, 4.1

[136] Jeffrey K Pinto and Dennis P Slevin. Critical factors in successful project implementation. *IEEE transactions on engineering management*, 1:22–27, 1987. 5.1

[137] David Pollard. *Convergence of stochastic processes*. Springer Science & Business Media, 2012. 1, 5.4.1

[138] Nikita Puchkin and Nikita Zhivotovskiy. Exponential savings in agnostic active learning through abstention. In *Conference on Learning Theory (COLT)*, pages 3806–3832. PMLR, 2021. 10.2

[139] J. Ross Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986. 4.6.3

[140] J. Ross Quinlan. Simplifying decision trees. *International Journal of Man-Machine Studies (IJMMS)*, 27(3):221–234, 1987. 4.4, 4.6.3

[141] J Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993. 4.1

[142] J. Ross Quinlan. Learning decision tree classifiers. *ACM Computing Surveys (CSUR)*, 28 (1):71–72, 1996. 4.1

[143] Jorge Reyes-Ortiz, Davide Anguita, Alessandro Ghio, Luca Oneto, and Xavier Parra. Human Activity Recognition Using Smartphones. UCI Machine Learning Repository, 2012. 4.6.2

[144] Ronald Rivest and Robert Sloan. A formal model of hierarchical concept-learning. *Information and Computation*, 114(1):88–114, 1994. 10.2

[145] Volker Roth. The generalized LASSO. *IEEE Transactions on Neural Networks*, 15(1): 16–28, 2004. 3.3

[146] Tim Roughgarden. The price of anarchy in games of incomplete information. In *Proceedings of the 13th ACM Conference on Electronic Commerce (EC)*, pages 862–879, 2012. 5.3

[147] Cynthia Rudin. Please stop explaining black box models for high stakes decisions. *Stat*, 1050:26, 2018. 4.1

[148] Bernhard Schölkopf, Ralf Herbrich, and Alex J Smola. A generalized representer theorem. In *Conference on Learning Theory (COLT)*, pages 416–426. Springer, 2001. 3.3

[149] Gideon Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, pages 461–464, 1978. 3.1

[150] Raimund Seidel. Small-dimensional linear programming and convex hulls made easy. *Discrete & Computational Geometry*, 6:423–434, 1991. 9.2, 9.4

[151] Shai Shalev-Shwartz. Online learning and online convex optimization. *Foundations and trends in Machine Learning*, 4(2):107–194, 2011. 1

[152] Dravyansh Sharma and Maxwell Jones. Efficiently learning the graph for semi-supervised learning. In *Uncertainty in Artificial Intelligence (UAI)*, pages 1900–1910. PMLR, 2023. 1.4, 1.4, 2.7

[153] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transac-*

tions on Pattern Analysis and Machine Intelligence (TPAMI), 22(8):888–905, 2000. **??**

[154] Nora H Sleumer. Output-sensitive cell enumeration in hyperplane arrangements. *Nordic Journal of Computing*, 6(2):137–147, 1999. 9.2.3

[155] Daniel 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. 1.4, 5, 6

[156] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015. 2.7

[157] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996. 3

[158] Ryan J Tibshirani. The lasso problem and uniqueness. *Electronic Journal of statistics*, 7: 1456–1490, 2013. A.2.1, 36, A.2.1, A.2.2

[159] Andrey Tikonov and Vasily Arsenin. Solutions of ill-posed problems. *New York: Winston*, 1977. 3

[160] Constantino Tsallis. Possible generalization of Boltzmann-Gibbs statistics. *Journal of Statistical Physics*, 52:479–487, 1988. 4.3

[161] Vladimir N Vapnik and Alexey Y Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16 (2):264, 1971. 5.4.1

[162] Richard S Varga. Matrix iterative analysis. *Prentice Hall Series in Automatic Computations*, 1962. 2.6.3

[163] Santosh Vempala. *The Random Projection Method*, volume 65. American Mathematical Soc., 2005. 6.1

[164] Martin Wainwright. Information-theoretic bounds on sparsity recovery in the high-dimensional and noisy setting. In *IEEE International Symposium on Information Theory*, pages 961–965. IEEE, 2007. 3.3

[165] Martin Wainwright. Sharp thresholds for high-dimensional and noisy sparsity recovery using L1-constrained quadratic programming (LASSO). *IEEE Transactions on Information Theory*, 55(5):2183–2202, 2009. 3.3

[166] Meng Wang, Weijie Fu, Shijie Hao, Dacheng Tao, and Xindong Wu. Scalable semi-supervised learning by efficient anchor graph regularization. *IEEE Transactions on Knowledge and Data Engineering*, 28(7):1864–1877, 2016. 2

[167] Gellért Weisz, Andras Gyorgy, and Csaba Szepesvári. LeapsAndBounds: A method for approximately optimal algorithm configuration. In *International Conference on Machine Learning (ICML)*, pages 5257–5265. PMLR, 2018. 1.3

[168] William H Wolberg, W Nick Street, and Olvi L Mangasarian. Machine learning techniques to diagnose breast cancer from image-processed nuclear features of fine needle

aspirates. *Cancer Letters*, 77(2-3):163–171, 1994. 4.6

[169] Yuhong Wu, Håkon Tjelmeland, and Mike West. Bayesian CART: Prior specification and posterior simulation. *Journal of Computational and Graphical Statistics (JCGS)*, 16(1): 44–66, 2007. 4.3.1

[170] Haifeng Xu. On the tractability of public persuasion with no externalities. In *Symposium on Discrete Algorithms (SODA)*, pages 2708–2727. SIAM, 2020. 2

[171] Ali Zeynali, Bo Sun, Mohammad Hajiesmaili, and Adam Wierman. Data-driven competitive algorithms for online knapsack and set cover. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 10833–10841, 2021. 1.3

[172] Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. *Advances in Neural Information Processing Systems*, 16(16):321–328, 2004. **??**

[173] Xiaojin Zhu and Andrew Goldberg. Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning*, 3(1):1–130, 2009. 2, 2.1

[174] Xiaojin Zhu, Zoubin Ghahramani, and John Lafferty. Semi-supervised learning using Gaussian fields and harmonic functions. In *International Conference on Machine Learning (ICML)*, pages 912–919, 2003. **??**, 2.1, 2.3.1, 2.4.2, 2.6.2, 2.6.2, 2.6.3, 2.6.3, 2.6.3

[175] Xiaojin Zhu, John Lafferty, and Ronald Rosenfeld. *Semi-supervised learning with graphs*. PhD thesis, Carnegie Mellon University, Language Technologies Institute, 2005. 2

[176] Yinglun Zhu and Robert Nowak. Active learning with neural networks: Insights from nonparametric statistics. In *Advances in Neural Information Processing Systems*, 2022. 10.2

[177] Hui Zou and Trevor Hastie. Regularization and variable selection via the Elastic Net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2): 301–320, 2005. 3.1, 1, 3.1, 3.2.2, A.2.1