Distributionally Adversarial Learning

by

Zhan Shi
B.E., University of Electronic Science and Technology of China, 2015

THESIS

Submitted as partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Chicago, 2021

Chicago, Illinois

Defense Committee:
Prof. Xinhua Zhang, Chair and Advisor
Prof. Ian Kash
Prof. Brian Ziebart
Prof. Lev Reyzin, Mathematics and Computer Science
Prof. Yaoliang Yu, University of Waterloo

ACKNOWLEDGMENTS

Firstly, I would like to express my sincere gratitude to my advisor Prof. Xinhua Zhang for his patience, enthusiasm, immense knowledge. During my research, he was always there to provide academic guidance and insight. He also inspired me to do more interesting research in different area of computer science. It would not be possible to conduct my doctoral study without his continuous support.

I also thank my thesis committee: Prof. Ian Kash, Prof. Lev Reyzin, Prof. Brian Ziebart, and Prof. Yaoliang Yu, for their insigtful comments and valuable suggestions. I thank my colleagues and friends in UIC for illuminating discussions and encouragement.

Last but not least, I would express a deep sense of gratitude to my family for their unconditional love and always supporting me spiritually throughout my life.

ZS

CONTRIBUTION OF AUTHORS

Chapter 2 presents a published manuscript [1] for which I was the primary author. Prof. Xinhua Zhang and Prof. Yaoliang Yu contributed to the analysis of the optimization algorithm and revising the manuscript.

Chapter 3 and Chapter 4 presents a published manuscript [2] for which I was the primary author. Dr. Zac Cranko contributed to the analysis of distributionally robust optimization. Prof. Xinhua Zhang contributed to the robust kernel machine and revising the manuscript. Prof. Richard Nock and Dr. Simon Kornblith contributed to discussions with respect to the work.

Chapter 5 presents a published manuscript [3] for which I was the primary author. Hongwei Jin contributed to the experiments and the analysis of certificates. Prof. Xinhua Zhang contributed to the analysis of certificates and revising the manuscript.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

CHAPTER **PAGE**

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES (Continued)

FIGURE **PAGE**

LIST OF ABBREVIATIONS

ADMM	Alternating Direction Method of Multipliers
APM	Adversarial Prediction Models
DCG	Discounted Cumulative Gain
DRO	Distributionally Robust Optimization
DRR	Distributionally Robust Risk
ERM	Empirical Risk Minimization
GCN	Graph Convolution Network
PGD	Projected Gradient Descent
PO	Polar Operator
RKHS	Reproducing Kernel Hilbert Space
SVM	Support Vector Machine
SVRG	Stochastic Variance Reduced Gradient

SUMMARY

Adversarial machines, where a learner competes against an adversary, have regained much recent interest in machine learning. This adversarial learning framework has motivated numerous ideas in cost-sensitive learning and model robustness. For example, the adversarial learning can be expressed as a robust minimax game to predict class labels under a cost-sensitive loss. However, the adversarial prediction under multivariate losses is computationally intractable due to the exponential number of variables. We first reduce those exponentially-sized problems to polynomially-sized convex-concave saddle point problems using appropriate sufficient statistics. Then we develop a new stochastic variance-reduced algorithm to efficiently solve them, which allows any Bregman divergence as a proximal function and achieves linear convergence rates. We verify the efficiency of our new optimization algorithm through extensive experiments on two example applications: LPboosting and multivariate prediction under F-score and Discounted Cumulative Gain.

We also investigate the robustness certificates under adversarial learning framework. As a popular adversarial learning method, distributional robust optimization has been shown to certify the robustness of models to adversarial attacks. However, existing work are either restricted to linear models or limited robustness. In this thesis we resolved these limitations by upper bounding the distributional robust risk with an empirical risk regularized by the Lipschitz constant of the model, including deep neural networks and kernel machines. As a key advantage of the resulting model, we showed that it also provides a certificate for adversarial

training. However, tightly enforcing the Lipschitz constant of neural networks is known to be difficult. Interestingly, by focusing on the kernel machines, we manage to show that the Lipschitz constant of functions in the Reproducing Kernel Hilbert Space induced by product kernel can be tightly enforced in polynomial time. We conduct extensive experiments to verify the theoretical findings and the robustness of Lipschitz regularized kernel machines.

The adversarial learning framework also enjoys the flexibility to be embedded into general loss functions. The resulting robust loss can significantly improve the robustness of graph convolution networks. However, current work only consider node feature attack or simple topological attack scenarios. We propose two certificates for graph convolution networks under complex topological attacks. The first certificate is based on Lagrange dualization by decoupling the constraint set. Our second certificate is a theoretically tightest certificate based on convex envelope, which can be efficiently computed by dynamic programming. We also develop a new attack algorithm to examine the tightness of the certificates. As a byproduct, the new attack algorithm will be used in conjunction with robust training, significantly increasing the fraction of graphs certified as robust.

CHAPTER 1

INTRODUCTION

This introduction chapter presents an overview of the empirical risk minimization (ERM) principle in statistical learning theory and the motivation for the adversarial learning framework. The main concepts will be covered in depth in later chapters.

1.1 Notation

Throughout this report, we adopt the conventions of extended arithmetics, whereby $\infty \cdot 0 = 0 \cdot \infty = 0/0 = 0$ and $\infty - \infty = -\infty + \infty = 1/0 = \infty$. We denote scalars with lower case letters (e.g. y and λ), vectors with bold face letters (e.g. $\mathbf{x}$ and $\boldsymbol{\theta}$), and matrix with capital letters (e.g. $\mathbf{X}$). Sets are in calligraphic font: $\mathcal{X}$, $\mathcal{Y}$. The set of real numbers is denoted by $\mathbb{R}$ and the set of non-negative real numbers is denoted by $\mathbb{R}_+$. The tth vector in a sequence of vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n$ is denoted by $\mathbf{x}_t$ while the ith element of a vector $\mathbf{x}$ is denoted by x_i. If $\mathbb{P}$ is a distribution on a set $\mathcal{X}$, then we denote as $\mathbb{E}_{\mathbb{P}}[\mathbf{x}]$ the expectation of the random variable $\mathbf{x}$ under distribution $\mathbb{P}$.

The inner product between vectors $\mathbf{x}$ and $\mathbf{w}$ is denoted by $\langle \mathbf{x}, \mathbf{w} \rangle$. A norm of a vector $\mathbf{x}$ is denoted by $\|\mathbf{x}\|$. The dual norm is defined through $\|\mathbf{x}\|_* = \sup\{\langle \mathbf{x}, \boldsymbol{\lambda} \rangle : \|\boldsymbol{\lambda}\| \leq 1\}$. For example, the ℓ_1 norm, $\|\mathbf{x}\|_1 = \sum_i |x_i|$, is dual to the ℓ_∞ norm, $\|\mathbf{x}\|_\infty = \max_i |x_i|$ and the Euclidean norm, $\|\mathbf{x}\|_2 = (\langle \mathbf{x}, \mathbf{x} \rangle)^{1/2}$, is dual to itself. Similarly, the inner product in a Reproducing Kernel Hilbert Space (RKHS) $\mathcal{F}$ is denoted by $\langle \cdot, \cdot \rangle_{\mathcal{F}}$, and the norm induced by the inner product is $\|\cdot\|_{\mathcal{F}}$.

The conjugate of a function $f(\mathbf{x})$ on $\mathbb{R}^n$ is defined as $f^*(\mathbf{x}) = \sup_{\boldsymbol{\lambda}} \langle \mathbf{x}, \boldsymbol{\lambda} \rangle - f(\boldsymbol{\lambda})$. The indicator function is defined through $[\![\cdot]\!] = 1$ if $\cdot$ is true, and 0 otherwise. The Lipschitz constant of a function f mapping from metric spaces $(\mathcal{X}, d_{\mathcal{X}})$ and $(\mathcal{Y}, d_{\mathcal{Y}})$ is denoted by $Lip(f) := \sup_{\mathbf{x} \neq \mathbf{x}'} \frac{d_{\mathcal{Y}}(f(\mathbf{x}), f(\mathbf{x}'))}{d_{\mathcal{X}}(\mathbf{x}, \mathbf{x}')}$.

1.2 Statistical Learning Theory

In statistical learning framework, we have following settings for a common learning problem:

- **Data-generation process:** Assuming $\mathbb{P}$ is a joint distribution over $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$ is called input space and $\mathcal{Y}$ is called output space (e.g. $\mathcal{Y} = \{0, 1\}$ in binary classification). Thus an input-output pair $(\mathbf{x}, y)$ is generated from this distribution $\mathbb{P}$.
- **The learner's input:** The learner has only access to a finite set $\mathcal{S} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ containing n input-output pairs in $\mathcal{X} \times \mathcal{Y}$. This is often called *training set* or *training examples*.
- **The learner's goal:** The learner is required to output a *prediction rule*, $f : \mathcal{X} \to \mathcal{Y}$, which can correctly predict the output of new input data. This function is also called *classifier* or *hypothesis*.

To measure the *error* or *risk* of a classifier, we define a *loss function* $\ell : \mathbb{R} \times \mathcal{Y} \to \mathbb{R}_+$ that quantifies the mismatch between the prediction of classifier $f(\mathbf{x})$ and the actual output y. Then the *expected risk* w.r.t. the data-generating distribution $\mathbb{P}$ is defined to be the expected loss of a classifier f, i.e., $\mathbb{E}_{\mathbb{P}} \ell(f(\mathbf{x}), y)$. Similarly, the *empirical risk* over the training set $\mathcal{S}$ is defined as $\mathbb{E}_{\hat{\mathbb{P}}_n} \ell(f(\mathbf{x}), y)$, where $\hat{\mathbb{P}}_n$ is the *empirical distribution* on the training set $\mathcal{S}$.

Empirical risk minimization (ERM) is a classical learning paradigm that aims to minimize the empirical risk $\mathbb{E}_{\hat{\mathbb{P}}_n}\ell(f(\mathbf{x}), y)$. Assuming the examples in training set are independently and identically distributed (i.i.d.) according to the true distribution $\mathbb{P}$, ERM can be formulated as an optimization problem:

$$\inf_{f\in\mathcal{F}}\left\{\mathbb{E}_{\hat{\mathbb{P}}_n}\ell(f(\mathbf{x}), y) = \frac{1}{n}\sum_{i=1}^{n}\ell(h(\mathbf{x}_i), y_i)\right\}, \tag{1.1}$$

where $\mathcal{F}$ is a hypothesis space.

Many popular models use ERM as a learning principle. For example, the square loss $\ell(f(\mathbf{x}), y) = \frac{1}{2}(f(\mathbf{x}) - y)^2$ is used in linear regression model, where the hypothesis space $\mathcal{F}$ is set to be all linear function $f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x}\rangle$; With the same hypothesis space, logistic regression uses cross entropy $\ell(f(\mathbf{x}), y) = -y\ln f(\mathbf{x}) - (1-y)\ln(1-f(\mathbf{x}))$ as its loss function. Support Vector Machine [4] is obtained when ℓ is the hinge loss $\ell(f(\mathbf{x}), y) = \max\{0, 1 - yf(\mathbf{x})\}$ and $\mathcal{F}$ is an RKHS.

On the other hand, the ERM learning often suffers from overfitting if the hypothesis space $\mathcal{F}$ is overly large. Instead, we would like to search over a restricted hypothesis space $\mathcal{F}$ with the hope that there exists a "good" classifier $f \in \mathcal{F}$ that can achieve a low *generalization error*, i.e., the difference between the expected and empirical risk. Hence, a fundamental question in statistical learning arises: over which hypothesis space ERM learning will achieve a low generalization error? This is answered in the seminal work of [5] by characterizing the hypothesis space via VC-dimension in the setup of binary classification for zero-one loss. Later, the fat-

shattering dimension [6–8], the Natarajan dimension [9] and Rademacher complexity [10] were proposed to characterize the learnability of a wide range of learning models, such as regression problems, multiclass learning and kernelized SVM.

Regularization is a standard technique to control the complexity of hypothesis space. It aims to minimize the empirical risk and a regularization function at the same time, which is commonly referred to as regularized ERM:

$$\inf_{f\in\mathcal{F}} \mathbb{E}_{\hat{\mathbb{P}}_n} \ell(f(\mathbf{x}), y) + \lambda\Omega(f), \tag{1.2}$$

where the regularization function $\Omega(f)$ penalizes the complexity of hypothesis f and λ controls the trade-off between the model complexity and the empirical risk. For example, when $\mathcal{F}$ is set to be the linear hypothesis space $\mathcal{F} = \{\langle \mathbf{w}, \mathbf{x}\rangle\}$, adding ℓ_1 regularization $\Omega(\mathbf{w}) = \|\mathbf{w}\|_1$ to linear regression recovers the classical Lasso (least absolute shrinkage and selection operator) model. ℓ_2 regularization $\Omega(\mathbf{w}) = \|\mathbf{w}\|_2^2$ is also commonly used in linear regression as well as logistic regression. These popular regularizations admit a probabilistic interpretation: ℓ_2 and ℓ_1 regularizations corresponds to Gaussian or Laplacian prior distribution on $\mathbf{w}$, respectively [11]. Nevertheless, most regularization methods in deep learning are heuristic so far, such as dropout and weight decay.

1.3 Adversarial Learning

Although ERM enjoys the theoretical guarantees, it still has limitations.

Cost-sensitive Learning. In many applications of machine learning, the loss function is cost-sensitive in terms of specific constraints. Taking the disease classification problem as an example, if an incorrect disease diagnosis leads to treatments that cause severe complications, then this mistake requires more penalty in the learning process. We can use a confusion cost matrix to represent different penalties for mistakes. However, such customized confusion cost matrix C are usually asymmetric, and directly minimizing the cost-sensitive loss is computational intractable due to its non-convexity.

Similarly, many problems in information retrieval and learning to ranking have a score matrix, where every entry measures the performance based on the combination of predictions it makes for *all* training examples. Examples include the precision when limited to k positive predictions (P@k), the harmonic mean of precision and recall (F-score), and the discounted cumulative gain (DCG) for measuring ranking quality. These multivariate measures are non-concave. Even worse, these problems often have exponentially many variables, leading to computational intractability.

Previous researchers either adopted reweighting methods [12] or modified the training criteria [13,14] to approximately minimize the cost-sensitive risk. However, these methods introduce significant sub-optimality of the solution.

Adversarial Attacks. Deep neural networks have achieved impressive results in a variety of complex machine learning tasks, such as object recognition [15,16], natural language processing [17], and speech recognition [18]. However, [19] discovered that neural networks are highly sensitive to *adversarial examples* in the image classification domain: With some imperceptible

perturbations, an image can be misclassified by neural network classifier. This property exposes the low robustness of neural networks and is harmful to safety-critical applications.

Similar to most deep learning models, Graph Convolution Networks (GCNs) are also vulnerable to adversarial attacks [20, 21]. These graph attacks can maliciously perturb node features [22–24] or topological structures (i.e. adding or removing edges) [21, 25] to induce specific errors. They pose serious challenges because adversaries are often omnipresent in their typical application scenarios [26–28]. Therefore, it is urgent to develop robust neural network models.

Adversarial Learning. As a complementary approach to ERM, adversarial learning framework has been proposed to solve the abovementioned issues. In adversarial learning, an adversary attempts to fool models by supplying malicious input while the learner competing against the adversary to minimize the loss function.

Adversarial learning can be naturally expressed as a minimax game [29] and used to solve cost-sensitive classification problems. For example, [30] formulated the cost-sensitive classification problem as a zero-sum game, where an optimal solution can be obtained. Many multivariate prediction problems in information retrieval and learning to ranking also fall into this framework [31]. However, most of those problems are computationally intractable due to the exponential number of variables. Our **first contribution**, which will be detailed in Chapter 2, is to reduce the problem size exponentially by using appropriate sufficient statistics. After that, we will propose a new efficient algorithm to solve the general convex-concave saddle-point optimization problems, including multivariate prediction problems [1].

Distributionally Robust Optimization (DRO) is another popular method under the adversarial learning framework and has been extensively studied in [32–35]. In particular, [36] analyzed the relation between DRO and adversarial attacks and provided a loose certificate of robustness to the adversarial attacks. [37] showed that the DROs with Wasserstein ambiguity sets admit tractable reformulations when the model is linear (e.g., linear regression, logistic regression, SVM). Moreover, they found that DROs are equivalent to a regularized ERM and the classical Robust Optimization [38] under mild conditions. As the **second contribution**, we generalize their work to a richer model class including deep neural networks and kernel machines in Chapter 3. Specifically, we leverage the McShane-Whitney extension theorem [39, 40] to upper bound DRRs by the standard empirical risk regularized with the Lipschitz constant of the model. More importantly, we further prove that this upperbound provides a robustness certificate for the *adversarial training* in [41].

Unfortunately, explicitly enforcing the Lipschitz constant of deep neural networks is generally hard because of the non-convexity. Existing work instead enforce an upper bound of Lipschitz constant through the operator norm (such as the matrix spectral norm) of each layer's weight matrix [42–45]. Although these techniques satisfy the Lipschitz constraint, [46] showed that norm-constrained networks have limited expressive power. We, instead, take a new path by approximating the model space, leveraging the improper learnability of ℓ_1-regularized neural networks [47, 48]. Indeed, these networks are contained in the RKHS of multi-layer inverse kernels [47], and relaxing the search to the RKHS only incurs a polynomial generalization error. Similar results have been conjectured for more general kernels such as Gaussian RBF

kernels [49]. Our **third contribution**, which we will detail in Chapter 4, is to prove that the Lipschitz constant of functions in the RKHS of *product kernels* can be computed with high probability in $O(1/\epsilon^2)$ time by using Nyström approximation [50, 51], and empirically this approximation is also effective for non-product kernels. We validate the superiority of our robust kernel machines than Lipschitz regularized neural networks in images classification tasks.

The adversarial learning framework also has the flexibility to be embedded into general loss functions. [22, 52] introduced a robust hinge loss: $\sum_{c \neq y} \max\{0, 1 + \max_{A \in \mathcal{A}}(z_c(A) - z_y(A))\}$, where $\mathcal{A}$ is the set of admissible perturbations and z is the logits from the last layer of GCNs. Adding this robust hinge loss to the standard cross entropy loss can significantly improve the robustness of GCNs because it enforces a larger margin (i.e., confidence) between true prediction $z_y(A)$ and wrong prediction $z_c(A)$. However, the margin $z_c(A) - z_y(A)$ is often intractable in GCNs due to the nonlinearity. Furthermore, the constraint set $\mathcal{A}$ can be complicated in different applications. [52] only considered the node feature attacks and [22] developed certificates for PageRank without symmetry constraint on the attacked affinity matrix. Our **fourth contribution** is to certify robustness of GCNs under topological perturbations with both local and global budgets. We also developed a new attack algorithm to characterize the tightness of our certificates. Using the efficient attack algorithm in conjunction with robust hinge loss, we verified the increased robustness of GCNs under topological attacks.

1.4 Outline

This thesis is divided into four main chapters.

Adversarial Prediction Models. In Chapter 2, we first introduce the *adversarial prediction models* (APM) and describe challenges in cost-sensitive learning in Section 2.1. We focus on the F-score and DCG prediction problems and reduce the dimension of optimization variables from 2^n to n^2 (where n is the number of samples) in Section 2.2. Then we propose a new efficient stochastic variance-reduced algorithm called Breg-SVRG for solving general convex-concave saddle-point problems in Section 2.3. The computational complexity for solving the multivariate prediction problems is analyzed in Section 2.4, along with the general linear rates of convergence for Breg-SVRG. Experimental results are presented in Section 2.5.

Distributionally Robust Optimization with Wasserstein Distance. Chapter 3 is devoted to DRO with Wasserstein distance. We start in Section 3.2 by introducing the Wasserstein distance and reviewing some existing work on solving the DRO with Wasserstein ball. In Section 3.3, we upper bound DRRs with an empirical risk regularized by Lipschitz constant for nonlinear models. As an application, we introduce the adversarial training technique used for solving the adversarial examples problem in neural networks in Section 3.4 and establish a connection between DRRs and adversarial training risk in Section 3.5.

Robust Kernel Machines. In Chapter 4, we first give the preliminaries on kernel machines and raise the problem on how to tightly enforce the Lipschitz constant in RKHS in Section 4.1. A natural sampling method of enforcing the Lipschitz constant usually has exponential sample complexity. To address this, we propose a new efficient algorithm in Section 4.2 and prove its polynomial sample complexity for product kernels in Section 4.3. The assumptions in the proof are checked for some commonly used kernels in Section 4.4. In the last Section 4.5, we

experimentally show the efficiency of the algorithm for enforcing the Lipschitz constant and the robustness achieved by the distributionally robust kernel machines.

Robustness of Graph Convolution Networks. In Chapter 5, we first review literature on the robustness of Graph Convolution Networks in Section 5.1, and then introduce a general threat model in Section 5.2. Given a trained GCN and the threat model, we develop the exact attacks and approximate attacks in Section 5.3. To certify the robustness of GCN, we propose our first certificate in Section 5.4 based on Lagrange duality. In addition, in Section 5.5.2, we provide the tightest certificate by leveraging convex envelop tool. Our experiments in Section 5.6 confirm the tightness of the certificates. When used in conjunction with robust training, it allows a more robust Graph Convolution Network.

CHAPTER 2

ADVERSARIAL PREDICTION MODELS

(Parts of this chapter were previously published as "Bregman Divergence for Stochastic Variance Reduction: Saddle-Point and Adversarial Prediction" [1], in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.)

2.1 Introduction

2.1.1 Adversarial Prediction Models

In a general multiclass classification problem, we assume $\mathcal{X} \in \mathbb{R}^d$ and $\mathcal{Y} = \{0, \ldots, k\}$. The predictor tries to estimate the conditional probability $\mathbb{P}_{\boldsymbol{\theta}}(p|\mathbf{x})$ parameterized by $\boldsymbol{\theta}$. So, the ERM learning in this problem is

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{(\mathbf{x},y)\sim\hat{\mathbb{P}}_n; p|\mathbf{x}\sim\mathbb{P}_{\boldsymbol{\theta}}} \ell\left(y, p\right). \tag{2.1}$$

Since it is NP-hard to minimize the empirical risk w.r.t. the zero-one loss [53], many different surrogate losses have been proposed to solve the multiclass classification problem [54–56]. Nevertheless, many cost-sensitive prediction problems, such as ordinal classification and disease classification, are intractable due to the non-convexity of loss function.

Previous researchers either adopted reweighting methods [12] or modified the training criteria [13,14] to approximately minimize the cost-sensitive risk. However, these methods introduce

significant sub-optimality of the solution. Motivated by [29], [30] formulated the cost-sensitive classification problem as a minimax problem

$$\min_{\mathbb{P}(p|\mathbf{x})} \max_{\mathbb{Q}(q|\mathbf{x})} \mathbb{E}_{\mathbf{x}\sim\hat{\mathbb{P}}_n;p|\mathbf{x}\sim\mathbb{P};q|\mathbf{x}\sim\mathbb{Q}} \ell(p,q) \tag{2.2}$$

$$s.t. \quad \mathbb{E}_{\mathbf{x}\sim\hat{\mathbb{P}}_n;q|\mathbf{x}\sim\mathbb{Q}} \left[\phi(\mathbf{x},q)\right] = \mathbb{E}_{(\mathbf{x},y)\sim\hat{\mathbb{P}}_n} \left[\phi(\mathbf{x},y)\right],$$

where the predictor tries to find the conditional probability $\mathbb{P}(p|\mathbf{x})$ that minimizes the empirical risk while the adversary conditional probability $\mathbb{Q}(q|\mathbf{x})$ seeks to maximize the loss. The constraint set in Equation 2.2 matches the cross-moments (feature function $\phi(\mathbf{x},y)$) of $\mathbb{Q}$ with those of the empirical distribution while keeping the same marginal $\mathbb{Q}(\mathbf{x})$ as the empirical marginal $\hat{\mathbb{P}}_n(\mathbf{x})$. If the feature function $\phi(\mathbf{x},y)$ is restrictive (e.g., high-order moments), then $\mathbb{Q}$ will approach to $\hat{\mathbb{P}}_n$ and Equation 2.2 collapses to ERM, leading to overfitting. On the contrary, the adversary may deviate significantly from the empirical distribution if $\phi(\mathbf{x},y)$ is overly simple, whereby $\mathbb{Q}$ may not preserve important statistics of the data.

[30] reformulated the minimax problem Equation 2.2 via strong duality [57, 58],

Theorem 1 ([30, Theorem 1]). *Let the loss ℓ be convex w.r.t. the first argument and concave w.r.t. the second argument. The constrained adversarial prediction minimax game Equation 2.2 is equivalent to a minimization on the empirical average of the value of many unconstrained minimax games:*

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{(\mathbf{x},y)\sim\hat{\mathbb{P}}_n} \left[\max_{\mathbb{Q}(q|\mathbf{x})} \min_{\mathbb{P}(p|\mathbf{x})} \mathbb{E}_{p|\mathbf{x}\sim\mathbb{P};q|\mathbf{x}\sim\mathbb{Q}} \left[\ell(p,q) + \boldsymbol{\theta}^T(\phi(\mathbf{x},q) - \phi(\mathbf{x},y))\right] \right], \tag{2.3}$$

where $\boldsymbol{\theta}$ is the Lagrange dual variable for the moment matching constraints.

2.1.2 Cost-sensitive Classification

For a cost-sensitive multiclass classification problem, the loss can be represented as the confusion cost matrix. For example, a classification task with four possible labels can have a confusion cost matrix

$$C = \begin{bmatrix} 0 & 1 & 3 & 2 \\ 3 & 0 & 3 & 2 \\ 1 & 2 & 0 & 5 \\ 5 & 0 & 3 & 0 \end{bmatrix},$$

where the (i, j)th entry indicates the cost for predicting i while the true label is j.

Let $\mathbf{p_x} = [\mathbb{P}(p = 1|\mathbf{x}) \ \cdots \ \mathbb{P}(p = k|\mathbf{x})]^T \in \mathbb{R}^k$ and $\mathbf{q_x} = [\mathbb{Q}(q = 1|\mathbf{x}) \ \cdots \ \mathbb{Q}(q = k|\mathbf{x})]^T \in \mathbb{R}^k$ for an input $\mathbf{x} \in \mathcal{X}$, then the dual adversarial prediction problem Equation 2.3 can be represented as

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{\hat{\mathbb{P}}_n(\mathbf{x})} \left[\max_{\mathbf{q_x} \in \Delta} \min_{\mathbf{p_x} \in \Delta} \mathbf{p}_\mathbf{x}^T C'_{\mathbf{x},\boldsymbol{\theta}} \mathbf{q_x} \right], \tag{2.4}$$

where $(C'_{\mathbf{x},\boldsymbol{\theta}})_{p,q} = C(p, q) + \boldsymbol{\theta}^T(\phi(\mathbf{x}, q) - \phi(\mathbf{x}, y))$ and Δ is a k-dimensional probability simplex. For the feature function, [30] used the second moment statistics. This cost-sensitive problem can be efficiently solved by applying stochastic gradient method on the outer minimization problem and linear programming on the inner unconstrained zero-sum game of size k^2.

2.1.3 Adversarial Prediction under Multivariate Loss

Given a training set $X = [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{d \times n}$ and the labels vector $\hat{\mathbf{y}} \in \{0,1\}^n$, the performance of an estimator $\mathbf{y} \in \{0,1\}^n$ can be measured by F-score as $F_1(\mathbf{y}, \hat{\mathbf{y}}) = \frac{2\langle \mathbf{y}, \hat{\mathbf{y}} \rangle}{\|\mathbf{y}\|_1 + \|\hat{\mathbf{y}}\|_1}$ and $F_1(\mathbf{0}, \mathbf{0}) = 1$. In learning to rank, DCG measures the performance of item rankings with k relevancy scores: $DCG(\boldsymbol{\sigma}, \mathbf{y}) = \sum_{i=1}^{n} \frac{2^{\mathbf{y}_{\boldsymbol{\sigma}(i)}} - 1}{\log_2(i+1)}$ where $\mathbf{y} \in \{0, 1, \ldots, k-1\}^n$ for n items and $\boldsymbol{\sigma}(i)$ denotes the ith ranked item. Figure 1 gives examples of score matrices under F-score and DCG measure when $n = 3$.

Let the feature function be $\phi(X, \mathbf{y}) = \frac{1}{n} X\mathbf{y}$, and the loss function $\ell(\mathbf{y}, \mathbf{z})$ be F-score. Then the multivariate prediction problem under F-score can be formulated as

$$\begin{aligned} \max_{q(\mathbf{z}|X) \in \Delta^{2^n}} \min_{p(\mathbf{y}|X) \in \Delta^{2^n}} \quad & \mathbb{E}_{\mathbf{y},\mathbf{z}} \frac{2\mathbf{y}^T\mathbf{z}}{\mathbf{1}^T\mathbf{y} + \mathbf{1}^T\mathbf{z}} \\ s.t. \quad & \mathbb{E}_{\mathbf{y} \sim p} \left(\frac{1}{n} X\mathbf{y} \right) = \frac{1}{n} X\hat{\mathbf{y}}. \end{aligned} \tag{2.5}$$

Different to Equation 2.4, here the distribution $p(\cdot)$ and $q(\cdot)$ are over the labeling on the entire training set, so they are in a 2^n dimensional probability simplex, Δ^{2^n}. Following the derivation in Theorem 1, the dual problem of Equation 2.5 can be written as

$$\max_{\boldsymbol{\theta}} \; -\frac{\lambda}{2} \|\boldsymbol{\theta}\|_2^2 + \frac{1}{n} \boldsymbol{\theta}^T X\hat{\mathbf{y}} + \min_{p \in \Delta^{2^n}} \max_{q \in \Delta^{2^n}} \mathop{\mathbb{E}}_{\mathbf{y} \sim p, \mathbf{z} \sim q} \left[\frac{2\mathbf{y}^T\mathbf{z}}{\mathbf{1}^T\mathbf{y} + \mathbf{1}^T\mathbf{z}} - \frac{1}{n} \boldsymbol{\theta}^T X\mathbf{y} \right], \tag{2.6}$$

F_1	000	001	010	011	100	101	110	111
000	1	0	0	0	0	0	0	0
001	0	1	0	2/3	0	2/3	0	1/2
010	0	0	1	2/3	0	0	2/3	1/2
011	0	2/3	2/3	1	0	1/2	1/2	4/5
100	0	0	0	0	1	2/3	2/3	1/2
101	0	2/3	0	1/2	2/3	1	1/2	4/5
110	0	0	2/3	1/2	2/3	1/2	1	4/5
111	0	1/2	1/2	4/5	1/2	4/5	4/5	1

DCG	000	001	010	011	100	101	110	111
123	0	1/2	$\frac{1}{\log_2 3}$	$1/2 + \frac{1}{\log_2 3}$	1	3/2	$1 + \frac{1}{\log_2 3}$	$3/2 + \frac{1}{\log_2 3}$
132	0	$\frac{1}{\log_2 3}$	1/2	$1/2 + \frac{1}{\log_2 3}$	1	$1 + \frac{1}{\log_2 3}$	2/3	$3/2 + \frac{1}{\log_2 3}$
213	0	1/2	1	3/2	$\frac{1}{\log_2 3}$	$1/2 + \frac{1}{\log_2 3}$	$1 + \frac{1}{\log_2 3}$	$3/2 + \frac{1}{\log_2 3}$
231	0	$\frac{1}{\log_2 3}$	1	$1 + \frac{1}{\log_2 3}$	1/2	$1/2 + \frac{1}{\log_2 3}$	2/3	$3/2 + \frac{1}{\log_2 3}$
312	0	1	1/2	3/2	$\frac{1}{\log_2 3}$	$1 + \frac{1}{\log_2 3}$	$1/2 + \frac{1}{\log_2 3}$	$3/2 + \frac{1}{\log_2 3}$
321	0	1	$\frac{1}{\log_2 3}$	$1 + \frac{1}{\log_2 3}$	1/2	3/2	$1/2 + \frac{1}{\log_2 3}$	$3/2 + \frac{1}{\log_2 3}$

Figure 1: Score matrix for F-score and DCG with training size $n = 3$

where we followed [31] to add an ℓ_2^2 regularizer on $\boldsymbol{\theta}$ penalizing the dual variables on the constraints over the training data. Similarly, DCG multivariate prediction problem can be formulated as

$$\max_{\boldsymbol{\theta}} \; -\frac{\lambda}{2}\|\boldsymbol{\theta}\|_2^2 + \frac{1}{n}\boldsymbol{\theta}^T X\hat{\mathbf{y}} + \min_{p(\mathbf{y})\in\Delta^{2^n}} \max_{q(\boldsymbol{\sigma})\in \mathrm{Per(n)}} \mathop{\mathbb{E}}_{\mathbf{y},\boldsymbol{\sigma}} \left[\sum_{j=1}^{n} \frac{2^{\mathbf{y}_{\boldsymbol{\sigma}(j)}}-1}{\log_2(j+1)} - \frac{1}{n}\boldsymbol{\theta}^T X\mathbf{y}\right], \tag{2.7}$$

where we assume each item has two relevancy scores, i.e., $\mathbf{y} = \{0,1\}^n$. Thus $p(\mathbf{y})$ is constrained to a 2^n dimensional probability simplex and $q(\boldsymbol{\sigma})$ is constrained to probability space of permutation of n.

It appears that solving Equation 2.6 and Equation 2.7 can be quite challenging, because the inner minimax problem has 2^n entries! A constraint sampling algorithm was adopted in [31] to address this challenge, although no formal guarantee was established. Note that we can maximize the outer unconstrained variable $\boldsymbol{\theta}$ (with dimension same as the number of features) relatively easily using, for instance, gradient ascent, provided that we can solve the inner minimax problem quickly—a significant challenge to which we turn our attention below.

2.2 Problem Size Reduction

Surprisingly, we show that the inner minimax problem in Equation 2.6 and Equation 2.7 can be significantly simplified. The key observation is that the expectation in the objective depends only on a few sufficient statistics of p and q.

F-score Game Reduction. Assuming $p(\{\mathbf{0}\}) = q(\{\mathbf{0}\}) = 0$, the inner minimax problem of Equation 2.6 can be rewritten as

$$\mathop{\mathbb{E}}_{\mathbf{y}\sim p,\mathbf{z}\sim q}\left[\frac{2\mathbf{y}^T\mathbf{z}}{\mathbf{1}^T\mathbf{y}+\mathbf{1}^T\mathbf{z}} - \frac{1}{n}\boldsymbol{\theta}^T X\mathbf{y}\right] \tag{2.8}$$

$$= -\frac{1}{n}\boldsymbol{\theta}^T X\mathbb{E}\left[\mathbf{y}\right] + \sum_{i=1}^{n}\sum_{j=1}^{n}\mathbb{E}\left(\frac{2\mathbf{y}^T\mathbf{z}}{\mathbf{1}^T\mathbf{y}+\mathbf{1}^T\mathbf{z}}[\![\mathbf{1}^T\mathbf{y}=i]\!][\![\mathbf{1}^T\mathbf{z}=j]\!]\right) \tag{2.9}$$

$$= -\frac{1}{n}\boldsymbol{\theta}^T X\mathbb{E}\left[\mathbf{y}\right] + \sum_{i=1}^{n}\sum_{j=1}^{n}\frac{2ij}{i+j}\cdot\underbrace{\frac{1}{i}\mathbb{E}\left(\mathbf{y}[\![\mathbf{1}^T\mathbf{y}=i]\!]\right)^T}_{\boldsymbol{\alpha}_i}\cdot\underbrace{\frac{1}{j}\mathbb{E}\left(\mathbf{z}[\![\mathbf{1}^T\mathbf{z}=j]\!]\right)}_{\boldsymbol{\beta}_j}, \tag{2.10}$$

Crucially, the variables $\boldsymbol{\alpha}_i$ and $\boldsymbol{\beta}_j$ are sufficient for re-expressing Equation 2.6, since

$$\mathbf{1}^T\boldsymbol{\alpha}_i = \frac{1}{i}\mathbb{E}\left(\mathbf{1}^T\mathbf{y}[\![\mathbf{1}^T\mathbf{y}=i]\!]\right) = \mathbb{E}[\![\mathbf{1}^T\mathbf{y}=i]\!] = p(\{\mathbf{1}^T\mathbf{y}=i\}), \tag{2.11}$$

$$\sum_i i\boldsymbol{\alpha}_i = \sum_i \mathbb{E}\left(\mathbf{y}[\![\mathbf{1}^T\mathbf{y}=i]\!]\right) = \mathbb{E}\mathbf{y}, \tag{2.12}$$

and similar equalities also hold for $\boldsymbol{\beta}_j$. In details, the inner minimax problem of Equation 2.6 simplifies to:

$$\min_{\boldsymbol{\alpha}\in S}\max_{\boldsymbol{\beta}\in S}\frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\big[\underbrace{\frac{2ijn^2}{i+j}\boldsymbol{\alpha}_i^T\boldsymbol{\beta}_j}_{f_{ij}(\boldsymbol{\alpha}_i,\boldsymbol{\beta}_j)} - i\boldsymbol{\theta}^T X\boldsymbol{\alpha}_i\big] + \Omega(\boldsymbol{\alpha}) - \Omega(\boldsymbol{\beta}), \tag{2.13}$$

$$\text{where } S = \{\boldsymbol{\alpha}\geq\mathbf{0} : \mathbf{1}^T\boldsymbol{\alpha}\mathbf{1}\leq 1, \forall i,\ \|i\boldsymbol{\alpha}_i\|_\infty \leq \|\boldsymbol{\alpha}_i\|_1\}, \quad \Omega(\boldsymbol{\alpha}) = \mu\textstyle\sum_{i,j}\alpha_{ij}\log(\alpha_{ij}). \tag{2.14}$$

Importantly, $\boldsymbol{\alpha} = [\boldsymbol{\alpha}_1; \ldots, \boldsymbol{\alpha}_n]$ (resp. $\boldsymbol{\beta}$) has n^2 entries, which is significantly smaller than the 2^n entries of $\mathbf{p}$ (resp. $\mathbf{q}$) in Equation 2.6. For later purpose we have also incorporated an entropy regularizer for $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ respectively in Equation 2.13.

To justify the constraint set S, note from Equation 2.10 and Equation 2.11 that for any distribution p of $\mathbf{y}$:

$$\text{since } \boldsymbol{\alpha} \geq 0 \text{ and } \mathbf{y} \in \{0,1\}^n,\ \|i\boldsymbol{\alpha}_i\|_\infty \leq \mathbb{E}\|\mathbf{y}[\![\mathbf{1}^T\mathbf{y} = i]\!]\|_\infty \leq \mathbb{E}[\![\mathbf{1}^T\mathbf{y} = i]\!] = \|\boldsymbol{\alpha}_i\|_1. \tag{2.15}$$

Conversely, for any $\boldsymbol{\alpha} \in S$, we can construct a distribution p such that $i\boldsymbol{\alpha}_{ij} = \mathbb{E}\left(y_j[\![\mathbf{1}^T\mathbf{y} = i]\!]\right) = p(\{1^T\mathbf{y} = i, y_j = 1\})$. Consider the set $S_i = \{\boldsymbol{\alpha}_i >= 0 : \|i\boldsymbol{\alpha}_i\|_\infty <= \|\boldsymbol{\alpha}_i\|_1\}$, It is clear that the extreme points of S_i are precisely the set of $\boldsymbol{\alpha}_i$ where i of the n elements are equal to $\|\boldsymbol{\alpha}_i\|_1/i$ (and the rest are 0). [Proof: if not, then there are at least two elements of $\boldsymbol{\alpha}_i$ that are strictly smaller than $\|\boldsymbol{\alpha}_i\|_1/i$; perturb these two elements to create a convex combination.]

Define $\mathbf{y}^*$ with $y^*_j = 1$ iff $\boldsymbol{\alpha}_{ij} \neq 0$. Clearly, $1^T\mathbf{y}^* = i$ (since the extreme point $\boldsymbol{\alpha}_i$ has exactly i nonzeros). put $p(\mathbf{y}^*) = \|\boldsymbol{\alpha}_i\|_1$ (and every other $\mathbf{y}$ with $1^T\mathbf{y} = i$ to be $p(\mathbf{y}) = 0$). We verify that $i\boldsymbol{\alpha}_{ij} = \|\boldsymbol{\alpha}_i\|_1 = p(\mathbf{y}^*) = p(1^T\mathbf{y} = i, \mathbf{y}_j = 1)$. If a point $\boldsymbol{\alpha}_i$ in S_i is a convex combination of the extreme points using weights $\{w_1, w_2, \cdots\}$, then the distribution $p(1^T\mathbf{y} = i)$ generating $\boldsymbol{\alpha}_i$ can be constructed as the convex combination of the inverse image of those extreme points, using the same weights $\{w_1, w_2, \cdots\}$ to combine, so as the whole distribution p.

DCG Game Reduction. Assuming $p(\{\mathbf{0}\}) = 0$, the inner minimax problem of Equation 2.7 can be rewritten as

$$
\begin{aligned}
&\min_{p(\mathbf{y})}\max_{q(\boldsymbol{\sigma})} \sum_{i=1}^{n}\sum_{\mathbf{y}:\|\mathbf{y}\|_1=i}\sum_{\boldsymbol{\sigma}} p(\mathbf{y})q(\boldsymbol{\sigma})\left[\sum_{j=1}^{n}\frac{2^{\mathbf{y}_{\boldsymbol{\sigma}(j)}}-1}{\log_2(j+1)} - \frac{1}{n}\boldsymbol{\theta}^T X\mathbf{y}\right] \\
=&\min_{p(\mathbf{y})}\max_{q(\boldsymbol{\sigma})} -\frac{1}{n}\boldsymbol{\theta}^T X\mathbb{E}[\mathbf{y}] + \sum_{i=1}^{n}\sum_{\mathbf{y}:\|\mathbf{y}\|_1=i} p(\mathbf{y})\sum_{j=1}^{n}\frac{1}{\log_2(j+1)}\sum_{\boldsymbol{\sigma}} q(\boldsymbol{\sigma})(2^{\mathbf{y}_{\boldsymbol{\sigma}(j)}}-1) \\
=&\min_{p(\mathbf{y})}\max_{q(\boldsymbol{\sigma})} -\frac{1}{n}\boldsymbol{\theta}^T X\mathbb{E}[\mathbf{y}] + \sum_{i=1}^{n}\sum_{\mathbf{y}:\|\mathbf{y}\|_1=i} p(\mathbf{y})\sum_{j=1}^{n}\frac{1}{\log_2(j+1)}\sum_{k=1}^{n}\sum_{\boldsymbol{\sigma}:\boldsymbol{\sigma}(j)=k} q(\boldsymbol{\sigma})(2^{\mathbf{y}_k}-1) \\
=&\min_{p(\mathbf{y})}\max_{q(\boldsymbol{\sigma})} -\frac{1}{n}\boldsymbol{\theta}^T X\mathbb{E}[\mathbf{y}] + \sum_{i=1}^{n}\sum_{j=1}^{n}\frac{1}{\log_2(j+1)}\sum_{\mathbf{y}:\|\mathbf{y}\|_1=i} p(\mathbf{y})\sum_{k=1}^{n}(2^{\mathbf{y}_k}-1)\underbrace{\sum_{\boldsymbol{\sigma}:\boldsymbol{\sigma}(j)=k} q(\boldsymbol{\sigma})}_{=:\boldsymbol{\beta}_{kj}\in\mathbb{R}} \\
=&\min_{p(\mathbf{y})}\max_{q(\boldsymbol{\sigma})} -\frac{1}{n}\boldsymbol{\theta}^T X\mathbb{E}[\mathbf{y}] + \sum_{i=1}^{n}\sum_{j=1}^{n}\frac{1}{\log_2(j+1)}\boldsymbol{\beta}_j^T\underbrace{\sum_{\mathbf{y}:\|\mathbf{y}\|_1=i} p(\mathbf{y})\mathbf{y}}_{=:i\boldsymbol{\alpha}_i\in\mathbb{R}^n} \\
=&\min_{\{\boldsymbol{\alpha}_i\}\in S_\alpha}\max_{\{\boldsymbol{\beta}_j\}\in S_\beta} \sum_{i=1}^{n}\sum_{j=1}^{n}\frac{i\boldsymbol{\alpha}_i^T\boldsymbol{\beta}_j}{\log_2(j+1)} - \frac{1}{n}\boldsymbol{\theta}^T X\sum_{i=1}^{n} i\boldsymbol{\alpha}_i. \qquad (2.16)
\end{aligned}
$$

where $\boldsymbol{\beta}_{kj}$ is the marginal probability of jth item of a permutation $\boldsymbol{\sigma}$ is k, i.e., $P(\boldsymbol{\sigma}(j) = k) = \boldsymbol{\beta}_{kj}$. And the domain of $\{\boldsymbol{\alpha}\}$ and $\{\boldsymbol{\beta}\}$ are

$$S_\alpha = \{\boldsymbol{\alpha} \geq \mathbf{0} : \mathbf{1}^T\boldsymbol{\alpha}\mathbf{1} \leq 1, \forall i,\ \|i\boldsymbol{\alpha}_i\|_\infty \leq \|\boldsymbol{\alpha}_i\|_1\}, \qquad (2.17)$$

$$S_\beta = \{\boldsymbol{\beta} \geq \mathbf{0} : \forall i, \mathbf{1}^T\boldsymbol{\beta}_{i\cdot} = 1, \forall j, \mathbf{1}^T\boldsymbol{\beta}_{\cdot j} = 1\}. \qquad (2.18)$$

Both domains are convex. The constraint set S_α has been justified as in F-score section. Similarly, to justify the constraint set S_β, it is obvious to see that any distribution q of $\boldsymbol{\sigma}$ satisfies S_β since:

$$\forall j, \quad \sum_{k=1}^{n} \sum_{\boldsymbol{\sigma}:\boldsymbol{\sigma}(j)=k} q(\boldsymbol{\sigma}) = \mathbf{1}^T \boldsymbol{\beta}_{\cdot j} = 1, \quad \boldsymbol{\beta} \geq \mathbf{0} \tag{2.19}$$

$$\forall i, \quad \sum_{k=1}^{n} \sum_{\boldsymbol{\sigma}:\boldsymbol{\sigma}(k)=i} q(\boldsymbol{\sigma}) = \mathbf{1}^T \boldsymbol{\beta}_{i\cdot} = 1, \quad \boldsymbol{\beta} \geq \mathbf{0}. \tag{2.20}$$

Conversely, for any $\boldsymbol{\beta} \in S_\beta$, we can construct a distribution q such that $\boldsymbol{\beta}_{kj} = P(\boldsymbol{\sigma}(j) = k)$ in the following algorithmic way:

1. Find the minimum value $\boldsymbol{\beta}_{ij}$ in $\boldsymbol{\beta}$, define $\Phi_{ij} = \{\boldsymbol{\sigma} : \boldsymbol{\sigma}(j) = i\}$. Represent permutations that are already assigned probability as set $\Phi_{ij}^{assigned} = \{\boldsymbol{\sigma} : \boldsymbol{\sigma}(j) = i, p(\boldsymbol{\sigma})\ \textit{is assigned}\}$, and the sum probability of $\Phi_{ij}^{assigned}$ as a. Then we can set $p(\boldsymbol{\sigma}) = (\boldsymbol{\beta}_{ij} - a)/|\Phi_{ij} \setminus \{\Phi_{ij}^{assigned}\}|$ for each $\boldsymbol{\sigma} \in \Phi_{ij} \setminus \{\Phi_{ij}^{assigned}\}$. This step in fact sets free variables of probability from the strongest constraint, that is, the minimum value $\boldsymbol{\beta}_{ij}$ in $\boldsymbol{\beta}$.

2. Check all other marginal probabilities $\boldsymbol{\beta}_{xy}$ in $\boldsymbol{\beta} \setminus \boldsymbol{\beta}_{ij}$: if there is only one permutation left unassigned in Φ_{xy}, then assign it as $\boldsymbol{\beta}_{xy}$ minus sum probability of set $\Phi_{xy}^{assigned}$. This step in fact avoids setting extra free variables of probability.

3. Remove $\boldsymbol{\beta}_{ij}$ in $\boldsymbol{\beta}$, back to 1 until all permutation probability are assigned.

Assume there are m queries, the number of returned documents in each query are $n_1, n_2, \ldots, n_m$. The overall DCG multivariate prediction problem is

$$\max_{\boldsymbol{\theta}} \frac{1}{M}\sum_{k=1}^{M}\left[-\frac{\lambda}{2}\left\|\boldsymbol{\theta}\right\|_2^2+\frac{\boldsymbol{\theta}^T}{n_k}\sum_{i=1}^{n_k}\hat{y}_i^k\mathbf{x}_i^k+\min_{\boldsymbol{\alpha}^k\in S_\alpha}\max_{\boldsymbol{\beta}^k\ \in S_\beta}\ \sum_{i=1}^{n_k}\sum_{j=1}^{n_k}\frac{i{\boldsymbol{\alpha}_i^k}^T\boldsymbol{\beta}_j^k}{\log_2(j+1)}-\frac{\boldsymbol{\theta}^T}{n_k}X^k\sum_{i=1}^{n_k}i\boldsymbol{\alpha}_i^k\right]$$

$$=\min_{\{\boldsymbol{\alpha}^k\}_M\in S_\alpha}\max_{\{\boldsymbol{\beta}^k\}_M\in S_\beta}\frac{1}{M}\sum_{k=1}^{M}\sum_{i=1}^{n_k}\sum_{j=1}^{n_k}\frac{i{\boldsymbol{\alpha}_i^k}^T\boldsymbol{\beta}_j^k}{\log_2(j+1)}+\max_{\boldsymbol{\theta}}-\frac{\lambda}{2}\left\|\boldsymbol{\theta}\right\|_2^2+\frac{\boldsymbol{\theta}^T}{M}\sum_{k=1}^{M}\frac{X^k\hat{y}^k-X^k\sum_{i=1}^{n_k}i\boldsymbol{\alpha}_i^k}{n_k}$$

$$=\min_{\{\boldsymbol{\alpha}^k\}_M\in S_\alpha}\max_{\{\boldsymbol{\beta}^k\}_M\in S_\beta}\ \mathop{\mathbb{E}}_{k}\sum_{i=1}^{n_k}\sum_{j=1}^{n_k}\frac{i{\boldsymbol{\alpha}_i^k}^T\boldsymbol{\beta}_j^k}{\log_2(j+1)}+\frac{1}{2\lambda}\left\|\mathop{\mathbb{E}}_{k}\frac{X^k\hat{y}^k-X^k\sum_{i=1}^{n_k}i\boldsymbol{\alpha}_i^k}{n_k}\right\|^2. \tag{2.21}$$

2.3 Stochastic Algorithm for Saddle Point Problem

In this section we focus on a composite saddle-point problem with convex-concave (saddle) functions K and M:

$$(x^*,y^*)=\arg\min_x\max_y K(x,y)+M(x,y), \tag{2.22}$$
$$\text{where}\quad K(x,y)=\frac{1}{n}\sum_{k=1}^{n}\psi_k(x,y).$$

Most commonly used supervised losses for linear models can be written as $g^\star(X\mathbf{w})$, where $g^\star$ is the Fenchel dual of a convex function g, X is the design matrix, and $\mathbf{w}$ is the model vector. So the regularized risk minimization can be naturally written as $\min_{\mathbf{w}}\max_{\boldsymbol{\alpha}}\boldsymbol{\alpha}^TX\mathbf{w}+\Omega(\mathbf{w})-g(\boldsymbol{\alpha})$, where Ω is a regularizer. This fits into our framework Equation 2.22 with a bilinear function K and a decoupled function M.

More importantly, the objective function in Equation 2.13 and Equation 2.16 fit naturally into this framework, with $\Omega(\boldsymbol{\alpha})-\Omega(\boldsymbol{\beta})$ and constraints corresponding to M, and the rest terms

to K. The entropy function Ω is convex w.r.t. the KL-divergence, which is in turn distance enforcing w.r.t. the ℓ_1 norm over the probability simplex [59].

Optimization for this specific form of saddle-point problems has been extensively studied. For example, [60] and [61] performed batch updates on $\mathbf{w}$ and stochastic updates on $\boldsymbol{\alpha}$, while [62] and [63] performed *doubly* stochastic updates on *both* $\mathbf{w}$ and $\boldsymbol{\alpha}$, achieving $O(\frac{1}{\epsilon})$ and $O(\log \frac{1}{\epsilon})$ rates respectively. The latter two also studied the more general form Equation 2.22. In the next section we propose the SVRG algorithm with Bregman divergence (Breg-SVRG) that (a) provably optimizes strongly convex saddle function with a linear convergence rate, and (b) adapts to the underlying geometry by choosing an appropriate Bregman divergence. Then, we apply Breg-SVRG to multivariate prediction problems under F-score and DCG measure, and achieve a factor of n speedup over a straightforward instantiation of [63].

2.3.1 Bregman Divergence and Saddle Functions

In this section we set up some notations, recall some background materials, and extend Bregman divergences to saddle functions, a key notion in our later analysis.

Bregman divergence. For any convex and differentiable function ψ over some closed convex set $C \subseteq \mathbb{R}^d$, its induced Bregman divergence is defined as:

$$\forall x \in \operatorname{int}(C), x' \in C, \quad \Delta_\psi(x', x) := \psi(x') - \psi(x) - \left\langle \nabla\psi(x), x' - x \right\rangle, \tag{2.23}$$

where $\nabla\psi$ is the gradient. Clearly, $\Delta_\psi(x', x) \geq 0$ since ψ is convex. We mention two familiar examples of Bregman divergence.

- Squared Euclidean distance: $\Delta_\psi(x', x) = \frac{1}{2}\|x' - x\|_2^2\,,\ \psi(x) = \frac{1}{2}\|x\|_2^2$.
- (Unnormalized) KL-divergence: $\Delta_\psi(x', x) = \sum_i x'_i \log \frac{x'_i}{x_i} - x'_i + x_i, \psi(x) = \sum_i x_i \log x_i$.

Strong convexity. Following [64] we call a function f ψ-convex if $f - \psi$ is convex, i.e., for all x, x'

$$f(x') \geq f(x) + \langle \partial f(x), x' - x \rangle + \Delta_\psi(x', x). \tag{2.24}$$

Smoothness. A function f is L-smooth w.r.t. a norm $\|\cdot\|$ if its gradient ∇f is L-Lipschitz continuous, i.e., for all x and x', $\|\nabla f(x') - \nabla f(x)\|_* \leq L\|x' - x\|$, The change of a smooth function, in terms of its induced Bregman divergence, can be upper bounded by the change of its input and lower bounded by the change of its slope, cf. Lemma 5 in Appendix A.1.

Saddle functions. Recall that a function $\phi(x, y)$ over $C_{\mathsf{z}} = C_{\mathsf{x}} \times C_{\mathsf{y}}$ is called a saddle function if it is convex in x for any $y \in C_{\mathsf{y}}$, and concave in y for any $x \in C_{\mathsf{x}}$. Given a saddle function ϕ, we call (x^*, y^*) its saddle point if

$$\forall x \in C_{\mathsf{x}},\ \forall y \in C_{\mathsf{y}}, \quad \phi(x^*, y) \leq \phi(x^*, y^*) \leq \phi(x, y^*), \tag{2.25}$$

or equivalently $(x^*, y^*) \in \arg\min_{x \in C_{\mathsf{x}}} \max_{y \in C_{\mathsf{y}}} \phi(x, y)$. Assuming ϕ is differentiable, we denote

$$\mathsf{G}_\phi(x, y) := [\partial_x \phi(x, y); -\partial_y \phi(x, y)]. \tag{2.26}$$

Note the negation sign due to the concavity in y. We can quantify the notion of "saddle": A function $f(x, y)$ is called ϕ-saddle iff $f - \phi$ is a saddle function, or equivalently, $\Delta_f(z', z) \geq \Delta_\phi(z', z)$ (see below). Note that any saddle function ϕ is 0-saddle and ϕ-saddle.

Bregman divergence for saddle functions. We now define the Bregman divergence induced by a saddle function ϕ: for $z = (x, y)$ and $z' = (x', y')$ in C_{z},

$$\Delta_\phi(z', z) := \Delta_{\phi_y}(x', x) + \Delta_{-\phi_x}(y', y) = \phi(x', y) - \phi(x, y') - \left\langle \mathsf{G}_\phi(z), z' - z \right\rangle, \tag{2.27}$$

where $\phi_y(x) = \phi(x, y)$ is a convex function of x for any fixed y, and similarly $\phi_x(y) = \phi(x, y)$ is a concave (hence the negation) function of y for any fixed x. The similarity between Equation 2.27 and the usual Bregman divergence Δ_ψ in Equation 2.23 is apparent. However, ϕ is never evaluated at z' but z (for G) and the cross pairs (x', y) and (x, y'). Key to our subsequent analysis is the following lemma that extends a result of [65] to saddle functions (proof in Appendix A.1).

Lemma 1. *Let f and g be ϕ-saddle and φ-saddle respectively, with one of them being differentiable. Then, for any $z = (x, y)$ and any saddle point (if exists)*

$$z^* := (x^*, y^*) \in \arg\min_x \max_y \{f(z) + g(z)\},$$

we have $f(x, y^*) + g(x, y^*) \geq f(x^*, y) + g(x^*, y) + \Delta_{\phi+\varphi}(z, z^*)$.

Geometry of norms. In the sequel, we will design two convex functions $\psi_{\mathsf{x}}(x)$ and $\psi_{\mathsf{y}}(y)$ such that their induced Bregman divergences are "distance enforcing" (a.k.a. 1-strongly

convex), that is, w.r.t. two norms $\|\cdot\|_{\mathsf{x}}$ and $\|\cdot\|_{\mathsf{y}}$ that we also design, the following inequality holds:

$$\Delta_{\mathsf{x}}(x,x') := \Delta_{\psi_{\mathsf{x}}}(x,x') \geq \frac{1}{2}\left\|x-x'\right\|_{\mathsf{x}}^2,\ \Delta_{\mathsf{y}}(y,y') := \Delta_{\psi_{\mathsf{y}}}(y,y') \geq \frac{1}{2}\left\|y-y'\right\|_{\mathsf{y}}^2. \tag{2.28}$$

Further, for $z=(x,y)$, we define

$$\Delta_{\mathsf{z}}(z,z') := \Delta_{\psi_{\mathsf{x}}-\psi_{\mathsf{y}}}(z,z') \geq \frac{1}{2}\left\|z-z'\right\|_{\mathsf{z}}^2, \quad \text{where} \quad \|z\|_{\mathsf{z}}^2 := \|x\|_{\mathsf{x}}^2 + \|y\|_{\mathsf{y}}^2 \tag{2.29}$$

When it is clear from the context, we simply omit the subscripts and write Δ, $\|\cdot\|$, and $\|\cdot\|_*$.

2.3.2 Breg-SVRG Algorithm

Algorithm 1: Breg-SVRG for Saddle-Point Problems

Initialize z_0 randomly. Set $\tilde{z} = z_0$.
for $s = 1, 2, \ldots$ **do**
  $\tilde{\mu} \leftarrow \tilde{\mu}^s := \nabla K(\tilde{z})$, $z_0 \leftarrow z_0^s := z_m$
  for $t = 1, \ldots, m$ **do**
    Randomly pick $\xi \in \{1, \ldots, n\}$.
    Compute v_t using Equation 2.31.
    Update z_t using Equation 2.32.
  $\tilde{z} \leftarrow \tilde{z}^s := \sum_{t=1}^m (1+\eta)^t z_t / \sum_{t=1}^m (1+\eta)^t$.

In this section we propose an efficient algorithm for solving the general saddle-point problem in Equation 2.22 and prove its linear rate of convergence. Our main assumption is:

Assumption 1. *There exist two norms* $\|\cdot\|_{\mathsf{x}}$ *and* $\|\cdot\|_{\mathsf{y}}$ *such that each* ψ_k *is a saddle function and L-smooth; M is* $(\psi_{\mathsf{x}}, \psi_{\mathsf{y}})$*-saddle; and* ψ_{x} *and* ψ_{y} *are distance enforcing.*

Note that w.l.o.g. we have scaled the norms so that the usual strong convexity parameter of M is 1. Recall we defined $\|z\|_{\mathsf{z}}$ and Δ_{z} in Equation 2.29. For saddle-point optimization, it is common to define a signed gradient $\mathsf{G}(z) := [\partial_x K(z); -\partial_y K(z)]$ (since K is concave in y). Recall $J = K + M$, and (x^*, y^*) is a saddle-point of J. Using Assumption 1, we measure the gap of an iterate $z_t = (x_t, y_t)$ as follows:

$$\epsilon_t = \epsilon(z_t) = J(x_t, y^*) - J(x^*, y_t) \geq \Delta(z_t, z^*) \geq \frac{1}{2}\|z_t - z^*\|^2 \geq 0. \tag{2.30}$$

Inspired by [63, 66, 67], we propose in Algorithm 1 a new stochastic variance-reduced algorithm for solving the saddle-point problem Equation 2.22 using Bregman divergences. The algorithm proceeds in epochs. In each epoch, we first compute the following stochastic estimate of the signed gradient $G(z_t)$ by drawing a random component from K:

$$v_t = \begin{pmatrix} v_x(z_t) \\ -v_y(z_t) \end{pmatrix} \quad \text{where} \quad \begin{cases} v_x(z_t) := \partial_x \psi_\xi(z_t) - \partial_x \psi_\xi(\tilde{z}) + \partial_x K(\tilde{z}) \\ v_y(z_t) := \partial_y \psi_\xi(z_t) - \partial_y \psi_\xi(\tilde{z}) + \partial_y K(\tilde{z}) \end{cases}. \tag{2.31}$$

Here $\tilde{z}$ is the pivot chosen after completing the previous epoch. We make two important observations: (1) By construction the stochastic gradient v_t is unbiased: $\mathbb{E}_\xi[v_t] = \mathsf{G}(z_t)$; (2) The expensive gradient evaluation $\partial K(\tilde{z})$ need only be computed once in each epoch since $\tilde{z}$

is held unchanged. If $\tilde{z} \to z^*$, then the variance of v_t would be largely reduced hence faster convergence may be possible.

Next, Algorithm 1 performs the following *joint* proximal update:

$$(x_{t+1}, y_{t+1}) = \arg\min_x \max_y \eta \langle v_x(z_t), x\rangle + \eta \langle v_y(z_t), y\rangle + \eta M(x,y) + \Delta(x, x_t) - \Delta(y, y_t), \quad (2.32)$$

where we have the flexibility in choosing a suitable Bregman divergence to better adapt to the underlying geometry. When $\Delta(x, x_t) = \frac{1}{2}\|x - x_t\|_2^2$, we recover the special case in [63]. However, to handle the asymmetry in a general Bregman divergence (which does not appear for the Euclidean distance), we have to choose the pivot $\tilde{z}$ in a significantly different way than [63, 66, 67].

We are now ready to present our main convergence guarantee for Breg-SVRG in Algorithm 1.

Theorem 2. *Let Assumption 1 hold, and choose a sufficiently small $\eta > 0$ such that $m := \left\lceil \log\left(\frac{1-\eta L}{18\eta L^2} - \eta - 1\right)/\log(1+\eta)\right\rceil \geq 1$. Then Breg-SVRG enjoys linear convergence in expectation:*

$$\mathbb{E}\epsilon(\tilde{z}^s) \leq (1+\eta)^{-ms}[\Delta(z^*, z_0) + c(Z+1)\epsilon(z_0)],$$

where $Z = \sum_{t=0}^{m-1}(1+\eta)^t$, $c = \frac{18\eta^2 L^2}{1-\eta L}$.

For example, we may set $\eta = \frac{1}{45L^2}$, which leads to $c = O(1/L^2)$, $m = \boldsymbol{\theta}\left(L^2\right)$, $(1+\eta)^m \geq \frac{64}{45}$, and $Z = O(L^2)$. Therefore, between epochs, the gap $\epsilon(\tilde{z}^s)$ decays (in expectation) by a factor of $\frac{45}{64}$, and each epoch needs to conduct the proximal update Equation 2.32 for $m = \boldsymbol{\theta}(L^2)$

number of times. (We remind that w.l.o.g. we have scaled the norms so that the usual strong convexity parameter is 1.) In total, to reduce the gap below some threshold ϵ, Breg-SVRG needs to call the proximal update Equation 2.32 $O(L^2 \log \frac{1}{\epsilon})$ number of times, plus a similar number of *component* gradient evaluations.

Discussions. As mentioned, Algorithm 1 and Theorem 2 extend those in [63] which in turn extend [66, 67] to saddle-point problems. However, [63, 66, 67] all heavily exploit the Euclidean structure (in particular symmetry) hence their proofs cannot be applied to an asymmetric Bregman divergence. Our innovations here include: (a) A new Pythagorean theorem for the newly introduced saddle Bregman divergence (Lemma 1). (b) A moderate extension of the variance reduction lemma in [67] to accommodate any norm (Appendix A.2). (c) A different pivot $\tilde{z}$ is adopted in each epoch to handle asymmetry. (d) A new analysis technique through introducing a crucial auxiliary variable that enables us to bound the function gap directly. See our proof in Appendix A.3 for more details. Compared with classical mirror descent algorithms [59, 62] that can also solve saddle-point problems with Bregman divergences, our analysis is fundamentally different and we achieve the significantly stronger rate $O(\log(1/\epsilon)$ than the sublinear $O(1/\epsilon)$ rate of [62], at the expense of a squared instead of linear dependence on L. Similar tradeoff also appeared in [63]. We will return to this issue in Section 2.4.

Variants and acceleration. Our analysis also supports to use different ξ in v_x and v_y. The standard acceleration methods such as universal catalyst [68] and non-uniform sampling can be applied directly (see Appendix A.5 where L, the largest smoothness constant over all pieces, is replaced by their mean).

2.4 Computational Complexity on Adversarial Prediction

The quadratic dependence on L, the smoothness parameter, in Theorem 2 reinforces the need to choose suitable Bregman divergences. In this section we illustrate how this can be achieved for the adversarial prediction problem in Section 2.1. As pointed out in [63], the factorization of K is important, and we consider three schemes: (a) $\psi_k = f_{ij}$; (b) $\psi_k = \frac{1}{n}\sum_{j=1}^{n} f_{k,j}$; and (c) $\psi_k = \frac{1}{n}\sum_{i=1}^{n} f_{i,k}$. W.l.o.g. let us fix the μ in Equation 2.14 to 1.

Comparison of smoothness constant. Both $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are n^2-dimensional, and the bilinear function f_{ij} can be written as $\boldsymbol{\alpha}^T A_{ij}\boldsymbol{\beta}$, where $A_{ij} \in \mathbb{R}^{n^2\times n^2}$ is an n-by-n block matrix, with the (i,j)-th block being $n^2\frac{2ij}{i+j}I$ and all other blocks being $\mathbf{0}$.

For scheme (a), the smoothness constant L_2 under ℓ_2 norm depends on the spectral norm of A_{ij}: $L_2 = \max_{i,j} n^2\frac{2ij}{i+j} = \Theta(n^3)$. In contrast the smoothness constant L_1 under ℓ_1 norm depends on the absolute value of the entries in A_{ij}: $L_1 = \max_{i,j} n^2\frac{2ij}{i+j} = \Theta(n^3)$; no saving is achieved.

For scheme (b), the bilinear function ψ_k corresponds to $\frac{1}{n}\boldsymbol{\alpha}^T\sum_{j=1}^{n} A_{kj}\boldsymbol{\beta}$. Then $L_1 = O(n^2)$ while

$$L_2^2 = \frac{1}{n^2}\max_k \max_{\mathbf{v}:\|\mathbf{v}\|_2=1} \sum\nolimits_{j=1}^{n} \left\|A_{kj}\mathbf{v}\right\|_2^2 = n^5. \tag{2.33}$$

Comparison of smoothness constant for the overall F-score problem. By strong duality, we may push the maximization over $\boldsymbol{\theta}$ to the innermost level of Equation 2.6, arriving at an overall problem in $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ only:

$$\min_{\{\boldsymbol{\alpha}_i\}\in S}\max_{\{\boldsymbol{\beta}_j\}\in S}\ \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\left[f_{ij}(\boldsymbol{\alpha}_i,\boldsymbol{\beta}_j)-\frac{i}{\lambda n}\mathbf{c}^T X\boldsymbol{\alpha}_i+\frac{ij}{2\lambda}\boldsymbol{\alpha}_i^T X^T X\boldsymbol{\alpha}_j+\frac{1}{2\lambda n^2}\left\|\mathbf{c}\right\|_2^2\right]. \tag{2.34}$$

where $\mathbf{c}=X\tilde{\mathbf{y}}$. The quadratic term w.r.t. $\boldsymbol{\alpha}$ can be written as $\frac{1}{2}\boldsymbol{\alpha}^T B\boldsymbol{\alpha}$, where $B\in\mathbb{R}^{n^2\times n^2}$ is an n-by-n block matrix, with its (i,j)-th block being $\frac{ij}{\lambda}X^TX$. And we assume each $\left\|\mathbf{x}_i\right\|_2\leq 1$. The full gradient over $\boldsymbol{\alpha}$ is $B\boldsymbol{\alpha}$ as B is a symmetric matrix, then we can individually split the gradient into $B_{ij}\boldsymbol{\alpha}$, where B_{ij} only keeps the (i,j)-th block and other blocks being 0, this splitting corresponds to scheme (a). Besides, we have factored splitting $\frac{1}{n}\sum_{j=1}^{n}B_{kj}\boldsymbol{\alpha}$ that corresponds to scheme (b). Note there is no splitting function f_{ij} corresponding to the splitting gradient. The smoothness constant can be bounded separately from A_{ij} and B_{ij}; see Equation A.104 in Appendix A.6.

For scheme (a), the smoothness constant square L_2^2 under ℓ_2 norm is upper bounded by the sum of spectral norm square of A_{ij} and B_{ij}. So $L_2^2\geq\max_{i,j}\left(\frac{ij}{\lambda}n\right)^2=\Omega(n^6)$, i.e. $L_2=\Theta(n^3)$. In contrast the smoothness constant square L_1^2 under ℓ_1 norm is at most the sum of square of maximum absolute value of the entries in A_{ij} and B_{ij}. Hence $L_1^2\leq\max_{i,j}\left(n^2\frac{2ij}{i+j}\right)^2+\max_{i,j}\left(\frac{ij}{\lambda}\right)^2=\Theta(n^6)$, i.e. $L_1=\Theta(n^3)$. So no saving is achieved here.

For scheme (b),

$$L_1^2 \le \frac{1}{n^2}\max_k \left[\max_{\mathbf{v}:\|\mathbf{v}\|_1=1} \left\| \sum_{j=1}^n A_{kj}\mathbf{v} \right\|_\infty^2 + \max_{\mathbf{v}:\|\mathbf{v}\|_1=1} \left\| \sum_{j=1}^n B_{kj}\mathbf{v} \right\|_\infty^2 \right] \quad \text{(by Equation A.104)} \tag{2.35}$$

$$\le \frac{1}{n^2}\max_k \max_j \left[\left(n^2(1+\frac{2kj}{k+j}) \right)^2 + \left(\frac{kj}{\lambda} \right)^2 \right] = n^4, \tag{2.36}$$

and by setting $\boldsymbol{\beta}$ to $\mathbf{0}$ in Equation A.102, we get $L_2^2 \ge n^5$ by Equation 2.33. Therefore, L_1^2 saves a factor of n compared with L_2^2. Similar results apply to scheme (c) too. We also tried non-uniform sampling, but it does not change the order in n. It can also be shown that if our scheme randomly samples n entries from $\{A_{ij}, B_{ij}\}$, the above L_1 and L_2 cannot be improved by further engineering the factorization.

Comparison of smoothness constant for DCG problem. Denoting $\mathbf{s}^k = \frac{X^k \tilde{y}^k - X^k \sum_{i=1}^{n_k} i\boldsymbol{\alpha}_i^k}{n_k}$, we can rewrite Equation 2.21 as:

$$\min_{\{\boldsymbol{\alpha}^k\}_M \in S_\alpha} \max_{\{\boldsymbol{\beta}^k\}_M \in S_\beta} \quad \frac{1}{M}\sum_{k=1}^M \left[\sum_{i=1}^{n_k}\sum_{j=1}^{n_k} \frac{i\boldsymbol{\alpha}_i^{kT}\boldsymbol{\beta}_j^k}{\log_2(j+1)} \right] + \frac{1}{2\lambda}\left\| \frac{1}{M}\sum_{k=1}^M \mathbf{s}^k \right\|^2. \tag{2.37}$$

We do uniform sampling on M queries,, the bilinear function f_s can be written as $\boldsymbol{\alpha}^{sT} A^s \boldsymbol{\beta}^s$, where $\boldsymbol{\alpha}^s$ and $\boldsymbol{\beta}^s$ are n_s^2-dimensional, $A^s \in \mathbb{R}^{n_s^2 \times n_s^2}$ is an n_s-by-n_s block matrix (n_s is the number of documents in sth query), with the (i,j)-th block being $\frac{i}{log(j+1)} I$.

The quadratic term w.r.t. $\boldsymbol{\alpha}$ is more tricky. Let us first stack all $\boldsymbol{\alpha}^s$ as a vector $\boldsymbol{\alpha}$ of length $\mathfrak{L} = \sum_{k=1}^{M} n_k^2$, so the second term in Equation 2.37 can be written as $\frac{1}{2}\boldsymbol{\alpha}^T B \boldsymbol{\alpha}$, where $B \in \mathbb{R}^{\mathfrak{L}\times\mathfrak{L}}$ is an M-by-M block matrix, with its (i,j)-th block being $B_{ij} \in \mathbb{R}^{n_i^2 \times n_j^2}$. Each B_{ij} itself is a block matrix with (x,y)-th block being $\frac{xy{X^i}^T X^j}{n_i n_j \lambda}$ $(x \le n_i, y \le n_j)$. And we assume each $\|\mathbf{x}_i\|_2 \le 1$. The full gradient over $\boldsymbol{\alpha}$ is $B\boldsymbol{\alpha}$ as B is a symmetric matrix, so we can split the gradient into $B^s\boldsymbol{\alpha}$, where B^s only keeps the s-th column and other columns being 0, s is uniformly sampled from M queries. Note there is no splitting function corresponding to this splitting gradient.

Let $n_{max} = \max_s n_s, \forall s \in \{1,\dots,M\}$, i.e., the largest number of documents in queries. Then L_1 and L_2 can be bounded as follows:

$$L_1^2 \le \max_{i,j \le n_{max}} \left((\frac{i}{\log(j+1)})^2 + 1 \right) = n_{max}^2, \tag{2.38}$$

$$L_2^2 \ge \max_{\mathbf{v}:\|\mathbf{v}\|_2=1} \left\| A^s \mathbf{v} \right\|_2^2 = n_{max}^3 \tag{2.39}$$

Therefore, L_1^2 saves a factor of n_{max} compared with L_2^2.

Computational complexity. We finally seek efficient algorithms for the proximal update Equation 2.32 used by Breg-SVRG. When $M(\boldsymbol{\alpha},\boldsymbol{\beta}) = \Omega(\boldsymbol{\alpha}) - \Omega(\boldsymbol{\beta})$ as in Equation 2.14, we can solve $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ separately as:

$$\min_{\boldsymbol{\alpha}} \sum_{ik} \alpha_{ik} \log(\alpha_{ik}/b_{ik}) - c_{ik}, \quad \text{s.t.} \quad \mathbf{1}^T \boldsymbol{\alpha} \mathbf{1} \leq 1, \ \forall i \ \forall k, \ 0 \leq i\alpha_{ik} \leq \mathbf{1}^T \boldsymbol{\alpha}_i. \tag{2.40}$$

where b_{ik} and c_{ik} are constants. In Appendix A.4 we designed an efficient "closed form" algorithm which finds an ϵ accurate solution in $O(n^2 \log^2 \frac{1}{\epsilon})$ time, which is also on par with that for computing the stochastic gradient in schemes (b) and (c). Although scheme (a) reduces the cost of gradient computation to $O(n)$, its corresponding smoothness parameter L_1^2 is increased by n^2 times, hence not worthwhile. We did manage to design an $\tilde{O}(n)$ algorithm for the proximal update in scheme (a), but empirically the overall convergence is rather slow.

If we use the Euclidean squared distance as the Bregman divergence, then a term $\|\boldsymbol{\alpha} - \boldsymbol{\alpha}_t\|_2^2$ needs to be added to the objective Equation 2.40. No efficient "closed form" solution is available, and so in experiments we simply absorbed M into K, and then the proximal update becomes the Euclidean projection onto S, which does admit a competitive $O(n^2 \log^2(1/\epsilon))$ time solution.

2.5 Experiments

Our major goal in this section is to show that empirically Entropy-SVRG (Breg-SVRG with KL divergence) is significantly more efficient than Euclidean-SVRG (Breg-SVRG with squared Euclidean distance) on some learning problems, especially those with an entropic regularizer and a simplex constraint.

2.5.1 Entropy Regularized LPBoost

We applied Breg-SVRG to an extension of LP Boosting using entropy regularization [69]. In a binary classification setting, the base hypotheses over the training set can be compactly represented as $U = (y_1\mathbf{x}_1, \ldots, y_n\mathbf{x}_n)^T$. Then the model considers a minimax game between a distribution $\mathbf{d} \in \Delta^n$ over training examples and a distribution $\mathbf{w} \in \Delta^p$ over the hypotheses:

$$\min_{\mathbf{d}\in\Delta^n, d_i\le\nu} \quad \max_{\mathbf{w}\in\Delta^p} \mathbf{d}^T U\mathbf{w} + \mu\Omega(\mathbf{d}) - \gamma\Omega(\mathbf{w}). \tag{2.41}$$

Here $\mathbf{w}$ tries to combine the hypotheses to maximize the edge (prediction confidence) $y_i\mathbf{x}_i^T\mathbf{w}$, while the adversary $\mathbf{d}$ tries to place more weights (bounded by ν) on "hard" examples to reduce the edge.

Settings. We experimented on the adult dataset from the UCI repository, which we partitioned into $n = 32,561$ training examples and 16,281 test examples, with $p = 123$ features. We set $\mu = \gamma = 0.01$ and $\nu = 0.1$ due to its best prediction accuracy. We tried a range of values of the step size η, and the best we found was 10^{-3} for Entropy-SVRG and 10^{-6} for Euclidean-SVRG (larger step size for Euclidean-SVRG fluctuated even worse). For both methods, $m = 32561/50$ gave good results.

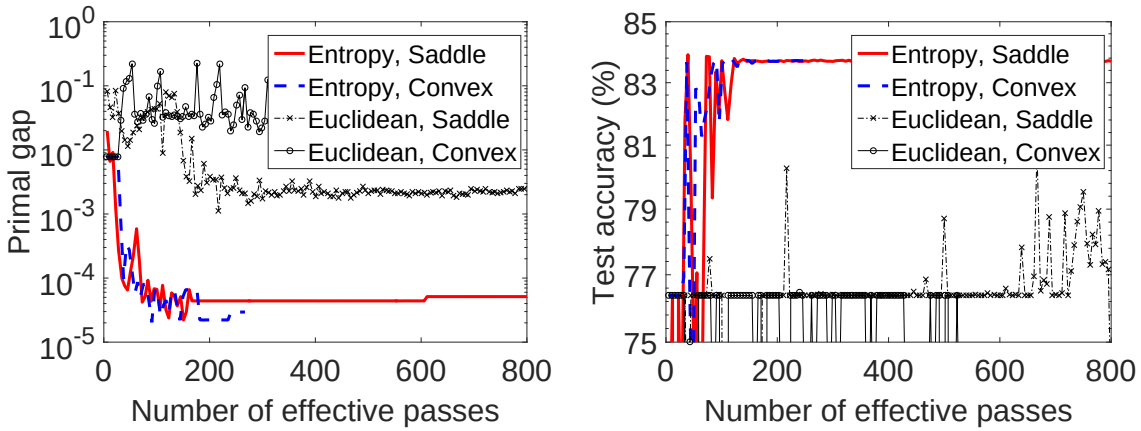
The stochastic gradient in $\mathbf{d}$ was computed by $U_{:j}w_j$, where $U_{:j}$ is the j-th column and j is randomly sampled. The stochastic gradient in $\mathbf{w}$ is $d_iU_{i:}^T$. We tried with $U_{ij}w_j$ and $U_{ij}d_i$ (scheme (a) in §2.4), but they performed worse. We also tried with the universal catalyst in the same form as [63], which can be directly extended to Entropy-SVRG. Similarly we used the

non-uniform sampling based on the ℓ_2 norm of the rows and columns of U. It turned out that the Euclidean-SVRG can benefit slightly from it, while Entropy-SVRG does not. So we only show the "accelerated" results for the former.

To make the computational cost comparable across machines, we introduced a counter called effective number of passes: #pass. Assume the proximal operator has been called #po number of times, then

$$\#\text{pass} \;:=\; \text{number of epochs so far} \;+\; \tfrac{n+p}{np} \cdot \#\text{po}. \tag{2.42}$$

We also compared with a "convex" approach. Given $\mathbf{d}$, the optimal $\mathbf{w}$ in Equation 2.41 obviously admits a closed-form solution. General saddle-point problems certainly do not enjoy such a convenience. However, we hope to take advantage of this opportunity to study the following question: suppose we solve Equation 2.41 as a convex optimization in $\mathbf{d}$ and the stochastic gradient were computed from the optimal $\mathbf{w}$, would it be faster than the saddle SVRG? Since solving $\mathbf{w}$ requires visiting the entire U, strictly speaking the term $\frac{n+p}{np}\cdot$#po in the definition of #pass in Equation 2.42 should be replaced by #po. However, we stuck with Equation 2.42 because our interest is whether a more accurate stochastic gradient in $\mathbf{d}$ (based on the optimal $\mathbf{w}$) can outperform doubly stochastic (saddle) optimization. We emphasize that this comparison is only for conceptual understanding, because generally optimizing the inner variable requires costly iterative methods.



(a) Primal gap v.s. #pass

(b) Test accuracy v.s. #pass

Figure 2: Entropy Regularized LPBoost on adult dataset

Results. Figure 2a demonstrated how fast the *primal gap* (with $\mathbf{w}$ optimized out for each $\mathbf{d}$) is reduced as a function of the number of effective passes. Methods based on entropic prox are clearly much more efficient than Euclidean prox. This corroborates our theory that for problems like Equation 2.41, Entropy-SVRG is more suitable for the underlying geometry (entropic regularizer with simplex constraints).

We also observed that using entropic prox, our doubly stochastic method is as efficient as the "convex" method, meaning that although at each iteration the $\mathbf{w}$ in saddle SVRG is not the optimal for the current $\mathbf{d}$, it still allows the overall algorithm to perform as fast as if it were. This suggests that for general saddle-point problems where no closed-form inner solution is available, our method will still be efficient and competitive. Note this "convex" method is similar to the optimizer used by [69].

Finally, we investigated the increase of test accuracy as more passes over the data are performed. Figure 2b shows that the entropic prox does allow the accuracy to be improved much faster than Euclidean prox. Again, the convex and saddle methods perform similarly.

As a final note, the Euclidean/entropic proximal operator for both $\mathbf{d}$ and $\mathbf{w}$ can be solved in either closed form, or by a 1-D line search based on partial Lagrangian. So their computational cost differ in the same order of magnitude as multiplication v.s. exponentiation, which is much smaller than the difference of #pass shown in Figure 2.

2.5.2 Adversarial Prediction with F-score

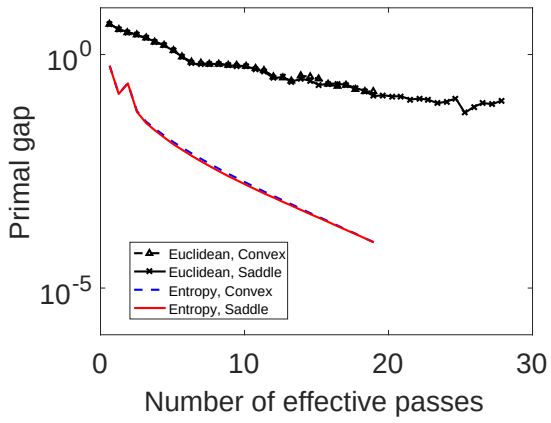
Datasets. Here we considered three datasets from the UCI repository.

The first dataset, adult, has 1000 training samples (239 pos and 761 neg), 1000 testing samples (237 pos and 763 neg). Each example has 120 features.

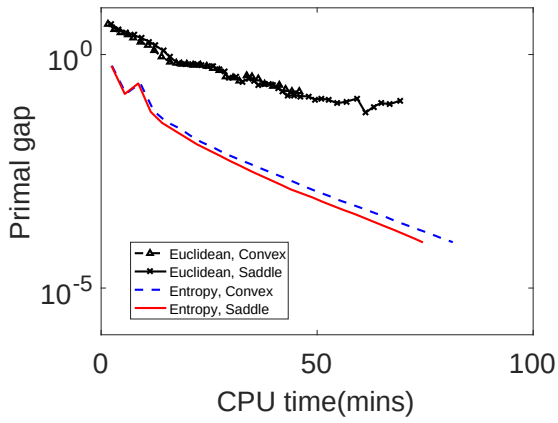
The second dataset, splice, has 1000 training samples (517 pos and 483 neg), 725 testing samples (377 pos and 348 neg). Each example has 61 features.

The third dataset is optdigits. We choose digit 5 versus all the other digits as digit 5 is relatively hard to classify. The dataset has 1000 training samples (518 pos and 482 neg), 1000 testing samples (518 pos and 482 neg). Each example has 56 features.

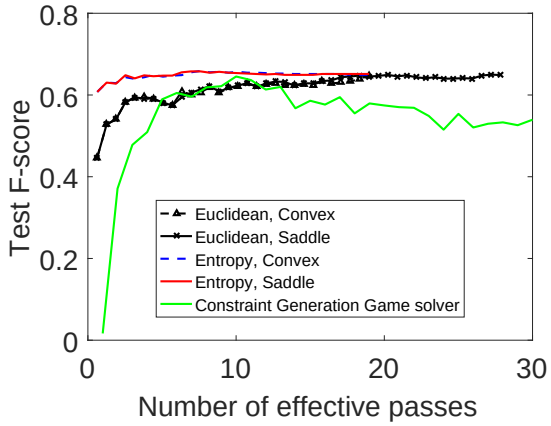
Methods. To apply saddle SVRG, we used strong duality to push the optimization over $\boldsymbol{\theta}$ to the inner-most level of Equation 2.6, and then eliminated $\boldsymbol{\theta}$ because it is a simple quadratic. So we ended up with the convex-concave optimization as shown in Equation 2.34, where the K part of Equation 2.13 is augmented with a quadratic term in $\boldsymbol{\alpha}$. The formulae for computing the stochastic gradient using scheme (b) are detailed in Appendix A.7.



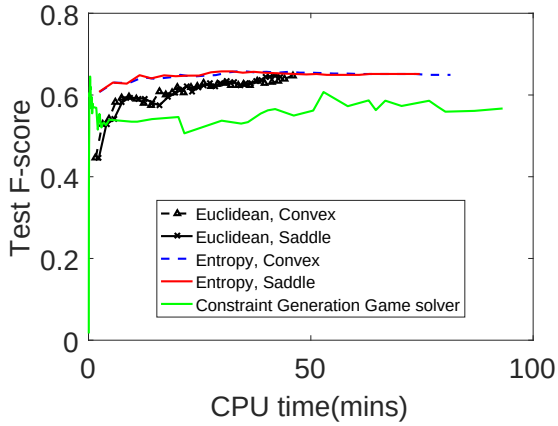
(a) Primal gap v.s. #pass



(b) Primal gap v.s. CPU time



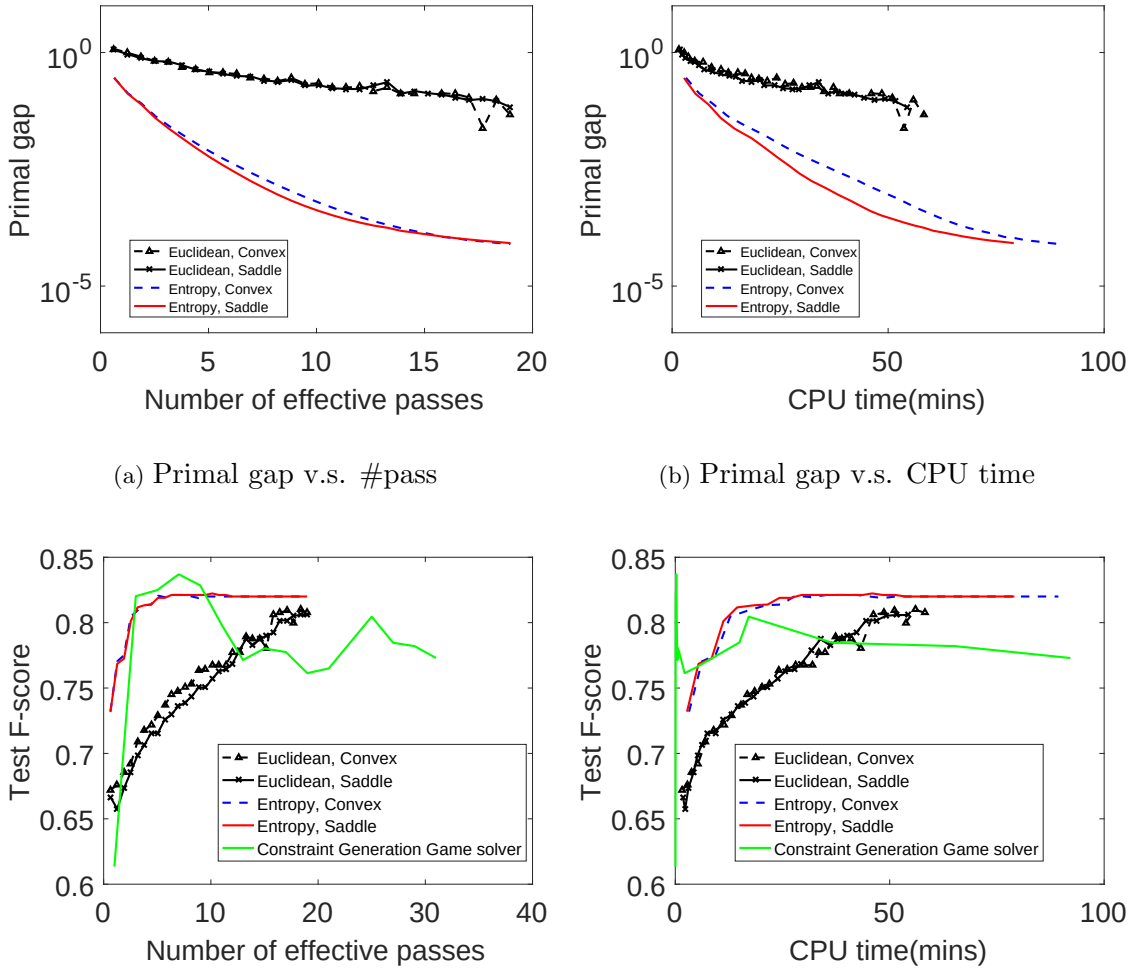
(c) Test F-score v.s. #pass



(d) Test F-score v.s. CPU time

Figure 3: F-score prediction on the adult dataset.

We also compared our method to the Constraint Generation Game solver. [30, 31, 70] employed L-BFGS to optimize over $\boldsymbol{\theta}$ where the gradient is gathered by solving the minimax game via double oracle method. In each iteration of double oracle algorithm, it iteratively finds best response actions based on the whole game matrix until finding the equilibrium. Therefore, the



(a) Primal gap v.s. #pass

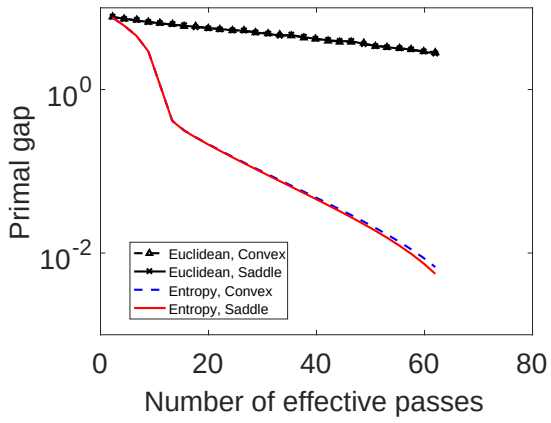
(b) Primal gap v.s. CPU time

(c) Test F-score v.s. #pass

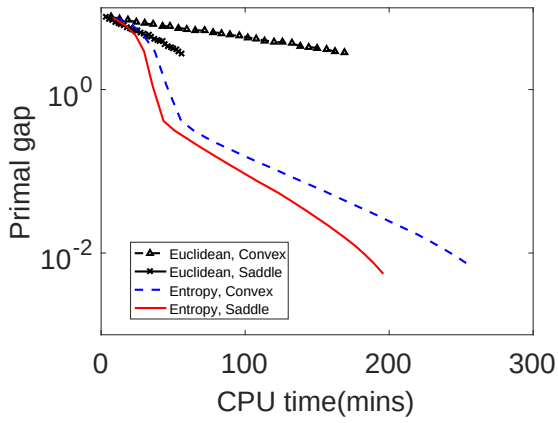
(d) Test F-score v.s. CPU time

Figure 4: F-score prediction on the splice dataset.

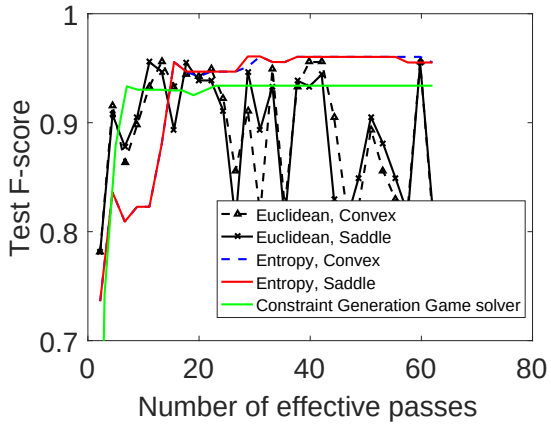
effective number of passes is at least the number of times that the best response function has been called. Here we treat each iteration of L-BFGS as one effective pass, which will better polish the plot of their method, but our result is still better than theirs in terms of effective number of passes.



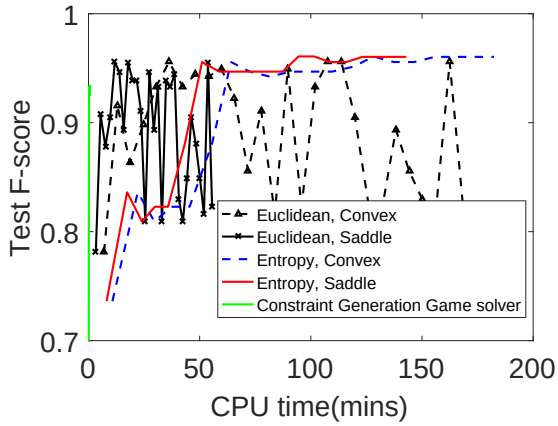
(a) Primal gap v.s. #pass



(b) Primal gap v.s. CPU time



(c) Test F-score v.s. #pass



(d) Test F-score v.s. CPU time

Figure 5: F-score prediction on the optdigits dataset.

Parameter setting. For splice and adult datasets we set strongly convex constant $\mu = 1$, regularizer constant $\lambda = 0.03$, epoch length $m = \sqrt{n}$ and minibatch size $mb = \sqrt{n}$. For optdigits dataset, in order to fit the data and have a better generalization, we set a lower regularizer constant $\lambda = 0.01$. Since each entry in the $n \times n$ matrix $\boldsymbol{\alpha}$ is relatively small when n is large,

		adult	splice	optdigits
	strongly convexity μ and γ	1		
	regularizer constant λ on $\boldsymbol{\theta}$	0.03	0.03	0.01
Euclidean	convex_svrg	1e-9	9e-10	1e-10
	saddle_svrg	1e-9	9e-10	1e-10
Entropy	convex_svrg	1e-2	1e-2	1e-2
	saddle_svrg	1e-2	1e-2	1e-2

TABLE I: Parameter setting on F-score experiments

we needed a relatively small step size, see Table I. For Constaint Generation Game solver, we find the best regularizer constant λ is 0.01.

Results. The results on the three datasets are shown in Figure 3, Figure 4, Figure 5, respectively. We truncated the #pass and CPU time in subplots (c) and (d) because the F-score has stabilized and we would rather zoom in to see the initial growing phase. In terms of primal gap versus #pass (subplot a), the entropy based method is significantly more effective than Euclidean methods on all three datasets. The CPU time comparisons (subplot b) follow the similar trend, except that the "convex methods" should be ignored because they are introduced only to compare #pass.

The F-score is noisy because, as is well known, it is not monotonic with the primal gap and glitches can appear. In subplots Figure 3c, Figure 3d, Figure 4d and Figure 4c, the entropy based

methods achieve higher F-score significantly faster than the plain Euclidean based methods on splice and adult datasets. In terms of the optdigits dataset, the F-score is noisy under Euclidean based methods while it is gradually increasing under the entropy based methods (Figure 5c and Figure 5d). Note that we used lower regularizer constant $\lambda = 0.01$ in this dataset, thereby it requires more #passes and CPU time to converge. At the beginning, Euclidean based methods are far away from the optimal solution, see Figure 5a, therefore, the F-score under Euclidean based methods is noisy. While the entropy based methods approach to the optimal solution with smaller primal gap, we see the F-score becomes stable after 20 effective passes in Figure 5c.

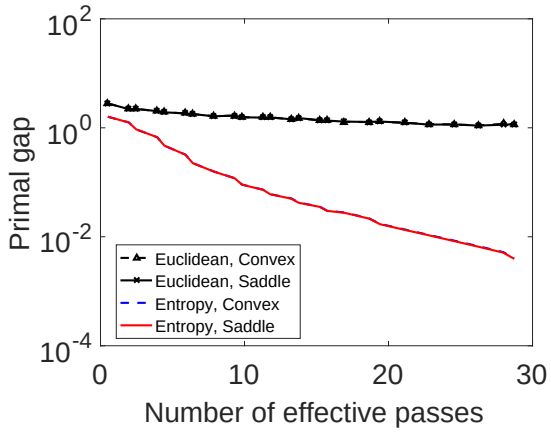
The F-score of Constraint Generation Game solver fluctuates a lot in subplots c and d, the reason is that the double oracle algorithm can only find approximate equilibrium under the limited number of expanding actions. Within reasonable time, our stochastic optimization methods achieves a higher and more stable F-score as it converges to the optimal equilibrium.

2.5.3 Adversarial Prediction with DCG

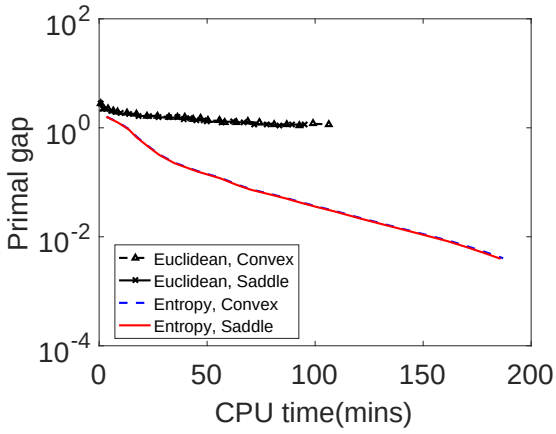
Datasets. We considered the query-document pairs from the million query TREC 2008 (MQ2008) dataset of LETOR4.0 [71] and Yahoo! Learn to Rank Challenge dataset [1]. However there are some invalid queries, in which all documents are 0 relevant. So we eliminated those invalid queries. MQ2008 has 354 training queries and 105 test queries (21.8 documents on average per query, 46 features per document). Yahoo has 500 queries, 200 test queries (28.03 documents on average per query, 700 features per document).

---

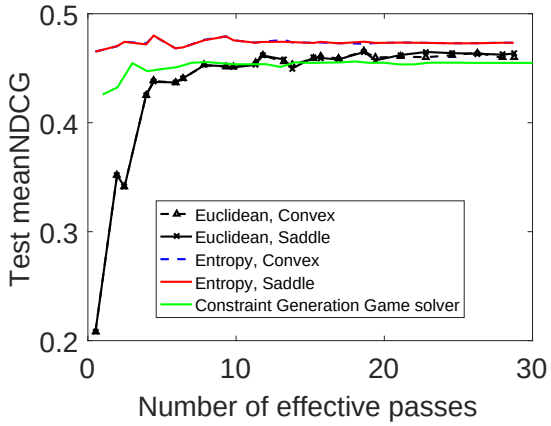
[1] Yahoo! Learn to Rank Challenge version 2.0 [https://webscope.sandbox.yahoo.com/]



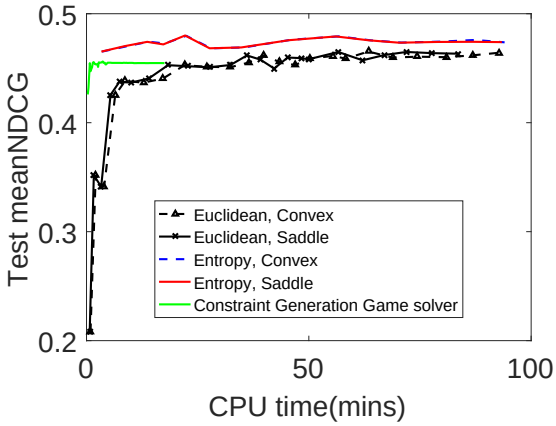
(a) Primal gap v.s. #pass



(b) Primal gap v.s. CPU time



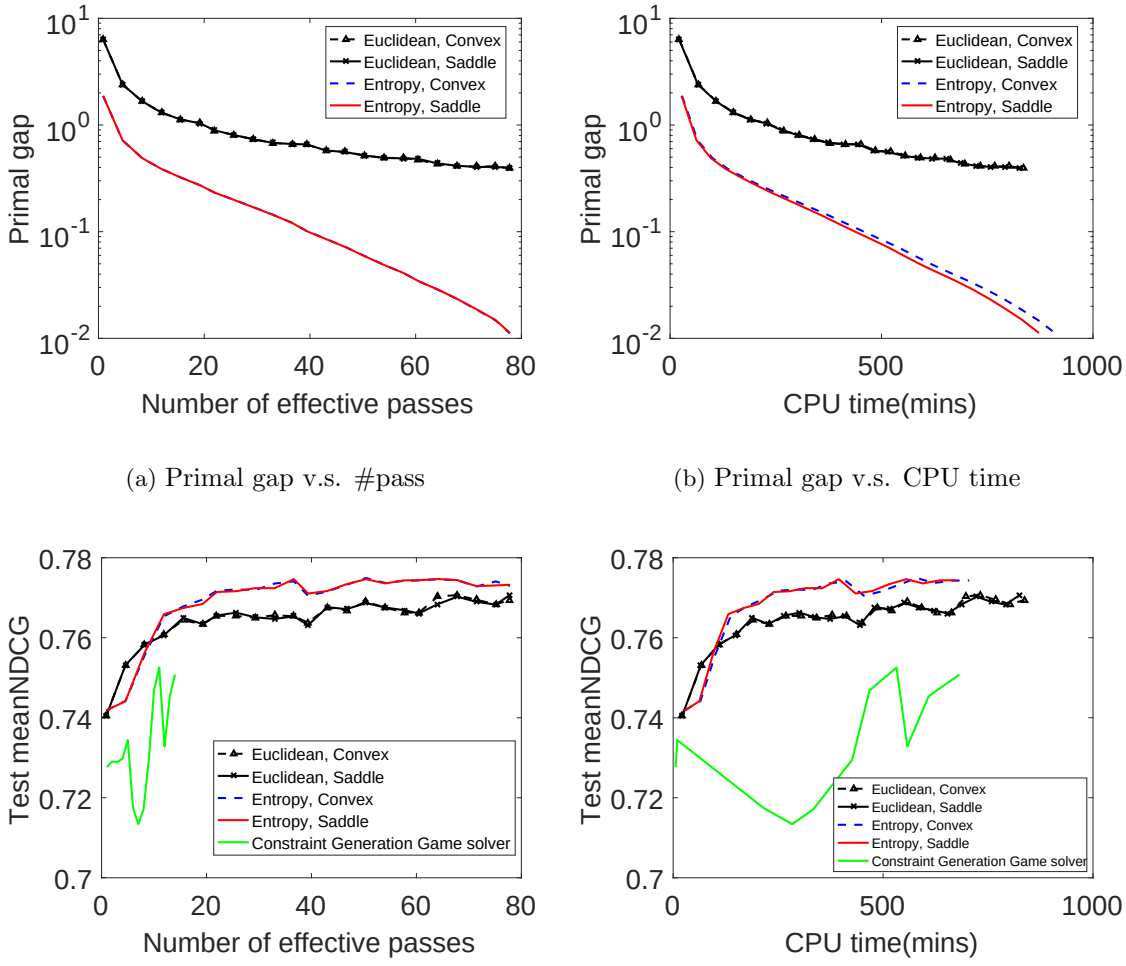
(c) Test F-score v.s. #pass



(d) Test F-score v.s. CPU time

Figure 6: NDCG prediction on the mq2008 dataset.

Methods. To apply saddle SVRG, we used strong duality to push the optimization over $\boldsymbol{\theta}$ to the inner minimax problem and eventually ended up with the convex-concave optimization as shown in Equation 2.21, The formulae for computing the stochastic gradient are detailed in



(a) Primal gap v.s. #pass

(b) Primal gap v.s. CPU time

(c) Test F-score v.s. #pass

(d) Test F-score v.s. CPU time

Figure 7: NDCG prediction on the Yahoo dataset.

A.8. To be consistent in each query, we measure performance using Normalized DCG (NDCG), which divides the realized DCG by the maximum possible DCG for the dataset.

Parameter setting. We set strongly convex constant $\mu = 1$, regularizer constant $\lambda = 0.01$, epoch length $m = \sqrt{n}$ and minibatch size $mb = \sqrt{n}$. The best step size we found for MQ2008

is 2 under entropy based methods and 8e-4 under Euclidean based methods. The best step size for Yahoo dataset is 0.3 under entropy based methods and 3e-4 for Euclidean based methods.

Results. The results on two datasets are shown in Figure 6, Figure 7, respectively. In terms of primal gap versus #pass (subplot a), the entropy based method is significantly more effective than Euclidean methods on both datasets. The CPU time comparisons (subplot b) follow the similar trend. In subplots Figure 6d and Figure 6c, the entropy based methods achieve higher meanNDCG significantly faster than the plain Euclidean based methods on mq2008 dataset. In terms of Yahoo dataset, after 10 effective passes (Figure 7c), the entropy based methods achieve higher meanNDCG significantly than the Euclidean based methods. Compared with Constraint Generation Game solver, our stochastic adversarial method can achieve better meanNDCG after stable.

CHAPTER 3

DISTRIBUTIONALLY ROBUST OPTIMIZATION WITH WASSERSTEIN DISTANCE

(Parts of this chapter were previously published as "Generalised Lipschitz Regularisation Equals Distributional Robustness" [1], in *International Conference on Machine Learning (ICML)*, 2021.)

3.1 Distributionally Robust Optimization

Distributionally robust optimization (DRO) minimizes the worst-case expected risk over a large set of distributions centered at the empirical distribution. It is in the form of following minimax problem

$$\inf_{f} \sup_{\mathbb{Q}\in\mathcal{B}(\hat{\mathbb{P}}_n)} \mathbb{E}_{\mathbb{Q}}\ell(f(\mathbf{x}), y), \tag{3.1}$$

where $\mathcal{B}$ is a set of distributions around the empirical distribution $\hat{\mathbb{P}}_n$. We call the distribution $\mathbb{Q} \in \mathcal{B}$ as *adversarial distribution*, and the worst-case expected risk w.r.t $\mathbb{Q}$ as *distributionally robust risk* (DRR). If $\mathcal{B}$ is large enough to contain the data-generating distribution $\mathbb{P}$, then we implicitly optimize for $\mathbb{P}$ via solving the DRO and the DRR provides an upper confidence bound on the expected risk. Thus, the constraint set $\mathcal{B}$ plays a key role in controlling the robustness of the model.

Many different choices of the constraint set $\mathcal{B}$ have been discussed in the literature. Here we introduce two main categories that are based on moment matching and distance function.

Constraining adversarial distribution by moment matching. [72] studied a restricted constraint set $\mathcal{B}$ that fixes the first and second moments of the class-conditional distribution. The DRR in that case can be explicitly upper bounded, yielding a simple convex optimization problem. Interestingly, the APM in Equation 2.2 can also be categorized as a DRO with the moment matching ambiguity set that matches the cross-moments of $\mathbb{Q}$ with those of the empirical distribution while keeping the same marginal $\mathbb{Q}(\mathbf{x})$ as the empirical marginal $\hat{\mathbb{P}}_n(\mathbf{x})$.

Although ambiguity sets consisting of all distributions that share low-order moments are computationally attractive, they are unsatisfactory from at least two aspects. First, the data-generating distribution may not share the same low-order moments with the empirical distribution when the number of training samples is limited. Second, when the number of training samples tends to infinity, the ambiguity sets fail to converge to a singleton [37, 73]. Thus, they preclude any asymptotic consistency results. For example, given a set of training examples drawn from the exponential distribution with $\lambda = 1/50$, we expect that when the number of data tends to infinity, the solution of mean-variance DRO f_{DRO} is consistent with the optimal classifier f^* that minimizes the expected risk under this exponential distribution. However, since the normal distribution with $\mu = \sigma = 50$ has the same mean and variance as the exponential distribution, the ambiguity set contains at least both distributions. As a result, f_{DRO} might be quite different as the optimal classifier f^*.

Constraining adversarial distribution by distance function. A standard way to address the abovementioned issue is to define a distance function $d(\mathbb{Q}, \hat{\mathbb{P}}_n)$ between the adversarial distribution $\mathbb{Q}$ and the empirical distribution $\hat{\mathbb{P}}_n$ [74]. Then the ambiguity set can be defined as

$$\mathcal{B}_{d,\rho}(\hat{\mathbb{P}}_n) = \left\{ \mathbb{Q} : d(\mathbb{Q}, \hat{\mathbb{P}}_n) \leq \rho \right\}, \tag{3.2}$$

where $\mathbb{Q} = \hat{\mathbb{P}}_n$ almost everywhere iff the radius $\rho = 0$.

Distance functions on probabilities have been widely studied in the statistical literature. A well known class of distance measure is the ϕ-divergences (e.g. KL-divergence, Hellinger). [75] investigated problems that naturally come with ϕ-divergence ambiguity sets and showed the tractability of Equation 3.1 for several choice of ϕ. However, ϕ-divergence ball $\mathcal{B}$ around $\hat{\mathbb{P}}_n$ contains only distributions with the same (finite) support as $\hat{\mathbb{P}}_n$. Hence, the data-generating distribution $\mathbb{P}$ is typically not in the ambiguity set $\mathcal{B}$.

Another popular class of distance functions is the integral probability metrics (IPM) [76]. Given a set of functions $\mathcal{F}$ from $\mathcal{X}$ to $\mathbb{R}$, the IPM is defined as

$$d_{\mathcal{F}}(\mathbb{P}, \mathbb{Q}) = \sup_{f \in \mathcal{F}} \mathbb{E}_{\mathbf{x} \sim \mathbb{P}} \left[f(x) \right] - \mathbb{E}_{\mathbf{x} \sim \mathbb{Q}} \left[f(x) \right]. \tag{3.3}$$

Choosing $\mathcal{F} = \{f : \|f\|_{\mathcal{H}} \leq 1\}$, where $\mathcal{H}$ is the RKHS associated with a kernel k, Equation 3.3 becomes the well-known Maximum Mean Discrepancy (MMD) metric. MMD can be expressed as the distance in $\mathcal{H}$ between mean embeddings: $d_{\mathcal{F}}(\mathbb{P}, \mathbb{Q}) = \|\mu_{\mathbb{P}} - \mu_{\mathbb{Q}}\|_{\mathcal{H}}$ where $\mu_{\mathbb{P}} = \mathbb{E}_{\mathbf{x} \sim \mathbb{P}} k(\mathbf{x}, \cdot)$ (resp. $\mu_{\mathbb{Q}}$). When k is a characteristic kernel, MMD is a well defined distance

function, because $d_{\mathcal{F}}$ is nonnegative and satisfies the triangular inequality, symmetric, and $d_{\mathcal{F}}(\mathbb{P}, \mathbb{Q}) = 0$ iff $\mathbb{P} = \mathbb{Q}$ [77]. Gaussian kernel is a commonly used characteristic kernel, which implicitly matches infinite order moments between the two distributions. [78] showed that the MMD between the empirical distribution $\hat{\mathbb{P}}_n$ and the data-generating distribution $\mathbb{P}$ shrinks as $O(n^{-1/2})$. This implies that properly scaled MMD ball constitutes natural confidence region for the data-generating distribution, which thereby provides an upper confidence bound on the expected risk. [79] showed that the DRR with MMD can be upper bounded by an empirical risk with the RKHS norm of ℓ_f and proposed a new regularizer for Gaussian kernel ridge regression by upper bounding the RKHS norm of ℓ_f. However, it is difficult to deal with the RKHS norm of ℓ_f in general.

When $\mathcal{F}$ is a set of all 1-Lipschitz functions, Equation 3.3 recovers the Kantorovich metric whose dual representation is called Wasserstein distance. Wasserstein ambiguity sets enjoy similar statistical properties as MMD ambiguity sets: the empirical distribution on n training samples converges in Wasserstein distance to the data-generating distribution $\mathbb{P}$ at rate $O(n^{-1/d})$, where d denotes the feature dimensions [80]. Although this decay rate is slower than the rate of MMD due to the typically large feature dimensions, [37] managed to break the curse of dimensionality on the generalization bound by combining the restrictions on hypothesis space with the convergence results of Wasserstein ball. More importantly, DRO with Wasserstein ambiguity sets have been extensively studied in [32–35]. In particular, [37] showed that the DROs with Wasserstein ambiguity sets Equation 3.1 admit tractable reformulations when the model is linear (e.g., linear regression, logistic regression, SVM). Moreover, they found that DROs are

equivalent to a regularized ERM and the classical Robust Optimization [38] under mild conditions. In the following sections, we will generalize their work to a richer model class including deep neural networks and kernel machines. Specifically, we leverage the McShane-Whitney extension theorem [39, 40] to upper bound DRRs by the standard empirical risk regularized with the Lipschitz constant of the model. More importantly, we further prove that this upperbound provides a robustness certificate for the *adversarial training* in [41].

3.2 Wasserstein Distance and Duality

For ease of notation, we denote the input-output pair $(\mathbf{x}, y)$ by $\mathbf{z}$, and the input-output space by $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$.

Definition 1. *The (first order) Wasserstein distance between two distributions* $\mathbb{P}$ *and* $\mathbb{Q}$ *supported on* $\mathcal{Z}$ *is defined as*

$$W_d(\mathbb{P}, \mathbb{Q}) := \inf_{\gamma \in \Pi(\mathbb{P}, \mathbb{Q})} \int_{\mathcal{Z} \times \mathcal{Z}} d(\mathbf{z}, \mathbf{z}') \mathrm{d}\gamma(\mathbf{z}, \mathbf{z}'),$$

where $\Pi(\mathbb{P}, \mathbb{Q})$ *is collection of all joint distributions on* $\mathcal{Z} \times \mathcal{Z}$ *with marginals* $\mathbb{P}$ *and* $\mathbb{Q}$ *respectively, and* d *is a metric on* $\mathcal{Z}$*.*

Wasserstein distance is a well-defined distance function [81]. Intuitively, Wasserstein distance is the minimal cost for moving the distribution $\mathbb{P}$ to $\mathbb{Q}$, where the cost for moving a unit

probability mass from $\mathbf{z}$ to $\mathbf{z}'$ is measured w.r.t. $d(\mathbf{z}, \mathbf{z}')$ (see [82] for more details). Based on Wasserstein distance, we define the ambiguity sets as follows,

$$\mathcal{B}_{d,\rho}(\hat{\mathbb{P}}_n) = \left\{ \mathbb{Q} : W_d(\mathbb{Q}, \hat{\mathbb{P}}_n) \leq \rho \right\}, \tag{3.4}$$

which contains all distributions on $\mathcal{Z}$ whose Wasserstein distance from $\hat{\mathbb{P}}_n$ does not exceed ρ. The DRO aims to minimize the DRR within this Wasserstein ball:

$$\inf_{f} \sup_{\mathbb{Q} \in \mathcal{B}_{d,\rho}(\hat{\mathbb{P}}_n)} \mathbb{E}_{\mathbb{Q}} \ell_f(\mathbf{z}). \tag{3.5}$$

where $\ell_f(\mathbf{z})$ denotes the loss incurred under prediction $f(\mathbf{z})$, i.e., $\ell(f(\mathbf{x}), y)$. [80] proved that the empirical distribution $\hat{\mathbb{P}}_n$ converges in Wasserstein distance to the data-generating distribution $\mathbb{P}$ at rate $O(n^{-1/d})$, where d denotes the feature dimensions. [37] leveraged this favorable statistical property to establish a generalization bound for DRO while traditionally generalization bounds are derived by measuring the complexity of the hypothesis space via VC dimensions or Rademacher complexities.

Despite the generality of Equation 3.5, the major challenge for solving the DRO stems from computational tractability, since the supremum is over an uncountably infinite dimension space of distributions. To tackle this problem, [32] and [33] showed that under certain conditions, the inner maximization problem of Equation 3.5 is equivalent to a finite dimensional convex problem. Later, [34] and [35] considered arbitrary distributions on Polish space and showed the tractability of DRO via strong duality. Here we recall the strong duality result from [34]:

Proposition 1. *Let loss function $\ell_f : \mathcal{Z} \mapsto \mathbb{R}$ and $d : \mathcal{Z} \times \mathcal{Z} \mapsto \mathbb{R}_+$ be continuous. For any distribution $\mathbb{P}$ and any ρ,*

$$\sup_{\mathbb{Q}\in\mathcal{B}_{d,\rho}(\mathbb{P})} \mathbb{E}_{\mathbb{Q}}\ell_f(\mathbf{z}) = \inf_{\lambda\geq 0}\left(\lambda\rho + \mathbb{E}_{\mathbf{z}\sim\mathbb{P}}\sup_{\mathbf{z}'\in\mathcal{Z}}\left(\ell_f(\mathbf{z}') - \lambda d(\mathbf{z},\mathbf{z}')\right)\right). \tag{3.6}$$

This strong duality result has been applied to various tasks such as specification of regularization parameter [83], design of transport cost [84], and selection of uncertainty region size for optimal confidence interval [85]. In particular, [37] showed that the supremum problem inside the dual form of Equation 3.6 can be further simplified:

Lemma 2 (Generalized from Lemma A.3 of [37])**.** *Let $(\mathcal{X}, \|\cdot\|)$ be a normed linear space and $f : \mathcal{X} \mapsto \mathbb{R}$ be convex. Then for $\mathbf{x}_0 \in \mathcal{X}, \lambda > 0$,*

$$\sup_{\mathbf{x}'\in\mathcal{X}}\left(f(\mathbf{x}) - \lambda\|\mathbf{x}-\mathbf{x}_0\|\right) = \begin{cases} f(\mathbf{x}_0) & \text{if } Lip(f) \leq \lambda, \\ +\infty & \text{otherwise.} \end{cases} \tag{3.7}$$

See proof in Appendix B.1.

Therefore, Proposition 1 together with Lemma 2 indicates that DRO with Wasserstein ball is computationally tractable. For example, when ℓ_f is logistic loss with linear hypotheses and the transportation metric $d(\mathbf{z},\mathbf{z}') = \|\mathbf{x}-\mathbf{x}'\| + \infty\cdot|y-y'|$, [37] showed that the DRO Equation 3.5 is equivalent to the regularized Logistic Regression: $\min_w \mathbb{E}_{\hat{\mathbb{P}}_n}\log(1+e^{-y\langle\mathbf{w},\mathbf{x}\rangle}) + \rho\|w\|_*$, since ℓ_f satisfies the convexity and Lipschitz continuity in Lemma 2. Similar arguments also apply to SVM where the hinge loss is convex and Lipschitz continuous.

A more interesting scenario is when the hypotheses space consists of nonlinear functions, such as neural networks or kernel machines. [37] developed lifted variants for kernel machines to accommodate nonlinear hypotheses. However, the perturbation of distribution is applied to $\Phi(\mathbf{x})$, where Φ is the implicit feature map of the RKHS. This still fall short of robustness with respect to distribution over $\mathcal{Z}$. On the other hand, [36] interpreted $\sup_{\mathbf{z}' \in \mathcal{Z}} (\ell_f(\mathbf{z}') - \lambda d(\mathbf{z}, \mathbf{z}'))$ in the dual form of Equation 3.6 as a surrogate loss which allows adversarial perturbations of the data $\mathbf{z}$, modulated by the penalty λ. Their approach is based on a simple insight that the surrogate loss is strongly-concave if the neural network f is L-smooth and $\lambda \geq L$. Therefore, solving $\inf_f \mathbb{E}_{\mathbf{z} \sim \mathbb{P}} \sup_{\mathbf{z}' \in \mathcal{Z}} (\ell_f(\mathbf{z}') - \lambda d(\mathbf{z}, \mathbf{z}'))$ is computationally efficient and provides an upper bound for the worst-case population performance. However, this method requires restricting the perturbation to be small (i.e., large enough penalty λ), which leaves unaddressed regimes of medium to large perturbations. Moreover, although the global solvability of the inner surrogate loss can be ensured by small enough perturbation, there is no practical procedure to compute the threshold.

3.3 Qualifying DRR for Nonlinear Models

In this section, we develop a novel certificate on distributional robustness for nonlinear models. Specifically, we will leverage the McShane-Whitney extension theorem [39,40] to upper bound DRRs by the standard empirical risk regularized with the Lipschitz constant of the model f.

Lemma 3 (McShane–Whitney [86, Thm. 2.1 (ii)])**.** *Let $(\mathcal{X}, d)$ be a metric space and $f : \mathcal{X} \to \mathbb{R}$. Then*

$$\forall \gamma \geq Lip(f): \quad \sup_{x \in \mathcal{X}} \Big(f(x) - \gamma d(x_0, x) \Big) = f(x_0). \tag{3.8}$$

Interestingly, Lemma 3 characterizes the supremum problem in the dual form of Equation 3.6 without assuming the convexity of function f. Therefore the DRR can be upper bounded for (nonconvex) nonlinear models.

Theorem 3. *Let $(\mathcal{Z}, d)$ be a Polish space with metric d. Assume $\ell_f : \mathcal{Z} \mapsto \mathbb{R}$ in Equation 3.5 is continuous. Then*

$$\sup_{\mathbb{Q} \in \mathcal{B}_{d,\rho}(\mathbb{P})} \mathbb{E}_{\mathbb{Q}} \ell_f(\mathbf{z}) \leq \rho Lip(\ell_f) + \mathbb{E}_{\mathbb{P}} \ell_f(\mathbf{z}). \tag{3.9}$$

If additionally $\mathcal{Z}$ is a linear space with d induced by a norm and ℓ_f is convex, then Equation 3.9 attains equality.

Proof. It is easily verified that d and ℓ_f satisfy the assumptions in Proposition 1. Then

$$\begin{aligned} \sup_{\mathbb{Q} \in \mathcal{B}_{d,\rho}(\mathbb{P})} \mathbb{E}_{\mathbb{Q}} \ell_f(\mathbf{z}) &= \inf_{\lambda \geq 0} \left(\lambda \rho + \mathbb{E}_{\mathbf{z} \sim \mathbb{P}} \sup_{\mathbf{z}' \in \mathcal{Z}} \big(\ell_f(\mathbf{z}') - \lambda d(\mathbf{z}, \mathbf{z}') \big) \right) \\ &\leq \inf_{\lambda \geq Lip(\ell_f)} \Big(\lambda \rho + \mathbb{E}_{\mathbf{z} \sim \mathbb{P}} \ell_f(\mathbf{z}) \Big) \\ &= \rho Lip(\ell_f) + \mathbb{E}_{\mathbf{z} \sim \mathbb{P}} \ell_f(\mathbf{z}), \end{aligned} \tag{3.10}$$

where the inequality follows from Lemma 3. To achieve equality in Equation 3.9, apply Lemma 2 to Equation 3.10 under the additional assumptions on data space and models. □

Remark 1. *For classification or regression tasks in machine learning, the Polish space $\mathcal{Z}$ usually admits a decomposition $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, where $(\mathcal{X}, \|\cdot\|)$ is a normed linear space and $(\mathcal{Y}, d_{\mathcal{Y}})$ is a metric space. Equip $\mathcal{Z}$ with metric $d(\mathbf{z}, \mathbf{z}') = \|\mathbf{x} - \mathbf{x}'\| + \kappa d_{\mathcal{Y}}(y, y')$, then we can apply the McShane-Whitney extension theorem independently on the input space $\mathcal{X}$:*

$$\begin{aligned}
\sup_{\mathbb{Q} \in \mathcal{B}_{d,\rho}(\mathbb{P})} \mathbb{E}_{\mathbb{Q}} \ell_f(\mathbf{z}) &= \inf_{\lambda \geq 0} \left(\lambda \rho + \mathbb{E}_{\mathbf{z} \sim \mathbb{P}} \sup_{\mathbf{z}' \in \mathcal{Z}} \left(\ell_f(\mathbf{z}') - \lambda d(\mathbf{z}, \mathbf{z}') \right) \right) \\
&= \inf_{\lambda \geq 0} \left(\lambda \rho + \mathbb{E}_{(\mathbf{x},y) \sim \mathbb{P}} \sup_{y' \in \mathcal{Y}} \sup_{\mathbf{x}' \in \mathcal{X}} \left(\ell(f(\mathbf{x}'), y') - \lambda \left\| \mathbf{x} - \mathbf{x}' \right\| - \kappa d_{\mathcal{Y}}(y, y') \right) \right) \\
&\leq \inf_{\lambda \geq \max_{y' \in \mathcal{Y}} Lip(\ell(f(\cdot), y'))} \Big(\lambda \rho + \mathbb{E}_{(\mathbf{x},y) \sim \mathbb{P}} \max_{y' \in \mathcal{Y}} \ell(f(\mathbf{x}), y') - \lambda \kappa d_{\mathcal{Y}}(y, y') \Big). \qquad (3.11)
\end{aligned}$$

When $\mathbb{P}$ is an empirical distribution on some training examples $(\mathbf{x}_i, y_i)_{i \in [n]}$, we minimize Equation 3.11 by solving the following constrained optimization problem

$$\left.\begin{array}{ll}
\min_{f, \lambda \geq 0, s \in \mathbb{R}^n} & \frac{1}{n} \sum_{i=1}^n s_i + \lambda \rho \\
\underset{\substack{\forall i = 1, \ldots, n \\ \forall y' \in \mathcal{Y}}}{\text{s.t.}} & \ell(f(\mathbf{x}_i), y') - \lambda \kappa d_{\mathcal{Y}}(y_i, y') \leq s_i, \\
& Lip(\ell(f(\cdot), y')) \leq \lambda.
\end{array}\right\}. \qquad (3.12)$$

Note that when $\kappa = \infty$, the constrains $\ell(f(\mathbf{x}_i), y') - \lambda \kappa d_{\mathcal{Y}}(y_i, y') \leq s_i, \forall\, y' \neq y_i$ are vacuous. Hence, Equation 3.12 recovers a classical ERM problem regularized by the Lipschitz constant of $\ell(f(\cdot), y')$, which implicitly assumes that every training example must have exactly one label.

Now we exemplify Remark 1 with two popular nonlinear machine learning models: neural networks and kernel machines.

Example 1. *(Neural Networks) Let the transportation metric* $d(\mathbf{z}, \mathbf{z}') = \|\mathbf{x} - \mathbf{x}'\|_\infty + \infty \cdot |y - y'|$. *Assuming the last layer of neural network* f *is softmax and* $f(x)$ *represents logits which is the output of the subnetwork before the softmax.* ℓ *is cross entropy loss. By the composition property of Lipschitz functions and the* 1*-Lipschitz continuity (w.r.t.* $\|\cdot\|_\infty$*) of cross entropy loss, we have* $Lip(\ell(f(\cdot), y')) \leq Lip(f)$. *Then*

$$\sup_{\mathbb{Q} \in \mathcal{B}_{d,\rho}(\hat{\mathbb{P}}_n)} \mathbb{E}_{\mathbb{Q}} \ell_f(\mathbf{z}) \leq \rho Lip(f) + \mathbb{E}_{\hat{\mathbb{P}}_n} \ell(f(\mathbf{x}), y).$$

Therefore, the robustness of neural networks depends on the Lipschitz constant of the model.

Example 2. *(Kernel Machines) Let the transportation metric* $d(\mathbf{z}, \mathbf{z}') = \|\mathbf{x} - \mathbf{x}'\|_2 + \infty \cdot |y - y'|$. *The kernel machine aims to solve*

$$\min_{f \in \mathcal{H}} \quad \mathbb{E}_{\hat{\mathbb{P}}_n} \ell(f(\mathbf{x}), y) + \frac{\lambda}{2} \|f\|_{\mathcal{H}}^2 ,$$

where $\mathcal{H}$ *is the RKHS induced by a kernel function* k. *By representer theorem, the optimal solution* f *must be in the form of* $f = \frac{1}{n} \sum_{i=1}^n \alpha_i k(\mathbf{x}^i, \cdot)$. *When* ℓ *is logistic loss, we obtain a similar result as Example 1,*

$$\sup_{\mathbb{Q} \in \mathcal{B}_{d,\rho}(\hat{\mathbb{P}}_n)} \mathbb{E}_{\mathbb{Q}} \ell_f(\mathbf{z}) \leq \rho Lip(f) + \mathbb{E}_{\hat{\mathbb{P}}_n} \ell(f(\mathbf{x}), y),$$

where the Lip(f) denotes the Lipschitz constant of model f in the RKHS.

3.4 Adversarial Examples in Neural Networks

In this section, we introduce the *adversarial example* problem in neural network models. Later, we will show how this critical problem can be solved under the DRO framework.

Deep neural networks have achieved impressive results in a variety of complex machine learning tasks, such as object recognition [15, 16], natural language processing [17], and speech recognition [18]. However, [19] discovered that neural networks are highly sensitive to *adversarial examples* in the image classification domain: With some imperceptible perturbations, an image can be misclassified by neural network classifier. This property exposes the low robustness of neural networks and is harmful to safety-critical applications.

Since then, there has been plenty of work in the arms race between attack and defense. Here we highlight some popular attacks. [19] formalized a standard targeted attack as an optimization problem: Given $c > 0$ and a target wrong label y', solve:

$$\min_{\boldsymbol{\delta}} c \|\boldsymbol{\delta}\|_2 + \ell(f(\mathbf{x}+\boldsymbol{\delta}), y'), \quad s.t.\ \mathbf{x}+\boldsymbol{\delta} \in [0,1]^d,\ f(\mathbf{x}+\boldsymbol{\delta}) = y', \tag{3.13}$$

where f is a neural network classifier and ℓ is the loss function. Then the authors used L-BFGS algorithm to find the minimum perturbation $\boldsymbol{\delta}$. Similarly, [87] replaced ℓ in Equation 3.13 with some margin-based loss function. They conducted experiments on different similarity measures including $\ell_0, \ell_2, \ell_\infty$ distance and achieved strong attacks.

In order to overcome the speed limitations of L-BFGS, [88] proposed a method called fast gradient sign (FGS), which takes only one small step in the direction of gradient: $\mathbf{x}_{adv} = \mathbf{x} + \epsilon \nabla_{\mathbf{x}} \ell(f(\mathbf{x}), y)$. Noticing that FGS is just a constrained maximization of the non-concave loss function, researchers in [41, 89, 90] have leveraged many efficient optimization algorithms to find the local maximum. In particular, the projected gradient descent (PGD) in [41] is a popular attack method as a multi-step variant of FGS: $\mathbf{x}^{t+1} = \prod_{\mathbf{x}+\mathcal{S}}(\mathbf{x}^t + \alpha \nabla_{\mathbf{x}} \ell(f(\mathbf{x}), y))$, where $\mathcal{S}$ is the set of allowed perturbations. PGD was found to produce superior results among all first-order adversaries, i.e., attacks that rely only on first-order information.

On the other hand, different types of model have been proposed to defend the adversarial examples. The first type of defense models apply pre-processing techniques on input data in order to reduce the impact of adversarial examples [91], or discard the data if it is detected as an adversarial example [92–94]. But, the pre-processing techniques are usually domain specific while adversarial detectors are vulnerable to iterative attacks. Another type of defenders are motivated by the phenomenon called *gradient masking* [95], which try to obfuscate the gradient of a classifier w.r.t. input data. However, [96] showed that most of the obfuscated gradients defense models are not as robust as they claimed [97–100].

There are several promising results on defending adversarial examples. From the robust optimization perspective, [41] proposed an *adversarial training* method,

$$\min_{f} \left\{ L(f; \mathcal{S}) := \mathbb{E}_{\hat{\mathbb{P}}_n} \left[\max_{\boldsymbol{\delta} \in \mathcal{S}} \ell(f(\mathbf{x} + \boldsymbol{\delta}), y) \right] \right\}, \tag{3.14}$$

where $\mathcal{S}$ is a perturbation set, e.g., $\mathcal{S} = \{\boldsymbol{\delta} : \|\boldsymbol{\delta}\|_\infty \leq 0.3\}$. We referred to the worst-case loss $L(f; \mathcal{S})$ as *adversarial training risk*. They empirically observed that the inner maximization problem of Equation 3.14 has well-concentrated loss values when using PGD attacker. Consequently, their experiments showed that adversarial training with PGD adversary yields certain robustness against all first-order adversaries. However, there is no guarantee that either L-BFGS or PGD can find the strongest attack (the argmax of $\boldsymbol{\delta}$ in Equation 3.13 and Equation 3.14), especially because the neural network f is not a convex function. So the adversarial training in Equation 3.14 is only solved approximately with no theoretical guarantee. Therefore, in the next chapter, we will develop a new approach based on DRO framework, which a) upper bounds the adversarial training risk $L(f; \mathcal{S})$ by DRR, and b) admits much more effective optimization recipes by dualization.

3.5 Certifying Adversarial Training by DRR

In this section, we prove that adversarial training risk is upper bounded by DRR, which in turn is upper bounded by the Lipschitz regularized empirical risk in Theorem 3. Similar results have been shown in [37] which are restricted to linear models while our results generalize to nonlinear models.

To begin with, we first define the push forward distribution from a distribution $\mathbb{P}$ on space $\mathcal{Z}$ by a mapping $a : \mathcal{Z} \mapsto \mathcal{T}$ as $a_{\#}\mathbb{P} := \mathbb{P} \circ a^{-1}$.

Lemma 4. *Let $a : \mathcal{Z} \to \mathcal{Z}$ be a transformation on input data; $\mathbb{P}$ is a distribution over $\mathcal{Z}$; $\mathbb{Q}$ is a push forward distribution from $\mathbb{P}$ by transformation a; and the transportation metric $d : \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}$. If d satisfies $\mathbb{E}_{\mathbf{z}\sim\mathbb{P}} d(\mathbf{z}, a(\mathbf{z})) \leq \rho$, then $\mathbb{Q} \in \mathcal{B}_{d,\rho}(\mathbb{P})$.*

Proof. Since $\mathbb{P} \times \mathbb{Q} \in \Pi(\mathbb{P}, \mathbb{Q})$, we have

$$W_d(\mathbb{P}, \mathbb{Q}) = \inf_{\gamma \in \Pi(\mathbb{P}, \mathbb{Q})} \int d \mathrm{d}\gamma \leq \int d \mathrm{d}\mathbb{P} \times \mathbb{Q} = \int d(\mathbf{z}, a(\mathbf{z})) \mathrm{d}\mathbb{P} \leq \rho, \tag{3.15}$$

hence $\mathbb{Q}_{\mathcal{Z}} \in \mathcal{B}_{d,\rho}(\mathbb{P}_{\mathcal{Z}})$. □

Remark 2. *Note that $\int d(\mathbf{z}, a(\mathbf{z})) \mathrm{d}\mathbb{P}$ in Equation 3.15 is a classical mass transport problem named Monge's problem. [101] proposed a relaxed version of the problem: $\inf_{\gamma \in \Pi(\mathbb{P}, \mathbb{Q})} \int d \mathrm{d}\gamma$, which allows the splitting of mass. Therefore, the first inequality in Equation 3.15 always hold. [102] proved that if $\mathbb{P}$ is non-atomic and d is continuous, then the infimum of Monge's problem equals the infimum of Kantorovich's problem (the infimum is always attainable due to the continuity of d [82]).*

Lemma 4 implies that the data distribution after any allowed perturbation on dataset belongs to a Wasserstein ball centered at empirical distribution. Therefore, we have

Theorem 4. *Let $\mathcal{S}$ be a perturbation set with ℓ_p norm, i.e., $\mathcal{S} = \left\{\delta : \|\delta\|_p \leq \rho\right\}$; transportation metric $d(x, y) = \|x - y\|_p$; $\hat{\mathbb{P}}_n$ be the empirical distribution over $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$; $\Gamma(\mathbf{z}) := \mathbf{z} + \mathcal{S}$ be the set of allowed perturbed data; $M_{\hat{\mathbb{P}}_n}(\Gamma) = \left\{a : \mathcal{Z} \to \mathcal{Z} : \ a(\mathbf{z}) \in \Gamma(\mathbf{z}), \mathbf{z} \sim \hat{\mathbb{P}}_n\right\}$ be collection of functions that perturb training examples; $\mathbb{Q}$ is a push forward distribution from $\hat{\mathbb{P}}_n$ by a transformation $a \in M_{\hat{\mathbb{P}}_n}(\Gamma)$. Also assume ℓ_f is continuous w.r.t. $\mathbf{z}$*

$$\mathbb{E}_{\hat{\mathbb{P}}_n}\left[\sup_{\delta \in \mathcal{S}} \ell_f(\mathbf{z} + \delta)\right] \leq \sup_{\mathbb{Q} \in \mathcal{B}_{d,\rho}(\hat{\mathbb{P}}_n)} \mathbb{E}_{\mathbb{Q}} \ell_f(\mathbf{z}). \tag{3.16}$$

Proof.

$$\mathbb{E}_{\hat{\mathbb{P}}_n}\left[\sup_{\delta\in\mathcal{S}} \ell_f(\mathbf{z}+\delta)\right] = \mathbb{E}_{\hat{\mathbb{P}}_n}\left[\sup_{\mathbf{z}'\in\Gamma(\mathbf{z})} \ell_f(\mathbf{z}')\right] = \sup_{a\in M_{\hat{\mathbb{P}}_n}(\Gamma)} \mathbb{E}_{\hat{\mathbb{P}}_n}\left[\ell_f \circ a(\mathbf{z})\right]$$

the last equality is from Theorem 6.1 of [103]. Then

$$d(\mathbf{z}, a(\mathbf{z})) = \|\mathbf{z} - a(\mathbf{z})\|_p = \|\delta\|_p \leq \rho.$$

Apply Lemma 4 to $a \in M_{\hat{\mathbb{P}}_n}(\Gamma)$, we have

$$\mathbb{E}_{\hat{\mathbb{P}}_n}\left[\sup_{\delta\in\mathcal{S}} \ell_f(\mathbf{z}+\delta)\right] = \sup_{a\in M_{\hat{\mathbb{P}}_n}(\Gamma)} \mathbb{E}_{\hat{\mathbb{P}}_n}\left[\ell_f \circ a(\mathbf{z})\right] \leq \sup_{\mathbb{Q}\in\mathcal{B}_{d,\rho}(\hat{\mathbb{P}}_n)} \mathbb{E}_{\mathbb{Q}}\ell_f(\mathbf{z}),$$

which proves Equation 3.16. A more general proof is given in [104]. □

Remark 3. *The inequality in Equation 3.16 stems from Lemma 4. Under the assumptions in Theorem 4 and compactness of data space, from [34, Prop. 2], there exists an optimal* $\mathbb{Q}^*$ *such that* $\mathbb{E}_{\mathbb{Q}^*}\ell_f(\mathbf{z}) = \sup_{\mathbb{Q}\in\mathcal{B}_{d,\rho}(\hat{\mathbb{P}}_n)} \mathbb{E}_{\mathbb{Q}}\ell_f(\mathbf{z})$. *As long as there exist a transport mapping* a *such that the push forward distribution* $a_{\#}\hat{\mathbb{P}}_n = \mathbb{Q}^*$, *the equality holds in Equation 3.16. However, this generally does not happen since* $\hat{\mathbb{P}}_n$ *is atomic and* $\mathbb{Q}^*$ *is typically non-atomic.*

Theorem 4 holds for commonly used neural networks because the composition of loss function and neural network function is continuous w.r.t. $\mathbf{z}$. The Wasserstein ball in the upperbound Equation 3.16 is determined by the perturbation set $\mathcal{S}$, where the transportation metric d on $\mathcal{Z}$ is the perturbation metric $(\|\cdot\|_p, \mathsf{Id})$ on $\mathcal{Z}$, and radius ρ of Wasserstein ambiguity set is the radius of allowed perturbation.

[37, Theorem 3.20] achieved a similar upper bound on robust optimization for only linear classifiers in binary cases, while Theorem 4 allows any nonlinear classifier as long as l_f is continuous.

Proposition 2. *Combining Theorem 3 and Theorem 4, the adversarial training risk is upper bounded by the Lipschitz regularized empirical risk,*

$$\mathbb{E}_{\hat{\mathbb{P}}_n}\left[\sup_{\delta\in\mathcal{S}} \ell_f(\mathbf{z}+\delta)\right] \leq \rho Lip(\ell_f) + \mathbb{E}_{\hat{\mathbb{P}}_n}\ell_f(\mathbf{z}). \tag{3.17}$$

Remark 4. *Since there is no efficient algorithm for globally minimizing the adversarial training risk Equation 3.14, we can instead solve its upperbound in Equation 3.17. If the transportation metric $d(\mathbf{z},\mathbf{z}') = \|\mathbf{x}-\mathbf{x}'\| + \infty\cdot|y-y'|$, then r.h.s of Equation 3.17 reduces to*

$$\min_{f\in\mathcal{F}} \quad \frac{1}{n}\sum_{i=1}^{n} \ell(f(\mathbf{x}_i), y) + \rho \max_{i\in[n]} Lip(\ell(f(\cdot), y_i)), \tag{3.18}$$

where the Lipschitz constant is measured w.r.t. $\|\cdot\|$. This is exactly same as the Lipschitz upperbound derived by [45].

The idea that there should be a relationship between the Lipschitz continuity of a neural network and its robustness to adversarial examples is not new. [19], in their landmark paper on adversarial examples [19], concluded with a section where they study the Lipschitz properties of ImageNet. Other papers have since attempted to formalize the connection between Lipschitz continuity and adversarial robustness: [42] used the Lipschitz constant of the network to provide a margin-based, local certification of adversarial robustness (Propositions 1 and 2) for multi-

class classification; and [105] bounded the adversarial risk (binary classification) in the special case of the logistic loss using a term similar to Lipschitz constant (Theorem 7). Additionally, the Lipschitz constant can also be related to model complexity [106] which lends a natural interpretation of the objective function in Equation 3.18 with respect to a trade-off between goodness of fit and parsimony. Therefore, the tightness in estimating Lipschitz constant will undoubtably affect the robustness.

Nevertheless, enforcing the Lipschitz constant in neural networks is known to be difficult. Assume a neural network f consists of l hidden layers as defined by $f(x) := W_l * \phi_l \circ \cdots \circ \phi_1(x)$, where each function ϕ_i can be broken down into the composition of a linear operator W_{i-1} and a nonlinear activation function σ, i.e., $\phi_i := \sigma_i \circ W_{i-1}$. Since Lipschitz constants for typically used activation functions are less than 1, the Lipschitz constant of f can be upper bounded as

$$Lip(f) \leq \|W_l\|_{op} \prod_{i=1}^{l} Lip(\phi_i) \leq \|W_l\|_{op} \prod_{i=1}^{l} \Big(Lip(\sigma_i) \cdot \|W_{i-1}\|_{op} \Big) \leq \prod_{i=0}^{l} \|W_i\|_{op}$$

where $\|W_i\|_{op}$ is the operator norm of W_i. [43, 45] placed bounds on each $\|W_i\|_{op}$, but only achieved little improvement on robustness. [46] showed that ReLU networks with simple norm constraints on W are unable to approximate simple functions such as absolute value. Their theory indicates that norm constraints only on weight matrices can jeopardize the expressiveness of neural networks. Instead, [107] converted the tight Lipschitz constant $Lip(W_1 \circ \sigma_1 \circ W_0)$ (two-layer neural network) into a semi-definite programming problem, and achieved impressive robust performance on the MNIST dataset. However, their method is hard to scale to deep

neural networks. [42] investigated the relation between margin, adversarial perturbation, and Lipschitz constant. Assuming the last layer of a neural network f is softmax, and $f(\mathbf{x})$ represents logits which is the output of the subnetwork before the softmax. The perturbation set is $\mathcal{S} = \{\|\delta\|_2 \leq \epsilon\}$ and the margin is defined as $M_{f,\mathbf{x}} := f(\mathbf{x})_t - \max_{i \neq t} f(x)_i$, where t is the true label of $\mathbf{x}$. They proved that if $M_{f,\mathbf{x}} \geq \sqrt{2} Lip(f)\epsilon$ then $f(\mathbf{x} + \delta) = t$ for all $\delta \in \mathcal{S}$. Thus, in the training procedure, they applied norm constraint to upper bound the Lipschitz constant, and added $\sqrt{2} Lip(f)\epsilon$ to all elements in logits except the index corresponding to t. In this way, they achieved large margin neural networks that have theoretical robustness guarantee. However, since their Lipschitz estimation is loose as we mentioned before, the resulting model is vulnerable to larger perturbation. [108] introduced an activation function called *GroupSort*, which takes an input vector $\mathbf{x}$, separates the elements into g groups, sorts each group into ascending order, and outputs the combined "group sorted" vector. This activation function can preserve Lipschitz constant between layers because its Jacobian is a permutation matrix. They showed that norm-constrained neural networks with GroupSort activation are more expressive than their ReLU counterparts. Combining *GroupSort* with Lipschitz margining training [42], they achieved improved robustness than ReLU counterparts.

CHAPTER 4

ROBUST KERNEL MACHINES

(Parts of this chapter were previously published as "Generalised Lipschitz Regularisation Equals Distributional Robustness" [1], in *International Conference on Machine Learning (ICML)*, 2021.)

4.1 Preliminaries

A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a *positive semi-definite kernel* if for all $n \in \mathbb{N}$ and all $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathcal{X}$, the $n \times n$ Gram matrix $K := (k(\mathbf{x}_i, \mathbf{x}_j))_{ij}$ is symmetric positive semi-definite. Example kernels on $\mathbb{R}^n \times \mathbb{R}^n$ include polynomial kernel of degree r, which are defined as $k(\mathbf{x}_1, \mathbf{x}_2) = (\langle \mathbf{x}_1, \mathbf{x}_2 \rangle + 1)^r$, as well as Gaussian kernels, defined as $k(\mathbf{x}_1, \mathbf{x}_2) = \exp(-\left\|\mathbf{x}_1 - \mathbf{x}_2\right\|_2^2 / (2\sigma^2))$. More comprehensive introductions to kernels are available in, e.g., [109, 110].

Given k, let $\mathcal{H}_0$ be the set of all finite linear combinations of functions in $k(\mathbf{x}, \cdot) : \mathbf{x} \in \mathcal{X}$, and endow on $\mathcal{H}_0$ an inner product as $\langle f, g \rangle_{\mathcal{H}} = \sum_{i=1}^{p} \sum_{j=1}^{q} \alpha_i \beta_j k(\mathbf{x}_i, \mathbf{y}_j)$ where $f(\cdot) = \sum_{i=1}^{p} \alpha_i k(\mathbf{x}_i, \cdot)$ and $f(\cdot) = \sum_{j=1}^{q} \beta_j k(\cdot, \mathbf{y}_j)$. Note $\langle f, g \rangle_{\mathcal{H}}$ is invariant to the form of expansion of f and g [109]. It is easy to show that $\mathcal{H}_0$ is an inner product space by the positive semi-definite properties of k. The completion of inner product space $\mathcal{H}_0$ under $\langle f, g \rangle_{\mathcal{H}}$ is denoted by the reproducing kernel Hilbert space (RKHS) $\mathcal{H}$ induced by k. For any $f \in \mathcal{H}$, the reproducing property implies

$f(\mathbf{x}) = \langle f, \phi(\mathbf{x}) \rangle_{\mathcal{H}}$, where $\phi(\mathbf{x}) := k(\mathbf{x}, \cdot) : \mathcal{X} \to \mathcal{H}$ is representer of evaluation functional. And we denote $\|f\|_{\mathcal{H}}^2 := \langle f, f \rangle_{\mathcal{H}}$.

Let us consider a Mercer's kernel k on a convex domain $\mathcal{X} \subseteq \mathbb{R}^d$ and assume the availability of l labeled examples for training $\{x^i, y^i\}_{i=1}^l$. The standard kernel method seeks a discriminant function f from $\mathcal{H}$ with the conventional form of finite kernel expansion $f(x) = \frac{1}{l} \sum_{i=1}^l \gamma_i k(x^i, \cdot)$, such that the regularized empirical risk can be minimized with the standard (hinge) loss and RKHS norm.

We note in passing that the function space of neural networks with sigmoid-like or ReLU-like activation is contained in the RKHS associated with stacked inverse kernel [47]. In this light, solving Lipschitz regularized kernel machines provides a robustness certificate while inheriting the expressive power of neural networks.

Therefore, our goal here is to additionally enforce that the learned f is smooth, i.e., its Lipschitz continuous constant falls below a prescribed threshold $L > 0$. Thanks to the convexity of X, this is equivalent to

$$\sup_{x \in \mathcal{X}} \|g(x)\|_2 \leq L + \epsilon, \text{ where } g(x) := \nabla f(x) \tag{4.1}$$

for some small $\epsilon > 0$. The standard RKHS norm constraint $\|f\|_{\mathcal{H}} \leq C$ indeed already induces an upper bound on $\|\nabla f(x)\|_2$ because by Example 3.23 of [37],

$$\sup_{x \in \mathcal{X}} \|\nabla f(x)\|_2 \leq \|f\|_{\mathcal{H}} \sup_{z>0} \frac{t(z)}{z} \tag{4.2}$$

$$\text{where } t(z) \geq \sup_{x,x' \in \mathcal{X}: \|x-x'\|_2 = z} \left\| k(x,\cdot) - k(x',\cdot) \right\|_{\mathcal{H}}.$$

For Gaussian kernel with bandwidth σ, $t(z) = \max\{\sigma^{-1}, 1\}z$. For exponential and inverse kernels, $t(z) = z$ [111]. [112] justified that the RKHS norm of a neural network can be used as a surrogate for Lipschitz regularization. But the quality of such a proxy, i.e., the gap in the inequality of Equation 4.2, can be loose as we will see later around Figure 9. Besides, C and L are independent constants. Hence, we must seek other means to enforce Equation 4.1.

4.2 Distributionally Robustness in Polynomial Time

In this section, we first present an algorithm that requires exponential complexity to enforce Equation 4.1. The analysis sheds light on the possibility of developing an effective algorithm to enforce Lipschitz constant. We will present a refined method with polynomial complexity for product kernels in the last subsection.

4.2.1 A Method with Exponential Cost

Since the direct evaluation of $\max_{x \in \mathcal{X}} \|g(x)\|_2$ is generally hard because g is neither convex nor concave, a natural idea is to uniformly sample n touchstone points $\{w^s\}_{s=1}^n$ from $\mathcal{X}$ and enforce the smoothness on these samples. For Gaussian kernel, we may sample from $\mathcal{N}(\mathbf{0}, \sigma^2 I)$

while for inverse kernel we may sample uniformly from the unit ball B. This leads to our training objective:

$$\min_{f \in \mathcal{H}} \quad \frac{1}{l} \sum_{i=1}^{l} \ell(f(x^i), y^i) + \frac{\lambda}{2} \|f\|_{\mathcal{H}}^2 \tag{4.3}$$

$$s.t. \quad \|g(w^s)\| \leq L, \quad \forall s \in [n]. \tag{4.4}$$

The key question is how large n (#sample) needs to be so that with high probability (w.h.p.) the optimal solution satisfies Equation 4.1? Unfortunately, this method may require $O(\frac{1}{\epsilon^d})$ samples to guarantee Equation 4.1 w.h.p. This is illustrated in Figure 8, where k is the polynomial kernel with degree 2 whose domain $\mathcal{X}$ is the unit ball B, and $f(x) = \frac{1}{2}(v^\top x)^2$. We seek to test whether the gradient $g(x) = (v^\top x)v$ has norm bounded by 1 for all $x \in B$, and we are only allowed to test whether $\|g(w^s)\| \leq 1$ for samples w^s that are drawn uniformly at random from B such as w_1 and w_2. This is equivalent to testing $\|v\| \leq 1$ because $g(x)$ is linear in x, and to achieve it at least one w^s must be from the ϵ ball around $v/\|v\|$ or $-v/\|v\|$, intersected with B. But the probability of hitting the blue shaded region decays exponentially with the dimensionality d. The problem, however, becomes trivial if we use the orthonormal basis $\{\tilde{w}_1, \tilde{w}_2\}$.

4.2.2 New Regularizer to Enforce Lipschitz Constant

The key insight from the above counter-example is that in fact $\|v\|$ can be easily computed by $\sum_{s=1}^{d}(v^\top \tilde{w}_s)^2$, where $\{\tilde{w}^s\}_{s=1}^{d}$ is the *orthonormal* basis computed from the Gram–Schmidt process on d random samples $\{w^s\}_{s=1}^{d}$ $(n = d)$. With probability 1, n samples drawn uniformly

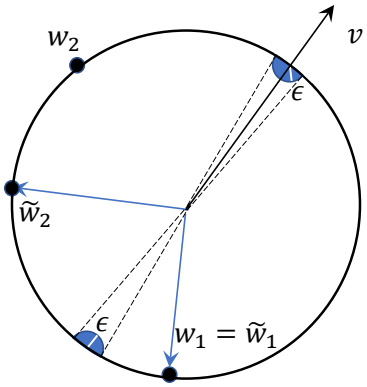


Figure 8: Illustration of exponential sample complexity for uniform sampling.

from B must span $\mathbb{R}^d$ as long as $n \geq d$, i.e., $\text{rank}(W) = d$ where $W = (w^1, \ldots, w^n)$. The Gram–Schmidt process can be effectively represented using a pseudo-inverse matrix (allowing $n > d$) as

$$\|v\|_2 = \left\|(W^\top W)^{-1/2} W^\top v\right\|_2, \tag{4.5}$$

where $(W^\top W)^{-1/2}$ is the square root of the pseudo-inverse of $W^\top W$. This is exactly the intuition underlying the Nyström approximation that we will leverage in the sequel.

The above example is over-simplified because in general $g(x)$ is not linear in x. Although $g_j(x) := \partial^j f(x) = \langle g_j, k(x, \cdot)\rangle_{\mathcal{H}}$ is linear in $k(x, \cdot)$, the evaluation elements $\{k(x, \cdot) : x \in \mathcal{X}\}$ only comprise a proper subset of the unit ball in RKHS. That means in general $\max_{x \in \mathcal{H}} |g_j(x)|$

is not equal to $\|g_j\|_{\mathcal{H}}$. Note here we assumed $g_j \in \mathcal{H}$ which is true for the kernels we consider, but not true in general. Indeed if $\sup_{x\in\mathcal{X}} k(x,x) = 1$, then

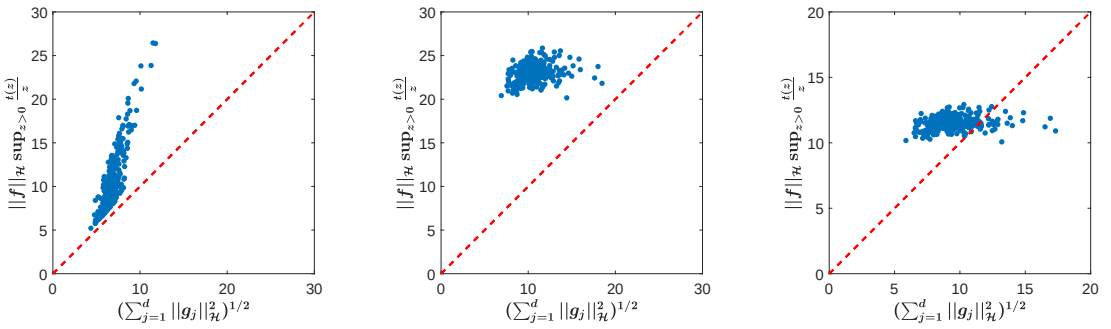
$$\sup_{x\in\mathcal{X}} \|g(x)\|_2^2 = \sup_{x\in\mathcal{X}} \sum_{j=1}^{d} g_j(x)^2 = \sup_{x\in\mathcal{X}} \sum_{j=1}^{d} \langle g_j, k(x,\cdot)\rangle_{\mathcal{H}}^2 \leq \sup_{x\in\mathcal{X}} \sum_{j=1}^{d} \|g_j\|_{\mathcal{H}}^2 \|k(x,\cdot)\|_{\mathcal{H}}^2 = \sum_{j=1}^{d} \|g_j\|_{\mathcal{H}}^2 . \tag{4.6}$$

Therefore we can strengthen the requirement Equation 4.1 as follows which effectively shrinks the hypothesis space:

$$\sum_{j=1}^{d} \|g_j\|_{\mathcal{H}}^2 \leq L^2 + \epsilon. \tag{4.7}$$

Why does $\sum_{j=1}^{d} \|g_j\|_{\mathcal{H}}^2$ tend to provide a *lower* (i.e., tighter) approximation of the Lipschitz constant than Equation 4.2? To gain some intuition, note that the latter takes **two** steps of relaxation: $|f(x) - f(x')| \leq \|f\|_{\mathcal{H}} \cdot \|k(x,\cdot) - k(x',\cdot)\|_{\mathcal{H}}$ and $\frac{\|k(x,\cdot)-k(x',\cdot)\|_{\mathcal{H}}}{\|x-x'\|_2} \leq \sup_{z>0} \frac{g_z}{z}$. They attain equality at potentially very different (x, x') pairs, and the former depends on f while the latter does not. In contrast, our bound in Equation 4.6 only relaxes once, leveraging the efficiently approximable partial derivatives g_j in Section 4.2.3.

Figure 9 compares the right-hand side of Equation 4.6 with $\|f\|_{\mathcal{H}} \max_{z>0} \frac{t(z)}{z}$ from Equation 4.2, in serving the role as an upper bound for $\sup_{x\in\mathcal{X}} \|\nabla f(x)\|_2$. The RHS of Equation 4.6 was computed by Equation 4.16 with 10000 randomly sampled landmark points $\{w^s\}$, and further increasing the sample size led to little difference. Smaller values mean a tighter bound. We randomly generated 300 functions $f = \sum_{i=1}^{100} \gamma_i k(x^i, \cdot)$, where γ_i were sampled uniformly



(a) Inverse kernel (b) Gaussian kernel with $\sigma = \frac{1}{2}$ (c) Gaussian kernel with $\sigma = 1$

Figure 9: Comparison of the right-hand side (RHS) of Equation 4.2 and Equation 4.6.

at random from $[-2, 2]$ and x^i from the MNIST dataset. Larger σ in Gaussian kernel means smoother functions, making the two bounds closer. Clearly the former is significantly lower, which motivated us to develop refined algorithms to approximate $\|g_j\|_{\mathcal{H}}^2$. The exact evaluation of $\|g_j\|_{\mathcal{H}}$ still remains challenging, but it is amenable to the Nyström approximation. Intuitively, $\|g_j\|_{\mathcal{H}} \approx \|\tilde{g}_j\|_2$, where $\tilde{g}_j \in \mathbb{R}^n$ is the Nyström approximation computed by

$$\tilde{g}_j := K^{-\frac{1}{2}} \begin{pmatrix} g_j(w^1) \\ \vdots \\ g_j(w^n) \end{pmatrix} = K^{-\frac{1}{2}} \begin{pmatrix} \langle g_j, k(w^1, \cdot) \rangle_{\mathcal{H}} \\ \vdots \\ \langle g_j, k(w^n, \cdot) \rangle_{\mathcal{H}} \end{pmatrix}, \quad \text{where} \quad K = [k(w^i, w^{i'})]_{i,i'} \tag{4.8}$$

$$= (Z^\top Z)^{-\frac{1}{2}} Z^\top g_j, \quad \text{where} \quad Z = (k(w^1, \cdot), k(w^2, \cdot), \ldots, k(w^n, \cdot)). \tag{4.9}$$

Here the Z "matrix" is just a notation since each column is a function in $\mathcal{H}$. The meaning of the "matrix-vector" multiplication $Z^\top g_j$ is obviously defined as $(g_j(w^1), \ldots, g_j(w^n))^\top$, and so

we just adopt this notation for simplicity. The term $(Z^\top Z)^{-\frac{1}{2}} Z^\top$ is effectively orthogonalizing Z, in exactly the same spirit as Equation 4.5.

We note in passing that although Nyström approximation has been extensively used in machine learning, its use has been restricted to approximating kernel Gram matrix [113] and PCA related analysis [114]. The general use of approximating an arbitrary function in the RKHS (other than $k(x, \cdot)$) has been left largely unstudied. Other methods such as Fourier transformation are not applicable here because they are inherently restricted to the approximation of $k(x, y)$ [115, 116].

Applying Nyström approximation Equation 4.8 to Equation 4.7, our training objective can be formulated as

$$\min_{f \in \mathcal{H}} \quad \frac{1}{l} \sum_{i=1}^{l} \ell(f(x^i), y^i) + \frac{\lambda}{2} \|f\|_{\mathcal{H}}^2 \tag{4.10}$$

$$s.t. \quad \left\| (Z^\top Z)^{-\frac{1}{2}} Z^\top G \right\|_F \leq L, \quad \text{where} \quad G = (g_1, \ldots, g_d). \tag{4.11}$$

This is a convex optimization problem which can be solved using the representer theorem. Since the mapping $f \mapsto \partial^j f(w^s)$ is a bounded linear functional for the kernels we consider here,

we can denote its corresponding representer as z_j^s: $\left\langle z_j^s, f \right\rangle_{\mathcal{H}} = \partial^j f(w^s) = g_j(w^s)$.[1] Then the optimal f must be in the form of

$$f = \frac{1}{l}\sum_{i=1}^{l} \gamma_i k(x^i, \cdot) + \sum_{s=1}^{n} \beta_s z_j^s. \tag{4.12}$$

For simplicity, we can also restrict our search to the span of $\{k(x^i, \cdot)\}$, i.e., setting all β_s to 0. To conclude, our optimization problem is:

$$\min_{\gamma \in \mathbb{R}^l} \quad \frac{1}{l}\sum_{i=1}^{l} \ell(f(x^i), y^i) + \frac{\lambda}{2}\|f\|_{\mathcal{H}}^2 \tag{4.13}$$

$$s.t. \quad \left\|(Z^\top Z)^{-\frac{1}{2}} Z^\top G\right\|_F \leq L \tag{4.14}$$

$$\text{where} \quad f = \frac{1}{l}\sum_{i=1}^{l} \gamma_i k(x^i, \cdot) \qquad \text{and} \qquad g_j = \partial^j f = \frac{1}{l}\sum_{i=1}^{l} \gamma_i \partial^{0,j} k(x^i, \cdot). \tag{4.15}$$

Although in general it is still hard to ensure $\sum_{j=1}^{d} \|g_j\|_{\mathcal{H}}^2 \leq L^2 + \epsilon$ by enforcing the convex constraint $\sum_{j=1}^{d} \|\tilde{g}_j\|_2^2 \leq L^2$ with a polynomial sample size, fortunately, we found that this can indeed be achieved for a quite general class of kernels that exhibit some decomposed structure.

4.2.3 A Coordinate-wise Nyström Approximation for Product Kernels

When the kernel is decomposable, we may exploit its structure to reduce the sample complexity. For example, periodic kernels, Gaussian kernels, and Laplace kernels are multiplicatively decomposed over the coordinates. We will consider $k(x, y) = \prod_{j=1}^{d} k_j(x_j, y_j)$ where k_j

---

[1] Trivially $z_s^j \neq g_j$, and $f \mapsto \partial^j f(w^s)$ being a bounded linear functional does not imply $g_j \in \mathcal{H}$.

is a base kernel for coordinate j and $\mathcal{X} = \prod_j \mathcal{X}_j$. Without loss of generality, we can assume k_j does not depend on j, and accordingly we denote $k(x, y) = \prod_{j=1}^{d} k_0(x_j, y_j)$ and $\mathcal{X} = \mathcal{X}_0^d$. Denote the RKHS on $\mathcal{X}_0$ induced by k_0 as $\mathcal{H}_0$.

The key benefit of this decomposition is that the derivative $\partial^{0,1}k(x, y)$ can be written as $\partial^{0,1}k_0(x_1, y_1)\prod_{j=2}^{d} k_0(x_j, y_j)$. Since $k_0(x_j, y_j)$ can be easily dealt with, approximation will be needed only for $\partial^{0,1}k_0(x_1, y_1)$. Applying this idea to $g_1 = \frac{1}{l}\sum_{a=1}^{l} \gamma_i \partial^{0,1}k(x^a, \cdot)$, we can derive

$$\|g_1\|^2 = \frac{1}{l^2}\sum_{a,b=1}^{l} \gamma_a\gamma_b \left\langle \partial^{0,1}k_0(x_1^a, \cdot), \partial^{0,1}k_0(x_1^b, \cdot)\right\rangle_{\mathcal{H}_0} \prod_{j=2}^{d} k_0(x_j^a, x_j^b). \tag{4.16}$$

Given samples $\{w_1^1, \ldots, w_1^n\}$ drawn from the distribution on $\mathcal{X}_0$, set

$$Z = (k_0(w_1^1, \cdot), k_0(w_1^2, \cdot), \ldots, k_0(w_1^n, \cdot)).$$

We can now use Nyström approximation to compute

$$\left\langle \partial^{0,1}k_0(x_1^a, \cdot), \partial^{0,1}k_0(x_1^b, \cdot)\right\rangle_{\mathcal{H}_0} = \partial^{0,1}k_0(x_1^a, \cdot)^\top Z(Z^\top Z)^{-1}Z^\top \partial^{0,1}k_0(x_1^b, \cdot) \tag{4.17}$$

$$= \begin{pmatrix} \partial^{0,1}k_0(x_1^a, w_1^1) \\ \vdots \\ \partial^{0,1}k_0(x_1^a, w_1^n) \end{pmatrix}^\top (Z^\top Z)^{-1} \begin{pmatrix} \partial^{0,1}k_0(x_1^b, w_1^1) \\ \vdots \\ \partial^{0,1}k_0(x_1^b, w_1^n) \end{pmatrix}. \tag{4.18}$$

As a result, the approximation bounds only need to be derived for each coordinate, making the sample complexity immune to the exponential dependency on the dimensionality d, which

we will show in the next section. Putting together $\sum_j \|g_j\|^2 \leq L^2$ leads to a convex constraint on $\boldsymbol{\gamma}$, and now samples w_j^s for coordinate j are drawn independently from those for coordinate j'.

4.3 Sample Complexity

We now state our result on sample complexity to achieve Equation 4.7, after introducing the basic constructs in kernel spectrum. Let k be a continuous kernel on a compact metric space $\mathcal{X}$, and ν be a finite Borel measure on $\mathcal{X}$ with $supp[\nu] = \mathcal{X}$. Recall from Chapter 4 of [110] that the integral operator for k and ν is defined by

$$T_k \;=\; I_k \circ S_k : L_2(\mathcal{X}, \nu) \to L_2(\mathcal{X}, \nu) \tag{4.19}$$

$$\text{where} \;\; S_k \;:\; L_2(\mathcal{X}; \nu) \to \mathcal{C}(\mathcal{X}), \quad (S_k f)(x) \;=\; \int k(x, y) f(y) d\nu(y), \quad f \in L_2(\mathcal{X}, \nu), \tag{4.20}$$

$$I_k \;:\; \mathcal{C}(\mathcal{X}) \hookrightarrow L_2(\mathcal{X}; \nu), \text{ inclusion operator.} \tag{4.21}$$

By the spectral theorem, if T_k is compact, then there is an at most countable orthonormal set (ONS) $\{\tilde{e}_j\}_{j \in J}$ of $L_2(\mathcal{X}, \nu)$ and $\{\lambda_j\}_{j \in J}$ with $\lambda_1 \geq \lambda_2 \geq \ldots > 0$ such that

$$Tf = \sum_{j \in J} \lambda_j \left\langle f, \tilde{e}_j \right\rangle_{L_2(\mathcal{X}; \nu)} \tilde{e}_j, \qquad f \in L_2(\mathcal{X}, \nu). \tag{4.22}$$

In particular, we have $\langle \tilde{e}_i, \tilde{e}_j \rangle_{L_2(\mathcal{X};\nu)} = \delta_{ij}$ (i.e., equals 1 if $i = j$, and 0 otherwise), and $T\tilde{e}_i = \lambda_i \tilde{e}_i$. Since $\tilde{e}_j$ is an equivalent class instead of a single function, we assign a set of continuous functions $e_j = \lambda_j^{-1} S_k \tilde{e}_j \in \mathcal{C}(\mathcal{X})$, which clearly satisfies

$$\langle e_i, e_j \rangle_{L_2(\mathcal{X};\nu)} = \delta_{ij}, \quad Te_j = \lambda_j e_j. \tag{4.23}$$

We will call λ_j and e_j as eigenvalues and eigenfunctions respectively, and $\{e_j\}_{j\in J}$ clearly forms an ONS. By Mercer's theorem,

$$k(x, y) = \sum_{j\in J} \lambda_j e_j(x) e_j(y), \tag{4.24}$$

and all functions in $\mathcal{H}$ can be represented by $\sum_{j\in J} a_j e_j$ where $\{a_j/\sqrt{\lambda_j}\} \in \ell^2(J)$. The inner product in $\mathcal{H}$ is equivalent to $\left\langle \sum_{j\in J} a_j e_j, \sum_{j\in J} b_j e_j \right\rangle_{\mathcal{H}} = \sum_{j\in J} a_j b_j/\lambda_j$. Therefore it is easy to see that

$$\varphi_j := \sqrt{\lambda_j} e_j, \qquad j \in J \tag{4.25}$$

is an orthonormal basis of $\mathcal{H}$, with Moreover, for all $f \in \mathcal{H}$ with $f = \sum_{j\in J} a_j e_j$, we have $\langle f, e_j \rangle_{\mathcal{H}} = a_j/\lambda_j$, $\langle f, \varphi_j \rangle_{\mathcal{H}} = a_j/\sqrt{\lambda_j}$, and

$$f = \sum_j \langle f, \varphi_j \rangle_{\mathcal{H}} \varphi_j = \sum_j \sqrt{\lambda_j} \langle f, e_j \rangle_{\mathcal{H}} \varphi_j = \sum_j \lambda_j \langle f, e_j \rangle_{\mathcal{H}} e_j. \tag{4.26}$$

Most kernels used in machine learning are infinite dimensional, i.e., $J = \mathbb{N}$. For convenience, we define $\Phi_m := (\varphi_1, \ldots, \varphi_m)$ and $\Lambda_m = \mathrm{diag}(\lambda_1, \ldots, \lambda_m)$.

4.3.1 General Sample Complexity and Assumptions on The Product Kernel

In this section, we first consider kernels k_0 with **scalar input**, i.e., $\mathcal{X}_0 \subseteq \mathbb{R}$. Assume there is a measure ν_0 on $\mathcal{X}_0$. This will serve as the basis for the more general product kernels in the form of $k(x, y) = \prod_{j=1}^{d} k_0(x_j, y_j)$ defined over $\mathcal{X}_0^d$.

Our proof is built upon the following two properties of the kernel in question.

Assumption 2. *Suppose $k_0(x, x) = 1$ and $\partial^{0,1} k_0(x, \cdot) \in \mathcal{H}_0$ for all $x \in \mathcal{X}_0$. For all $\epsilon > 0$, there exists $N_\epsilon \in \mathbb{N}$ such that the tail energy of $\partial^{0,1} k_0(x, \cdot)$ beyond the N_ϵ-th eigenpair is less than ϵ, uniformly for all $x \in \mathcal{X}_0$. That is,*

$$N_\epsilon := \inf_m \{ \left\| \partial^{0,1} k_0(x, \cdot) - \Phi_m \Phi_m^\top \partial^{0,1} k_0(x, \cdot) \right\|_{\mathcal{H}_0} < \epsilon \text{ for all } x \in \mathcal{X}_0 \tag{4.27}$$

$$\text{and } \left\| k_0(x, \cdot) - \Phi_m \Phi_m^\top k_0(x, \cdot) \right\|_{\mathcal{H}_0} < \epsilon \text{ for all } x \in \mathcal{X}_0 \} < \infty. \tag{4.28}$$

Note since all $f \in \mathcal{H}_0$ can be written as $\sum_{j \in J} a_j e_j$ with $\{a_j / \sqrt{\lambda_j}\} \in \ell^2(J)$, there must be a large enough m such that $\left\| f - \Phi_m \Phi_m^\top f \right\|_{\mathcal{H}_0} \leq \epsilon$. So the key assumption here is that this holds *uniformly* for all $x \in \mathcal{X}_0$. Our sample complexity will be almost linear in N_ϵ.

Assumption 3. *Under Assumption 2, the RKHS inner product of $\partial^{0,1} k_0(x, \cdot)$ with $\{e_i : i \in N_\epsilon\}$ is uniformly bounded by a constant M_ϵ for all $x \in \mathcal{X}_0$:*

$$M_\epsilon := \sup_{x \in \mathcal{X}_0} \max_{i \in [N_\epsilon]} \left| \left\langle \partial^{0,1} k_0(x, \cdot), e_i \right\rangle_{\mathcal{H}_0} \right| < \infty. \tag{4.29}$$

Moreover $e_i(x)$ is also bounded over $i \in N_\epsilon$:

$$Q_\epsilon := \sup_{x \in \mathcal{X}_0} \max_{i \in [N_\epsilon]} |e_i(x)| < \infty. \tag{4.30}$$

In the next subsection, we will examine how three different kernels satisfy/unsatisfy the Assumptions 2 and 3, and what the value of N_ϵ is. For each case, we will specify ν_0 on $\mathcal{X}_0$, and the measure on $\mathcal{X}_0^d$ is trivially $\nu = \nu_0^d$.

With Assumptions 2 and 3, our main result of sample complexity can be stated as follows in an informal way.

Theorem 5. *Suppose k_0 and ν_0 satisfy Assumptions 2 and 3. Let $\{w^s\}_{s=1}^n$ be sampled i.i.d. from ν. Then the optimal solution to Equation 4.13 with coordinate-wise Nyström approximation Equation 4.16 and Equation 4.17 satisfies Equation 4.7 with probability $1-\delta$ if $n \geq \tilde{\Theta}\left(\frac{1}{\epsilon^2} d^2 N_\epsilon^2 M_\epsilon^2 Q_\epsilon^2 \log \frac{dN_\epsilon}{\delta}\right)$. Here $\tilde{\Theta}$ hides all poly-log terms.*

To develop a formal statement, we first provide the sample complexity for approximating the partial derivatives.

Theorem 6. *Suppose $\{w^s\}_{s=1}^n$ are drawn iid from ν_0 on $\mathcal{X}_0$, where ν_0 is the uniform distribution on $[-v/2, v/2]$ for periodic kernels or periodized Gaussian kernels. Let*

$$Z := (k_0(w^1, \cdot), \quad k_0(w^2, \cdot), \ldots, \quad k_0(w^n, \cdot)),$$

and $g_1 = \frac{1}{l}\sum_{a=1}^{l} \gamma_a g_1^a$*:* $\mathcal{X}_0^d \to \mathbb{R}$*, where* $\|\boldsymbol{\gamma}\|_\infty \leq c_1$ *and*

$$g_1^a(y) = \partial^{0,1}k(x^a, y) = h_1^a(y_1)\prod_{j=2}^{d} k_0(x_j^a, y_j) \quad \textit{with} \quad h_1^a(\cdot) := \partial^{0,1}k_0(x_1^a, \cdot). \tag{4.31}$$

Given $\epsilon \in (0, 1]$*, let* $\Phi_m = (\varphi_1, \ldots \varphi_m)$ *where* $m = N_\epsilon$*. Then with probability* $1-\delta$*, the following holds when the sample size* $n = \max(N_\epsilon, \frac{5}{3\epsilon^2}N_\epsilon Q_\epsilon^2 \log\frac{2N_\epsilon}{\delta})$*:*

$$\|g_1\|_{\mathcal{H}}^2 \leq \frac{1}{l^2}\boldsymbol{\gamma}^\top K_1 \boldsymbol{\gamma} + 3c_1\left(1 + 2\sqrt{N_\epsilon}M_\epsilon\right)\epsilon, \tag{4.32}$$

$$\textit{where} \quad (K_1)_{a,b} = (h_1^a)^\top Z(Z^\top Z)^{-1}Z^\top h_1^b \prod_{j=2}^{d} k_0(x_j^a, x_j^b). \tag{4.33}$$

Then we obtain the formal statement of sample complexity, as stated in the following corollary, by combining all the coordinates from Theorem 6.

Corollary 1. *Applying the results in Equation 4.32 for coordinates from 1 to d and using the union bound, we have that with sample size* $n = \max(N_\epsilon, \frac{5}{3\epsilon^2}N_\epsilon Q_\epsilon^2 \log\frac{2N_\epsilon}{\delta})$*, the following holds with probability* $1 - d\delta$*,*

$$\sum_{j=1}^{d}\|g_j\|_{\mathcal{H}}^2 \leq \frac{1}{l^2}\sum_{j=1}^{d}\boldsymbol{\gamma}^\top K_j\boldsymbol{\gamma} + 3c_1 d\left(1 + 2\sqrt{N_\epsilon}M_\epsilon\right)\epsilon. \tag{4.34}$$

Equivalently, if N_ϵ, M_ϵ and Q_ϵ are constants or poly-log terms of ϵ which we treat as constant, then to ensure $\sum_{j=1}^{d} \|g_j\|_{\mathcal{H}}^2 \leq \frac{1}{l^2} \sum_{j=1}^{d} \boldsymbol{\gamma}^\top K_j \boldsymbol{\gamma} + \epsilon$ with probability $1-\delta$, the sample size needs to be

$$n = \frac{15}{\epsilon^2} c_1^2 d^2 \left(1 + 2\sqrt{N_\epsilon} M_\epsilon\right)^2 N_\epsilon Q_\epsilon^2 \log \frac{2dN_\epsilon}{\delta}. \tag{4.35}$$

Remark 5. *The first term on the right-hand side of Equation 4.34 is explicitly upper bounded by L in our training objective. In the case of Theorem 7, the values of Q_ϵ, N_ϵ, and M_ϵ lead to a $\tilde{O}(\frac{d^2}{\epsilon^2})$ sample complexity. If we further zoom into the dependence on the period v, then note that N_ϵ is almost a universal constant while $M_\epsilon = \frac{\sqrt{2}\pi}{v}(N_\epsilon - 1)$. So overall, n depends on v by $\frac{1}{v^2}$. This is not surprising because smaller period means higher frequency, hence more samples are needed.*

Remark 6. *Corollary 1 uses the same set of samples $\{w^s\}_{s=1}^n$ for all coordinates. When coordinates differ in their domains, we can draw different sets of samples for them. The sample complexity hence grows by d times as we only use a weak union bound. More refined analysis could save us a factor of d as these sets of samples are independent of each other.*

4.4 Checking the Assumptions for Different Kernels

In this section, we will check the Assumption 2 and Assumption 3 for three different types of kernels: periodic kernels, Gaussian kernels and Non-product kernels.

4.4.1 Case 1: Periodic Kernels

Periodic kernels on $\mathcal{X}_0 := \mathbb{R}$ are translation invariant, and can be written as $k_0(x, y) = \kappa(x - y)$ where $\kappa : \mathbb{R} \to \mathbb{R}$ is a) periodic with period v; b) even, with $\kappa(-t) = \kappa(t)$; and c) normalized with $\kappa(0) = 1$. A general treatment was given by [117], and an example was given by David MacKay in [118]:

$$k_0(x, y) = \exp\left(-\frac{1}{2\sigma^2}\sin\left(\frac{\pi}{v}(x - y)\right)^2\right). \tag{4.36}$$

We define ν_0 to be a uniform distribution on $[-\frac{v}{2}, \frac{v}{2}]$, and let $\omega_0 = 2\pi/v$.

Since κ is symmetric, we can simplify the Fourier transform of $\kappa(t)\delta_v(t)$, where $\delta_v(t) = 1$ if $t \in [-v/2, v/2]$, and 0 otherwise:

$$F(\omega) = \frac{1}{\sqrt{2\pi}}\int_{-v/2}^{v/2}\kappa(t)\cos(\omega t)\,\mathrm{d}t. \tag{4.37}$$

It is now easy to observe that thanks to periodicity and symmetry of κ, for all $j \in \mathbb{Z}$,

$$\frac{1}{v}\int_{-v/2}^{v/2} k_0(x, y)\cos(j\omega_0 y)\,\mathrm{d}y = \frac{1}{v}\int_{-v/2}^{v/2}\kappa(x - y)\cos(j\omega_0 y)\,\mathrm{d}y \tag{4.38}$$

$$= \frac{1}{v}\int_{x-v/2}^{x+v/2}\kappa(z)\cos(j\omega_0(x - z))\,\mathrm{d}z \quad (\text{note } \cos(j\omega_0(x - z)) \text{ also has period } v) \tag{4.39}$$

$$= \frac{1}{v}\int_{-v/2}^{v/2}\kappa(z)[\cos(j\omega_0 x)\cos(j\omega_0 z) + \sin(j\omega_0 x)\sin(j\omega_0 z))\,\mathrm{d}z \quad (\text{by periodicity}) \tag{4.40}$$

$$= \frac{1}{v}\cos(j\omega_0 x)\int_{-v/2}^{v/2}\kappa(z)\cos(j\omega_0 z)\,\mathrm{d}z \quad (\text{by symmetry of } \kappa) \tag{4.41}$$

$$= \frac{\sqrt{2\pi}}{v}F(j\omega_0)\cos(j\omega_0 x). \tag{4.42}$$

And similarly,

$$\frac{1}{v}\int_{-v/2}^{v/2} k_0(x,y)\sin(j\omega_0 y)\,\mathrm{d}y = \frac{\sqrt{2\pi}}{v}F(j\omega_0)\sin(j\omega_0 x). \tag{4.43}$$

Therefore the eigenfunctions of the integral operator T_k are

$$e_0(x) = 1, \qquad e_j(x) := \sqrt{2}\cos(j\omega_0 x), \qquad e_{-j}(x) := \sqrt{2}\sin(j\omega_0 x) \quad (j \geq 1) \tag{4.44}$$

and the eigenvalues are $\lambda_j = \frac{\sqrt{2\pi}}{v}F(j\omega_0)$ for all $j \in \mathbb{Z}$ with $\lambda_{-j} = \lambda_j$. An important property our proof will rely on is that

$$e_j'(x) = -j\omega_0 e_{-j}(x), \quad \text{for all } j \in \mathbb{Z}. \tag{4.45}$$

Applying Mercer's theorem in Equation 4.24 and noting $\kappa(0) = 1$, we derive $\sum_{j\in\mathbb{Z}} \lambda_j = 1$.

Checking the Assumptions 2 and 3. The following theorem summarizes the assumptions and conclusions regarding the satisfaction of Assumptions 2 and 3. Again we focus on the case of $\mathcal{X} \subseteq \mathbb{R}$.

Theorem 7. *Suppose the periodic kernel with period v has eigenvalues λ_j that satisfies*

$$\lambda_j(1+j)^2\max(1,j^2)(1+\delta(j\geq 1)) \leq c_6 \cdot c_4^{-j}, \quad \textit{for all } j \geq 0, \tag{4.46}$$

where $c_4 > 1$ and $c_6 > 0$ are universal constants. Then Assumption 2 holds with

$$N_\epsilon = 1 + 2\lfloor n_\epsilon \rfloor, \quad \textit{where} \quad n_\epsilon := \log_{c_4}\left(\frac{2.1c_6}{\epsilon^2}\max\left(1, \frac{v^2}{4\pi^2}\right)\right). \tag{4.47}$$

In addition, Assumption 3 holds with $Q_\epsilon = \sqrt{2}$ and $M_\epsilon = \frac{2\sqrt{2}\pi}{v}\lfloor n_\epsilon \rfloor = \frac{\sqrt{2}\pi}{v}(N_\epsilon - 1)$.

For example, if we set $v = \pi$ and $\sigma^2 = 1/2$ in the kernel in Equation 4.36, elementary calculation shows that the condition Equation 4.46 is satisfied with $c_4 = 2$ and $c_6 = 1.6$.

4.4.2 Case 2: Gaussian Kernels

Gaussian kernels $k(x, y) = \exp(-\|x - y\|^2/(2\sigma^2))$ are obviously product kernels with $k_0(x_1, y_1) = \kappa(x_1 - y_1) = \exp(-(x_1 - y_1)^2/(2\sigma^2))$. It is also translation invariant. The spectrum of Gaussian kernel k_0 on $\mathbb{R}$ is known; see, e.g., Chapter 4.3.1 of [119] and Section 4 of [120]. Let ν be a Gaussian distribution $\mathcal{N}(0, \sigma^2)$. Setting $\epsilon^2 = \alpha^2 = (2\sigma^2)^{-1}$ in Eq 12 and 13 of [121], the eigenvalue and eigenfunctions are (for $j \geq 0$):

$$\lambda_j = c_0^{-j-1/2}, \quad \text{where} \quad c_0 = \frac{1}{2}(3 + \sqrt{5}) \tag{4.48}$$

$$e_j(x) = \frac{5^{1/8}}{2^{j/2}}\exp\left(-\frac{\sqrt{5} - 1}{4}\frac{x^2}{\sigma^2}\right)\frac{1}{\sqrt{j!}}H_j\left(\sqrt[4]{1.25}\,\frac{x}{\sigma}\right), \tag{4.49}$$

where H_j is the Hermite polynomial of order j.

Although the eigenvalues decay exponentially fast, the eigenfunctions are not uniformly bounded in the L_∞ sense. Although the latter can be patched if we restrict x to a bounded set,

the above closed-form of eigen-pairs will no longer hold, and the analysis will become rather challenging.

To resolve this issue, we resort to the period-ization technique proposed by [117]. Consider $\kappa(x) = \exp(-x^2/(2\sigma^2))$ when $x \in [-v/2, v/2]$, and then extend κ to $\mathbb{R}$ as a periodic function with period v. Again let ν be the uniform distribution on $[-v/2, v/2]$. As can be seen from the discriminant function $f = \frac{1}{l}\sum_{i=1}^{l} \gamma_i k(x^i, \cdot)$ in Equation 4.15, as along as our training and test data both lie in $[-v/4, v/4]$, the modification of κ outside $[-v/2, v/2]$ does not effectively make any difference. Although the term $\partial^{0,1} k_0(x_1^a, w_1^1)$ in Equation 4.18 may possibly evaluate κ outside $[-v/2, v/2]$, it is only used for testing the gradient norm bound of κ.

With this periodized Gaussian kernel, it is easy to see that $Q_\epsilon = \sqrt{2}$. If we standardize by $\sigma = 1$ and set $v = 5\pi$ as an example, it is not hard to see that Equation 4.46 holds with $c_4 = 1.25$ and $c_6 = 50$. The expressions of N_ϵ and M_ϵ then follow from Theorem 7 directly.

4.4.3 Case 3: Non-product Kernels

The above analysis has been restricted to product kernels. But in practice, there are many useful kernels that are not decomposable. A prominent example is the inverse kernel: $k(x, y) = (2 - x^\top y)^{-1}$. In general, it is extremely challenging to analyze eigenfunctions, which are commonly *not* bounded [122, 123], i.e., $\sup_{i\to\infty} \sup_x |e_i(x)| = \infty$. The opposite was (incorrectly) claimed in Theorem 4 of [117] by citing an incorrect result in [124, p. 145], which was later corrected by [122] and Steve Smale. Indeed, uniform boundedness is not known even for Gaussian kernels with uniform distribution on $[0, 1]^d$ [125], and [126, Theorem 5] showed the unboundedness for Gaussian kernels with uniform distribution on the unit sphere when $d \geq 3$.

Here we only present the limited results that we have obtained on the eigenvalues of the integral operator of inverse kernels with a uniform distribution on the unit ball. The analysis of eigenfunctions is left for future work. Specifically, in order to drive the eigenvalue λ_i below ϵ, i must be at least $d^{\lceil \log_2 \frac{1}{\epsilon} \rceil + 1}$. This is a quasi-quadratic bound if we view d and $1/\epsilon$ as two large variables. The detailed results are relegated to Appendix C.3.

4.5 Experiments

In this section, we show the results of numerical evaluations. First, we validate the efficiency of the Lipschitz constant enforcing algorithm proposed in Section 4.2. Then we evaluated the robustness of Lipschitz regularized kernel machines through extensive experiments.

4.5.1 Efficiency of Enforcing Lipschitz Constant

The six different ways to train SVMs with Lipschitz regularization are summarized in Algorithm 2.

Finding the exact $\arg\max_{x \in X} \left\| \nabla f^{(i)}(x) \right\|$ is intractable, so we used a local maximum found by L-BFGS with 15 random initializations as the Lipschitz constant of the current solution $f^{(i)}$ ($L^{(i)}$ in step 6). The solution found by L-BFGS is also used as the new greedy point added in step 5b.

Furthermore, the kernel expansion $f(x) = \frac{1}{l} \sum_{a=1}^{l} \gamma_a k(x^a, \cdot)$ can lead to high cost in optimization (our experiment used $l = 10000$), and therefore we used *another* Nyström approximation for the kernels. We randomly sampled 2000 landmark points, and based on them we computed the Nyström approximation for each $k(x^a, \cdot)$, denoted as $\tilde{\phi}(x^a) \in \mathbb{R}^{2000}$. Then $f(x)$ can be written as $\frac{1}{l} \sum_{a=1}^{l} \gamma_a \tilde{\phi}(x^a)^\top \tilde{\phi}(x)$. Defining $w = \frac{1}{l} \sum_{a=1}^{l} \gamma_a \tilde{\phi}(x^a)$, we can equivalently

Algorithm 2: Training binary SVMs by enforcing Lipschitz constant L

1 Initialize the constraint set S by some random samples from X.

2 **for** $i = 1, 2, \ldots$ **do**

3 Train SVM using one of the following constraints:

(1) **Brute-force:** $\|\nabla f(w)\|_2^2 \leq L^2,\ \forall\ w \in S$

(2) **Nyström holistic:** $\sum_{j=1}^{d} \|\tilde{g}_j\|_2^2 \leq L^2$ using S as the set $\{w^1, \ldots, w^n\}$ in Equation 4.8

(3) **Nyström coordinate wise:** $\sum_{j=1}^{d} L_n^2(g_j) \leq L^2$ using S as the set $\{w^1, \ldots, w^n\}$ in Equation 4.17

4 Let the trained SVM be $f^{(i)}$.

5 Find a new w to add to S by one of the following methods:

(a) **Random:** randomly sample w from X.

(b) **Greedy:** find $\arg\max_{x \in X} \left\|\nabla f^{(i)}(x)\right\|$ (approximately) by L-BFGS with 15 random initializations.

6 Record $L^{(i)} = \max_{x \in X} \left\|\nabla f^{(i)}(x)\right\|$.

optimize over w, and the RKHS norm bound on f can be equivalently imposed as the ℓ_2-norm bound on w.

To summarize, Nyström approximation is used in two different places: one for approximating the kernel function, and one for computing $\|g_j\|_{\mathcal{H}}$ either holistically or coordinate wise. For the former, we randomly sampled 2000 landmark points; for the latter, we used greedy selection as option b in step 5 of Algorithm 2.

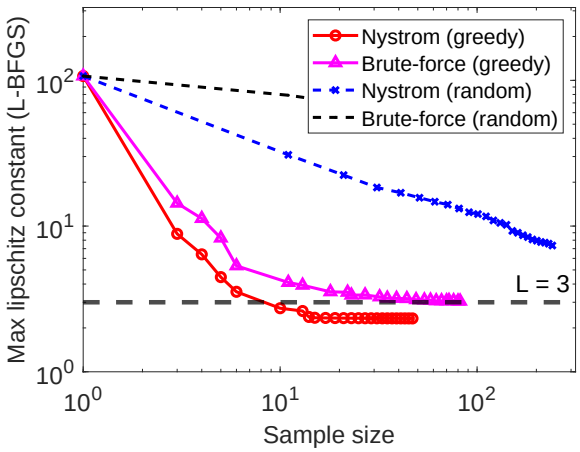


Figure 10: Comparison of efficiency in enforcing Lipschitz constant by various methods

Figure 10 plots how fast the regularization on gradient norm becomes effective when more and more points w are added to the constraint set. We call them "samples" although it is not so random in the greedy method, modulo the random initialization of BFGS within the greedy

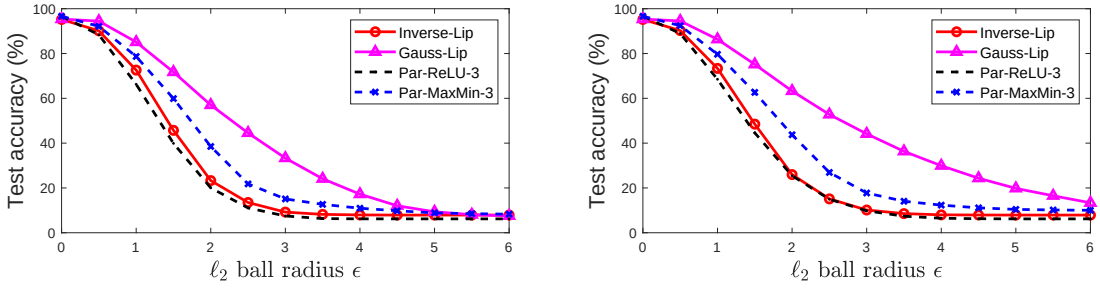
method. The horizontal axis is the loop index i in Algorithm 2, and the vertical axis is $L^{(i)}$ therein, which is the estimation of the Lipschitz constant of the current solution $f^{(i)}$. We used 400 random examples (200 images of digit 1 and 200 images of digit 0) in the MNIST dataset and set $L = 3$ and RKHS norm $\|f\|_{\mathcal{H}} \leq \infty$ for all algorithms. Inverse kernel is used, hence no results are shown for coordinate-wise Nyström.

Clearly the Nyström algorithm is more efficient than the Brute-force algorithm, and the greedy method significantly reduces the number of samples for both algorithms. In fact, Nyström with greedy selection eventually fell below the prespecified L, because of the gap in Equation 4.6.

4.5.2 Robustness

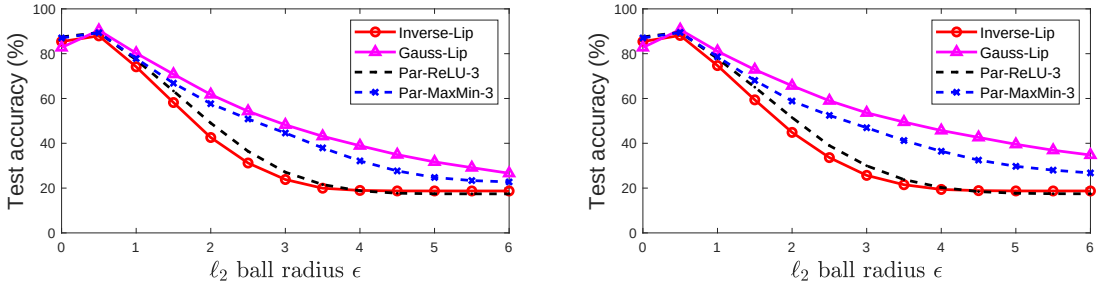
We studied the empirical robustness and accuracy of the proposed Lipschitz regularization technique for adversarial training of kernel methods, under both Gaussian kernel and inverse kernel. Comparison will be made with state-of-the-art defense algorithms under effective attacks.

Datasets. We tested on two datasets: MNIST and Fashion-MNIST, and used 10000 examples for training, and 10000 examples for test. Both datasets have 10 classes, and all the classes get almost the same number of examples, both in training and test sets. Each image is represented as a 784 dimensional feature vector, with each feature/pixel normalized to $[0, 1]$. To eliminate the complication of data normalization "inside" the training algorithms (e.g., inverse kernel), we normalized all examples to unit ℓ_2 norm for *all* algorithms.



(a) PGD attack (b) FGS attack

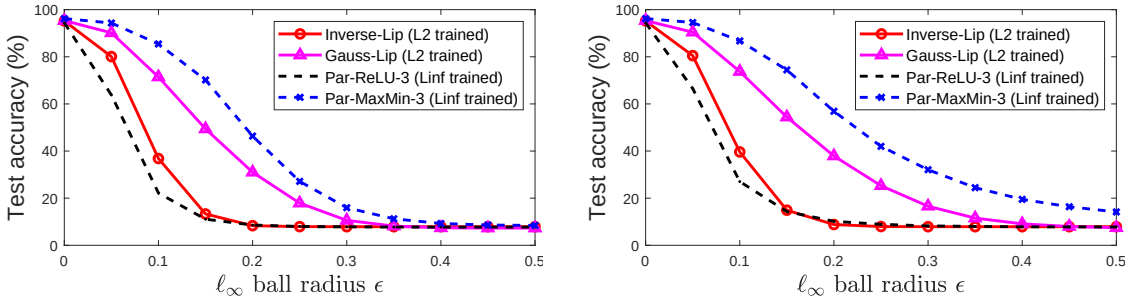
Figure 11: Test accuracy on MNIST under PGD and FGS attacks with ℓ_2 norm radius.



(a) PGD attack (b) FGS attack

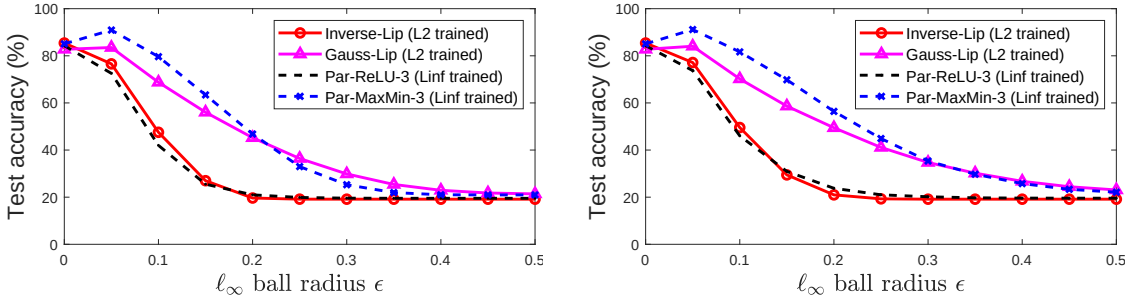
Figure 12: Test accuracy on Fashion-MNIST under PGD and FGS attacks with ℓ_2 norm radius.

Attacks. To evaluate the robustness of the trained model, we attacked them on test examples using the Projected Gradient Descent method with 10 steps [41, PGD] and the Fast Gradient Sign method [88, FGS], where the perturbation δ is constrained within an ℓ_2 or ℓ_∞ ball. To evaluate the robustness, we scaled the perturbation radius ϵ from 0.05 to 0.5 for ℓ_∞ ball, and from 0.5 to 6 for ℓ_2 ball (when $\epsilon = 6$, the average coordinate magnitude $|\delta_i|$ is 0.214).



(a) PGD attack (b) FGS attack

Figure 13: Test accuracy on MNIST under PGD and FGS attacks with ℓ_∞ norm radius.



(a) PGD attack (b) FGS attack

Figure 14: Test accuracy on Fashion-MNIST under PGD and FGS attacks with ℓ_∞ norm radius.

New images after perturbation are renormalised to unit ℓ_2 norm before fed into *all* the trained models for testing.

Algorithms. We compared four training algorithms. The Parseval networks orthonormalise the weight matrices to enforce the Lipschitz constant [45]. We used three hidden layers with 1024 units and ReLU activation (Par-ReLU). Also considered is the Parseval network with

MaxMin activations (Par-MaxMin), which has been shown to enjoy much improved robustness [108]. Both algorithms can be customized for ℓ_2- or ℓ_∞-norm attacks, and naturally we trained them under the corresponding norms. Using cross-entropy loss, they two constitute strong baselines for adversarial learning.

Both Gaussian and inverse kernel machines applied Lipschitz regularisation by randomly and greedily selecting $\{w^s\}$, and they will be referred to as Gauss-Lip and Inverse-Lip, respectively. In practice, we observed that Gauss-Lip with the coordinate-wise Nyström approximation ($L_n(g_j)$ from Equation 4.17) can approximate $\sum_j \|g_j\|^2_{\mathcal{H}}$ with a much smaller number of samples than if using the holistic approximation as in Equation 4.8. Furthermore, we found an even more efficient approach. Inside the iterative training algorithm, we used L-BFGS to find the input that yields the steepest gradient under the current solution, and then added it to the set $\{w_s\}$ (which was initialized with 15 random points). Although L-BFGS is only a local solver, this greedy approach empirically reduces the number of samples by an order of magnitude. See the empirical convergence results in Appendix 4.5.1. Its theoretical analysis is left for future investigation. We also applied this greedy approach to Inverse-Lip.

Extending binary kernel machines to multiclass. Kernel methods employed the one-vs-all strategy with 10 binary base classifiers, each using logistic loss, and enforced the same Lipschitz constant L. After learning a binary classifier $f^\tau := \sum_a \boldsymbol{\alpha}^\tau_a k(x^a, \cdot)$ for each class $\tau \in [10]$, we stacked the weights $\boldsymbol{\alpha}^\tau$ into a matrix $W = \left[\boldsymbol{\alpha}^1, \cdots, \boldsymbol{\alpha}^{10}\right]$, which is analogous to the output layer of the 10-class neural network. As a result, the architectures of kernel method and neural network differ only in the hidden layers, with the former using kernel mapping while the latter

using fully connected layers. The Lipschitz constant of kernel machines can therefore be computed on the same footing as neural networks, namely $\hat{L} := \min_{x \in \mathcal{X}} |||[\nabla f^1(x), \cdots, \nabla f^{10}(x)]|||_{sp}$ (spectral norm). In our experiment, we first trained $\{f^\tau\}$ in kernel methods independently with 2000 landmarks, under a prescribed value of L for all f^τ. Then we evaluated $\hat{L}$ by L-BFGS, and used it to train Par-ReLU and Par-MaxMin.

As pointed out by [42], solely enforcing the Lipschitz constant only achieves suboptimal robustness. So we followed their Lipschitz-margin approach by adding a margin to f^τ for all classes τ except the ground truth class, before feeding them to the loss. This treatment was applied to all the four algorithms. For each ϵ, we tried margins [4] for the kernel machines, $14 \times [4]$ for Par-ReLU, and $14 \times [7]$ for Par-MaxMin. Then we reported the highest accuracy achieved among them.

Parameter selection. We used $L^2 = 20000$ for both Gauss-Lip and Inverse-Lip. The RKHS norm C was set to $10^{2.5}$ so that they had enough expressiveness. After training was completed, we computed that $\hat{L} = 140$ and used it for training ℓ_2-based Par-ReLU and Par-MaxMin. Gauss-Lip achieved high accuracy and robustness on a validation set with bandwidth $\sigma = 0.4$ (for MNIST) and $\sigma = 0.3$ (for Fashion-MNIST). ℓ_∞-based Par-ReLU and Par-MaxMin used $\hat{L} = 1000$ as in [108].

Results. Figure 11 and Figure 12 show how the test accuracy decays as an increasing amount of perturbation (ϵ) in ℓ_2 norm is added to the test images. Clearly Gauss-Lip achieves higher accuracy and robustness than Par-ReLU and Par-MaxMin on both MNIST and Fashion-

MNIST under both PGD and FGS attacks. In contrast, Inverse-Lip only performs similarly to Par-ReLU.

The results for ℓ_∞-norm attacks are showed in Figure 13 and Figure 14. Here our ℓ_2 trained Gauss-Lip significantly outperforms the ℓ_∞ trained Par-ReLU on MNIST, although it is not as robust as Par-MaxMin. This is expected because the RKHS norm of kernel methods is designed for the ℓ_2 norm, and its extension to ℓ_∞ is left for future work. On the positive side, as shown in Figure 13a and Figure 13b for Fashion-MNIST, our ℓ_2 trained Gauss-Lip achieves similar robustness as ℓ_∞ trained Par-MaxMin when $\epsilon > 0.3$. And Gauss-Lip easily outperforms ℓ_∞ trained Par-ReLU, which demonstrated similar robustness as Inverse-Lip on both datasets and attacks.

CHAPTER 5

ROBUSTNESS OF GRAPH CONVOLUTION NETWORKS

(Parts of this chapter were previously published as "Certified Robustness of Graph Convolution Networks for Graph Classification under Topological Attacks " [1], in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.)

5.1 Introduction

Graph convolution networks [127] have been demonstrated to be successful at modeling graph structured data such as interactive networks [128] and protein social networks [129]. In this thesis, we focus on graph classification task, which aims to predict the class label of a graph given its attributes. GCNs, along with other models such as graph kernels [130, 131], have achieved strong performance in graph classification [132–135].

However, similar to most deep learning models in image domain, adversarial attacks also exist for GCNs that maliciously perturb the data to induce specific errors [20, 21]. These include topological attacks (i.e., adding or removing edges) [21, 25] and node attacks (i.e., perturbing node features) [22–24]. They exposes serious security concern in many graph-related applications, such as recommendation system [26], web search [27] and fraud detection [28]. Therefore, it is urgent to analyze the worst-case robustness of the model.

Since the strongest attack is often intractable to compute, constructing certificates of robustness becomes necessary in order to theoretically guarantee the immunity to *any* admissible

attack. Unfortunately, there are only few results on graph applications. [52] developed certificates for PageRank and label/feature propagation under topological perturbation, and [136] developed them for community detection. [22] certified GCNs against node feature perturbations for node classification.

However, when extended to GCNs for graph classification under topological attacks, all of previous techniques face significant challenges or slackness. First of all, the set of admissible topological perturbations may have a complex structure. For example, [136] and [137] only considered the global budgets, while the local budget for each node is also an important consideration. Extending them to such a setting is nontrivial. Furthermore, undirected graphs naturally enforce symmetry on adjacency matrix, which is not considered by existing work. Secondly, graph classification requires pooling together the hidden representation of *all* nodes, which hinders the usage of k-hop subgraph in node classification [22, 138]. Thirdly, GCNs have a specific normalized graph Laplacian structure which introduces a new type of nonlinearity. This structure can not be simply treated as generic function because it often leads to loose certificates.

Our certificate is the first to address all these challenges. Specifically, our first certificate is based on dualization (Section 5.4) which decouples the symmetry constraint and local/global constraints. As a result, given a trained GCN, it can efficiently verify that no topological perturbation can change the graph prediction. Because dualization is not tight, we further propose a certificate based on convex envelope (Section 5.5), which can be computed efficiently through dynamic programming. To characterize the tightness, we also developed a new attack algo-

rithm (Section 5.3). Experiments confirm empirically that both the attack and the certificate are often tight (Section 5.6). As a byproduct, the attack algorithm can be used in a robust hinge loss, hence facilitates the robust training for GCN.

Related Work. Convex envelope relaxation of the ReLU activation [139] is a popular method for certifying the robustness of neural networks in image domain [140–143]. In order to apply the convex envelop tool to GCNs, we need to first deal with the nonlinearity introduced by the normalization in the graph Laplacian. A more general certificate is based on curvature or Lipschitz continuous constant [45,108,144], as we introduced in section 3.5. However, estimating the local or global Lipschitz constant or engineering Lipschitz layers over the discrete domain are complicated. Besides, quantifying the slackness in these relaxations is also generally hard, an issue that has been mitigated by our tightest convex envelop certificate.

Randomized smoothing transforms a classifier into a new smoothed classifier by adding noise to the input, where the resulting smoothed classifier is certified to be robust to the corresponding noise [145–147]. [136, 137] extended it to graph application by injecting discrete noises. However, it is hard to design the noise distribution under both local *and* global budgets.

5.2 Threat Model

We consider the task of graph classification. The training set consists of pairs of (G_k, y_k), where G_k is a directed or undirected graph, and $y_k \in [C] := \{1, \ldots, C\}$ denotes its label for multiclass classification. The vertices of a graph G with n nodes are denoted as $1, 2, \ldots, n$. The topology of G is represented by its adjacency matrix A, where $A_{ij} = 1$ if there exists an edge from i to j, and is 0 otherwise. If the graph is undirected, then A is symmetric. We denote the

i-th row and i-th column of A as $A_{i:}$ and $A_{:i}$, respectively. For simplicity, we set $A_{ii} = 0$, i.e., no self-loop, although self-loops setting can be easily accommodated by our method.

Each node has its own features $x_i \in \mathbb{R}^d$, and we stack them into a matrix $X := (x_1, \ldots, x_n)^\top \in \mathbb{R}^{n \times d}$. So a graph G is uniquely characterized by the tuple (A, X). The graph convolution network uses a weight matrix $\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}$ to mix the hidden representation of nodes, where $\hat{A} = A + I$ with I being the identity matrix, and $\hat{D}$ is a diagonal matrix whose diagonal is $\hat{A}\mathbf{1}$ ($\mathbf{1}$ is a vector of all ones). For simplicity, we will use $\hat{D}^{-1}\hat{A}$ instead of $\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}$ in our development, as it does not lead to significant difference in performance [52].

The one-layer GCN for graph classification tries to minimize the following empirical risk:

$$\min_{W,U} \mathbb{E}_{(G,y)\sim\tilde{p}}\, \ell(\text{pooling}(\sigma(\hat{D}^{-1}\hat{A}XW))U, y). \tag{5.1}$$

where $W \in \mathbb{R}^{d \times d'}$ is the hidden-layer weight matrix, $U \in \mathbb{R}^{d' \times c}$ is the output-layer weight matrix, ℓ is a loss function, σ is the activation function, and $\tilde{p}$ is the empirical distribution of graph-label pairs. The pooling function aggregates the hidden representation of all nodes in a graph. Such pooling functions include i) maximum: $H \mapsto (\max(H_{:1}), \ldots, \max(H_{:d'}))$; ii) attention: $H \mapsto \alpha^\top H$, where $\alpha_i \geq 0$ and $\mathbf{1}^\top\alpha = 1$; iii) averaging: $H \mapsto \frac{1}{n}\mathbf{1}^\top H$, where $H = \sigma(\hat{D}^{-1}\hat{A}XW)$. In this thesis, we consider attention pooling where α is allowed to depend on X but *not* on A, which obviously subsumes average pooling with $\alpha = n^{-1}\mathbf{1}$.

The learned GCN can be attacked by perturbing the feature matrix X, which is a continuous problem and has been well studied [22]. In contrast, attacking the graph topology A

under involved constraints is much more challenging because it leads to a discrete optimization problem: $\max_{A\in\mathcal{A}} \ell(\text{pooling}(\sigma(\hat{D}^{-1}\hat{A}XW))U, y)$, where $\mathcal{A}$ is the admissible set of perturbed graphs in $\{0,1\}^{n\times n}$ with $A_{ii}=0$. Here we consider the following examples of $\mathcal{A}$:

$\mathcal{A}^1$: For each node, at most δ_l incident edges can be added or removed. Formally, we can write it as $\left\|A_{i:} - A^{ori}_{i:}\right\|_1 \le \delta_l$, where A^{ori} is the original adjacency matrix of graph. δ_l can also vary for different nodes i. A can be either symmetric or asymmetric in this case.

$\mathcal{A}^2$: Across all nodes, at most $2\delta_g$ directed edges (or δ_g undirected edges) can be added or removed. That is, $\left\|A - A^{ori}\right\|_1 := \sum_i \left\|A_{i:} - A^{ori}_{i:}\right\|_1 \le 2\delta_g$.

Furthermore, for modeling undirected graphs, we use the symmetry constraint $\mathcal{A}^3 := \{A \in \mathbb{R}^{n\times n} : A^\top = A\}$, which is a convex set. These three constraints can be intersected. For example, the intersected local and global budget can be denoted via the superscript, e.g., $\mathcal{A}^{1+3} := \mathcal{A}^1 \cap \mathcal{A}^3$.

Margin and Certificate. Given W and U, let $z_c = \text{pooling}(\sigma(\hat{D}^{-1}\hat{A}XW))U_{:c}$ be the logit for class c. The worst-case margin with attention pooling is defined as

$$\min_{A\in\mathcal{A}} \min_c z_y - z_c \quad = \quad \min_c \min_{A\in\mathcal{A}} F_c(A), \tag{5.2}$$

$$\text{where} \quad F_c(A) := \sum\nolimits_{i=1}^{n} \underbrace{\alpha_i \sigma\left((\hat{A}_{i:}\mathbf{1})^{-1}\hat{A}_{i:}XW\right)(U_{:y} - U_{:c})}_{=:f_{c,i}(A_{i:})}. \tag{5.3}$$

If there is exists an $A \in \mathcal{A}$ and c such that $F_c(A)$ is negative, then the GCN is vulnerable. On the other hand, if the minimal value of the margin is positive, then the trained model must be robust. However, finding the minimum of $F_c(A)$ can be hard due to the discrete constraints.

In practice, if any *lower bound* on the minimum value of $F_c(A)$ is positive, then it serves as a **certificate of robustness**. To lighten notation, we will suppress the subscript c, and just write $F(A)$ and $f_i(A_{i:})$.

GCNs, different from other neural networks, usually do not benefit from more than two hidden layers [21, 22, 127, 128]. Therefore, our certificate will be presented under one hidden layer. For completeness, we also detail the extension to multiple hidden layers in our paper [3].

5.3 Attack Algorithms

To start with, we first look at the upper bound of $F_c(A)$, i.e., an attack algorithm. In Section 5.3.1, we first develop an exact attack algorithm to solve Equation 5.3 under $\mathcal{A}^1$ and $\mathcal{A}^2$. However the exact solutions can't apply when the symmetry constraint $\mathcal{A}^3$ is involved. Thus, we design an approximate attack algorithm based on Alternating direction method of multipliers (ADMM) in section 5.3.2.

5.3.1 Exact Attacks

we notice that $F(A)$ can be decomposed over $A_{i:}$ across nodes i under $\mathcal{A}^1$, allowing each $A_{i:}$ to be solved separately. Using this amenable structure, we show that minimizing $F(A)$ over $\mathcal{A}^1$ and $\mathcal{A}^2$ admits a closed-form solution, which hence provides an exact attacks.

When σ is the linear activation, each $A_{i:}$ in $f_i(A_{i:})$ can be solved by sorting. Recall from Equation 5.3 that $f_i(A_{i:}) = \alpha_i \sigma\left((\hat{A}_{i:}\mathbf{1})^{-1}\hat{A}_{i:}XW\right)(U_{:y} - U_{:c})$. As σ is identity, we can write it as $f_i(A_{i:}) = (\hat{A}_{i:}\mathbf{1})^{-1}\hat{A}_{i:}\pi$ for some vector π. Let v_k encode whether A_{ik} changes upon A_{ik}^{ori} (1 for true and 0 for false). Assume the local budget is δ_l for node i, then we have $\mathbf{1}^\top v = \delta_l$.

Denote $a = (A_{i:}^{ori})^\top$, then $A_{i:}^\top = v + a - 2a \circ v$. The optimization of $f_i(A_{i:})$ can now be written as

$$\min_v f(v) = \frac{\beta^\top v + c_1}{a^\top v + c_2}, \quad s.t. \quad v \in \{0,1\}^n,\ \mathbf{1}^\top v = \delta_l,\ v_i = 0, \tag{5.4}$$

for some vector β and scalar c_1, c_2. Now we only need to enumerate all possible values of $s := a^\top v$ from 0 to $\min(j, \mathbf{1}^\top a)$ in the denominator. Let $I_+ := \{k \in [n] : a_k = 1\}$ and $I_- = [n] \backslash (\{i\} \cup I_+)$. Then it is trivial to optimize the numerator under $a^\top v = s$ by computing

$$\min_{v_k : k \in I_+} \sum_{k \in I_+} \beta_k v_k, \quad \text{s.t.} \quad v_k \in \{0,1\},\ \sum_{k \in I_+} v_k = s, \tag{5.5}$$

$$\min_{v_k : k \in I_-} \sum_{k \in I_-} \beta_k v_k, \quad \text{s.t.} \quad v_k \in \{0,1\},\ \sum_{k \in I_-} v_k = j - s. \tag{5.6}$$

Both can be computed by sorting $\{\beta_k : k \in I_+\}$ and $\{\beta_k : k \in I_-\}$. The sorting results can be precomputed and used for all values of s. As a result, the overall complexity of optimizing $f_i(A_{i:})$ over $\mathcal{P}_i$ is $O(n \log n)$. This strategy applies to $\mathcal{A}^2$ by vectorizing the whole matrix A.

When σ is not linear, $f_i(A_{i:})$ can not be simply reformulated as Equation 5.4. But we can enumerate all the possible selections of δ_l neighbors among the $n-1$ ones, which still admits polynomial complexity $O(\binom{n}{\delta_l})$.

The above exact attack works on $\mathcal{A}^1$, now let us consider $\mathcal{A}^{1+2}$. Notice that attack on each node is a subproblem of attacking on entire graph, this naturally motivates a dynamic programming algorithm. First apply the above technique for $\mathcal{A}^1$ to precompute

$$a_i(j) := \min\{f_i(A_{i:}) : A_{i:} \text{ satisfies } \left\|A_{i:} - A^{ori}_{i:}\right\|_1 = j\}, \quad \forall\ i \in [n],\ j \in [0, \delta_l]. \tag{5.7}$$

Then minimizing $F(A)$ over $\mathcal{A}^{1+2}$ is equivalent to minimizing $\sum_i a_i(k_i)$ subject to $k_i \in [0, \delta_l]$ and $\sum_{i=1}^n k_i \leq 2\delta_g$. To this end, we denote $s_i(j)$ as the lowest possible value of $F(A)$ under $\sum_{i'=1}^i k_{i'} = j$ and $k_{i'} = 0$ for all $i' > i$. Then our dynamic programming updates it from $i = 1$ to n, as shown in Algorithm 3, Clearly the computational cost for the dynamic programming is $O(n\delta_g\delta_l)$, and the storage cost is $O(n\delta_g)$. Besides, computing $\{a_i(j) : i \in [n],\ j \in [0, \delta_l]\}$ costs $O(\delta_l^2 n + n^2 \log n)$ when σ is identity, otherwise $O(\binom{n}{\delta_l} n)$.

Algorithm 3: Dynamic programming for minimizing $F(A)$ under $\mathcal{A}^{1+2}$

Initialize by $s_0(0) = 0$ and $c_0 = 0$. Precompute $a_i(j)$ for all $i \in [n]$ and $j \in [0, \delta_l]$.

for $i = 1, 2, \ldots, n$ **do**

  $c_i \leftarrow \min\{c_{i-1} + \delta_l, 2\delta_g\}$.

  $s_i(j) \leftarrow \min_k\{s_{i-1}(j-k) + a_i(k) : k \in [0, \delta_l], j - k \in [0, c_{i-1}]\}$ for all $j \in [0, c_i]$.

Pick $j_n \leftarrow \arg\min_j s_n(j)$.

for $i = n, n-1, \ldots, 1$ **do**

  $k_i \leftarrow \arg\min_k\{s_{i-1}(j_t - k) + a_i(k) : k \in [0, \delta_l], j_i - k \in [0, c_{i-1}]\}$ and $j_{i-1} \leftarrow j_i - k_i$.

  Recover the optimal $A_{i:}$ based on the argmin in the definition of $a_i(k_i)$ in Equation 5.7.

Return A

Unfortunately, polytime exact solutions are not available to $\mathcal{A}^{1+3}$, $\mathcal{A}^{2+3}$, and $\mathcal{A}^{1+2+3}$ even for linear activation. Instead, we resort to solve them approximately in the next section.

5.3.2 Approximate Attacks

ADMM has been widely utilized in convex and nonconvex optimization to deal with complex constraints or objectives. Noting that Algorithm 3 is efficient for $\mathcal{A}^{1+2}$ and $\mathcal{A}^{1+2+3}$ can be decomposed as $\mathcal{A}^{1+2} \cap \mathcal{A}^3$, We extend ADMM to this discrete setting:

$$\min_{A,B} \; F(A) + \delta(A \in \mathcal{A}^{1+2}) + \delta(B \in \mathcal{A}^3), \quad s.t. \;\; A = B. \tag{5.8}$$

Here $\delta(\cdot) = 0$ if $\cdot$ is true, and ∞ otherwise. Then introduce the augmented Lagrangian with a small positive constant μ and Frobenious norm $\|\cdot\|_F$:

$$\mathcal{L}(A, B, \Lambda) := F(A) + \delta(A \in \mathcal{A}^{1+2}) + \delta(B \in \mathcal{A}^3) - \mathrm{tr}(\Lambda^\top (A - B)) + \tfrac{1}{2\mu} \|A - B\|_F^2 , \tag{5.9}$$

In the end, ADMM loops as follows

$$A_{t+1} \leftarrow \arg\min_A \mathcal{L}(A, B_t, \Lambda_t) = \arg\min_{A \in \mathcal{A}^{1+2}} F(A) - \mathrm{tr}(\Lambda_t^\top A) + \tfrac{1}{2\mu} \|A - B_t\|_F^2 \tag{5.10}$$

$$B_{t+1} \leftarrow \arg\min_B \mathcal{L}(A_{t+1}, B, \Lambda_t) = \arg\min_{B \in \mathcal{A}^3} \left\{ \mathrm{tr}(\Lambda_t^\top B) + \tfrac{1}{2\mu} \|A_{t+1} - B\|_F^2 \right\} \tag{5.11}$$

$$\Lambda_{t+1} \leftarrow \Lambda_t + \tfrac{1}{\mu}(B_{t+1} - A_{t+1}) \tag{5.12}$$

Equation 5.10 is solvable by Algorithm 3 as $x^2 = x$ for $x \in \{0, 1\}$. Equation 5.11 has a closed-form solution: $\frac{1}{2}(A_t + A_t^\top - \mu\Lambda_t - \mu\Lambda_t^\top)$.

Although ADMM in this discrete domain is not guaranteed to find the global optimum, it works well on minimizing $F(A)$ approximately in our experiments, and it trivially provides an upper bound for $F(A)$ under $A \in \mathcal{A}^{1+2+3}$.

5.4 Certifying Robustness by Lagrange Duality

The exact solution in the above section defines a lower bound (i.e., certificate of robustness) for $F(A)$ in the asymmetric constraints while the approximate attack only indicates an upper bound with the additional symmetric constraint. We will next develop a lower bound for $F(A)$ under $\mathcal{A}^{1+3}$, $\mathcal{A}^{2+3}$, and $\mathcal{A}^{1+2+3}$. To this end, we will decompose $\mathcal{A}^{1+2+3}$ into $\mathcal{A}^{1+2} \cap \mathcal{A}^3$, and resort to weak duality:

$$\min_{A \in \mathcal{A}^{1+2+3}} F(A) = \min_{A \in \mathcal{A}^{1+2}} \max_{\Lambda} F(A) + \operatorname{tr}(\Lambda^\top (A - A^\top)) \tag{5.13}$$

$$\geq \max_{\Lambda} \min_{A \in \mathcal{A}^{1+2}} \left\{ F(A) + \operatorname{tr}((\Lambda^\top - \Lambda)A) \right\}. \tag{5.14}$$

Fixing Λ, because $\operatorname{tr}((\Lambda^\top - \Lambda)A)$ is decomposed over the rows of A, the inner optimization can be exactly solved by Algorithm 3 for both linear and nonlinear activations, With the optimal A^*, the supergradient in Λ can be evaluated via Danskin's theorem: $A^* - (A^*)^\top$. This approach generalizes directly to $\mathcal{A}^{1+3}$ and $\mathcal{A}^{2+3}$: just use $\mathcal{A}^1$ and $\mathcal{A}^2$ in Equation 5.14 respectively.

5.5 Certifying Robustness by Convex Envelope

The dualization based certificate relies on the efficient computation of $a_i(j)$ in Equation 5.7, which can be expensive for ReLU. Besides, dualization is not the tightest lower bound. In this section, we show that both of the issues can be resolved by convexification of the problem:

substituting the objective function with its tightest convex minorant, i.e., the convex envelope [148], and approximating the feasible domain $\mathcal{A}$ with the smallest enclosing convex set (convex hull).

Thanks to the amenable structures in $F(A)$ and $\mathcal{A}$, $F(A)$ can be extended from $\mathcal{A}$ to its convex hull $\operatorname{co}\mathcal{A}$ as a convex function $F^{\circ}(Z)$ that matches F on $\mathcal{A}$ (i.e., $F^{\circ}(A) = F(A)$ for $A \in \mathcal{A}$). As a result, the optimization

$$\min F^{\circ}(Z), \quad s.t. \quad Z \in \operatorname{co}\mathcal{A}, \tag{5.15}$$

can serve as a bona fide certificate of robustness.

However, the explicit form of $\operatorname{co}\mathcal{A}$ only exists for $\mathcal{A}^1$ and $\mathcal{A}^2$. Once $\mathcal{A}^1$ and $\mathcal{A}^2$ are intersected with $\mathcal{A}^3$, it is nontrivial to explicitly express the $\operatorname{co}\mathcal{A}$ in terms of linear inequalities. In these cases, projected gradient descent algorithm is also not applicable due to the difficulty of projection to $\operatorname{co}\mathcal{A}$. Alternatively, we adopted the conditional gradient algorithm [149, CG,]. It builds a first-order Tayler approximation of the objective function F° at the current solution Z_t, and then maximizing the *linear function* over $\operatorname{co}(\mathcal{A})$ (polar operator). A key benefit is that the polar operator is equivalent to maximizing a linear function over $\mathcal{A}$. Therefore, CG avoids the projection to $\operatorname{co}(\mathcal{A})$ and the resulting polar operator can be efficiently solved as shown in Section 5.5.1.

1. Initialize Z_0 to any arbitrary element in $\operatorname{co}\mathcal{A}$ (or just in $\mathcal{A}$).
2. For $t = 1, 2, \ldots$
3. **PO**: find $B_t \in \arg\max_{Z \in \operatorname{co}\mathcal{A}} \operatorname{tr}(R^\top Z) = \arg\max_{A \in \mathcal{A}} \operatorname{tr}(R^\top A)$, where $R = -\nabla F^\circ(Z_{t-1})$.
4. Update $Z_t \leftarrow (t+2)^{-1}(2B_t + tZ_{t-1})$.

The final optimal solution to Equation 5.15 can be found in $O(1/\epsilon)$ iterations by CG whenever the PO is exactly computable. Section 5.5.1 will show that the PO in Step 3 can be solved i) exactly over the convex hull of $\mathcal{A}^{1+2}$, $\mathcal{A}^{1+3}$, or $\mathcal{A}^{2+3}$; and ii) with $\frac{1}{2}$-approximation guarantee over the convex hull of $\mathcal{A}^{1+2+3}$. Sections 5.5.1 and 5.5.2 will address the efficient computation of the gradient of $F^\circ(A)$ for linear and ReLU activation, respectively. .

5.5.1 Convexification of $\mathcal{A}$ and its polar operators

In this section, we show that the PO in CG can be efficiently computed when $\mathcal{A}$ consists of the intersection of $\mathcal{A}^1$, $\mathcal{A}^2$, and/or $\mathcal{A}^3$. Let V_{ij} encode whether A_{ij} changes upon A^{ori} (1 for yes and 0 for no). Then $A = \frac{1}{2}[(2A^{ori} - E) \circ (-2V + E) + E]$, where E is a matrix of all ones, and $\circ$ represents the elementwise product. Now the PO problem can be converted to $\arg\max_{V \in \mathcal{A}_\circ} \operatorname{tr}(J^\top V) + \operatorname{tr}(R^\top A^{ori})$, where $\mathcal{A}_\circ$ is the $\mathcal{A}$ with A^{ori} set to the zero matrix, and $J = R - 2R \circ A^{ori}$.

Given J, maximizing $\operatorname{tr}(J^\top V)$ over $V \in \mathcal{A}_\circ^{2+3}$ is exactly the same problem as Equation 5.11. Maximizing over $\mathcal{A}_\circ^{1+3}$ is a maximum weight b-matching problem ($b = \delta_l$) with a fully connected graph, and its exact solution can be found in $O(n^4)$ time [150]. Maximization over $\mathcal{A}_\circ^{1+2}$ can be solved by dynamic programming almost identical to Algorithm 3. Simply redefine $a_t(j)$ therein

Algorithm 4: Greedy algorithm for solving the polar operator under $\mathcal{A}_\circ^{1+2+3}$

Initialize $V = \mathbf{0}$ and set $\hat{J} = (J + J^\top)/2$.

Sort the indices $\mathcal{I} := \{(i,j) : j > i, \hat{J}_{ij} > 0\}$ in a descending order of $\hat{J}_{ij}$.

for $(i,j) \in \mathcal{I}$ *(in the sorted order)* **do**

  Set $\hat{V} = V$, followed by $\hat{V}_{ij} \leftarrow 1$ and $\hat{V}_{ji} \leftarrow 1$. If $\hat{V} \in \mathcal{A}^{1+2+3}$, then $V \leftarrow \hat{V}$.

Return V

by the negative of the sum of the largest j elements in $J_{t:}$ with $a_t(0) = 0$. It costs $O(n\delta_g\delta_l)$ in time, and $O(n\delta_g)$ in space.

Given J, maximizing $\mathrm{tr}(J^\top V)$ over $\mathcal{A}_\circ^{1+2}$ can be solved by dynamic programming almost identical to Algorithm 3. Maximizing over $V \in \mathcal{A}_\circ^{2+3}$ is exactly the same problem as Equation 5.11. Maximizing over $\mathcal{A}_\circ^{1+3}$ is a maximum weight b-matching problem ($b = \delta_l$) with a fully connected graph, which costs $O(n^4)$ in time for the exact solution.

However, the PO for $\mathcal{A}_\circ^{1+2+3}$ is NP-hard. We found a $\frac{1}{2}$-approximate solution by greedily adding edges to V as shown in Algorithm 4. The time complexity is $O(n^2 \log n)$. The analysis of bound and implied certificate is summarized in Theorem 8.

Theorem 8. *(Proof relegated to Appendix D.1) Suppose Algorithm 4 returns V^*, and CG returns a solution Z_t. Then $\mathrm{tr}(J^\top V^*) \geq \frac{1}{2} \max_{V \in \mathcal{A}_\circ^{1+2+3}} \mathrm{tr}(J^\top V)$, and a certificate can be derived as*

$$\min_{A\in\mathcal{A}^{1+2+3}} F(A) \geq F^{\circ}(Z_t) + \operatorname{tr}(R^{\top}Z_t) - 2\operatorname{tr}(J^{\top}V^*) - \operatorname{tr}(R^{\top}A^{ori}). \tag{5.16}$$

The resulting certificate may be not tight due to the $\frac{1}{2}$-approximate of PO. It is thus worthwhile to slightly relax the domain in exchange for the factor of 2. The most natural relaxation is $\mathcal{C} := \operatorname{co}\mathcal{A}^1 \cap \operatorname{co}\mathcal{A}^2 \cap \mathcal{A}^3$, which can be expressed by a set of linear inequalities: $\{Z \in [0,1]^{n\times n} : Z^{\top} = Z, Z_{ii} = 0, \left\|Z - A^{ori}\right\|_1 \leq 2\delta_g, \left\|Z_{i:} - A^{ori}_{i:}\right\|_1 \leq \delta_l\}$ (see Appendix D.2). Hence, PO on $\mathcal{C}$ simply optimizes a linear function over linear constraints.

5.5.2 Convexification of $F(A)$ for linear activation

The convex hull constraint issue has been solved by efficient polar operator, now let us convexify $F(A)$. We first consider the identity activation σ. Although f_i is defined on the discrete domain, we can extend it on a continuous domain as follows:

$$h_i(z) := \begin{cases} f_i(z) & \text{if } z \in \mathcal{P}_i := \{z \in \{0,1\}^n : z_i = 0,\ \left\|z - A^{ori}_{i:}\right\|_1 \leq \delta_l\} \\ \infty & \text{otherwise} \end{cases}. \tag{5.17}$$

So the convex envelope of h_i over $\operatorname{co}\mathcal{P}_i$ can be written as

$$h_i^{**}(z) = \sup_r \left\{ r^{\top}z - \sup_{w\in\mathcal{P}_i} \left\{ r^{\top}w - f_i(w) \right\} \right\} = \max_r \min_{w\in\mathcal{P}_i} r^{\top}z - r^{\top}w + f_i(w). \tag{5.18}$$

It is straightforward to evaluate $h_i^{**}(z)$. Given r, the optimal w can be found efficiently by Algorithm 3. This objective is concave in r, thus solvable by, e.g., bundle method. So the convex envelope of $F(A)$ over $\operatorname{co}\mathcal{A}^1 = \prod_i \operatorname{co}\mathcal{P}_i$ is

$$F_1^\circ(Z) := \sum\nolimits_i h_i^{**}(Z_{i:}), \qquad Z \in \operatorname{co}\mathcal{A}^1. \tag{5.19}$$

Since all points in $\mathcal{P}_i$ must be an extreme point of $\operatorname{co}\mathcal{P}_i$, $F_1^\circ(Z) = F(Z)$ for all $Z \in \mathcal{A}^1$. Although F_1° is derived from $\mathcal{A}^1$, it can be optimized over any subset of $\operatorname{co}\mathcal{A}^1$, e.g., $\operatorname{co}\mathcal{A}^{1+2+3}$.

When intersecting with the global budget, we have an extended function $H(Z) := \sum_i f_i(Z_{i:})$ if $Z \in \mathcal{A}^{1+2}$, and is ∞ otherwise. Then the convex envelope of $F(A)$ over $\operatorname{co}\mathcal{A}^{1+2}$ is

$$F_{1+2}^\circ(Z) := H^{**}(Z) = \sup_R \left\{ \operatorname{tr}(R^\top Z) - \sup_{W \in \mathcal{A}^{1+2}} \left\{ \operatorname{tr}(R^\top W) - \sum\nolimits_i f_i(W_{i:}) \right\} \right\}. \tag{5.20}$$

Clearly, $F_{1+2}^\circ(Z) = F(Z)$ for all $Z \in \mathcal{A}^{1+2}$. **Optimization recipe** Given R, the optimal W can be efficiently found by Algorithm 3. The whole objective in Equation 5.20 is again concave in R, hence solvable by bundle method. By Danskin's theorem, the optimal R is the gradient of $F_{1+2}^\circ(Z)$ in Equation 5.20, which can be fed to the polar operator (PO) for CG.

5.5.3 Convexification of $F(A)$ for ReLU activation

When σ is ReLU, $f_i(A_{i:}) = (\hat{A}_{i:}\mathbf{1})^{-1}[\hat{A}_{i:}XW]_+ v$, where $v := \alpha_i(U_{:y} - U_{:c})$, and $[z]_+ := \max\{0, z\}$ is applied element-wise to a vector. The nonlinearity in ReLU activation makes the

convexification hard. Recall the technique used in [22, 139], the ReLU unit $\hat{x} := [x]_+$ over $x \in [l, u]$ can be approximated via the following intervals:

$$\hat{x} \in [[x]_+, u(x-l)/(u-l)] \;\text{ if }\; l \cdot u < 0, \qquad \hat{x} = x \;\text{ if }\; l \geq 0, \qquad \hat{x} = 0 \;\text{ if }\; u \leq 0. \tag{5.21}$$

Both l and u of the j-th entry of $\hat{A}_{i:}XW$ can be estimated under $\left\|A_{i:} - A^{ori}_{i:}\right\|_1 \leq \delta_l$ (Appendix D.3). Since $f_i(A_{i:})$ is to be minimized, minimizing $v_j\hat{x}_{ij}$ under Equation 5.21 with $x = x_{ij} = \hat{A}_{i:}XW_{:j}$ yields

$$\min_{\hat{x}_{ij} \text{ meets } \textit{Equation } 5.21} v_j\hat{x}_{ij} = \begin{cases} v_j \cdot \sigma^-_{ij}(x_{ij}) := v_j u_{ij}(\hat{A}_{i:}XW_{:j} - l_{ij})/(u_{ij} - l_{ij}) & \text{if } j \in \mathcal{N}_i \\ v_j \cdot \sigma^+_{ij}(x_{ij}) := v_j[\hat{A}_{i:}XW_{:j}]_+ & \text{otherwise} \end{cases}, \tag{5.22}$$

$$\text{where} \quad \mathcal{N}_i := \{j : v_j < 0 \;\wedge\; l_{ij} \cdot u_{ij} < 0\}. \tag{5.23}$$

Letting e_i be the canonical vector for coordinate i, we can lower bound $f_i(A_{i:})$ by $g_i(A_{i:})$, where

$$g_i(z) := (\mathbf{1}^\top z + 1)^{-1}\hat{g}_i(z), \quad z \in \mathcal{P}_i \tag{5.24}$$

$$\text{where} \quad \hat{g}_i(z) := \sum\nolimits_{j \notin \mathcal{N}_i} v_j\sigma^+_{ij}((z+e_i)^\top XW_{:j}) + \sum\nolimits_{j \in \mathcal{N}_i} v_j\sigma^-_{ij}((z+e_i)^\top XW_{:j}). \tag{5.25}$$

σ_{ij}^+ is in fact the original ReLU, which yields the greater of 0 and an affine function. σ_{ij}^- is affine, hence can be handled with ease. So $\hat{g}_i$ is convex. We extend g_i to $\operatorname{co}\mathcal{P}_i$:

$$h_i(z) = g_i(z) = (\mathbf{1}^\top z + 1)^{-1}\hat{g}_i(z) \quad \text{if } z \in \operatorname{co}\mathcal{P}_i, \qquad \text{and } h_i(z) = \infty \quad \text{otherwise.} \tag{5.26}$$

Then the convex envelope of h_i over $\operatorname{co}\mathcal{P}_i$ can be derived as

$$h_i^{**}(z) = \sup_r \left\{ r^\top z - \sup_{w \in \operatorname{co}\mathcal{P}_i} \left\{ r^\top w - (\mathbf{1}^\top w + 1)^{-1}\hat{g}_i(w) \right\} \right\} \tag{5.27}$$

$$= \max_r \min_{\alpha \in [0,n]} \min_{w : w \in \operatorname{co}\mathcal{P}_i,\ \mathbf{1}^\top w = \alpha} \{ r^\top z - r^\top w + (\alpha + 1)^{-1}\hat{g}_i(w) \}. \tag{5.28}$$

Given r and α, the inner-most optimization over w is convex because $\hat{g}_i$ is convex. The optimization over α is one dimensional, hence can be solved by enumerating over an ϵ-grid, which enjoys $O(1/\epsilon)$ complexity. The formula of the derivative in α is relegated to Appendix D.4. Eventually, the r is solvable by bundle method. The final convexified objective is $F_{1+2}^\circ(Z) := \sum_i h_i^{**}(Z_{i:})$.

5.6 Experiments

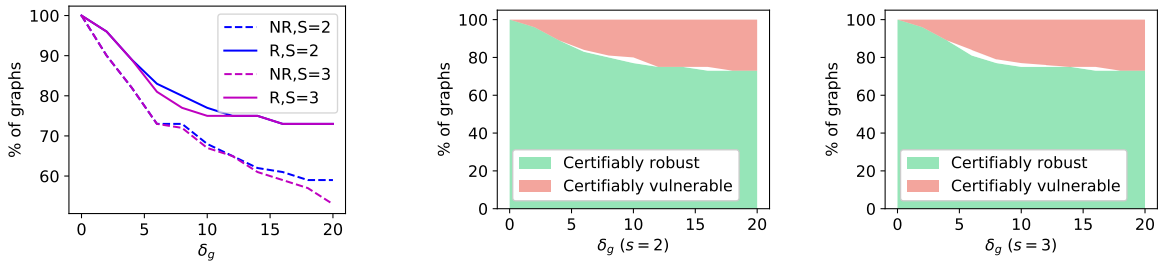
Our experiment aims to evaluate the certificate of robustness for GCNs in graph classification, with an emphasis on its tightness. This is facilitated by the ADMM that provides an upper bound under topological attacks. We will also apply the certificate to the models trained with a robust regularizer. All code and data are available at `https://github.com/RobustGraph/RoboGraph`.

Datasets. We tested on four public datasets that are commonly used in graph classification: Enzymes, NCI1, MUTAG, and PROTEINS [151]. Their properties are summarized in Appendix

D.5. We used 30% of the whole dataset for training, 20% for validation, and 50% for testing. The test set has a higher fraction because our focus is on attacking the model on the test set. Due to space limit, we only report the result on Enzymes, relegating to Appendix D.5 the result of the other two datasets. Enzymes has 600 graphs, $C = 6$ classes, $d = 21$ node features. The median number of node over all graphs is 32, while that of edge is 120.

Training algorithms and settings. We used a single hidden layer and the latent dimensionality was set to 32. Average pooling was also deployed. In [22], the local budget was set to 1% of the total local degree of freedom (i.e., number of node features). This is clearly inapplicable to our setting because most graphs do not even have 100 nodes; indeed, the median #node is 32. Following [52], we varied the local budget for each node v in a graph G by $\delta_l(v) = \max(0, d_v - \max(\{d_v : v \in G\}) + s)$, where d_v is the degree of v in the original graph, $\max(\{d_v : v \in G\})$ is the max of the node degrees in G, and s called the *local attack strength* that is varied from 1, 2, etc. A lower value of s results in a more restrictive budget. The underlying rationale is to endow a larger local budget for higher degree nodes, because they often require more perturbations to make a difference [21, 22]. Empirically we observed that when $s = 3$, about 60% of the nodes in each graph get a positive budget thanks to ties in d_v, and that rate goes to nearly 100% when $s = 4$.

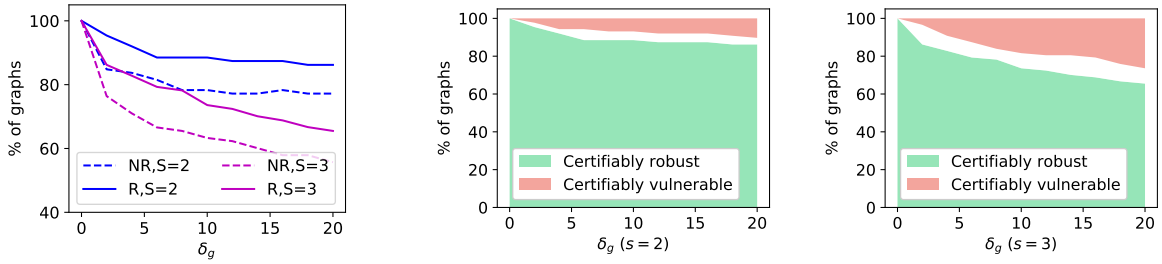
As shown by [22, 52], a hinge loss added to Equation 5.1 can significantly improve the robustness without degrading the accuracy: $\lambda \sum_{c \neq y} \max\{0, 1 + \max_{A \in \mathcal{A}}(z_c(A) - z_y(A))\}$, adapted from node classification to graph classification. So in practice, it is almost always advisable to use it, computation permitting. In our experiment, we set $\lambda = 0.5$, $s^{tr} = 3$, and $\delta_g^{tr} = \infty$,



(a) Fraction of graphs certified robust with $s \in \{2, 3\}$, under robust training (**R**) and non-robust training (**NR**)

(b) Fraction of graphs that are certified as robust (lower green area) and vulnerable (upper red area, percentage $= 100 - y$-axis). Left: $s = 2$, right: $s = 3$ for attack. Both are under robust training.

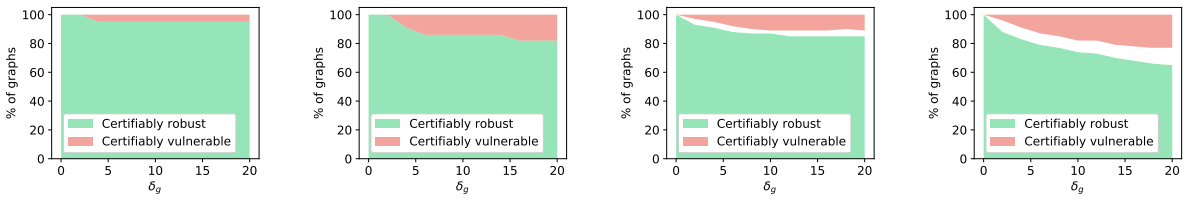
Figure 15: Certificates under **linear** activation on Enzymes. s is the local attack strength at *testing*.



(a) Fraction of graphs certified robust

(b) Fraction of graphs that are certified as robust and vulnerable

Figure 16: Certificates under **ReLU** activation on Enzymes, with other settings identical to Figure 15



(a) Linear ($s = 2$)

(b) Linear ($s = 3$)

(c) ReLU ($s = 2$)

(d) ReLU ($s = 3$)

Figure 17: Certificate for Enzymes with 80% for training.

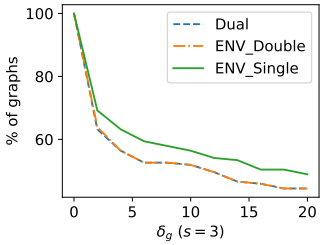


Figure 18: Comparison of convex certificates on ReLU

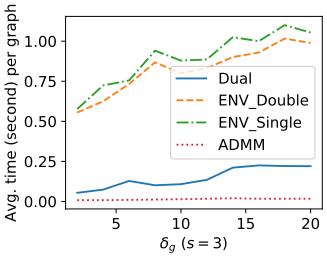


Figure 19: Computational time for certificates/attacks on ReLU

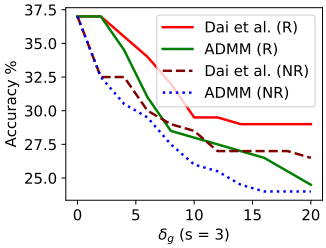


Figure 20: Test accuracy under various attacks

because they produced the highest robustness while not hurting the test accuracy. Note that the value of s and δ_g for test data can be different from s^{tr} and δ_g^{tr} (i.e., different $\mathcal{A}$). Our work claims no novelty in robust training, and it was leveraged to demonstrate the effectiveness of our certificate in a more realistic setting.

Performance of the certificates. Figure 15a demonstrates the percentage of graphs that are certified robust by the dualization method, where the activation function is linear. Figure 16a shows a similar result, but for ReLU activation using the convex envelope approach from Section 5.5.3, optimized over $\operatorname{co}\mathcal{A}^1 \cap \operatorname{co}\mathcal{A}^2 \cap \mathcal{A}^3$. The global budget δ_g is varied from 1 to 20. It is clear that robust training significantly increases the fraction of graphs that are certifiably robust. Naturally, $s = 2$ allows more graphs to be certified robust. Since node features are not subject to attack, the certified percentage does not drop to 0 even for large δ_g, akin to Figure 4a of [52].

Tightness of the certificates. To better examine the tightness of our certificates, Figure 15b plots the percentage of graphs that are certified as robust (lower green area, same as the

'R' curves Figure 15a) and vulnerable (by ADMM, upper red area). It is for linear activation under robust training, and that for ReLU is in Figure 16b. Clearly the gap is almost zero for linear activation, while that for ReLU is also quite small, below 20% which is similar to [22]. So the gap stems from the approximation in ReLU, and any improvement in this respect can benefit a number of certificate algorithms like ours.

In addition, we compared three convex certificates for ReLU: ENV_Single (used above) is the convex envelop based on the single linear approximation (for $\mathcal{N}_i$) as in Equation 5.22. We also adopted the double linear approximation (for $\bar{\mathcal{N}}_i$) from [140], which enables both dualization (Dual) and convex envelope (ENV_Double) methods. Figure 18 shows ENV_Single yields clearly superior (higher) certificate.

Performance on larger training sets. We additionally experimented on Enzyme with 80%, 10%, 10% for training, validation and testing, respectively. The small size of test data led to marked variations in the gap plot, and it is unclear how to "average" them. So we plotted a typical result in Figure 17, showing the fraction of certifiably robust/vulnerable for both linear and ReLU activations. Compared with Figure 15b and 16b where 30% graphs were used for training, the tightness here appears similar, or slightly better under the linear activation.

Computational efficiency. The two convex envelope based methods are more costly (Figure 19). Since single linear approximation keeps the nonsmooth $[\cdot]_+$ function in Equation 5.22 for $\bar{\mathcal{N}}_i$, it is slightly more expensive than double linear approximation.

To test the scalability to graphs with a larger number of nodes and edges, we examined another dataset DD [152], where the median number of node and edge per graph is 284 and 716, respectively. We set $\delta_g = 20$ and $s = 3$. For the certificates with linear activation, Enzyme

(median #node = 32, median #edge = 120) takes 0.37 seconds per graph on average, while DD takes 7.3 seconds. For the certificates with ReLU activation, Enzyme takes 1 second per graph on average, while DD takes 28 seconds. This is consistent with the fact that the computational cost depends quadratically on the number of nodes (in practice, a bit lower than that due to implementation details).

Different datasets have different number of classes. To facilitate comparison, the reported time cost is for each class. The number of edges in the original graph does not affect the computational cost much, because the attacker can both delete **and** add edges.

Comparison with other structural attacks. While not being our key focus, we also compared ADMM with Dai et al. [20], which provides an effective topological attack to a flexible range of applications including graph classification. [21,22,25,52] are not applicable to structural attacks on GCNs for graph classification. As can be seen from Figure 20, robust training (R) mitigates the drop of accuracy compared with non-robust training (NR), and ADMM finds more effective attacks under both robust and non-robust training.

CHAPTER 6

CONCLUSIONS AND FUTURE DIRECTIONS

6.1 Conclusions

This thesis investigates algorithms and robustness certificates under adversarial learning framework. First of all, we showed that adversarial prediction under multivariate losses (F-score and DCG) can be solved much faster than they used to be. We first reduced the problem size exponentially by using appropriate sufficient statistics. To solve the resulting convex-concave saddle point problems, we proposed an efficient Breg-SVRG algorithm that is compatible with any Bregman divergence and proved its linear convergence rate. This new algorithm benefits many applications including multivariate prediction under F-score and DCG measure as well as the LPboost problems. Our experiments showed the superiority of Breg-SVRG algorithm over the state-of-the-art solvers.

In addition, we studied the distributionally robust optimization with Wasserstein ambiguity sets, which gives asymptotic consistency solutions. We leveraged the McShane-Whitney extension theorem to upper bound DRRs by the standard empirical risk regularized with the Lipschitz constant of the model. Besides, we demonstrated that DRRs upperbounds the adversarial training risk, which thereby provides a theoretical robustness certificate for popular neural network models. Although explicitly enforcing the Lipschitz constant of deep neural networks is generally hard, we found an algorithm that can tightly enforce the Lipschitz con-

stant of functions in the RKHS that contains certain neural networks. We proved the algorithm enjoys polynomial sample complexity for product kernels. Then we experimentally validated efficiency of the algorithm in enforcing Lipschitz constant. Besides, we verified the robustness of proposed kernel machines through extensive experiments compared with the state-of-the-art Parseval networks.

Last but not least, we investigated the robustness certificate of graph convolution networks for graph classification under topological attacks. We first developed an exact and efficient attack algorithm for local budgets and global budgets. When the symmetric constraint is involved, we used ADMM to obtain an upper bound. To certify the robustness of GCNs, we proposed our first certificate is based on Lagrange duality. However this certificate relies on the efficient attack algorithm which is expensive for ReLU activation. Besides, dualization is not the tightest lower bound. To overcome the obstacles, we convexificated the discrete problem and applied conditional gradient algorithm to solve the resulting optimization. Our experiments confirm the tightness of the certificates. Using the certificates in conjunction with robust training, it significantly improved the robustness of GCNs under different datasets.

6.2 Future Directions

We plan to pursue the following directions by leveraging the proposed methods in this thesis.

Multivariate prediction: Our approach to reducing the problem size for DCG measure in Section 2.2 only applies to simple cases where the relevancy score of returned item is either 0 (irrelevant) or 1 (relevant). In practice, the returned items in a query may have many levels of relevancy. Thus we are interested in reducing the problem size of DCG with multiple relevancy

scores, e.g., $\{0, 1, 2, 3, 4\}$ where 4 indicates the highest relevancy and 0 indicates the lowest relevancy. To achieve this, we may need more statistics to fully express the problem structure. As a result, the resulting problems may have more variables than n^2. As long as the problem size is polynomial, it is still amenable to our Breg-SVRG algorithm.

Optimization algorithms: In Section 2.3, we established a linear convergence rate for the Breg-SVRG algorithm on saddle-point optimization. In addition to SVRG, other algorithms such as SAGA [153], SAG [154] and MISO [155] also enjoy linear convergence rates by exploiting the finite-sum structure of the optimization problem. Therefore, it will be natural to extend Bregman divergence to those algorithms. Hence, solving problems with specific underlying geometries will benefit from non-Euclidean norms and Bregman divergences as we saw in Section 2.5.

Sample complexity: So far we only proved the polynomial sample complexity of the Lipschitz constant enforcing algorithm for product kernels. Our experiments showed that the Lipschitz constant can also be efficiently enforced for RKHS with inverse kernel. In Appendix C.3, we have analyzed the eigenvalue of inverse kernel which indicate the quasi-polynomial sample complexity. We will further analyze the uniform boundedness of its eigenfunctions.

On the other hand, the empirical convergence results in Figure 10 show that the greedy approach significantly reduces the number of samples. Investigating the reason behind this phenomena is an interesting future work. And this may trigger more ideas on the greedy sampling methods.

Experiments on robustness: Our distributionally robust kernel machines are motivated by [47], who showed that the RKHS with stacked inverse kernel subsumes certain neural network function space. Although our experiments show the superiority of kernel machines than Parseval networks, the robustness performance is still worse than the adversarial training [41]. One conjecture is that our upper bound is too loose, which we can examine empirically as mentioned above. Also noticing that [47] put many constraints on the norm of weight matrices in neural networks, which has already shrunk the neural network space. Therefore, another reasonable guess is that the expressive power of RKHS is still limited. It will be interesting to explore how the expressiveness of RKHS with different kernels related to the expressiveness of neural networks, and how the expressiveness affects the robustness of kernel machines.

Consideration of κ: The Lipschitz regularized ERMs in Chapter 3 are obtained by assuming κ in Equation 3.12 equals ∞, which implicitly assumes that every training example must have an exact label. This "hard label" assumption is quite strong in real applications because a label usually reflects the confidence level of classifying an object. [156] introduced a defensive distillation network that is trained with soft labels, i.e., a probability distribution over all classes given by a softmax function. It is straightforward to the see the benefit of using class probabilities instead of hard labels because the probabilities not only provide provide the correct class, but also encode additional information about each class. Although their defensive distillation networks can be fooled by strong attackers such as PGD and C&W attack [87], the idea of soft labels is still valuable. Indeed, the κ in our model plays the role of controlling the softness of labels. [37] showed that using a reasonable κ chosen from cross validation increases

robustness for linear models. In this light, we believe that tuning the κ will improve the robustness of our distributionally robust kernel machines. Investigating how different value of κ affects the robustness of model and how to tune it adaptively will be an interesting topic.

Adversarial training for kernel machines: Instead of minimizing the Lipschitz regularized empirical risk, we may directly apply the adversarial example training mechanism in [41] on kernel machines. The adversary will first find strong attacks by PGD method and add them to the training set, and then we minimize the adversarial training risk where the model is kernel machines. This procedure could be efficiently done by using Nyström approximation. We would not expect the adversarially trained kernel models to perform better than the adversarially trained neural networks. However, the difference between these two models will be interesting.

MMD ambiguity sets: MMD is well-known for its efficient estimation via the kernel mean embedding and fast convergence properties. [79] showed that

$$\sup_{\mathbb{Q}:\mathrm{MMD}(\mathbb{Q},\mathbb{P})\leq\rho} \mathbb{E}_{\mathbb{Q}}\ell_f(\mathbf{z}) \leq \mathbb{E}_{\mathbf{z}\sim\mathbb{P}}\ell_f(\mathbf{z}) + \rho\,\|\ell_f\|_{\mathcal{H}}$$

by assuming $\ell_f \in \mathcal{H}$. This result is similar to our Theorem 3 built on Wasserstein distance. The key difference is that their regularization term is the RKHS norm of ℓ_f while ours penalizes

the Lipschitz constant of ℓ_f. In fact, the RKHS norm of ℓ_f is closely related to the Lipschitz constant of ℓ_f:

$$\begin{aligned} Lip(\ell_f) &= \sup_{\mathbf{z}\neq\mathbf{z}'} \frac{|\ell_f(\mathbf{z}) - \ell_f(\mathbf{z}')|}{d_{\mathbf{z}}(\mathbf{z},\mathbf{z}')} \\ &\leq \sup_{\mathbf{z}\neq\mathbf{z}'} \frac{\|\ell_f\|_{\mathcal{H}} \|k(\mathbf{z},\cdot) - k(\mathbf{z}',\cdot)\|_{\mathcal{H}}}{d_{\mathbf{z}}(\mathbf{z},\mathbf{z}')} \\ &\leq \|\ell_f\|_{\mathcal{H}} \sup_{\mathbf{z}\neq\mathbf{z}'} \frac{\|k(\mathbf{z},\cdot) - k(\mathbf{z}',\cdot)\|_{\mathcal{H}}}{d_{\mathbf{z}}(\mathbf{z},\mathbf{z}')}. \end{aligned}$$

If the feature mapping $k(\mathbf{z},\cdot)$ is Lipschitz continuous measured by $d_{\mathbf{z}}$, then the RKHS norm of ℓ_f upperbounds the Lipschitz constant of ℓ_f by a factor of constant. To compute $\|\ell_f\|_{\mathcal{H}}$, we can use the Nyström approximation as in the Section 4.2. However, the sampling strategy is unclear yet because the sampling space is $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$. Designing the kernel on space $\mathcal{Z}$ will play a key role in simplifying the problem and developing efficient algorithms.

Furthermore, figuring out how MMD differs from Wasserstein distance is also a trending topic in the research of generative adversarial network [157–160].

APPENDICES

Copyright Policy of Neural Information Processing Systems

The Neural Information Processing Systems conference's acronym changed from NIPS to NeurIPS in 2018. All NIPS/NeurIPS authors retain copyright of their work. You will need to sign a nonexclusive license giving the NIPS/NeurIPS foundation permission to publish the work. Ultimately, however, you can do whatever you like with the content, including having the paper as a chapter of your thesis.

Copyright Policy of the International Conference on Machine Learning

The Proceedings of Machine Learning Research is a series that publishes machine learning research papers presented at workshops and conferences. Each volume is separately titled and associated with a particular workshop or conference and will be published online on the PMLR web site. Authors retain copyright.

Appendix A

ANALYSIS FOR BREG-SVRG ALGORITHM

A.1 Proof of Lemma 1

The following result that sanwiches the Bregman divergence induced by an L-smooth convex function is well-known:

Lemma 5. *If f is convex, and L-smooth w.r.t. $\|\cdot\|$, then*

$$\tfrac{1}{2L}\left\|\nabla f(x')-\nabla f(x)\right\|_*^2 \le \Delta_f(x',x) \le \tfrac{L}{2}\left\|x'-x\right\|^2. \tag{A.1}$$

Proof. The second inequality is obvious. To show the first inequality, define

$$g(x') := \Delta_f(x',x) = f(x') - f(x) - \left\langle \nabla f(x), x'-x\right\rangle. \tag{A.2}$$

Then g is minimized at x with $g(x)=0$, and g is also L-smooth. Therefore for any $\bar{x}$

$$g(\bar{x}) \le g(x') + \left\langle \nabla g(x'), \bar{x}-x'\right\rangle + \frac{L}{2}\left\|\bar{x}-x'\right\|^2 \le g(x') + \left\|\nabla g(x')\right\|_* \left\|\bar{x}-x'\right\| + \frac{L}{2}\left\|\bar{x}-x'\right\|^2. \tag{A.3}$$

Now take minimization over $\bar{x}$ on both sides:

$$0 = g(x) \le g(x') - \frac{1}{2L}\left\|\nabla g(x')\right\|_*^2. \tag{A.4}$$

Appendix A (Continued)

Plugging in the definition of g and noticing $\nabla g(x') = \nabla f(x') - \nabla f(x)$, we get the first inequality.

□

The following result is crucial for our later analysis, and extends a result of [65].

Lemma 1. *Let f and g be ϕ-saddle and φ-saddle respectively, with one of them being differentiable. Then, for any $z = (x, y)$ and any saddle point (if exists)*

$$z^* := (x^*, y^*) \in \arg\min_x \max_y \{f(z) + g(z)\},$$

we have $f(x, y^*) + g(x, y^*) \geq f(x^*, y) + g(x^*, y) + \Delta_{\phi+\varphi}(z, z^*)$.

Proof. We first recall the following slight generalization of a result of [65]:

Claim. Let h and k be respectively ψ_1- and ψ_2-convex, with one of them being differentiable.

Let $x^* \in \arg\min_x h(x) + k(x)$, then for all x, $h(x) + k(x) \geq h(x^*) + k(x^*) + \Delta_{\psi_1+\psi_2}(x, x^*)$.

Indeed, using the optimality of x^*, we have $\mathbf{0} \in \partial(h + k)(x^*) = \partial h(x^*) + \partial k(x^*)$, where the last equality is due to the differentiable assumption (on one of h and k). Since h is ψ_1-convex and k is ψ_2-convex, we have

$$h(x) \geq h(x^*) + \langle x - x^*, \partial h(x^*)\rangle + \Delta_{\psi_1}(x, x^*) \tag{A.5}$$

$$k(x) \geq k(x^*) + \langle x - x^*, \partial k(x^*)\rangle + \Delta_{\psi_2}(x, x^*). \tag{A.6}$$

Adding the above two inequalities and noting that $\Delta_{\psi_1} + \Delta_{\psi_2} = \Delta_{\psi_1+\psi_2}$ completes the proof of our claim.

Appendix A (Continued)

Now to prove Lemma 1, we note that if (x^*, y^*) is a saddle point of $f + g$, then $x^* \in \arg\min_x f(x, y^*) + g(x, y^*)$ and also $y^* \in \arg\min_y -f(x^*, y) - g(x^*, y)$. Note also that if f is ϕ-saddle, then $f_y(x) = f(x, y)$ is ϕ_y-convex. Similarly, if g is φ-saddle, then $-g_x(y) = -g(x, y)$ is $(-\varphi_x)$-convex. Applying the above claim twice we have:

$$f_{y^*}(x) + g_{y^*}(x) \geq f_{y^*}(x^*) + g_{y^*}(x^*) + \Delta_{\phi_{y^*}+\varphi_{y^*}}(x, x^*) \tag{A.7}$$

$$-f_{x^*}(y) - g_{x^*}(y) \geq -f_{x^*}(y^*) - g_{x^*}(y^*) + \Delta_{-\phi_{x^*}-\varphi_{x^*}}(y, y^*). \tag{A.8}$$

Adding the above two equations and noting that $\Delta_{\phi_{y^*}+\varphi_{y^*}}(x, x^*)+\Delta_{-\phi_{x^*}-\varphi_{x^*}}(y, y^*) = \Delta_{\phi+\varphi}(z, z^*)$ completes our proof. □

A.2 Bregman Divergence for Convex SVRG

Prior to saddle-point optimization, it is illustrative to see how variance reduction methods can be extended to Bregman divergence in convex optimization. Let us consider a proximal objective

$$J(x) = P(x) + \Omega(x) = \tfrac{1}{n}\textstyle\sum_{k=1}^{n} \psi_k(x) + \Omega(x).$$

Appendix A (Continued)

Here each ψ_k is convex and L-smooth (w.r.t. some norm), and Ω is Δ-convex for some Bregman divergence Δ. Breg-SVRG extends the vanilla SVRG by employing the following proximal operator [161] which we assume is efficiently computable:

$$x_{t+1} = \arg\min_x \left\{ \eta \langle v_t, x \rangle + \eta\Omega(x) + \Delta(x, x_t) \right\}, \text{ where } v_t = \nabla\psi_\xi(x_t) - \nabla\psi_\xi(\tilde{x}) + \tilde{\mu}. \quad \text{(A.9)}$$

Here ξ is sampled uniformly at random from $\{1, \dots, n\}$, $\tilde{x}$ is the pivot found after completing the last epoch, and $\tilde{\mu} = \nabla P(\tilde{x})$. The whole procedure, which we call Breg-SVRG, is detailed in Algorithm 5. To ease notation, the x_t here always refers to the t-th step of the current epoch s, and we will include the epoch index s only when necessary.

Let us define the gap $\epsilon(x) := J(x) - J(x_*)$ for some x_* that minimizes J. Our first convergence result for Algorithm 5 is as follows:

Theorem 9. *Assume each ψ_k is convex and L-smooth wrt $\|\cdot\|$, and P and Ω are $(\gamma\Delta)$- and $(\lambda\Delta)$-convex wrt some Bregman divergence Δ, respectively. Let η be sufficiently small such that $m := \left\lceil \log(\frac{1}{8\eta L} - \frac{1}{8} - \rho) / \log\rho \right\rceil \geq 1$. Then Breg-SVRG enjoys linear convergence in expectation:*

$$\mathbb{E}\epsilon(\tilde{x}^s) \leq \rho^{-ms}[\Delta(x_*, x_0) + c(Z+1)\epsilon(x_0)],$$

where $\rho = \frac{1+\eta\lambda}{1-\eta\gamma}, c = \frac{8\eta^2 L}{(1-\eta L)(1-\eta\gamma)}$ and $Z = \sum_{t=0}^{m-1} \rho^t$.

For example, we may set $\eta = \frac{1}{18L}$, which leads to $c = \frac{4}{153L}$, $m = \frac{\log\left(\frac{9}{8} - \frac{\lambda}{18L}\right)}{\log\left(1 + \frac{\lambda}{18L}\right)} = \Theta\left(\frac{L}{\lambda}\right)$, $(1+\eta\lambda)^m = \frac{9}{8} - \frac{\lambda}{18L} \geq \frac{9}{8} - \frac{1}{18} = \frac{77}{72}$, and $Z = \frac{9L}{4\lambda} - 1$. Therefore, between epochs, the gap decays by a factor of $\frac{72}{77}$, and each epoch needs to call Equation A.9 for $\Theta(L/\lambda)$ times. In total, to

Appendix A (Continued)

Algorithm 5: SVRG with Bregman Divergence

Initialize x_0 randomly. Set $\tilde{x} = x_0$.

for $s = 1, 2, \ldots$ **do**

$\tilde{\mu} \leftarrow \tilde{\mu}^s := \nabla P(\tilde{x})$, $x_0 \leftarrow x_0^s := x_m^{s-1}$

for $t = 1, \ldots, m$ **do**

Randomly pick $\xi \in \{1, \ldots, n\}$.

Compute v_t using Equation A.9.

Update x_{t+1} using Equation A.9.

Denote $x_m^s = x_m$.

$\tilde{x} \leftarrow \tilde{x}^s := \frac{\sum_{t=1}^m (1+\eta\lambda)^t x_t}{\sum_{t=1}^m (1+\eta\lambda)^t}$.

reduce the gap below some tolerance ϵ, the proximal operator Equation A.9 needs to be called for $O\left(\frac{L}{\lambda} \log \frac{1}{\epsilon}\right)$ times. If the norm $\|\cdot\|$ is chosen to be Euclidean, then the above guarantee reduces to that of SVRG [67]. The condition number L/λ, however, can change significantly w.r.t. the chosen norm (which reflects the underlying problem geometry).

We need the following variance reduction lemma that extends a result in [67] to any norm.

Lemma 6. *The variance of v_t can be bounded by:* $\mathbb{E}_\xi \|v_t - \nabla P(x_t)\|_*^2 \leq 16L \cdot [\epsilon(x_t) + \epsilon(\tilde{x})]$.

Proof. Clearly, for any norm, $\|a+b\|^2 \leq 2(\|a\|^2 + \|b\|^2)$. Besides, for any random variable X and norm $\|\cdot\|$, $\mathbb{E}\|X - \mathbb{E}[X]\|^2 \leq 2\mathbb{E}[\|X\|^2 + \|\mathbb{E}X\|^2] \leq 4\mathbb{E}\|X\|^2$. It bounds the "variance" of a random variable, under any norm, by four times its "second moment."

Appendix A (Continued)

Using these two inequalities and conditional on x_t, we have

$$\begin{aligned}
&\mathbb{E}_\xi \left\| v_t - \nabla P(x_t) \right\|_*^2 = \mathbb{E}_\xi \| (\nabla \psi_\xi(x_t) - \nabla \psi_\xi(\tilde{x})) - (\nabla P(x_t) - \nabla P(\tilde{x})) \|_*^2 \\
&\leq 4 \cdot \mathbb{E}_\xi \left\| \nabla \psi_\xi(x_t) - \nabla \psi_\xi(\tilde{x}) \right\|_*^2 = 4 \mathbb{E}_\xi \| \nabla \psi_\xi(x_t) - \nabla \psi_\xi(x_*) - (\nabla \psi_\xi(\tilde{x}) - \nabla \psi_\xi(x_*)) \|_*^2 \\
&\leq 8 \cdot \mathbb{E}_\xi \left\| \nabla \psi_\xi(x_t) - \nabla \psi_\xi(x_*) \right\|_*^2 + 8 \cdot \mathbb{E}_\xi \left\| \nabla \psi_\xi(\tilde{x}) - \nabla \psi_\xi(x_*) \right\|_*^2 . \qquad \text{(A.10)}
\end{aligned}$$

We can next invoke Lemma 5 to upper bound the first part of Equation A.10:

$$\tfrac{1}{2L} \mathbb{E}_\xi \left\| \nabla \psi_\xi(x_t) - \nabla \psi_\xi(x_*) \right\|_*^2 \leq \mathbb{E} \Delta_{\psi_\xi}(x_t, x_*) = \Delta_P(x_t, x_*) \leq \epsilon(x_t),$$

where the last inequality is due to (a special case of) Lemma 1. The second part of Equation A.10 can be bounded similarly. □

Proof of Theorem 9. We apply Lemma 1 to the update Equation A.9, with $g = \eta\Omega(x) + \Delta(x, x_t)$, $\phi = 0$, and $\varphi = \lambda\Delta$:

$$\begin{aligned}
&\eta \langle v_t, x_{t+1} \rangle + \eta\Omega(x_{t+1}) + \Delta(x_{t+1}, x_t) && \text{(A.11)} \\
&\leq \eta \langle v_t, x^* \rangle + \eta\Omega(x^*) + \Delta(x^*, x_t) - \eta\Delta_\Omega(x^*, x_{t+1}) - \Delta(x^*, x_{t+1}) && \text{(A.12)} \\
&\leq \eta \langle v_t, x^* \rangle + \eta\Omega(x^*) + \Delta(x^*, x_t) - \eta\lambda\Delta(x^*, x_{t+1}) - \Delta(x^*, x_{t+1}). && \text{(A.13)}
\end{aligned}$$

Appendix A (Continued)

Therefore

$$\eta\Omega(x_{t+1}) + (1+\eta\lambda)\Delta(x^*, x_{t+1}) \tag{A.14}$$

$$\leq \Delta(x^*, x_t) + \eta\left\langle v_t, x^* - x_{t+1}\right\rangle + \eta\Omega(x^*) - \Delta(x_{t+1}, x_t) \tag{A.15}$$

$$\leq \Delta(x^*, x_t) + \eta\left\langle v_t, x^* - x_{t+1}\right\rangle + \eta\Omega(x^*) - \frac{1}{2}\|x_{t+1} - x_t\|^2, \tag{A.16}$$

where the last inequality is because Δ is distance enforcing w.r.t. the norm $\|\cdot\|$. Since J is L-smooth, we obtain

$$0 \leq P(x_t) - P(x_{t+1}) + \langle \nabla P(x_t), x_{t+1} - x_t\rangle + \frac{L}{2}\left\|x_{t+1} - x_t\right\|^2. \tag{A.17}$$

Multiplying the above by $\eta > 0$ and adding to Equation A.16 we get

$$\eta\Omega(x_{t+1}) + (1+\eta\lambda)\Delta(x^*, x_{t+1}) \tag{A.18}$$

$$\leq \Delta(x^*, x_t) + \eta\left\langle v_t, x^* - x_{t+1}\right\rangle + \eta\Omega(x^*) - \frac{1-\eta L}{2}\left\|x_{t+1} - x_t\right\|^2 \tag{A.19}$$

$$+ \eta P(x_t) - \eta P(x_{t+1}) + \eta\left\langle \nabla P(x_t), x_{t+1} - x_t\right\rangle \tag{A.20}$$

$$= \Delta(x^*, x_t) + \eta\left\langle v_t - \nabla P(x_t), x_t - x_{t+1}\right\rangle - \frac{1-\eta L}{2}\left\|x_{t+1} - x_t\right\|^2 \tag{A.21}$$

$$+ \eta P(x_t) - \eta P(x_{t+1}) + \eta\Omega(x^*) + \eta\left\langle v_t, x^* - x_t\right\rangle \tag{A.22}$$

$$\leq \Delta(x^*, x_t) + \frac{\eta^2}{2(1-\eta L)}\left\|v_t - \nabla P(x_t)\right\|_*^2 \tag{A.23}$$

$$+ \eta[\langle v_t, x^* - x_t\rangle + P(x_t) - P(x_{t+1}) + \Omega(x^*)]. \tag{A.24}$$

Appendix A (Continued)

Conditional on x_t we take expectation over ξ on both sides:

$$(1+\eta\lambda)\mathbb{E}\Delta(x^*, x_{t+1}) \leq (1-\eta\gamma)\Delta(x^*, x_t) + \frac{\eta^2}{2(1-\eta L)}\mathbb{E}\left\|v_t - \nabla P(x_t)\right\|_*^2 \tag{A.25}$$

$$+ \eta[J(x^*) - \mathbb{E}J(x_{t+1})], \tag{A.26}$$

where we have also used the assumption that P is $(\gamma\Delta)$-convex. Using Lemma 6 we take expectation over x_t again on both sides, leading to

$$\rho\Delta_{t+1} \leq \Delta_t + c\left(\delta_t + \tilde{\delta}^{s-1}\right) - \kappa\delta_{t+1}. \tag{A.27}$$

where $\rho := \frac{1+\eta\lambda}{1-\eta\gamma}$, $c := \frac{8\eta^2 L}{(1-\eta L)(1-\eta\gamma)}$, $\kappa := \frac{\eta}{1-\eta\gamma}$, $\delta_t := \mathbb{E}\epsilon(x_t)$, $\Delta_t := \mathbb{E}\Delta(x^*, x_t)$, $\tilde{\delta}^{s-1} := \mathbb{E}\epsilon(\tilde{x}^{s-1})$. Multiplying both sides by ρ^t and telescoping from $t = 0$ to $m-1$, we obtain

$$\rho^m\Delta_m \leq \Delta_0 + c\sum_{t=1}^{m}\rho^{t-1}\delta_{t-1} + c\tilde{\delta}^{s-1}\sum_{t=1}^{m}\rho^{t-1} - \kappa\sum_{t=1}^{m}\rho^{t-1}\delta_t. \tag{A.28}$$

Rearranging, we get

$$\rho^m\Delta_m + c\rho^m\delta_m + (\kappa - c\rho)\sum_{t=1}^{m}\rho^{t-1}\delta_t \leq \Delta_0 + c\delta_0 + c\tilde{\delta}^{s-1}\sum_{t=1}^{m}\rho^{t-1}. \tag{A.29}$$

Now define the representer of epoch s as

$$\tilde{x}^s = \frac{1}{Z}\sum_{t=1}^{m}\rho^{t-1}x_t, \quad \text{where} \quad Z = \sum_{t=1}^{m}\rho^{t-1}. \tag{A.30}$$

Appendix A (Continued)

Note that $\rho > 1$ hence the most recent iterate gets a bigger weight. Also, we can equivalently use $\tilde{x}^s = \frac{1}{Z'}\sum_{t=1}^{m} \rho^t x_t$ where $Z' = \sum_{t=1}^{m} \rho^t$, see Algorithm 5. Then, noting that J is convex and $\tilde{\delta}^s = \mathbb{E}[J(\tilde{x}^s) - J(x^*)]$, we obtain

$$\rho^m(\Delta_m + c\delta_m) + (\kappa - c\rho)Z\tilde{\delta}^s \leq \rho^m(\Delta_m + c\delta_m) + (\kappa - c\rho)\sum_{t=1}^{m} \rho^{t-1}\delta_t \tag{A.31}$$

$$\leq (\Delta_0 + c\delta_0) + cZ\tilde{\delta}^{s-1}. \tag{A.32}$$

Now pick m such that

$$\rho^m = \frac{(\kappa - c\rho)Z}{cZ} = \frac{\kappa - c\rho}{c} = \frac{1 - \eta L}{8\eta L} - \frac{1 + \eta\lambda}{1 - \eta\gamma} = \frac{1}{8\eta L} - \frac{1}{8} - \frac{1 + \eta\lambda}{1 - \eta\gamma}. \tag{A.33}$$

Therefore,

$$\mathbb{E}\Delta(x^*, x_m^s) + c(\mathbb{E}J(x_m^s) - J(x^*)) + cZ(J(\tilde{x}^s) - J(x^*)) \tag{A.34}$$

$$\leq \rho^{-m}[\mathbb{E}\Delta(x^*, x_m^{s-1}) + c(\mathbb{E}J(x_m^{s-1}) - J(x^*)) + cZ(J(\tilde{x}^{s-1}) - J(x^*))]. \tag{A.35}$$

So there is a decay of factor ρ^{-m} between epochs. Set $\eta = \frac{\alpha}{L}$ and we obtain

$$\rho^m = \frac{1}{8\alpha} - \frac{1}{8} - \frac{L + \alpha\lambda}{L - \alpha\gamma} > 1 \tag{A.36}$$

Appendix A (Continued)

for α sufficiently small. Moreover, $\rho = \frac{L+\alpha\lambda}{L-\alpha\gamma}$ hence

$$m = \frac{\log\left(\frac{1}{8\alpha} - \frac{1}{8} - \frac{L+\alpha\lambda}{L-\alpha\gamma}\right)}{\log\frac{L+\alpha\lambda}{L-\alpha\gamma}} = \Theta\left(\frac{1}{\alpha}\log\frac{1}{\alpha}\cdot\frac{L}{\lambda+\gamma}\right). \tag{A.37}$$

So between epochs, we decrease the gap by a constant factor that is strictly smaller than 1, and the number of iterations per epoch is $\Theta\left(\frac{L}{\lambda+\gamma}\right)$. In total, to find an ϵ accurate solution, the computational cost is $\Theta\left(\frac{L}{\lambda+\gamma}\log\frac{1}{\epsilon}\right)$. □

A.3 Rates for Proximal Saddle-Point Optimization in Section 2.3

The proof of [63] relies on resolvent operators, which is inherently restricted to the Euclidean norm. Besides, their bound is on $\|z_t - z^*\|^2$, and it was claimed that "the convex minimization analysis does not apply and we use the notion of monotone operators to prove convergence". We show here that by introducing an auxiliary variable, our analysis in Appendix A.2 can be largely reused for SVRG with Bregman divergence in saddle-point problems, and the bound is directly on function values.

Theorem 2. *Let Assumption 1 hold, and choose a sufficiently small $\eta > 0$ such that $m := \left\lceil \log\left(\frac{1-\eta L}{18\eta L^2} - \eta - 1\right)/\log(1+\eta)\right\rceil \geq 1$. Then Breg-SVRG enjoys linear convergence in expectation:*

$$\mathbb{E}\epsilon(\tilde{z}^s) \leq (1+\eta)^{-ms}[\Delta(z^*, z_0) + c(Z+1)\epsilon(z_0)],$$

where $Z = \sum_{t=0}^{m-1}(1+\eta)^t$, $c = \frac{18\eta^2 L^2}{1-\eta L}$.

Appendix A (Continued)

Proof. Our key innovation in analysis is the introduction of an auxiliary variable: $u_t = \begin{pmatrix} \partial_x K(x_t, y^*) \\ -\partial_y K(x^*, y_t) \end{pmatrix}$.

Note that $\mathbb{E}_\xi v_t \neq u_t$. Recall that

$$\epsilon_t^M := \epsilon^M(z_t) := M(x_t, y^*) - M(x^*, y_t), \qquad \epsilon_t^K := \epsilon^K(z_t) := K(x_t, y^*) - K(x^*, y_t) \tag{A.38}$$

$$\epsilon_t^K := \epsilon^K(z_t) := K(x_t, y^*) - K(x^*, y_t) \in \mathbb{R} \tag{A.39}$$

$$\epsilon_t = \epsilon(z_t) = J(x_t, y^*) - J(x^*, y_t) = [J(x_t, y^*) - J(x^*, y^*)] + [J(x^*, y^*) - J(x^*, y_t)] \tag{A.40}$$

$$\geq \Delta(z_t, z^*) \geq \tfrac{1}{2} \|z_t - z^*\|^2 \geq 0. \tag{A.41}$$

The first step of our proof is to invoke Lemma 1 on the update Equation 2.32:

$$(1+\eta)\Delta(z^*, z_{t+1}) \leq \Delta(z^*, z_t) - \eta\epsilon_{t+1}^M - \Delta(z_{t+1}, z_t) + \eta \langle v_t, z^* - z_t \rangle + \eta \langle v_t, z_t - z_{t+1} \rangle$$

$$= \Delta(z^*, z_t) - \eta\epsilon_{t+1}^M - \Delta(z_{t+1}, z_t) + \eta \langle v_t, z^* - z_t \rangle + \eta \langle v_t - u_t, z_t - z_{t+1} \rangle + \eta \langle u_t, z_t - z_{t+1} \rangle .$$

It is easy to bound $\langle u_t, z_t - z_{t+1} \rangle$ as K is L-smooth:

$$\langle u_t, z_t - z_{t+1} \rangle = \langle \partial_x K(x_t, y^*), x_t - x_{t+1} \rangle - \langle \partial_y K(x^*, y_t), y_t - y_{t+1} \rangle$$

$$\leq K(x_t, y^*) - K(x_{t+1}, y^*) + \tfrac{L}{2} \|x_t - x_{t+1}\|^2 + K(x^*, y_{t+1}) - K(x^*, y_t) + \tfrac{L}{2} \|y_t - y_{t+1}\|^2$$

$$= \epsilon_t^K - \epsilon_{t+1}^K + \tfrac{L}{2} \|z_t - z_{t+1}\|^2 . \tag{A.42}$$

Appendix A (Continued)

So we can proceed by

$$
\begin{aligned}
&(1+\eta)\Delta(z^*, z_{t+1}) \\
\leq &\Delta(z^*, z_t) - \eta\epsilon_{t+1} + \eta\epsilon_t^K + \eta\left\langle v_t, z^* - z_t\right\rangle + \eta\left\langle v_t - u_t, z_t - z_{t+1}\right\rangle - \tfrac{1-\eta L}{2}\left\|z_t - z_{t+1}\right\|^2 \\
\leq &\Delta(z^*, z_t) - \eta\epsilon_{t+1} + \eta\epsilon_t^K + \eta\left\langle v_t, z^* - z_t\right\rangle + \frac{\eta^2\left\|v_t - u_t\right\|_*^2}{2(1-\eta L)}.
\end{aligned}
$$

Take expectation over ξ on both sides (conditional on z_t). Since $\mathbb{E}_\xi[v_t] = \mathsf{G}(z_t)$, we may apply the inequality $K(x, y') - K(x', y) \leq \langle \mathsf{G}(z), z - z'\rangle$:

$$
(1+\eta)\mathbb{E}\Delta(z^*, z_{t+1}) \leq \Delta(z^*, z_t) - \eta\mathbb{E}\epsilon_{t+1} + \frac{\eta^2}{2(1-\eta L)}\mathbb{E}\left\|v_t - u_t\right\|_*^2. \tag{A.43}
$$

Finally we bound $\mathbb{E}\left\|v_t - u_t\right\|_*^2$:

$$
\mathbb{E}\left\|v_t - u_t\right\|_*^2 = \mathbb{E}\left\|v_t - \mathsf{G}(z_t) + \mathsf{G}(z_t) - u_t\right\|_*^2 \leq 2\mathbb{E}\left\|v_t - \mathsf{G}(z_t)\right\|_*^2 + 2\left\|\mathsf{G}(z_t) - u_t\right\|_*^2. \tag{A.44}
$$

Notice that by the L-smoothness of K,

$$
\begin{aligned}
\left\|\mathsf{G}(z_t) - u_t\right\|_*^2 &= \left\|\partial_x K(x_t, y_t) - \partial_x K(x_t, y^*)\right\|_*^2 + \left\|\partial_y K(x_t, y_t) - \partial_y K(x^*, y_t)\right\|_*^2 \\
&\leq L^2\left\|y_t - y^*\right\|^2 + L^2\left\|x_t - x^*\right\|^2. \qquad \text{(A.45)}
\end{aligned}
$$

Appendix A (Continued)

Again using $\mathbb{E}\left\|X-\mathbb{E}[X]\right\|^2 \leq 4\mathbb{E}\left\|X\right\|^2$ and L-smoothness of ψ_k,

$$\begin{aligned}\mathbb{E}\left\|v_t-\mathsf{G}(z_t)\right\|_*^2 &= \mathbb{E}\|\nabla\psi_\xi(z_t) - \nabla\psi_\xi(\tilde{z}) - \mathbb{E}[\nabla\psi_\xi(z_t) - \nabla\psi_\xi(\tilde{z})]\|_*^2 \\ &\leq 4\mathbb{E}\left\|\nabla\psi_\xi(z_t) - \nabla\psi_\xi(\tilde{z})\right\|_*^2 \leq 4L^2\left\|z_t-\tilde{z}\right\|^2 \\ &\leq 8L^2\left\|z_t-z^*\right\|^2 + 8L^2\left\|\tilde{z}-z^*\right\|^2. \end{aligned} \tag{A.46}$$

Plug Equation A.45 and Equation A.46 into Equation A.44, and then into Equation A.43. Using Equation A.38, we finally arrive at (expectation on ξ)

$$(1+\eta)\mathbb{E}\Delta(z^*, z_{t+1}) \leq \Delta(z^*, z_t) - \eta\mathbb{E}\epsilon_{t+1} + \frac{18\eta^2L^2}{1-\eta L}(\epsilon_t + \epsilon(\tilde{z}^{s-1})). \tag{A.47}$$

Taking expectation of the whole history on both sides, we obtain

$$\rho\Delta_{t+1} \leq \Delta_t + c'\left(\delta_t + \tilde{\delta}^{s-1}\right) - \eta\delta_{t+1}. \tag{A.48}$$

where $\rho := 1+\eta$, $c' := \frac{18\eta^2L^2}{1-\eta L}$, $\delta_t := \mathbb{E}\epsilon(z_t)$, $\Delta_t := \mathbb{E}\Delta(z^*, z_t)$, $\tilde{\delta}^{s-1} := \mathbb{E}\epsilon(\tilde{z}^{s-1})$. This has exactly the same shape as Equation A.27, and therefore the rest derivation is almost identical, except that in c', we have η^2L^2 rather that η^2L as under Equation A.27. So almost all the derivation can be shared. Let us set $\eta = \frac{1}{45L^2}$, and we obtain

$$\rho^m = \frac{\eta - c\rho}{c} = \frac{45-1/L}{18} - \frac{1}{45L^2} - 1 \geq \frac{45-1}{18} - \frac{1}{45} - 1 = \frac{64}{45}. \tag{A.49}$$

Appendix A (Continued)

Since $\rho = 1 + \frac{1}{45L^2}$, we derive

$$m = \frac{\log\left(\frac{45-1/L}{18} - \frac{1}{45L^2} - 1\right)}{\log\left(1 + \frac{1}{45L^2}\right)} = \Theta\left(L^2\right). \tag{A.50}$$

So between epochs, the decay is by a factor of $\frac{45}{64}$, and the number of iterations per epoch is $\Theta(L^2)$. The total computational cost is therefore $O\left(L^2 \log \frac{1}{\epsilon}\right)$. □

A.4 Efficient Proximal Operator for Solving Equation 2.40

Given a set of variables $\{\boldsymbol{\alpha}_i\}$ which are *not* necessarily in S, we need to project it to S based on Bregman divergence. Here we show how this can be done in $O(n^2)$ time for both Euclidean and entropic projections.

A.4.1 Euclidean projection to S

Given a set $\{\boldsymbol{\alpha}_k\}$, the projection to S requires solving

$$\min_{\{\mathbf{x}_k\}} \tfrac{1}{2}\textstyle\sum_k \|\mathbf{x}_k - \boldsymbol{\alpha}_k\|_2^2 \text{ s.t. } \mathbf{x}_k \in C_k,\ \sum_k \mathbf{1}^T\mathbf{x}_k \le 1 \tag{A.51}$$

$$\text{where} \quad C_k := \{\mathbf{x} \in [0, \tfrac{r_k}{k}]^n : r_k \ge 0,\ \mathbf{1}^T\mathbf{x} = r_k\} \tag{A.52}$$

Introducing a Lagrange variable ρ that corresponds to the last constraint, we obtain the partial Lagrangian:

$$\max_{\rho \ge 0} -\rho + \sum_k \min_{\mathbf{x}_k \in C_k} \left\{ \frac{1}{2}\|\mathbf{x}_k - \boldsymbol{\alpha}_k\|_2^2 + \rho \mathbf{1}^T\mathbf{x}_k \right\}. \tag{A.53}$$

Appendix A (Continued)

Since the optimal $\mathbf{x}_k$ is unique by strong convexity, we can solve ρ by any smooth solver such as BFGS, proximal bundle method (PBM, http://napsu.karmitsa.fi/proxbundle/), or even bi-section. Given a ρ and its optimal $\mathbf{x}_k(\rho)$, the gradient in ρ can be easily computed as $-1 + \sum_k \mathbf{1}^T \mathbf{x}_k(\rho)$. Therefore it suffices to optimize $\mathbf{x}_k$ separately. In the sequel, we will first present the optimization procedure without worrying about the computational cost. After that, we will show how to reduce the complexity to $\tilde{O}(n)$.

Fixing ρ, the optimal $\mathbf{x}_k$ can be found by solving the following problem. Here we dropped all subscripts k to lighten the notation.

$$\min_{r \geq 0} \min_{\mathbf{x} \in [0, \frac{r}{k}]^n, \, \mathbf{1}^T \mathbf{x} = r} \frac{1}{2} \|\mathbf{x} - \boldsymbol{\alpha}\|_2^2 + \rho \mathbf{1}^T \mathbf{x}. \tag{A.54}$$

Introducing a Lagrange multiplier μ for the $\mathbf{1}^T \mathbf{x} = r$ constraint, we dualize the inner problem as

$$\begin{aligned} & \min_{r \geq 0} \min_{\mathbf{x} \in [0, \frac{r}{k}]^n} \max_{\mu} \frac{1}{2} \|\mathbf{x} - \boldsymbol{\alpha}\|_2^2 + \rho \mathbf{1}^T \mathbf{x} - \mu(\mathbf{1}^T \mathbf{x} - r) \\ = & \min_{r \geq 0} \max_{\mu} \left\{ \mu r + \min_{\mathbf{x} \in [0, \frac{r}{k}]^n} \frac{1}{2} \|\mathbf{x} - \boldsymbol{\alpha}\|_2^2 + \rho \mathbf{1}^T \mathbf{x} - \mu \mathbf{1}^T \mathbf{x} \right\} \qquad \text{(A.55)} \\ = & \min_{r \geq 0} \max_{\mu} \left\{ \mu r + \min_{\mathbf{y} \in [0,1]^n} \frac{1}{2} \left\| \tfrac{r}{k} \mathbf{y} - \boldsymbol{\alpha} \right\|_2^2 + (\rho - \mu) \tfrac{r}{k} \mathbf{1}^T \mathbf{y} \right\}. \qquad \text{(A.56)} \end{aligned}$$

Appendix A (Continued)

Now r does not appear in the constraint and so the once we have the optimal μ and $\mathbf{y}$ (or optimal $\mathbf{x}$ based on which we get the optimal $\mathbf{y} = \frac{k}{r}\mathbf{x}$), the gradient in r can be written as

$$\mu + \mathbf{y}^T(\tfrac{r}{k}\mathbf{y} - \boldsymbol{\alpha})/k + (\rho - \mu)\mathbf{1}^T\mathbf{y}/k. \tag{A.57}$$

which can be calculated in constant time via our efficient update rule.

Given r and μ, the optimal $\mathbf{x}$ admits a closed form

$$\mathbf{x} = \text{MED}(\boldsymbol{\alpha} + \mu\mathbf{1} - \rho\mathbf{1}, \mathbf{0}, \tfrac{r}{k}\mathbf{1}), \tag{A.58}$$

where MED stands for the elementwise median. Given r, the optimal μ is the one that ensures $\mathbf{1}^T\mathbf{x} = r$ (*not* necessarily unique). Since each x in Equation A.58 is non-decreasing in μ, a simple bi-section search can find such a $\mu(r)$ by probing $O(\log n)$ values of μ. With $\mu(r)$ in hand, the optimal $\mathbf{x}(r)$ for the inner problem in Equation A.54 can be recovered by Equation A.58.

In hindsight, we observe that although the optimal $\mathbf{x}$ in Equation A.54 is unique, the objective function in r is not necessarily smooth because r also appears in the constraints of $\mathbf{x}$. Therefore we resort to a nonsmooth solver (e.g. PBM) for optimizing over r.

The overall procedure for Euclidean projection is summarized in Algorithm 6. We assumed without loss of generality that for each $\boldsymbol{\alpha}_k$ all its elements are already sorted increasingly. The binary search over μ can be refined, with μ only probing kink points corresponding to the entries in $\boldsymbol{\alpha}_k$. This will ensure the bi-section terminates in $O(\log \min\{n, 1/\epsilon\})$ iterations.

Appendix A (Continued)

A.4.2 Entropic projection to S

Given a set $\{\boldsymbol{\alpha}_k\}$, the entropic projection to S requires solving

$$\min_{\{\mathbf{x}_k\}} \sum_{ks} x_{ks} \log \frac{x_{ks}}{\alpha_{ks}} + \alpha_{ks} - x_{ks} \tag{A.59}$$
$$s.t.\ \mathbf{x}_k \in C_k,\ \sum_{ks} x_{ks} \le 1 \quad \text{where} \quad C_k := \{\mathbf{x} \in [0, \frac{r_k}{k}]^n : r_k \ge 0,\ \mathbf{1}^T\mathbf{x} = r_k\}$$

Introducing a Lagrange variable ρ that corresponds to the last constraint, we obtain the partial Lagrangian:

$$\max_{\rho \ge 0} -\rho + \sum_{ks} \min_{\mathbf{x}_k \in C_k} \left\{ x_{ks} \log \frac{x_{ks}}{\alpha_{ks}} + \alpha_{ks} - x_{ks} + \rho x_{ks} \right\}. \tag{A.60}$$

Since the optimal $\mathbf{x}_k$ is unique by strong convexity, we can solve ρ by any smooth solver such as BFGS, PBM, or even bi-section. Given a ρ and its optimal $\mathbf{x}_k(\rho)$, the gradient in ρ can be easily computed as $-1 + \sum_k \mathbf{1}^T\mathbf{x}_k(\rho)$. Therefore it suffices to optimize $\mathbf{x}_k$ separately.

Fixing ρ, the optimal $\mathbf{x}_k$ can be found by solving the following problem. Here we dropped all subscripts k to lighten the notation.

$$\min_{r \ge 0} \min_{\mathbf{x} \in [0, \frac{r}{k}]^n,\, \mathbf{1}^T\mathbf{x} = r} \sum_s \{Q + \rho x_s\} \quad \text{where} \quad Q = x_s \log \frac{x_s}{\alpha_s} + \alpha_s - x_s \tag{A.61}$$

Appendix A (Continued)

Introducing a Lagrange multiplier μ for the $\mathbf{1}^T\mathbf{x} = r$ constraint, we dualize the inner problem as

$$\min_{r\geq 0}\max_{\mu}\left\{\mu r + \min_{x_s\in[0,\frac{r}{k}]}\sum_s\{Q+\rho x_s - \mu x_s\}\right\} \tag{A.62}$$

$$= \min_{r\geq 0}\max_{\mu}\left\{\mu r + \min_{y_s\in[0,1]}\sum_s\{Q_y + (\rho-\mu)\frac{r}{k}y_s\}\right\} \tag{A.63}$$

$$\text{where } Q_y = \frac{r}{k}y_s\log\frac{y_s r}{\alpha_s k} + \alpha_s - \frac{r}{k}y_s$$

the gradient in r can be written as

$$\mu + \sum_s \frac{y_s}{k}\log\frac{y_s r}{\alpha_s k} + (\rho-\mu)\frac{y_s}{k}. \tag{A.64}$$

which can be calculated in constant time via our efficient update rule.

Given r and μ, the optimal $\mathbf{x}$ admits a closed form

$$\mathbf{x} = \text{MIN}\left(\boldsymbol{\alpha}\exp(\mu-\rho), \frac{r}{k}\mathbf{1}\right), \tag{A.65}$$

where MIN stands for the elementwise minimum. Given r, the optimal μ is the one that ensures $\mathbf{1}^T\mathbf{x} = r$ (*not* necessarily unique). Since each x in Equation A.65 is non-decreasing in μ, a simple bi-section search can find such a $\mu(r)$ by probing $O(\log n)$ values of μ. With $\mu(r)$ in hand, the optimal $\mathbf{x}(r)$ for the inner problem in Equation A.61 can be recovered by Equation A.65.

The overall procedure for Entropic projection is summarized in Algorithm 7. We assumed without loss of generality that for each $\boldsymbol{\alpha}_k$ all its elements are already sorted increasingly. The

binary search over μ can be refined, with μ only probing kink points corresponding to the entries in $\boldsymbol{\alpha}_k$. This will ensure the bi-section terminates in $O(\log n) < O(\log 1/\epsilon)$ iterations.

A.5 Rates for Proximal Saddle-Point Optimization (Non-uniform)

Problem. We consider the following problem

$$(x^*, y^*) = \arg\min_x \max_y K(x,y) + M(x,y), \quad \text{where} \quad K(x,y) = \sum_{i \in J} \psi_i(x,y). \tag{A.66}$$

Assumption 2. We assume each ψ_i is a saddle function that is L_i-smooth as follows:

$$L_i = \sup_{z \neq z'} \left\| B_i(z) - B_i(z') \right\|_* \quad s.t. \left\| z - z' \right\| \leq 1, \tag{A.67}$$

$$\text{where} \quad B_i(z) = [\partial_x \psi_i(x,y); -\partial_y \psi_i(x,y)]$$

and K is L_{avg}-smooth:

$$L_{avg} = \sup_{z \neq z'} \left\| B(z) - B(z') \right\|_* \quad s.t. \left\| z - z' \right\| \leq 1, \tag{A.68}$$

$$\text{where} \quad B(z) = [\partial_x \psi(x,y); -\partial_y \psi(x,y)]$$

Then we can define $\bar{L}$ adapted to our sampling schemes:

$$\bar{L}(\pi)^2 = \sup_{z \neq z'} \sum_{i \in J} \frac{1}{\pi_i} \left\| B_i(z) - B_i(z') \right\|_*^2 \quad s.t. \left\| z - z' \right\| \leq 1, \tag{A.69}$$

$$\text{where} \quad B_i(z) = [\partial_x \psi_i(x,y); -\partial_y \psi_i(x,y)]$$

Appendix A (Continued)

and π is a probability vector that sums to 1. We always have the bound:

$$L_{avg} \leq \sum_{i=1}^{J} L_i \tag{A.70}$$

$$L_{avg}^2 \leq \bar{L}(\pi)^2 \leq (\sum_{i=1}^{J} \frac{L_i^2}{\pi_i}) \leq L_{max}^2 \times \sum_{i=1}^{J} \frac{1}{\pi_i}. \tag{A.71}$$

Algorithm. Let us define a variant of variance-reduced stochastic gradient for saddle-point problems:

$$v_t := [v_x(z_t); -v_y(z_t)], \tag{A.72}$$

$$\text{where} \quad v_x(z_t) := \frac{1}{\pi_\xi}(\partial_x \psi_\xi(z_t) - \partial_x \psi_\xi(\tilde{z})) + \partial_x K(\tilde{z}) \tag{A.73}$$

$$v_y(z_t) := \frac{1}{\pi_\xi}(\partial_y \psi_\xi(z_t) - \partial_y \psi_\xi(\tilde{z})) + \partial_y K(\tilde{z}). \tag{A.74}$$

Here $\tilde{z}$ is the pivot chosen after completing the last epoch, ξ is randomly choose from probability vector π. Clearly, $\mathbb{E}_\xi[v_t] = \mathsf{G}(z_t)$ (unbiased). The stochastic algorithm then performs the proximal update at each step:

$$(x_{t+1}, y_{t+1}) = \arg\min_x \max_y \eta \langle v_x(z_t), x \rangle + \eta \langle v_y(z_t), y \rangle + \eta M(x, y) + \Delta(x, x_t) - \Delta(y, y_t). \tag{A.75}$$

Theorem 10. *With the above modification, the same guarantee in Theorem 2 with L (in fact, this L is $\max_{i \in J} L_i$) replaced by $\bar{L} = \bar{L}(\pi)$ holds.*

Appendix A (Continued)

Proof. Our key innovation in analysis is the introduction of an auxiliary variable:

$$u_t = \begin{pmatrix} \partial_x K(x_t, y^*) \\ -\partial_y K(x^*, y_t) \end{pmatrix}. \tag{A.76}$$

Note that $\mathbb{E}_\xi v_t \neq u_t$. The first step of our proof is to invoke Lemma 1 on the update Equation A.75:

$$(1+\eta)\Delta(z^*, z_{t+1}) \leq \Delta(z^*, z_t) - \eta\epsilon_{t+1}^M - \Delta(z_{t+1}, z_t) + \eta \langle v_t, z - z_t \rangle + \eta \langle v_t, z_t - z_{t+1} \rangle \tag{A.77}$$

$$= \Delta(z^*, z_t) - \eta\epsilon_{t+1}^M - \Delta(z_{t+1}, z_t) + \eta \langle v_t, z - z_t \rangle \tag{A.78}$$

$$+ \eta \langle v_t - u_t, z_t - z_{t+1} \rangle + \eta \langle u_t, z_t - z_{t+1} \rangle .$$

It is easy to bound $\langle u_t, z_t - z_{t+1} \rangle$ as K is L_{avg}-smooth:

$$\langle u_t, z_t - z_{t+1} \rangle = \langle \partial_x K(x_t, y^*), x_t - x_{t+1} \rangle - \langle \partial_y K(x^*, y_t), y_t - y_{t+1} \rangle \tag{A.79}$$

$$\leq K(x_t, y^*) - K(x_{t+1}, y^*) + \tfrac{L_{avg}}{2} \|x_t - x_{t+1}\|^2$$

$$+ K(x^*, y_{t+1}) - K(x^*, y_t) + \tfrac{L_{avg}}{2} \|y_t - y_{t+1}\|^2 \tag{A.80}$$

$$= \epsilon_t^K - \epsilon_{t+1}^K + \tfrac{L_{avg}}{2} \|z_t - z_{t+1}\|^2 \tag{A.81}$$

$$= \epsilon_t^K - \epsilon_{t+1}^K + \tfrac{\bar{L}}{2} \|z_t - z_{t+1}\|^2 . \tag{A.82}$$

The last inequality is due to $L_{avg}^2 \leq \bar{L}^2$.

Appendix A (Continued)

So we can proceed by

$$(1+\eta)\Delta(z^*, z_{t+1}) \leq \Delta(z^*, z_t) - \eta\epsilon_{t+1} + \eta\epsilon_t^K + \eta\left\langle v_t, z^* - z_t\right\rangle + \eta\left\langle v_t - u_t, z_t - z_{t+1}\right\rangle \quad \text{(A.83)}$$

$$- \tfrac{1-\eta\bar{L}}{2}\left\|z_t - z_{t+1}\right\|^2$$

$$\leq \Delta(z^*, z_t) - \eta\epsilon_{t+1} + \eta\epsilon_t^K + \eta\left\langle v_t, z^* - z_t\right\rangle + \frac{\eta^2\left\|v_t - u_t\right\|_*^2}{2(1-\eta\bar{L})}. \quad \text{(A.84)}$$

Take expectation over ξ on both sides (conditional on z_t). Since $\mathbb{E}_\xi[v_t] = \mathsf{G}(z_t)$, we apply the inequality $K(x, y') - K(x', y) \leq \langle \mathsf{G}(z), z - z'\rangle$ and get:

$$(1+\eta)\mathbb{E}\Delta(z^*, z_{t+1}) \leq \Delta(z^*, z_t) - \eta\mathbb{E}\epsilon_{t+1} + \frac{\eta^2}{2(1-\eta\bar{L})}\mathbb{E}\left\|v_t - u_t\right\|_*^2. \quad \text{(A.85)}$$

Finally we bound $\mathbb{E}\left\|v_t - u_t\right\|_*^2$:

$$\mathbb{E}\left\|v_t - u_t\right\|_*^2 = \mathbb{E}\left\|v_t - \mathsf{G}(z_t) + \mathsf{G}(z_t) - u_t\right\|_*^2$$

$$\leq 2\mathbb{E}\left\|v_t - \mathsf{G}(z_t)\right\|_*^2 + 2\left\|\mathsf{G}(z_t) - u_t\right\|_*^2. \quad \text{(A.86)}$$

Notice that by the L-smoothness of K,

$$\left\|\mathsf{G}(z_t) - u_t\right\|_*^2 = \left\|\partial_x K(x_t, y_t) - \partial_x K(x_t, y^*)\right\|_*^2 + \left\|\partial_y K(x_t, y_t) - \partial_y K(x^*, y_t)\right\|_*^2 \quad \text{(A.87)}$$

$$\leq L_{avg}^2\left\|y_t - y^*\right\|^2 + L_{avg}^2\left\|x_t - x^*\right\|^2 \quad \text{(A.88)}$$

$$\leq \bar{L}^2\left\|y_t - y^*\right\|^2 + \bar{L}^2\left\|x_t - x^*\right\|^2. \quad \text{(A.89)}$$

Appendix A (Continued)

Again using the definition Equation A.69,

$$\begin{aligned}\mathbb{E}\left\|v_t-\mathsf{G}(z_t)\right\|_*^2 &= \mathbb{E}\|\frac{1}{\pi_\xi}(\nabla\psi_\xi(z_t)-\nabla\psi_\xi(\tilde{z}))-\mathbb{E}[\frac{1}{\pi_\xi}(\nabla\psi_\xi(z_t)-\nabla\psi_\xi(\tilde{z}))]\|_*^2\\ &\le 4\mathbb{E}\left\|\frac{1}{\pi_\xi}(\nabla\psi_\xi(z_t)-\nabla\psi_\xi(\tilde{z}))\right\|_*^2\\ &\le 4\bar{L}^2\left\|z_t-\tilde{z}\right\|^2 \quad \text{(by } \bar{L}\text{-smoothness)}\\ &\le 8\bar{L}^2\left\|z_t-z^*\right\|^2+8\bar{L}^2\left\|\tilde{z}-z^*\right\|^2. \end{aligned} \tag{A.90}$$

Plug Equation A.88 and Equation A.90 into Equation A.86, and then into Equation A.85. Using Equation A.38, we finally arrive at (expectation is only over ξ)

$$(1+\eta)\mathbb{E}\Delta(z^*,z_{t+1}) \le \Delta(z^*,z_t)-\eta\mathbb{E}\epsilon_{t+1}+\frac{18\bar{L}^2\eta^2}{1-\eta\bar{L}}(\epsilon_t+\epsilon(\tilde{z}^{s-1})). \tag{A.91}$$

Taking expectation of the whole history on both sides, we obtain

$$\rho\Delta_{t+1} \le \Delta_t + c'\left(\delta_t+\tilde{\delta}^{s-1}\right)-\eta\delta_{t+1}. \tag{A.92}$$

where $\rho := 1+\eta$, $c' := \frac{18\bar{L}^2\eta^2}{1-\eta\bar{L}}$, $\delta_t := \mathbb{E}\epsilon(z_t)$, $\Delta_t := \mathbb{E}\Delta(z^*,z_t)$, $\tilde{\delta}^{s-1} := \mathbb{E}\epsilon(\tilde{z}^{s-1})$. This has exactly the same shape as Equation A.27, and therefore the rest derivation is almost identical, except that in c', we have $\bar{L}^2\eta^2$ rather that $\eta^2 L_Q$ as under Equation A.27. So almost all the derivation can be shared. Let us set $\eta=\frac{1}{45\bar{L}^2}$, and we obtain

$$\rho^m = \frac{\eta-c\rho}{c} = \frac{45-1/\bar{L}}{18}-\frac{1}{45\bar{L}^2}-1 \ge \frac{45-1}{18}-\frac{1}{45}-1 = \frac{64}{45}. \tag{A.93}$$

Appendix A (Continued)

Since $\rho = 1 + \frac{1}{45\bar{L}^2}$, we derive

$$m = \frac{\log\left(\frac{45-1/\bar{L}}{18} - \frac{1}{45\bar{L}^2} - 1\right)}{\log\left(1 + \frac{1}{45\bar{L}^2}\right)} = \Theta\left(\bar{L}^2\right). \tag{A.94}$$

So between epochs, the decay is by a factor of $\frac{45}{64}$, and the number of iterations per epoch is $\Theta(\bar{L}^2)$. The total computational cost is therefore $O\left(\bar{L}^2 \log \frac{1}{\epsilon}\right)$.

In F-score game,

$$\min_{\boldsymbol{\alpha}\in S} \max_{\boldsymbol{\beta}\in S} \sum_{i=1}^{n}\sum_{j=1}^{n} \Big[\underbrace{\tfrac{2ij}{i+j}\boldsymbol{\alpha}_i^T\boldsymbol{\beta}_j - \frac{i}{\lambda n^3}\mathbf{c}^T X\boldsymbol{\alpha}_i + \frac{ij}{2\lambda n^2}\boldsymbol{\alpha}_i^T X^T X\boldsymbol{\alpha}_j}_{f_{ij}(\boldsymbol{\alpha}_i,\boldsymbol{\beta}_j)} + \frac{1}{2\lambda n^4}\left\|\mathbf{c}\right\|_2^2 \Big], \tag{A.95}$$

where $\mathbf{c} = X\tilde{\mathbf{y}}$. Both $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are n^2-dimensional, and the bilinear function f_{ij} can be written as $\boldsymbol{\alpha}^T A_{ij}\boldsymbol{\beta}$, where $A_{ij} \in \mathbb{R}^{n^2\times n^2}$ is an n-by-n block matrix, with the (i,j)-th block being $\frac{2ij}{i+j}I$ and all other blocks being $\mathbf{0}$. The linear terms can be absorbed into the regularizer Ω without affecting the smoothness parameter.

The quadratic term w.r.t. $\boldsymbol{\alpha}$ can be written as $\boldsymbol{\alpha}^T B_{ij}\boldsymbol{\alpha}$, where $B_{ij} \in \mathbb{R}^{n^2\times n^2}$ is an n-by-n block matrix, with its (i,j)-th block being $\frac{ij}{2\lambda n^2}X^T X$ and all other blocks being $\mathbf{0}$. And we assume each $\left\|\mathbf{x}_i\right\|_2 \le 1$. The smoothness constant can be bounded separately from A_{ij} and B_{ij}; see Equation A.104 in Appendix A.6.

we consider three schemes: (a) $\psi_{ij} = f_{ij}$; (b) $\psi_{i\cdot} = \sum_{j=1}^{n} f_{i,j}$; and (c) $\psi_{\cdot j} = \sum_{i=1}^{n} f_{i,j}$.

Note these are just splitting schemes of objectives, the corresponding derivatives B_{ij} and smoothness constant $\bar{L}$ are listed later.

Appendix A (Continued)

For scheme (a), the smoothness constant square ${L_{max}}_2^2$ under ℓ_2 norm is upper bounded by the sum of spectral norm square of A_{ij} and B_{ij}. So ${L_{max}}_2^2 \geq \max_{i,j} \left(\frac{ij}{2\lambda n^2} n\right)^2 = \Omega(n^2)$. In contrast the smoothness constant square ${L_{max}}_1^2$ under ℓ_1 norm is at most the sum of square of maximum absolute value of the entries in A_{ij} and B_{ij}. Hence ${L_{max}}_1^2 \leq \max_{i,j} \left(\frac{2ij}{i+j}\right)^2 + \max_{i,j} \left(\frac{ij}{2\lambda n^2}\right)^2 = \Theta(n^2)$. So no saving is achieved here.

For scheme (b), $\psi_{i\cdot}$ corresponds to $\boldsymbol{\alpha}^T \sum_{j=1}^n A_{kj}\boldsymbol{\beta} + \boldsymbol{\alpha}^T \sum_{j=1}^n B_{kj}\boldsymbol{\alpha}$. Then

$$L_{max}{}_1^2 \leq \max_k \left[\max_{\mathbf{v}:\|\mathbf{v}\|_1=1} \left\| \sum_{j=1}^n A_{kj}\mathbf{v} \right\|_\infty^2 + \max_{\mathbf{v}:\|\mathbf{v}\|_1=1} \left\| \sum_{j=1}^n B_{kj}\mathbf{v} \right\|_\infty^2 \right] \quad \text{(by Equation A.104)} \tag{A.96}$$

$$\leq \max_k \max_j \left[\left(\frac{2kj}{k+j}\right)^2 + \left(\frac{kj}{2n^2}\right)^2 \right] = n^2, \tag{A.97}$$

$$L_{max}{}_2^2 \geq \max_k \left[\max_{\mathbf{v}:\|\mathbf{v}\|_2=1} \sum_{j=1}^n \left\| B_{kj}\mathbf{v} \right\|_2^2 \right] = n^3. \tag{A.98}$$

Similar results apply to scheme (c) too. It can also be shown that if our scheme randomly samples n entries from $\{A_{ij}, B_{ij}\}$, the above L_{max1} and L_{max2} cannot be improved by further engineering the factorization.

1. Euc-Uniform-Individual:

$\pi_{ij} = \frac{1}{n^2}, \quad {L_{ij}}_2^2 \leq {L_{max}}_2^2 = n^2, \quad B_{ij}(z) = [\partial_x \psi_{ij}(x,y); -\partial_y \psi_{ij}(x,y)] = [A_{ij}\boldsymbol{\beta}, -A_{ij}^T\boldsymbol{\alpha}]$

Appendix A (Continued)

$$\bar{L}(\pi)^2 = \sup_{z \neq z'} \sum_{ij} n^2 \left\| B_{ij}(z) - B_{ij}(z') \right\|_*^2 \leq n^2 \sum_{ij} {L_{ij}}_2^2 = \Theta(n^6)$$

2. Euc-NonUniform-Individual:

$\pi_{ij} = \frac{{L_{ij}}_2^2}{\sum_{ij} {L_{ij}}_2^2}, \quad {L_{ij}}_2^2 \leq {L_{max}}_2^2 = n^2, \quad B_{ij}(z) = [\partial_x \psi_{ij}(x,y); -\partial_y \psi_{ij}(x,y)]$

$$\bar{L}(\pi)^2 = \sup_{z \neq z'} \sum_{ij} \frac{\sum_{ij} {L_{ij}}_2^2}{{L_{ij}}_2^2} \left\| B_{ij}(z) - B_{ij}(z') \right\|_*^2 \leq n^2 \sum_{ij} {L_{ij}}_2^2 = \Theta(n^6)$$

3. Ent-Uniform-Individual:

$\pi_{ij} = \frac{1}{n^2}, \quad {L_{ij}}_1^2 \leq {L_{max}}_1^2 = n^2, \quad B_{ij}(z) = [\partial_x \psi_{ij}(x,y); -\partial_y \psi_{ij}(x,y)] = [A_{ij}\boldsymbol{\beta}, -A_{ij}^T \boldsymbol{\alpha}]$

$$\bar{L}(\pi)^2 = \sup_{z \neq z'} \sum_{ij} n^2 \left\| B_{ij}(z) - B_{ij}(z') \right\|_*^2 \leq n^2 \sum_{ij} {L_{ij}}_1^2 = \Theta(n^6)$$

4. Ent-NonUniform-Individual:

$$\bar{L}(\pi)^2 = \sup_{z \neq z'} \sum_{ij} \frac{\sum_{ij} {L_{ij}}_1^2}{{L_{ij}}_1^2} \left\| B_{ij}(z) - B_{ij}(z') \right\|_*^2 \leq n^2 \sum_{ij} {L_{ij}}_1^2 = \Theta(n^6)$$

5. Euc-Uniform-factored:

$p_i = q_j = \frac{1}{n}, \quad {L_{ij}}_2^2 \leq {L_{max}}_2^2 = n^3, \quad B_{ij}(z) = [p_j \partial_x \psi_{i\cdot}(x,y); -q_i \partial_y \psi_{\cdot j}(x,y)]$

Appendix A (Continued)

$$\bar{L}(\pi)^2 = \sup_{z \neq z'} \sum_{ij} \frac{1}{p_i q_j} \left\| B_{ij}(z) - B_{ij}(z') \right\|_*^2 \leq \sum_{ij} {L_{ij}}_2^2 = \Theta(n^5)$$

6. Euc-NonUniform-factored:

$p_i = q_j = \frac{{L_{ij}}_2^2}{\sum_{ij} {L_{ij}}_2^2}, \quad {L_{ij}}_2^2 \leq {L_{max}}_2^2 = n^3, \quad B_{ij}(z) = [p_j \partial_x \psi_{i\cdot}(x,y); -q_i \partial_y \psi_{\cdot j}(x,y)]$

$$\bar{L}(\pi)^2 = \sup_{z \neq z'} \sum_{ij} \frac{1}{p_i q_j} \left\| B_{ij}(z) - B_{ij}(z') \right\|_*^2 \leq \sum_{ij} {L_{ij}}_2^2 = \Theta(n^5)$$

7. Ent-Uniform-factored:

$\pi_{ij} = p_i q_j = \frac{1}{n}\frac{1}{n} = \frac{1}{n^2}, \quad {L_{ij}}_1^2 \leq {L_{max}}_1^2 = n^2, \quad B_{ij}(z) = [p_j \partial_x \psi_{i\cdot}(x,y); -q_i \partial_y \psi_{\cdot j}(x,y)]$

$$\bar{L}(\pi)^2 = \sup_{z \neq z'} \sum_{ij} \frac{1}{p_i q_j} \left\| B_{ij}(z) - B_{ij}(z') \right\|_*^2 \leq \sum_{ij} {L_{ij}}_1^2 = \Theta(n^4)$$

8. Ent-NonUniform-factored:

$p_i = q_j = \frac{{L_{ij}}_1^2}{\sum_{ij} {L_{ij}}_1^2}, \quad {L_{ij}}_1^2 \leq {L_{max}}_1^2 = n^2, \quad B_{ij}(z) = [p_j \partial_x \psi_{i\cdot}(x,y); -q_i \partial_y \psi_{\cdot j}(x,y)]$

$$\bar{L}(\pi)^2 = \sup_{z \neq z'} \sum_{ij} \frac{1}{p_i q_j} \left\| B_{ij}(z) - B_{ij}(z') \right\|_*^2 \leq \sum_{ij} {L_{ij}}_1^2 = \Theta(n^4)$$

Appendix A (Continued)

From above eight results of $\bar{L}^2$, we see entropy and factored splitting does help in the adversarial prediction, however non-uniform does not help. □

A.6 Bounding Smoothness Constant in Section 2.4

Let us consider both $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ as n^2 dimensional vectors. Denote $\mathbf{z} = \begin{pmatrix} \boldsymbol{\alpha} \\ \\ \boldsymbol{\beta} \end{pmatrix}$. Then the bilinear part of $\frac{1}{n^2}\sum_{ij} f_{ij}(\boldsymbol{\alpha}_i, \boldsymbol{\beta}_j)$ can be written as $F(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{1}{2}\boldsymbol{\alpha}^T A \boldsymbol{\beta} = \frac{1}{2}\mathbf{z}^T \begin{pmatrix} \mathbf{0} & A \\ \\ A^T & \mathbf{0} \end{pmatrix} \mathbf{z}$.

Then $\nabla F(\mathbf{z}) = \begin{pmatrix} \mathbf{0} & A \\ \\ A^T & \mathbf{0} \end{pmatrix} \mathbf{z} = \begin{pmatrix} A\boldsymbol{\beta} \\ \\ A^T\boldsymbol{\alpha} \end{pmatrix}$

Recall that $\|\mathbf{z}\|^2 = \|\boldsymbol{\alpha}\|^2 + \|\boldsymbol{\beta}\|^2$ and similarly for their dual norms. We could use subscripts for these norms to highlight that $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, and $\mathbf{z}$ employ different norms. But we omit these subscripts because they are clear from the context.

So the L^2 of F can be computed as

$$L^2 = \max_{\|\mathbf{z}\| \le 1} \left\| \begin{pmatrix} A\boldsymbol{\beta} \\ \\ A^T\boldsymbol{\alpha} \end{pmatrix} \right\|_*^2 = \max_{\|\boldsymbol{\alpha}\|^2 + \|\boldsymbol{\beta}\|^2 \le 1} \|A\boldsymbol{\beta}\|_*^2 + \left\|A^T\boldsymbol{\alpha}\right\|_*^2. \tag{A.99}$$

The objective function here is obviously convex in $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ jointly. Since we are maximizing, the optimal solution must be attained at some extreme points of the domain. There can be only two types of extreme points: a) $\|\boldsymbol{\alpha}\| = 1$ and $\boldsymbol{\beta} = \mathbf{0}$; and b) $\boldsymbol{\alpha} = \mathbf{0}$ and $\|\boldsymbol{\beta}\| = 1$. So

$$L^2 = \max\left\{ \max_{\|\boldsymbol{\alpha}\|=1} \left\|A^T\boldsymbol{\alpha}\right\|_*^2, \max_{\|\boldsymbol{\beta}\|=1} \|A\boldsymbol{\beta}\|_*^2 \right\}, \tag{A.100}$$

Appendix A (Continued)

where the first term corresponds to case a), and the second term to case b). It is not hard to see from definition that these two terms are equal, both being exactly $\max_{\|\boldsymbol{\alpha}\|=\|\boldsymbol{\beta}\|=1} \boldsymbol{\alpha}^T A\boldsymbol{\beta}$.

Now let us add the quadratic term in $\boldsymbol{\alpha}$, so that $G(\boldsymbol{\alpha},\boldsymbol{\beta}) = \frac{1}{2}\boldsymbol{\alpha}^T A\boldsymbol{\beta} + \frac{1}{2}\boldsymbol{\alpha}^T B\boldsymbol{\alpha}$. Then

$$\nabla G(\mathbf{z}) = \begin{pmatrix} B & A \\ A^T & \mathbf{0} \end{pmatrix} \mathbf{z} = \begin{pmatrix} B\boldsymbol{\alpha} + A\boldsymbol{\beta} \\ A^T\boldsymbol{\alpha} \end{pmatrix} \tag{A.101}$$

So we can compute the L of G by:

$$L^2 = \max_{\|\mathbf{z}\|\le 1} \left\| \begin{pmatrix} B\boldsymbol{\alpha} + A\boldsymbol{\beta} \\ A^T\boldsymbol{\alpha} \end{pmatrix} \right\|_*^2 = \max_{\|\boldsymbol{\alpha}\|^2+\|\boldsymbol{\beta}\|^2\le 1} \|B\boldsymbol{\alpha} + A\boldsymbol{\beta}\|_*^2 + \left\|A^T\boldsymbol{\alpha}\right\|_*^2 \tag{A.102}$$

$$\le \max\left\{ \max_{\|\boldsymbol{\alpha}\|=1} \|B\boldsymbol{\alpha}\|_*^2 + \left\|A^T\boldsymbol{\alpha}\right\|_*^2,\ \max_{\|\boldsymbol{\beta}\|=1} \|A\boldsymbol{\beta}\|_*^2 \right\} \tag{A.103}$$

$$\le \max_{\|\boldsymbol{\alpha}\|=1} \|B\boldsymbol{\alpha}\|_*^2 + \max_{\|\boldsymbol{\alpha}\|=1} \left\|A^T\boldsymbol{\alpha}\right\|_*^2. \tag{A.104}$$

where the last equality again used the fact that $\max_{\|\boldsymbol{\alpha}\|=1} \left\|A^T\boldsymbol{\alpha}\right\|_*^2 = \max_{\|\boldsymbol{\beta}\|=1} \|A\boldsymbol{\beta}\|_*^2$, and $\max_{\boldsymbol{\alpha}}\{f(\boldsymbol{\alpha}) + g(\boldsymbol{\alpha})\} \le \max_{\boldsymbol{\alpha}} f(\boldsymbol{\alpha}) + \max_{\boldsymbol{\alpha}} g(\boldsymbol{\alpha})$.

So now bounding $\max_{\|\boldsymbol{\alpha}\|=1} \|B\boldsymbol{\alpha}\|_*^2$ can be done in the similar way as bounding $\max_{\|\boldsymbol{\alpha}\|=1} \left\|A^T\boldsymbol{\alpha}\right\|_*^2$.

A.7 Stochastic Updates of SVRG on F-score Game

The original problem for F-score is

$$\min_{\{\boldsymbol{\alpha}_i\}\in S} \max_{\{\boldsymbol{\beta}_j\}\in S} \left\{ \frac{1}{n^2}\sum_{i=1}^n\sum_{j=1}^n f_{ij}(\boldsymbol{\alpha}_i,\boldsymbol{\beta}_j) + \max_{\boldsymbol{\theta}} -\frac{\lambda}{2}\|\boldsymbol{\theta}\|_2^2 + \frac{\boldsymbol{\theta}^T}{n} X\tilde{\mathbf{y}} - \frac{\boldsymbol{\theta}^T}{n} X\sum_{i=1}^n i\boldsymbol{\alpha}_i \right\}. \tag{A.105}$$

Appendix A (Continued)

Then the optimal $\boldsymbol{\theta}$ admits a closed form solution

$$\boldsymbol{\theta}(\boldsymbol{\alpha}) = \frac{1}{\lambda n} X \left(\tilde{\mathbf{y}} - \sum_{i=1}^{n} i\boldsymbol{\alpha}_i \right). \tag{A.106}$$

Plugging it back and denoting $\mathbf{c} = X\tilde{\mathbf{y}}$, we arrive at the overall problem in $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ only:

$$\min_{\{\boldsymbol{\alpha}_i\} \in S} \max_{\{\boldsymbol{\beta}_j\} \in S} \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} \left[f_{ij}(\boldsymbol{\alpha}_i, \boldsymbol{\beta}_j) - \frac{i}{\lambda n} \mathbf{c}^T X \boldsymbol{\alpha}_i + \frac{ij}{2\lambda} \boldsymbol{\alpha}_i^T X^T X \boldsymbol{\alpha}_j + \frac{1}{2\lambda n^2} \|\mathbf{c}\|_2^2 \right]. \tag{A.107}$$

1. Full gradient over $\boldsymbol{\alpha}$ in each epoch is

$$\begin{aligned} G(\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\beta}}) &= \nabla \left[\frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij}(\hat{\boldsymbol{\alpha}}_i, \hat{\boldsymbol{\beta}}_j) \right] - \frac{X^T \mathbf{c} \cdot (1:n)}{\lambda n^2} + \frac{X^T X (\sum_{j=1}^{n} j\hat{\boldsymbol{\alpha}}_j) \cdot (1:n)}{\lambda n^2} \\ &= \nabla \left[\frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij}(\hat{\boldsymbol{\alpha}}_i, \hat{\boldsymbol{\beta}}_j) \right] - X^T \boldsymbol{\theta}(\hat{\boldsymbol{\alpha}}) * \frac{(1:n)}{n} \end{aligned}$$

2. The stochastic gradient over $\boldsymbol{\alpha}$ is

$$g(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \nabla \left[\frac{1}{n} \sum_{i=1}^{n} f_{ij}(\boldsymbol{\alpha}_i, \boldsymbol{\beta}_j) \right] - \frac{X^T \mathbf{c} \cdot (1:n)}{\lambda n^2} + \frac{X^T X j \boldsymbol{\alpha}_j \cdot (1:n)}{\lambda n}$$

3. The delayed stochastic gradient over anchor variable $\hat{\boldsymbol{\alpha}}$ is

$$g(\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\beta}}) = \nabla \left[\frac{1}{n} \sum_{i=1}^{n} f_{ij}(\hat{\boldsymbol{\alpha}}_i, \hat{\boldsymbol{\beta}}_j) \right] - \frac{X^T \mathbf{c} \cdot (1:n)}{\lambda n^2} + \frac{X^T X j \hat{\boldsymbol{\alpha}}_j \cdot (1:n)}{\lambda n}$$

Appendix A (Continued)

Note $\mathbb{E}[g] = G$. So the euclidean SVRG update in each iteration is

$$\boldsymbol{\alpha}^{t+1} = \boldsymbol{\alpha}^t - \eta\Big[G(\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\beta}}) + g(\boldsymbol{\alpha}^t, \boldsymbol{\beta}^t) - g(\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\beta}})\Big] \tag{A.108}$$

A.8 Stochastic Updates of SVRG on DCG Game

The original problem for DCG game is

$$\max_{\boldsymbol{\theta}} \ \frac{1}{M}\sum_{k=1}^{M}\left[-\frac{\lambda}{2}\left\|\boldsymbol{\theta}\right\|_2^2 + \frac{\boldsymbol{\theta}^T}{n_k}\sum_{i=1}^{n_k}\tilde{y}_i^k\mathbf{x}_i^k + \min_{\boldsymbol{\alpha}^k\in S_\alpha}\max_{\boldsymbol{\beta}^k\ \in S_\beta}\ \sum_{i=1}^{n_k}\sum_{j=1}^{n_k}\frac{i{\boldsymbol{\alpha}_i^k}^T\boldsymbol{\beta}_j^k}{\log_2(j+1)} - \frac{\boldsymbol{\theta}^T}{n_k}X^k\sum_{i=1}^{n_k}i\boldsymbol{\alpha}_i^k\right]. \tag{A.109}$$

Then the optimal $\boldsymbol{\theta}$ admits a closed form solution

$$\boldsymbol{\theta}\Big(\{\boldsymbol{\alpha}^k\}_M\Big) = \frac{1}{M\lambda}\sum_{k=1}^{M}\frac{X^k\tilde{y}^k - X^k\sum_{i=1}^{n_k}i\boldsymbol{\alpha}_i^k}{n_k}. \tag{A.110}$$

Plugging it back and denoting $\mathbf{s}^k = \frac{X^k\tilde{y}^k - X^k\sum_{i=1}^{n_k}i\boldsymbol{\alpha}_i^k}{n_k}$, we arrive at the overall problem in $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ only:

$$\min_{\{\boldsymbol{\alpha}^k\}_M\in S_\alpha}\max_{\{\boldsymbol{\beta}^k\}_M\in S_\beta}\ \frac{1}{M}\sum_{k=1}^{M}\sum_{i=1}^{n_k}\sum_{j=1}^{n_k}\frac{i{\boldsymbol{\alpha}_i^k}^T\boldsymbol{\beta}_j^k}{\log_2(j+1)} + \frac{1}{2\lambda}\left\|\frac{1}{M}\sum_{k=1}^{M}\mathbf{s}^k\right\|^2. \tag{A.111}$$

$$= \min_{\{\boldsymbol{\alpha}^k\}_M\in S_\alpha}\max_{\{\boldsymbol{\beta}^k\}_M\in S_\beta}\ \frac{1}{M}\sum_{k=1}^{M}\psi_k\Big(\{\boldsymbol{\alpha}^k\}_M, \boldsymbol{\beta}^k\Big). \tag{A.112}$$

Appendix A (Continued)

where

$$\psi_k\Big(\{\boldsymbol{\alpha}^k\}_M, \boldsymbol{\beta}^k\Big) = \sum_{i=1}^{n_k}\sum_{j=1}^{n_k} \frac{i{\boldsymbol{\alpha}_i^k}^T\boldsymbol{\beta}_j^k}{\log_2(j+1)} + \frac{{\mathbf{s}^k}^T\sum_{l=1}^{M}\mathbf{s}^l}{2\lambda M}$$

Thus, the gradient over $\boldsymbol{\alpha}$ is

1. The stochastic gradient over $\{\boldsymbol{\alpha}^k\}_M^t$ at iteration t (note here the gradient is a tensor) is

$$\begin{aligned}
&\nabla\psi_k\Big(\{\boldsymbol{\alpha}^k\}_M^t, \boldsymbol{\beta}^k\Big) = \nabla_{\{\boldsymbol{\alpha}^k\}_M^t}\Big[\sum_{i=1}^{n_k}\sum_{j=1}^{n_k} \frac{i{\boldsymbol{\alpha}_i^k}^T\boldsymbol{\beta}_j^k}{\log_2(j+1)} + \frac{{\mathbf{s}^k}^T\sum_{l=1}^{M}\mathbf{s}^l}{2\lambda M}\Big] \\
=&\Big[\frac{\nabla_{\boldsymbol{\alpha}^1}({\mathbf{s}^k}^T\mathbf{s}^1)}{2\lambda M}, \quad \frac{\nabla_{\boldsymbol{\alpha}^2}({\mathbf{s}^k}^T\mathbf{s}^2)}{2\lambda M}, \ldots, \\
&\quad \ldots, \frac{\nabla_{\boldsymbol{\alpha}^k}({\mathbf{s}^k}^T\mathbf{s}^k)}{2\lambda M} + \nabla_{\boldsymbol{\alpha}^k}\Big[\sum_{i=1}^{n_k}\sum_{j=1}^{n_k} \frac{i{\boldsymbol{\alpha}_i^k}^T\boldsymbol{\beta}_j^k}{\log_2(j+1)}\Big] + \frac{\nabla_{\boldsymbol{\alpha}^k}({\mathbf{s}^k}^T\sum_{l=1}^{M}\mathbf{s}^l)}{2\lambda M}, \ldots, \frac{\nabla_{\boldsymbol{\alpha}^M}({\mathbf{s}^k}^T\mathbf{s}^M)}{2\lambda M}\Big]
\end{aligned}$$

2. The delayed stochastic gradient over anchor variable $\hat{\boldsymbol{\alpha}}$ is

$$\begin{aligned}
&\nabla\psi_k\Big(\{\hat{\boldsymbol{\alpha}}^k\}_M, \hat{\boldsymbol{\beta}}^k\Big) \\
=&\Big[\frac{\nabla_{\hat{\boldsymbol{\alpha}}^1}({\hat{\mathbf{s}}^k}^T\hat{\mathbf{s}}^1)}{2\lambda M}, \quad \frac{\nabla_{\hat{\boldsymbol{\alpha}}^2}({\hat{\mathbf{s}}^k}^T\hat{\mathbf{s}}^2)}{2\lambda M}, \ldots, \\
&\quad \ldots, \frac{\nabla_{\hat{\boldsymbol{\alpha}}^k}({\hat{\mathbf{s}}^k}^T\hat{\mathbf{s}}^k)}{2\lambda M} + \nabla_{\hat{\boldsymbol{\alpha}}^k}\Big[\sum_{i=1}^{n_k}\sum_{j=1}^{n_k} \frac{i{\hat{\boldsymbol{\alpha}}_i^k}^T\hat{\boldsymbol{\beta}}_j^k}{\log_2(j+1)}\Big] + \frac{\nabla_{\hat{\boldsymbol{\alpha}}^k}({\hat{\mathbf{s}}^k}^T\sum_{l=1}^{M}\hat{\mathbf{s}}^l)}{2\lambda M}, \ldots, \frac{\nabla_{\hat{\boldsymbol{\alpha}}^M}({\hat{\mathbf{s}}^k}^T\hat{\mathbf{s}}^M)}{2\lambda M}\Big]
\end{aligned}$$

Appendix A (Continued)

3. Full gradient in each epoch is

$$\begin{aligned}\hat{\mu} &= \frac{1}{M}\sum_{k=1}^{M}\nabla\psi_k\Big(\{\hat{\boldsymbol{\alpha}}^k\}_M, \hat{\boldsymbol{\beta}}^k\Big) \\ &= \frac{1}{M}\Bigg\{\nabla_{\hat{\boldsymbol{\alpha}}^k}\Bigg[\sum_{i=1}^{n_k}\sum_{j=1}^{n_k}\frac{i\hat{\boldsymbol{\alpha}}_i^{k^T}\hat{\boldsymbol{\beta}}_j^k}{\log_2(j+1)}\Bigg] + \frac{\nabla_{\hat{\boldsymbol{\alpha}}^k}(\hat{\mathbf{s}}^{k^T}\sum_{l=1}^{M}\hat{\mathbf{s}}^l)}{\lambda M}\Bigg\}_{k=1}^{M} \\ &= \frac{1}{M}\Bigg\{\nabla_{\hat{\boldsymbol{\alpha}}^k}\Bigg[\sum_{i=1}^{n_k}\sum_{j=1}^{n_k}\frac{i\hat{\boldsymbol{\alpha}}_i^{k^T}\hat{\boldsymbol{\beta}}_j^k}{\log_2(j+1)}\Bigg] + \frac{X^{k^T}\sum_{l=1}^{M}\hat{\mathbf{s}}^l}{\lambda M} * \frac{(1:n_k)}{n_k}\Bigg\}_{k=1}^{M} \\ &= \frac{1}{M}\Bigg\{\nabla_{\hat{\boldsymbol{\alpha}}^k}\Bigg[\sum_{i=1}^{n_k}\sum_{j=1}^{n_k}\frac{i\hat{\boldsymbol{\alpha}}_i^{k^T}\hat{\boldsymbol{\beta}}_j^k}{\log_2(j+1)}\Bigg] - X^{k^T}\boldsymbol{\theta}(\hat{\boldsymbol{\alpha}}) * \frac{(1:n_k)}{n_k}\Bigg\}_{k=1}^{M}\end{aligned}$$

The gradient over $\boldsymbol{\beta}$ is

1. The stochastic gradient over $\boldsymbol{\beta}^k$ at iteration t (note here the gradient is a tensor) is

$$\nabla\psi_k\Big(\{\boldsymbol{\alpha}^k\}_M^t, \{\boldsymbol{\beta}^k\}^t\Big) = \Bigg[0, 0, \ldots, \nabla_{\boldsymbol{\beta}^k}\Bigg[\sum_{i=1}^{n_k}\sum_{j=1}^{n_k}\frac{i\boldsymbol{\alpha}_i^{k^T}\boldsymbol{\beta}_j^k}{\log_2(j+1)}\Bigg], \ldots, 0\Bigg]$$

2. The delayed stochastic gradient over anchor variable $\hat{\boldsymbol{\beta}}$ is

$$\nabla\psi_k\Big(\{\hat{\boldsymbol{\alpha}}^k\}_M, \{\hat{\boldsymbol{\beta}}^k\}^t\Big) = \Bigg[0, 0, \ldots, \nabla_{\hat{\boldsymbol{\beta}}^k}\Bigg[\sum_{i=1}^{n_k}\sum_{j=1}^{n_k}\frac{i\hat{\boldsymbol{\alpha}}_i^{k^T}\hat{\boldsymbol{\beta}}_j^k}{\log_2(j+1)}\Bigg], \ldots, 0\Bigg]$$

3. Full gradient in each epoch is

$$\hat{\mu} = \frac{1}{M}\Bigg\{\nabla_{\hat{\boldsymbol{\beta}}^k}\Bigg[\sum_{i=1}^{n_k}\sum_{j=1}^{n_k}\frac{i\hat{\boldsymbol{\alpha}}_i^{k^T}\hat{\boldsymbol{\beta}}_j^k}{\log_2(j+1)}\Bigg]\Bigg\}_{k=1}^{M}$$

Appendix A (Continued)

So the euclidean SVRG update each iteration $t+1$ is

$$\begin{bmatrix} \boldsymbol{\alpha}_k^{t+1} \\ \boldsymbol{\beta}_k^{t+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\alpha}_k^{t} \\ \boldsymbol{\beta}_k^{t} \end{bmatrix} - \eta \Big[\hat{\mu} + \nabla \psi_k \Big(\{\boldsymbol{\alpha}^k\}_M^t, \{\boldsymbol{\beta}^k\}^t \Big) - \nabla \psi_k \Big(\{\hat{\boldsymbol{\alpha}}^k\}_M, \{\hat{\boldsymbol{\beta}}^k\}^t \Big) \Big], \forall k \in \{1, \ldots, M\}$$

Appendix A (Continued)

Algorithm 6: Euclidean projection of $\{\boldsymbol{\alpha}_k\}$ on S

$\rho^* = \text{minimize}(\texttt{obj\_rho}, [0, +\infty))$

$[\sim, \sim, \{\mathbf{x}_k\}] = \texttt{obj\_rho}(\rho^*)$.

Return $\{\mathbf{x}_k\}$

Function $[f, g, \{\mathbf{x}_k\}] = \texttt{obj\_rho}(\rho)$

for $k = 1, \ldots, n$ **do**

  $r_k = \text{minimize}(\texttt{@(r)obj\_r}(\texttt{r}, k, \rho), [0, +\infty))$

  $[f_k, \sim, \mathbf{x}_k] = \texttt{obj\_r}(r_k, k, \rho)$

  ▷ $\mathbf{x}_k = \mathbf{x}_k(r_k) = \mathbf{x}_k(\rho)$

$f = \rho - \sum_{k=1}^n f_k$

$g = 1 - \sum_{k=1}^n \mathbf{1}^T \mathbf{x}_k$

end function

Function $[f, g, \mathbf{x}] = \texttt{obj\_r}(r, k, \rho)$

$\mu_{\min} = \rho - \max_s \alpha_{ks}, \;\; \mu_{\max} = \rho - \min_s \alpha_{ks} + \frac{r}{k}$

while *true* **do**

  $\mu = (\mu_{\min} + \mu_{\max})/2$ ▷ bi-section search

  $\mathbf{x} = \text{MED}(\boldsymbol{\alpha}_k + \mu\mathbf{1} - \rho\mathbf{1}, \mathbf{0}, \frac{r}{k}\mathbf{1})$

  if $\mathbf{1}^T\mathbf{x} > r + 10^{-5}$ **then**

    $\mu_{\max} = \mu$

  else if $\mathbf{1}^T\mathbf{x} < r - 10^{-5}$ **then**

    $\mu_{\min} = \mu$

  else

    break

$f = \frac{1}{2}\|\mathbf{x} - \boldsymbol{\alpha}_k\|_2^2 + \rho\mathbf{1}^T\mathbf{x}$ ▷ Now $\mathbf{x} = \mathbf{x}_k(r)$

$g = \mu + \mathbf{y}^T(\frac{r}{k}\mathbf{y} - \boldsymbol{\alpha})/k + (\rho - \mu)\mathbf{1}^T\mathbf{y}/k$ ▷ Now $\mu = \mu_k(r)$

end function

Appendix A (Continued)

Algorithm 7: Entropic projection of $\{\boldsymbol{\alpha}_k\}$ on S

$\rho^* = \text{minimize}(\texttt{obj\_rho}, [0, +\infty))$

$[\sim, \sim, \{\mathbf{x}_k\}] = \texttt{obj\_rho}(\rho^*)$.

Return $\{\mathbf{x}_k\}$

Function $[f, g, \{\mathbf{x}_k\}] = \texttt{obj\_rho}(\rho)$

for $k = 1, \ldots, n$ **do**

  $r_k = \text{minimize}(\texttt{@(r)obj\_r}(\texttt{r}, k, \rho), [0, +\infty))$

  $[f_k, \sim, \mathbf{x}_k] = \texttt{obj\_r}(r_k, k, \rho)$

  ▷ $\mathbf{x}_k = \mathbf{x}_k(r_k) = \mathbf{x}_k(\rho)$

$f = \rho - \sum_{k=1}^n f_k$

$g = 1 - \sum_{k=1}^n \mathbf{1}^T \mathbf{x}_k$

end function

Function $[f, g, \mathbf{x}] = \texttt{obj\_r}(r, k, \rho)$

$\mu_{\min} = -50, \;\; \mu_{\max} = \rho + \log(\frac{r}{k * \min_s \alpha_{ks}})$

while *true* **do**

  $\mu = (\mu_{\min} + \mu_{\max})/2$ ▷ bi-section search

  $\mathbf{x} = \text{MIN}(\boldsymbol{\alpha} \exp(\mu - \rho), \frac{r}{k}\mathbf{1})$

  if $\mathbf{1}^T\mathbf{x} > r + 10^{-5}$ **then**

    $\mu_{\max} = \mu$

  else if $\mathbf{1}^T\mathbf{x} < r - 10^{-5}$ **then**

    $\mu_{\min} = \mu$

  else

    break

$f = \rho \mathbf{1}^T\mathbf{x} + \sum_s Q$ ▷ Now $\mathbf{x} = \mathbf{x}_k(r)$

$g = \mu + \sum_s \frac{y_s}{k} \log \frac{y_s r}{\alpha_s k} + (\rho - \mu)\frac{y_s}{k}$ ▷ Now $\mu = \mu_k(r)$

end function

Appendix B

PROOFS IN CHAPTER 3

B.1 Proof of Lemma 2

Lemma 2 (Generalized from Lemma A.3 of [37]). *Let $(\mathcal{X}, \|\cdot\|)$ be a normed linear space and $f : \mathcal{X} \mapsto \mathbb{R}$ be convex. Then for $\mathbf{x}_0 \in \mathcal{X}, \lambda > 0$,*

$$\sup_{\mathbf{x}' \in \mathcal{X}} \left(f(\mathbf{x}) - \lambda \left\| \mathbf{x} - \mathbf{x}_0 \right\| \right) = \begin{cases} f(\mathbf{x}_0) & \text{if } Lip(f) \leq \lambda, \\ +\infty & \text{otherwise.} \end{cases} \tag{3.7}$$

See proof in Appendix B.1.

Proof. Since f is closed proper convex, $f = f^{**}$ [162, Thm. 3.44, p. 214]. Then, using the bidual form of the norm

$$\forall x \in X : \quad -\gamma \left\| x \right\| = -\gamma \sup_{\|g\|_* \leq 1} \left\langle g, x \right\rangle = \inf_{\|g\|_* \leq \gamma} - \left\langle g, x \right\rangle, \tag{B.1}$$

Appendix B (Continued)

we have

$$\sup_{x\in X}\left(f(x)-\gamma\left\|x-x_0\right\|\right)=\sup_{x\in X}\left(\sup_{x^*\in\operatorname{dom}f^*}\left\langle x^*,x\right\rangle-f^*(x)-\gamma\left\|x-x_0\right\|\right) \tag{B.2}$$

$$\overset{Equation\ B.1}{=}\sup_{x\in X}\sup_{x^*\in\operatorname{dom}f^*}\left(\inf_{\left\|g^*\right\|_*\leq\gamma}\left\langle x^*,x\right\rangle-f^*(x)-\left\langle g^*,x-x_0\right\rangle\right) \tag{B.3}$$

$$=\sup_{x\in X}\sup_{x^*\in\operatorname{dom}f^*}\inf_{\left\|g^*\right\|_*\leq\gamma}\left(\left\langle x^*-g^*,x\right\rangle+\left\langle g^*,x_0\right\rangle-f^*(x^*)\right) \tag{B.4}$$

$$=\inf_{\left\|g^*\right\|_*\leq\gamma}\sup_{x^*\in\operatorname{dom}f^*}\sup_{x\in X}\left(\left\langle x^*-g^*,x\right\rangle+\left\langle g^*,x_0\right\rangle-f^*(x^*)\right), \tag{B.5}$$

where the exchange the infimum and supremum follows from [162, Thm. 1.86, p. 59]. Thus

$$\inf_{\left\|g^*\right\|_*\leq\gamma}\sup_{x^*\in\operatorname{dom}f^*}\sup_{x\in X}\left(\left\langle x^*-g^*,x\right\rangle+\left\langle g^*,x_0\right\rangle-f^*(x^*)\right) \tag{B.6}$$

$$=\sup_{x^*\in\operatorname{dom}f^*}\inf_{\left\|g^*\right\|_*\leq\gamma}\left(\infty(1-[\![x^*=g^*]\!])+\left\langle g^*,x_0\right\rangle-f^*(x^*)\right), \tag{B.7}$$

$$=\sup_{x^*\in\operatorname{dom}f^*}\left(\infty\left(1-[\![\left\|x^*\right\|_*\leq\gamma]\!]\right)+\left\langle x^*,x_0\right\rangle-f^*(x^*)\right) \tag{B.8}$$

$$=\sup_{x^*\in\operatorname{dom}f^*}\left(\left\langle x^*,x_0\right\rangle-f^*(x^*)\right)+\infty[\![\exists g^*\in\operatorname{dom}f^*\left\|g^*\right\|_*>\gamma]\!] \tag{B.9}$$

$$=f(x_0)+\infty[\![Lip(f)>\gamma]\!], \tag{B.10}$$

as claimed. □

Appendix C

ANALYSIS IN CHAPTER 4

C.1 Proof of Sample Complexity for Product Kernel

We now prove Theorem 6 by using the Assumptions 2 and 3. We will assume that $\mathcal{X} = \mathcal{X}_0^d$ and $k(x, y) = \prod_i k_0(x_i, y_i)$.

Proof of Theorem 6. Let $\epsilon' := (1 + 2\sqrt{m}M_\epsilon)\,\epsilon$. Since $\left\langle g_1^a, g_1^b \right\rangle_{\mathcal{H}} = \left\langle h_1^a, h_1^b \right\rangle_{\mathcal{H}_0} \prod_{j=2}^d k_0(x_j^a, x_j^b)$ and $\left|k_0(x_j^a, x_j^b)\right| \leq 1$, it suffices to show that for all $a, b \in [l]$,

$$\left| \left\langle h_1^a, h_1^b \right\rangle_{\mathcal{H}_0} - (h_1^a)^\top Z (Z^\top Z)^{-1} Z^\top h_1^b \right| \leq 3\epsilon'. \tag{C.1}$$

Towards this end, it is sufficient to show that for any $h(\cdot) = \theta_x \partial^{0,1} k_0(x, \cdot) + \theta_y \partial^{0,1} k_0(y, \cdot)$ where $x, y \in \mathcal{X}_0$ and $|\theta_x| + |\theta_y| \leq 1$, we have

$$\left| h^\top Z (Z^\top Z)^{-1} Z^\top h - \|h\|_{\mathcal{H}_0}^2 \right| \leq \epsilon'. \tag{C.2}$$

Appendix C (Continued)

This is because, if so, then

$$\left|\left\langle h_1^a, h_1^b \right\rangle_{\mathcal{H}_0} - (h_1^a)^\top Z(Z^\top Z)^{-1} Z^\top h_1^b\right| \tag{C.3}$$

$$= \left|\frac{1}{2}\left(\left\|h_1^a + h_1^b\right\|_{\mathcal{H}_0}^2 - \|h_1^a\|_{\mathcal{H}_0}^2 - \left\|h_1^b\right\|_{\mathcal{H}_0}^2\right) - \frac{1}{2}\left[(h_1^a + h_1^b)^\top Z(Z^\top Z)^{-1} Z^\top (h_1^a + h_1^b)\right.\right. \tag{C.4}$$

$$\left.\left. -(h_1^a)^\top Z(Z^\top Z)^{-1} Z^\top h_1^a - (h_1^b)^\top Z(Z^\top Z)^{-1} Z^\top h_1^b\right]\right| \tag{C.5}$$

$$\leq \frac{1}{2}\left(4\epsilon' + \epsilon' + \epsilon'\right) = 3\epsilon'. \tag{C.6}$$

The rest of the proof is devoted to Equation C.2. Since $n \geq m$, the SVD of $\Lambda_m^{-1/2}\Phi_m^\top Z$ can be written as $U\Sigma V^\top$, where $UU^\top = U^\top U = V^\top V = I_m$ (m-by-m identity matrix), and $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_m)$. Define

$$\boldsymbol{\alpha} = n^{-1/2} V U^\top \Lambda_m^{-1/2} \Phi_m^\top h. \tag{C.7}$$

Consider the optimization problem $o(\boldsymbol{\alpha}) := \frac{1}{2}\|Z\boldsymbol{\alpha} - h\|_{\mathcal{H}_0}^2$. It is easy to see that its minimal objective value is $o^* := \frac{1}{2}\|h\|_{\mathcal{H}_0}^2 - \frac{1}{2}h^\top Z(Z^\top Z)^{-1}Z^\top h$. So

$$0 \leq 2o^* = \|h\|_{\mathcal{H}_0}^2 - h^\top Z(Z^\top Z)^{-1} Z^\top h \leq 2o(\boldsymbol{\alpha}). \tag{C.8}$$

Appendix C (Continued)

Therefore to prove Equation C.2, it suffices to bound $o(\boldsymbol{\alpha}) = \|Z\boldsymbol{\alpha} - h\|_{\mathcal{H}_0}$. Since $\sqrt{n}\Phi_m\Lambda^{1/2}UV^\top\boldsymbol{\alpha} = \Phi_m\Phi_m^\top h$, we can decompose $\|Z\boldsymbol{\alpha} - h\|_{\mathcal{H}_0}$ by

$$\|Z\boldsymbol{\alpha} - h\|_{\mathcal{H}_0} \leq \left\|(Z - \Phi_m\Phi_m^\top Z)\boldsymbol{\alpha}\right\|_{\mathcal{H}_0} + \left\|(\Phi_m\Phi_m^\top Z - \sqrt{n}\Phi_m\Lambda_m^{1/2}UV^\top)\boldsymbol{\alpha}\right\|_{\mathcal{H}_0} \tag{C.9}$$

$$+ \left\|\Phi_m\Phi_m^\top h - h\right\|_{\mathcal{H}_0}. \tag{C.10}$$

The last term $\left\|\Phi_m\Phi_m^\top h - h\right\|_{\mathcal{H}_0}$ is clearly below ϵ because by Assumption 2 and $m = N_\epsilon$

$$\begin{aligned}
&\left\|\Phi_m\Phi_m^\top h - h\right\|_{\mathcal{H}_0} \\
\leq& |\theta_x| \left\|\Phi_m\Phi_m^\top \partial^{0,1}k_0(x,\cdot) - \partial^{0,1}k_0(x,\cdot)\right\|_{\mathcal{H}_0} + |\theta_y| \left\|\Phi_m\Phi_m^\top \partial^{0,1}k_0(y,\cdot) - \partial^{0,1}k_0(y,\cdot)\right\|_{\mathcal{H}_0} \\
\leq& (|\theta_x| + |\theta_y|)\epsilon \leq \epsilon.
\end{aligned} \tag{C.11}$$

We will next bound the first two terms on the right-hand side of Equation C.9.

(1) By Assumption 2, $\left\|k_0(w^s,\cdot) - \Phi_m\Phi_m^\top k_0(w^s,\cdot)\right\|_{\mathcal{H}_0} \leq \epsilon$, hence $\left\|(Z - \Phi_m\Phi_m^\top Z)\boldsymbol{\alpha}\right\|_{\mathcal{H}_0} \leq \epsilon\sqrt{n}\,\|\boldsymbol{\alpha}\|_2$. To bound $\|\boldsymbol{\alpha}\|_2$, note all singular values of $VU^\top$ are 1, and so Assumption 3 implies that for all $i \in [m]$,

$$\left|\lambda_j^{-1/2}\langle\varphi_j, h\rangle_{\mathcal{H}_0}\right| = \left|\langle e_j, h\rangle_{\mathcal{H}_0}\right| = \left|\left\langle e_j, \theta_x\partial^{0,1}k_0(x,\cdot) + \theta_y\partial^{0,1}k_0(y,\cdot)\right\rangle_{\mathcal{H}_0}\right| \tag{C.12}$$

$$\leq \sup_{x\in\mathcal{X}}\left|\left\langle e_j, \partial^{0,1}k(x,\cdot)\right\rangle_{\mathcal{H}_0}\right| \leq M_\epsilon. \tag{C.13}$$

Appendix C (Continued)

As a result,

$$\left\|(Z-\Phi_m\Phi_m^\top Z)\boldsymbol{\alpha}_j\right\|_{\mathcal{H}_0} \le \epsilon n^{1/2}\cdot n^{-1/2}\left\|\Lambda_m^{-1/2}\Phi_m^\top h\right\| \le \epsilon\sqrt{m}M_\epsilon. \tag{C.14}$$

(2) We first consider the concentration of the matrix $R := \frac{1}{n}\Lambda_m^{-1/2}\Phi_m^\top ZZ^\top\Phi_m\Lambda_m^{-1/2} \in \mathbb{R}^{m\times m}$. Clearly,

$$\mathop{\mathbb{E}}_{\{w_s\}}[R_{ij}] = \mathop{\mathbb{E}}_{\{w_s\}}\left[\frac{1}{n}\sum_{s=1}^n e_i(w_s)e_j(w_s)\right] = \int e_i(x)e_j(x)\,\mathrm{d}\nu(x) = \delta_{ij}. \tag{C.15}$$

By matrix Bernstein theorem [163, Theorem 1.6.2], we have $\Pr\left(\|R-I_m\|_{sp}\le\epsilon\right)\ge 1-\delta$ when $n\ge O(.)$. This is because $\|(e_1(x),\ldots,e_m(x))\|^2\le mQ_\epsilon^2$, $\left\|\mathbb{E}_{\{w_s\}}[RR^\top]\right\|_{sp}\le mQ_\epsilon^2/n$, and

$$\Pr\left(\|R-I_m\|_{sp}\le\epsilon\right)\ge 1-2m\exp\left(\frac{-\epsilon^2}{\frac{mQ_\epsilon^2}{n}\left(1+\frac{2}{3}\epsilon\right)}\right)\ge 1-2m\exp\left(\frac{-\epsilon^2}{\frac{5mQ_\epsilon^2}{3n}}\right)\ge 1-\delta, \tag{C.16}$$

where the last step is by the definition of n. Since $R=\frac{1}{n}U\Sigma^2U^\top$, this means with probability $1-\delta$, $\left\|\frac{1}{n}U\Sigma^2U^\top - I_m\right\|_{sp}\le\epsilon$. So for all $i\in[m]$,

$$\left|\frac{1}{n}\sigma_i^2-1\right|\le\epsilon \qquad \text{which implies} \qquad \left|\frac{1}{\sqrt{n}}\sigma_i-1\right| < \epsilon\left|\frac{1}{\sqrt{n}}\sigma_i+1\right|^{-1}\le\epsilon. \tag{C.17}$$

Appendix C (Continued)

Moreover, $\lambda_1 \leq 1$ since $k_0(x,x) = 1$. It then follows that

$$\left\|(\Phi_m\Phi_m^\top Z - \sqrt{n}\Phi_m\Lambda_m^{1/2}UV^\top)\boldsymbol{\alpha}\right\|_{\mathcal{H}_0} \tag{C.18}$$

$$= \left\|\Phi_m\Lambda_m^{1/2}U\Sigma V^\top \frac{1}{\sqrt{n}} VU^\top\Lambda_m^{-1/2}\Phi_m^\top h - \sqrt{n}\Phi_m\Lambda_m^{1/2}UV^\top\frac{1}{\sqrt{n}}VU^\top\Lambda_m^{-1/2}\Phi_m^\top h\right\|_{\mathcal{H}_0} \tag{C.19}$$

$$= \left\|\Lambda_m^{1/2}U\left(\frac{1}{\sqrt{n}}\Sigma - I_m\right)U^\top\Lambda_m^{-1/2}\Phi_m^\top h\right\|_2 \qquad (\text{because } \Phi_m^\top\Phi_m = I_m) \tag{C.20}$$

$$\leq \sqrt{\lambda_1}\max_{i\in[m]}\left|\frac{1}{\sqrt{n}}\sigma_i - 1\right|\left\|\Lambda_m^{-1/2}\Phi_m^\top h\right\|_2 \tag{C.21}$$

$$\leq \epsilon\sqrt{m}M_\epsilon \qquad (\text{by } \textit{Equation C.17}, \textit{Equation C.12}, \text{ and } \lambda_1 \leq 1). \tag{C.22}$$

Combining (1) and (2), we arrive at the desired bound in Equation 4.32. □

C.2 Proof of Theorem 7

Theorem 7. *Suppose the periodic kernel with period v has eigenvalues λ_j that satisfies*

$$\lambda_j(1+j)^2\max(1,j^2)(1+\delta(j\geq 1)) \leq c_6\cdot c_4^{-j}, \quad \textit{for all } j \geq 0, \tag{4.46}$$

where $c_4 > 1$ and $c_6 > 0$ are universal constants. Then Assumption 2 holds with

$$N_\epsilon = 1 + 2\lfloor n_\epsilon \rfloor, \quad \textit{where} \quad n_\epsilon := \log_{c_4}\left(\frac{2.1c_6}{\epsilon^2}\max\left(1,\frac{v^2}{4\pi^2}\right)\right). \tag{4.47}$$

In addition, Assumption 3 holds with $Q_\epsilon = \sqrt{2}$ and $M_\epsilon = \frac{2\sqrt{2}\pi}{v}\lfloor n_\epsilon\rfloor = \frac{\sqrt{2}\pi}{v}(N_\epsilon - 1)$.

Appendix C (Continued)

Proof of Theorem 7. First we show that $h(x) := \partial^{0,1} k_0(x_0, x)$ is in $\mathcal{H}_0$ for all $x_0 \in \mathcal{X}_0$. Since $k_0(x_0, x) = \sum_{j\in\mathbb{Z}} \lambda_j e_j(x_0) e_j(x)$, we derive

$$h(x) = \sum_{j\in\mathbb{Z}} \lambda_j e_j(x_0) \partial^1 e_j(x) = \sum_{j\in\mathbb{Z}} \lambda_j e_j(x_0) \left(-j\omega_0 e_{-j}(x)\right) = \omega_0 \sum_{j\in\mathbb{Z}} \lambda_j j e_{-j}(x_0) e_j(x). \quad \text{(C.23)}$$

$h(x)$ is in $\mathcal{H}$ if the sequence $\lambda_j j e_{-j}(x_0)/\sqrt{\lambda_j}$ is square summable. This can be easily seen by Equation 4.46:

$$\omega_0^{-2} \|h\|_{\mathcal{H}_0}^2 = \sum_j \lambda_j j^2 e_{-j}^2(x_0) = \sum_{j\in\mathbb{Z}} \lambda_j j^2 e_{-j}^2(x_0) \quad \text{(C.24)}$$

$$= \sum_{j\in\mathbb{Z}} \lambda_j j^2 e_{-j}^2(x_0) = \lambda_0 + 2\sum_{j\geq 1} j^2 \lambda_j \leq \frac{2c_4 c_5}{c_4 - 1}. \quad \text{(C.25)}$$

Finally to derive N_ϵ, we reuse the orthonormal decomposition of $h(x)$ in Equation C.23. For a given set of j values A where $A \subseteq \mathbb{Z}$, we denote as Φ_A the "matrix" whose columns enumerate the φ_j over $j \in A$. Let us choose

$$A := \left\{ j : \lambda_j \max(1, j^2)(1 + j^2)(1 + \delta(j \geq 1)) \geq \min(1, w_0^{-2}) \frac{\epsilon^2}{2.1} \right\}. \quad \text{(C.26)}$$

Appendix C (Continued)

If $j \in A$, then $-j \in A$. Letting $\mathbb{N}_0 = \{0, 1, 2, \ldots\}$, we note $\sum_{j\in\mathbb{N}_0} \frac{1}{1+j^2} \leq 2.1$. So

$$\left\|h - \Phi_A \Phi_A^\top h\right\|_{\mathcal{H}_0}^2 = w_0^2 \sum_{j\in\mathbb{Z}\setminus A} \lambda_j j^2 e_{-j}^2(x_0) \tag{C.27}$$

$$= w_0^2 \sum_{j\in\mathbb{N}_0\setminus A} \lambda_j j^2 \left[(e_j^2(x) + e_{-j}^2(x))\delta(j \geq 1) + \delta(j = 0)\right] \tag{C.28}$$

$$= w_0^2 \sum_{j\in\mathbb{N}_0\setminus A} \lambda_j j^2 (1 + \delta(j \geq 1)) \tag{C.29}$$

$$= w_0^2 \sum_{j\in\mathbb{N}_0\setminus A} \left\{\lambda_j j^2 (1 + j^2)(1 + \delta(j \geq 1)) \frac{1}{1 + j^2}\right\} \tag{C.30}$$

$$\leq \frac{\epsilon^2}{2.1} \sum_{j\in\mathbb{N}_0} \frac{1}{1 + j^2} = \frac{\epsilon^2}{2.1} \sum_{j\in\mathbb{N}_0} \frac{1}{1 + j^2} \leq \epsilon^2. \tag{C.31}$$

Similarly, we can bound $\left\|k_0(x_0, \cdot) - \Phi_A \Phi_A^\top k_0(x_0, \cdot)\right\|_{\mathcal{H}_0}$ by

$$\left\|k_0(x_0, \cdot) - \Phi_A \Phi_A^\top k_0(x_0, \cdot)\right\|_{\mathcal{H}_0}^2 \tag{C.32}$$

$$= \sum_{j\in\mathbb{Z}\setminus A} \lambda_j e_j^2(x_0) \leq \sum_{j\in\mathbb{Z}\setminus A} \lambda_j \max(1, j^2) e_j^2(x_0) \tag{C.33}$$

$$= \sum_{j\in\mathbb{N}_0\setminus A} \lambda_{\boldsymbol{\alpha}} \max(1, j^2)[\left(e_j^2(x) + e_{-j}^2(x)\right) \delta(j \geq 1) + \delta(j = 0)] \tag{C.34}$$

$$= \sum_{j\in\mathbb{N}_0\setminus A} \left\{\lambda_j \max(1, j^2)(1 + j^2)(1 + \delta(j \geq 1)) \frac{1}{1 + j^2}\right\} \tag{C.35}$$

$$\leq \frac{1}{2.1}\epsilon^2 \sum_{j\in\mathbb{N}_0} \frac{1}{1 + j^2} \leq \epsilon^2. \tag{C.36}$$

Appendix C (Continued)

To upper bound the cardinality of A, we consider the conditions for $j \notin A$. Thanks to the conditions in Equation 4.46, we know that any j satisfying the following relationship cannot be in A:

$$c_6 \cdot c_4^{-|j|} < \min(1, w_0^{-2}) \frac{\epsilon^2}{2.1} \qquad \Leftrightarrow \qquad c_4^{-|j|} < \frac{1}{2.1 \cdot c_6} \min\left(1, \frac{4\pi^2}{v^2}\right) \epsilon^2. \tag{C.37}$$

So $A \subseteq \{j : |j| \leq n_\epsilon\}$, which yields the conclusion Equation 4.47. Finally $Q_\epsilon \leq \sqrt{2}$, and to bound M_ϵ, we simply reuse Equation C.23. For any j with $|j| \leq n_\epsilon$,

$$\left|\langle h, e_j \rangle_{\mathcal{H}}\right| \leq \omega_0 \left|j e_{-j}(x_0)\right| \leq \frac{2\pi}{v}\sqrt{2}\,\lfloor n_\epsilon \rfloor = \frac{\sqrt{2}\pi}{v}(N_\epsilon - 1). \qquad \square$$

C.3 Analysis of the Eigenvalues of Inverse Kernel RKHS

It is quite straightforward to give an explicit characterization of the functions in $\mathcal{H}$. The Taylor expansion of z^{-1} at $z = 2$ is $\frac{1}{2}\sum_{i=0}^{\infty}(-\frac{1}{2})^i x^i$. Using the standard multi-index notation with $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_d) \in (\mathbb{N} \cup \{0\})^d$, $|\boldsymbol{\alpha}| = \sum_{i=1}^d \alpha_i$, and $\mathbf{x}^{\boldsymbol{\alpha}} = x_1^{\alpha_1} \ldots x_d^{\alpha_d}$, we derive

$$k(\mathbf{x}, \mathbf{y}) = \frac{1}{2 - \mathbf{x}^\top \mathbf{y}} = \frac{1}{2}\sum_{k=0}^{\infty}\left(-\frac{1}{2}\right)^k (-\mathbf{x}^\top \mathbf{y})^k = \sum_{k=0}^{\infty} 2^{-k-1} \sum_{\boldsymbol{\alpha}:|\boldsymbol{\alpha}|=k} C_{\boldsymbol{\alpha}}^k \mathbf{x}^{\boldsymbol{\alpha}} \mathbf{y}^{\boldsymbol{\alpha}} \tag{C.38}$$

$$= \sum_{\boldsymbol{\alpha}} 2^{-|\boldsymbol{\alpha}|-1} C_{\boldsymbol{\alpha}}^{|\boldsymbol{\alpha}|} \mathbf{x}^{\boldsymbol{\alpha}} \mathbf{y}^{\boldsymbol{\alpha}}, \tag{C.39}$$

where $C_{\boldsymbol{\alpha}}^k = \frac{k!}{\prod_{i=1}^d \alpha_i!}$. So we can read off the feature mapping for $\mathbf{x}$ as

$$\phi(\mathbf{x}) = \{w_{\boldsymbol{\alpha}} \mathbf{x}^{\boldsymbol{\alpha}} : \boldsymbol{\alpha}\}, \quad \text{where} \quad w_{\boldsymbol{\alpha}} = 2^{-\frac{1}{2}(|\boldsymbol{\alpha}|+1)} C_{\boldsymbol{\alpha}}^{|\boldsymbol{\alpha}|}, \tag{C.40}$$

Appendix C (Continued)

and the functions in $\mathcal{H}$ are

$$\mathcal{H} = \left\{ f = \sum_{\boldsymbol{\alpha}} \theta_{\boldsymbol{\alpha}} w_{\boldsymbol{\alpha}} \mathbf{x}^{\boldsymbol{\alpha}} : \|\boldsymbol{\theta}\|_{\ell_2} < \infty \right\}. \tag{C.41}$$

Note this is just an intuitive "derivation" while a rigorous proof for Equation C.41 can be constructed in analogy to that of Theorem 1 in [164].

C.3.1 Background of eigenvalues of a kernel

We now use Equation C.41 to find the eigenvalues of inverse kernel.

Now specializing to our inverse kernel case, let us endow a uniform distribution over the unit ball B: $p(x) = V_d^{-1}$ where $V_d = \pi^{d/2}\Gamma(\frac{d}{2}+1)^{-1}$ is the volume of B, with Γ being the Gamma function. Then λ is an eigenvalue of the kernel if there exists $f = \sum_{\boldsymbol{\alpha}} \theta_{\boldsymbol{\alpha}} w_{\boldsymbol{\alpha}} \mathbf{x}^{\boldsymbol{\alpha}}$ such that $\int_{\mathbf{y}\in B} k(\mathbf{x},\mathbf{y}) p(\mathbf{y}) f(\mathbf{y})\, \mathrm{d}\mathbf{y} = \lambda f(\mathbf{x})$. This translates to

$$V_d^{-1} \int_{\mathbf{y}\in B} \sum_{\boldsymbol{\alpha}} w_{\boldsymbol{\alpha}}^2 \mathbf{x}^{\boldsymbol{\alpha}} \mathbf{y}^{\boldsymbol{\alpha}} \sum_{\boldsymbol{\beta}} \theta_{\boldsymbol{\beta}} w_{\boldsymbol{\beta}} \mathbf{y}^{\boldsymbol{\beta}}\, \mathrm{d}\mathbf{y} = \lambda \sum_{\boldsymbol{\alpha}} \theta_{\boldsymbol{\alpha}} w_{\boldsymbol{\alpha}} \mathbf{x}^{\boldsymbol{\alpha}}, \qquad \forall\, \mathbf{x} \in B. \tag{C.42}$$

Since B is an open set, that means

$$w_{\boldsymbol{\alpha}} \sum_{\boldsymbol{\beta}} w_{\boldsymbol{\beta}} q_{\boldsymbol{\alpha}+\boldsymbol{\beta}} \theta_{\boldsymbol{\beta}} = \lambda \theta_{\boldsymbol{\alpha}}, \qquad \forall\, \boldsymbol{\alpha}, \tag{C.43}$$

Appendix C (Continued)

where

$$q_{\boldsymbol{\alpha}} = V_d^{-1} \int_{\mathbf{y}\in B} \mathbf{y}^{\boldsymbol{\alpha}}\, \mathrm{d}\mathbf{y} = \begin{cases} \frac{2\prod_{i=1}^{d}\Gamma\left(\frac{1}{2}\alpha_i+\frac{1}{2}\right)}{V_d\cdot(|\boldsymbol{\alpha}|+d)\cdot\Gamma\left(\frac{1}{2}|\boldsymbol{\alpha}|+\frac{d}{2}\right)} & \text{if all } \alpha_i \text{ are even} \\ 0 & \text{otherwise} \end{cases}. \tag{C.44}$$

In other words, λ is the eigenvalue of the infinite dimensional matrix $Q = [w_{\boldsymbol{\alpha}} w_{\boldsymbol{\beta}} q_{\boldsymbol{\alpha}+\boldsymbol{\beta}}]_{\boldsymbol{\alpha},\boldsymbol{\beta}}$,

C.3.2 Bounding the eigenvalues

To bound the eigenvalues of Q, we resort to the majorization results in matrix analysis. Since k is a PSD kernel, all its eigenvalues are nonnegative, and suppose they are sorted decreasingly as $\lambda_1 \geq \lambda_2 \geq \ldots$. Let the row corresponding to $\boldsymbol{\alpha}$ have ℓ_2 norm $r_{\boldsymbol{\alpha}}$, and let them be sorted as $r_{[1]} \geq r_{[2]} \geq \ldots$. Then by [165, 166], we have

$$\prod_{i=1}^{n} \lambda_i \;\leq\; \prod_{i=1}^{n} r_{[i]}, \quad \forall\; n \geq 1. \tag{C.45}$$

So our strategy is to bound $r_{\boldsymbol{\alpha}}$ first. To start with, we decompose $q_{\boldsymbol{\alpha}+\boldsymbol{\beta}}$ into $q_{\boldsymbol{\alpha}}$ and $q_{\boldsymbol{\beta}}$ via Cauchy-Schwartz:

$$q_{\boldsymbol{\alpha}+\boldsymbol{\beta}}^2 = V_d^{-2} \left(\int_{\mathbf{y}\in B} \mathbf{y}^{\boldsymbol{\alpha}+\boldsymbol{\beta}}\, \mathrm{d}\mathbf{y} \right)^2 \leq V_d^{-2} \int_{\mathbf{y}\in B} \mathbf{y}^{2\boldsymbol{\alpha}}\, \mathrm{d}\mathbf{y} \cdot \int_{\mathbf{y}\in B} \mathbf{y}^{2\boldsymbol{\beta}}\, \mathrm{d}\mathbf{y} = q_{2\boldsymbol{\alpha}} q_{2\boldsymbol{\beta}}. \tag{C.46}$$

To simplify notation, we consider without loss of generality that d is an even number, and denote the integer $b := d/2$. Now $V_d = \pi^b / b!$. Noting that there are $\begin{pmatrix} k+d-1 \\ k \end{pmatrix}$ values of $\boldsymbol{\beta}$

Appendix C (Continued)

such that $|\boldsymbol{\beta}| = k$, we can proceed by (fix below by changing $\begin{pmatrix} k+d \\ k \end{pmatrix}$ into $\begin{pmatrix} k+d-1 \\ k \end{pmatrix}$, or no need because the former upper bounds the latter)

$$r_{\boldsymbol{\alpha}}^2 = w_{\boldsymbol{\alpha}}^2 \sum_{\boldsymbol{\beta}} w_{\boldsymbol{\beta}}^2 q_{\boldsymbol{\alpha}+\boldsymbol{\beta}}^2 \leq w_{\boldsymbol{\alpha}}^2 q_{2\boldsymbol{\alpha}} \sum_{\boldsymbol{\beta}} w_{\boldsymbol{\beta}}^2 q_{2\boldsymbol{\beta}} = w_{\boldsymbol{\alpha}}^2 q_{2\boldsymbol{\alpha}} \sum_{k=0}^{\infty} 2^{-k-1} \sum_{\boldsymbol{\beta}:|\boldsymbol{\beta}|=k} C_{\boldsymbol{\beta}}^k q_{2\boldsymbol{\beta}} \tag{C.47}$$

$$\leq w_{\boldsymbol{\alpha}}^2 q_{2\boldsymbol{\alpha}} \sum_{k=0}^{\infty} 2^{-k-1} \begin{pmatrix} k+d \\ d \end{pmatrix} \max_{|\boldsymbol{\beta}|=k} C_{\boldsymbol{\beta}}^k q_{2\boldsymbol{\beta}} \tag{C.48}$$

$$= w_{\boldsymbol{\alpha}}^2 q_{2\boldsymbol{\alpha}} \sum_{k=0}^{\infty} 2^{-k-1} \begin{pmatrix} k+d \\ d \end{pmatrix} \max_{|\boldsymbol{\beta}|=k} \frac{k!}{\prod_{i=1}^d \beta_i!} \cdot \frac{2\prod_{i=1}^d \Gamma(\beta_i + \frac{1}{2})}{V_d \cdot (2k+d) \cdot \Gamma(k + \frac{d}{2})} \tag{C.49}$$

$$= w_{\boldsymbol{\alpha}}^2 q_{2\boldsymbol{\alpha}} V_d^{-1} \sum_{k=0}^{\infty} 2^{-k} \begin{pmatrix} k+d \\ d \end{pmatrix} \frac{k!}{(2k+d)\Gamma(k+\frac{d}{2})} \cdot \max_{|\boldsymbol{\beta}|=k} \prod_{i=1}^d \frac{\Gamma(\beta_i + \frac{1}{2})}{\beta_i!} \tag{C.50}$$

$$< w_{\boldsymbol{\alpha}}^2 q_{2\boldsymbol{\alpha}} \cdot \frac{b!}{\pi^b d!} \cdot \sum_{k=0}^{\infty} 2^{-k-1} \frac{(k+d)!}{(k+b)!} \qquad (\text{since } \Gamma(\beta_i + \tfrac{1}{2}) < \Gamma(\beta_i + 1) = \beta_i!). \tag{C.51}$$

The summation over k can be bounded by

$$\sum_{k=0}^{\infty} 2^{-k-1} \frac{(k+d)!}{(k+b)!} = \frac{1}{2} b! \left(2^d + \begin{pmatrix} d \\ b \end{pmatrix} \right) \leq \frac{1}{2} \left(b! 2^d + 2^b \right) \leq b! 2^d, \tag{C.52}$$

Appendix C (Continued)

where the first equality used the identity $\sum_{k=1}^{\infty} 2^{-k} \begin{pmatrix} d+k \\ b \end{pmatrix} = 2^d$. Letting $l := |\boldsymbol{\alpha}|$, we can continue by

$$r_{\boldsymbol{\alpha}}^2 < w_{\boldsymbol{\alpha}}^2 q_{2\boldsymbol{\alpha}} \cdot \frac{b!}{\pi^b d!} b! 2^d = 2^{-l-1} \frac{l!}{\prod_{i=1}^d \alpha_i!} \frac{2\prod_{i=1}^d \Gamma\left(\alpha_i + \frac{1}{2}\right)}{V_d \cdot (2l+d) \cdot \Gamma(l+b)} \frac{(b!)^2 2^d}{\pi^b d!} \tag{C.53}$$

$$\leq 2^{-l+d} \pi^{-2b} \frac{l!(b!)^3}{d!(l+b-1)!(2l+d)} \qquad (\text{since } \Gamma(\alpha_i + \tfrac{1}{2}) < \Gamma(\alpha_i + 1) = \alpha_i!) \tag{C.54}$$

$$\leq 2^{-l+b-1} \pi^{-2b} \begin{pmatrix} l+b \\ l \end{pmatrix}^{-1} \qquad (\text{since } \frac{(b!)^2}{d!} \leq 2^{-b}). \tag{C.55}$$

This bound depends on $\boldsymbol{\alpha}$, not directly on $\boldsymbol{\alpha}$. Letting $n_l = \begin{pmatrix} l+d-1 \\ l \end{pmatrix}$ and $N_L = \sum_{l=0}^{L} n_l = \begin{pmatrix} d+L \\ L \end{pmatrix}$, it follows that

$$\sum_{l=0}^{L} l n_l = \sum_{l=1}^{L} \frac{l(l+d)!}{d! \cdot l!} = (d+1) \sum_{l=1}^{L} \frac{(l+d)!}{(d+1)!(l-1)!} \tag{C.56}$$

$$= (d+1) \sum_{l=1}^{L} \begin{pmatrix} l+d \\ d+1 \end{pmatrix} = (d+1) \begin{pmatrix} L+d+1 \\ d+2 \end{pmatrix}. \tag{C.57}$$

Appendix C (Continued)

Now we can bound λ_{N_L} by

$$\lambda_{N_L}^{N_L} \leq \prod_{i=1}^{N_L} \lambda_i \leq \prod_{l=0}^{L} \left(2^{-l+b-1} \pi^{-2b} \binom{l+b}{l}^{-1} \right)^{n_l} \tag{C.58}$$

$$\Rightarrow \quad \log \lambda_{N_L} \leq N_L^{-1} \sum_{l=0}^{L} n_l \left(-(l-b+1)\log 2 - 2b \log \pi - \log \binom{l+b}{l} \right) \tag{C.59}$$

$$\leq -N_L^{-1} \cdot \log 2 \cdot \sum_{l=0}^{L} l n_l \qquad (\text{since } \log 2 < 2 \log \pi \text{ as the coefficients of } b) \tag{C.60}$$

$$= -\binom{d+L+1}{d+1}^{-1} \cdot \log 2 \cdot (d+1) \binom{d+L+1}{d+2} \tag{C.61}$$

$$= -\frac{d+1}{d+2} L \log 2 \tag{C.62}$$

$$\approx -L \log 2 \tag{C.63}$$

$$\Rightarrow \quad \lambda_{N_L} \leq 2^{-L}. \tag{C.64}$$

This means that the eigenvalue $\lambda_i \leq \epsilon$ provided that $i \geq N_L$ where $L = \lceil \log_2 \frac{1}{\epsilon} \rceil$. Since $N_L \leq d^{L+1}$, that means it suffices to choose i such that

$$i \geq d^{\lceil \log_2 \frac{1}{\epsilon} \rceil + 1}. \tag{C.65}$$

This is a quasi-polynomial bound. It seems tight because even in Gaussian RBF kernel, the eigenvalues follow the order of $\lambda_{\boldsymbol{\alpha}} = O(c^{-|\boldsymbol{\alpha}|})$ for some $c > 1$ [167, p.A742].

Appendix D

ANALYSIS IN CHAPTER 5

D.1 Proof of Theorem 8.

The proof for the first part follows from [168], which showed that if the constraint set is a k-extendible system, then the greedy algorithm can find a $\frac{1}{k}$-approximate solution. So it suffices to check that $\mathcal{A}_{\circ}^{1+2+3}$ is 2-extendible.

Let T be a finite set and $\mathcal{L}$ be a collection of subsets of T. Then $(T, \mathcal{L})$ is 2-extendible if

- For all $C \subseteq D$, if $D \in \mathcal{L}$ then $C \in \mathcal{L}$;
- Suppose $C \subseteq D \in \mathcal{L}$, and x be such that $x \notin C$ and $C \cup \{x\} \in \mathcal{L}$. Then there exists $Y \subseteq D \backslash C$ such that $|Y| \leq k$ and $D \backslash Y \cup \{x\} \in \mathcal{L}$.

Consider the obvious bijection between $V \in \mathcal{A}_{\circ}^{1+2+3}$ and a graph of n nodes where two different nodes i and j are connected by an undirected edge if, and only if, $(X_{ji} =) X_{ij} = 1$. So let T be the set of all possible (non-self) undirected edges, and $\mathcal{L}$ be the set of undirected graphs with at most δ_g edges and each node has degree at most δ_l. Clearly such $(T, \mathcal{L})$ satisfies the first condition. To check the second condition, let x be an edge (i, j). If $x \in D$, then the condition holds trivially with $Y = \emptyset$. Otherwise, $x \notin D$, hence $x \notin C$. Note that the degree of i and j in the graph C must be strictly less than δ_l, because otherwise $C \cup \{x\}$ would not be in $\mathcal{L}$ (i.e., not valid). If the degree of i in D is δ_l, then we can find an edge $e_1 \in D \backslash C$ that is incident to

Appendix D (Continued)

i, and add e_1 to Y. Similarly if the degree of j in D is δ_l, then we can find an edge $e_2 \in D \backslash C$ that is incident to j, and add e_2 to Y. Clearly $|Y| \leq 2$, $Y \subseteq D \backslash C$, and $D \backslash Y \cup \{x\} \in \mathcal{L}$.

To prove the second part, note by convexity, $F(A) = F^\circ(A) \geq F^\circ(Z_t) + \operatorname{tr}((A - Z_t)^\top \nabla F^\circ(Z_t))$. Therefore

$$\min_{A \in \mathcal{A}} F(A) \geq F^\circ(Z_t) + \operatorname{tr}(R^\top Z_t) - \max_{A \in \mathcal{A}} \operatorname{tr}(R^\top A), \quad \text{where} \quad R = -\nabla F^\circ(Z_t). \tag{D.1}$$

If the PO can be solved exactly, then the right-hand side can be evaluated exactly, leading to a slightly tighter certificate than $F^\circ(Z_t)$. However, if the PO is not tractable, then let V^* be the result returned by the greedy algorithm, and it follows from Theorem 8 that

$$\max_{A \in \mathcal{A}^{1+2+3}} \operatorname{tr}(R^\top A) = \max_{V \in \mathcal{A}_\circ^{1+2+3}} \operatorname{tr}(R^\top ((-2A^{ori} + E) \circ V + A^{ori})) \tag{D.2}$$

$$\leq 2 \operatorname{tr}(J^\top V^*) + \operatorname{tr}(R^\top A^{ori}), \quad \text{where} \quad J = R \circ (-2A^{ori} + E). \tag{D.3}$$

Plugging it into Equation D.1 and we get a feasible certificate, though at a price of 2.

D.2 Representing $\operatorname{co} \mathcal{A}^1 \cap \operatorname{co} \mathcal{A}^2 \cap \mathcal{A}^3$ by linear inequalities.

We first show that

$$\operatorname{co} \mathcal{D} = \{z \in [0,1]^n : \|z - \alpha\|_1 \leq \delta_l\}, \quad \text{where} \quad \mathcal{D} := \{z \in \{0,1\}^n : \|z - \alpha\|_1 \leq \delta_l\}. \tag{D.4}$$

Clearly, the right-hand side subsumes $\mathcal{D}$ and is convex. Therefore it subsumes $\operatorname{co} \mathcal{D}$. To show the converse direction, it suffices to show that for any $\gamma \in \mathbb{R}^n$, $\max_{z \in z \in [0,1]^n : \|z - \alpha\|_1 \leq \delta_l} \gamma^\top z$ can

Appendix D (Continued)

be attained at an integral solution (hence in $\mathcal{D}$). To this end, without loss of generality, suppose $\alpha_1 = \ldots = \alpha_m = 0$ and $\alpha_{m+1} = \ldots = \alpha_n = 1$, where $m \in \{0, 1, \ldots, n\}$. Then $\|z - \alpha\|_1 \leq \delta_l$ can be written as $\sum_{i=1}^{m} z_i - \sum_{i=m+1}^{n} z_i \leq \delta_l - n + m$. So this linear inequality, along with $z_i \leq 1$, can be written as $Pz \leq (1, \ldots, 1, \delta_l - n + m)^\top$, where

$$P := \begin{pmatrix} 1 & & & & & \\ & 1 & & & & \\ & & 1 & & & \\ & & & 1 & & \\ & & & & 1 & \\ & & & & & 1 \\ 1 & \ldots & 1 & -1 & \ldots & -1 \end{pmatrix} \in \mathbb{R}^{(n+1)\times n}. \tag{D.5}$$

Here the last row has m ones, followed by $n - m$ copies of -1. Now we can partition the rows of P into two groups R_1 and R_2, where R_1 consists of the first m rows, and R_2 consists of the remaining $n - m + 1$ rows. Now obviously each column contains at most two nonzero entries. For the first m columns, the two nonzero entries have the same sign, with one belonging to R_1 and the other to R_2. For the last $n - m + 1$ columns, both the 1 and -1 belong to R_2. So P is totally unimodular [169]. Finally noting that $\delta_l - n + m$ is integral, there must be an optimal solution that is integral.

Appendix D (Continued)

Algorithm 8: l and u of the j-th entry of $\hat{A}_{i:}XW$

Initialize $V^l = V^u = \mathbf{0}$ and set $J = (1 - 2A^{ori}_{i:}) \circ (XW)^\top_{:j}$.

Sort the indices $\mathcal{I} := \{1, \ldots, n\}$ in an ascending order of J.

Let $k = 1$

while $J[\mathcal{I}[k]] < 0$ **and** $\mathbf{1}^\top V^l < \delta_l$ **and** $\mathbf{1}^\top V^l < \delta_g$ **do**

  If $\mathcal{I}[k] \neq i$, set $V^l[\mathcal{I}[k]] = 1$

  $k = k + 1$

Return V^l

$l = [(1 - 2A^{ori}_{i:}) \circ (XW)^\top_{:j}]V^l + A^{ori}_{i:}(XW)_{:j} + (XW)_{i,j}$

Let $k = n$

while $J[\mathcal{I}[k]] > 0$ **and** $\mathbf{1}^\top V^u < \delta_l$ **and** $\mathbf{1}^\top V^u < \delta_g$ **do**

  If $\mathcal{I}[k] \neq i$, set $V^u[\mathcal{I}[k]] = 1$

  $k = k - 1$

Return V^u

$u = [(1 - 2A^{ori}_{i:}) \circ (XW)^\top_{:j}]V^u + A^{ori}_{i:}(XW)_{:j} + (XW)_{i,j}$

Appendix D (Continued)

D.3 Lower and upper bound for ReLU approximation

Both l and u of the j-th entry of $\hat{A}_{i:}XW$ can be easily estimated under $\left\|A_{i:} - A^{ori}_{i:}\right\|_1 \leq \delta_l$. Let $V = |A_{i:} - A^{ori}_{i:}|$, so that $V_j = 0$ if $A_{i:}$ makes no change to A^{ori}_{ij}, and $V_j = 1$ otherwise (adding or removing an edge). Then

$$l = \min_{\left\|A_{i:} - A^{ori}_{i:}\right\|_1 \leq \delta_l} A_{i:}(XW)_{:j} + (XW)_{i,j} \tag{D.6}$$

$$= \min_{\mathbf{1}^\top V \leq \delta_l} \frac{(2A^{ori}_{i:} - 1) \circ (-2V^\top + 1) + 1}{2}(XW)_{:j} + (XW)_{i,j} \tag{D.7}$$

$$= \min_{\mathbf{1}^\top V \leq \delta_l} [(1 - 2A^{ori}_{i:}) \circ (XW)^\top_{:j}]V + A^{ori}_{i:}(XW)_{:j} + (XW)_{i,j} \tag{D.8}$$

Now the detailed algorithm can be derived and is presented in Algorithm 8.

D.4 Derivative of Equation 5.28 in α.

Note we can change variable by $\theta = |w - A^{ori}_{i:}|$, so that $\theta_j = 0$ if w makes no change to A^{ori}_{ij}, and $\theta_j = 1$ otherwise (adding or removing an edge). Then $w = \frac{1}{2}[(2A^{ori}_{i:} - 1) \circ (-2\theta + 1) + 1]$. So $\mathbf{1}^\top w = \alpha$ can be translated into a linear constraint on θ, which we denote as $\beta^\top \theta = \eta_\alpha$. Now $w \in \operatorname{co} \mathcal{P}_i$ is equivalent to $\theta_j \in [0, 1]$ and $\mathbf{1}^\top \theta \in [0, \delta_l]$. Write out the Lagrangian for the minimization over w:

$$J(\alpha) := \min_{\theta : \mathbf{1}^\top \theta \leq \delta_l, \beta^\top \theta = \eta_\alpha, \theta_j \in [0,1]} (\alpha + 1)^{-1}\kappa(\theta) - \gamma^\top \theta \tag{D.9}$$

$$= \min_{\theta} \max_{\lambda \geq 0, \mu, \rho_j \geq 0, \xi_j \geq 0} (\alpha + 1)^{-1}\kappa(\theta) - \gamma^\top \theta + \lambda(\mathbf{1}^\top \theta - \delta_l) + \mu(\beta^\top \theta - \eta_\alpha) \tag{D.10}$$

$$+ \sum_j \rho_j(\theta_j - 1) - \sum_j \xi_j \theta_j. \tag{D.11}$$

Appendix D (Continued)

Taking partial derivative with respect to θ_j, we have

$$(\alpha+1)^{-1}\nabla_j\kappa(\theta)-\gamma_j+\lambda+\mu\beta_j+\rho_j-\xi_j=0. \tag{D.12}$$

If $\theta_j \in (0,1)$, then $\rho_j=\xi_j=0$, and

$$(\alpha+1)^{-1}\nabla_j\kappa(\theta)-\gamma_j+\lambda+\mu\beta_j=0. \tag{D.13}$$

So as long as there are two indices j which satisfy $\theta_j \in (0,1)$, we can solve for (λ,μ). If $\mathbf{1}^\top\theta<\delta_l$, we can further simplify by $\lambda=0$. In practice, we can collect all such j and find the least square solution of (λ,μ). With that, we can compute $J'(\alpha)=-\kappa(\theta)(\alpha+1)^{-2}-\mu\frac{\partial}{\partial\alpha}\eta_\alpha$.

D.5 Experiment

Here we provide detailed experiments for all datasets. Although part of the results for Enzymes have been shown in Section 5.6, we will further provide the results when the attack strength is $s=4$ at testing. The observations and conclusions from all datasets are similar to what is presented in Section 5.6. The properties of all datasets are summarized in Table II.

D.5.1 Comparing activation and pooling functions

Table III compares graph classification accuracy under various activations and pooling functions where 30% data were used for training, 20% for validation, and 50% for testing. There is one hidden layer with $d'=64$ hidden nodes. All settings were run for 10 times to obtain mean and standard deviation. The best result of each row is marked in boldface. We notice that the performance of (linear, ReLU) activation, in conjunction with various pooling methods, can

Appendix D (Continued)

TABLE II: Datasets used in experiments.

dataset	#graphs	#labels	# node features	median #node	median #edge
Enzymes	600	6	21	32	120
NCI1	4110	2	37	27	58
PROTEINS	1113	2	4	26	98
MUTAG	188	2	7	17	38

be quite mixed. No combination is uniformly the best. In particular, we considered average pooling (avg), max pooling (max), and attention pooling with

- att_node: the attention weights α were trained as functions of node features only;
- att_topo: the attention weights α were trained also using the graph Laplacian [170].

As a result, it is meaningful and useful to study the robustness certificate and attack for all combinations of activation and pooling.

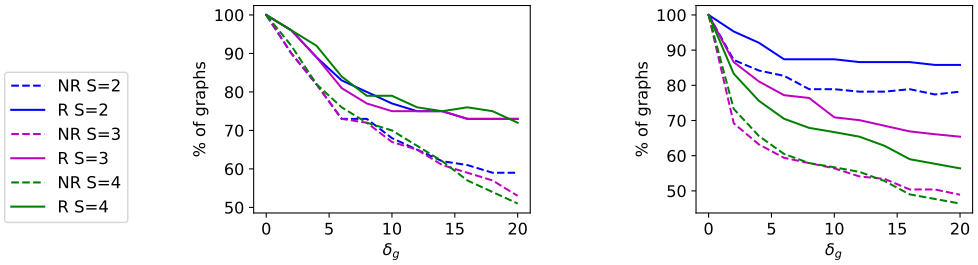
Appendix D (Continued)

dataset	activation	pooling			
		avg	max	att_topo	att_node
Enzymes	ReLU	**31.6 ± .5**	29.8 ± .4	29.5 ± 1.1	19.9 ± 1.3
	Linear	29.1 ± 1.7	**30.3 ± .0**	30.1 ± .8	21.3 ± 4.4
NCI1	ReLU	65.0 ± 0.3	62.5 ± .0	**67.6 ± .2**	63.0 ± .1
	Linear	58.3 ± .0	62.5 ± .3	61.4 ± .0	**63.2 ± .2**
PROTEINS	ReLU	**67.4 ± 1.2**	66.9 ± 1.6	66.0 ± .0	64.9 ± .0
	Linear	64.1 ± 3.3	**69.5 ± .4**	65.5 ± .1	62.9 ± 2.3
MUTAG	ReLU	68.8 ± 1.5	66.1 ± .8	70.2 ± .4	**73.3 ± 3.5**
	Linear	65.3 ± .0	65.7 ± .8	69.4 ± 2.0	**74.1 ± 2.0**

TABLE III: Comparison of graph classification accuracy under various activations and pooling functions.

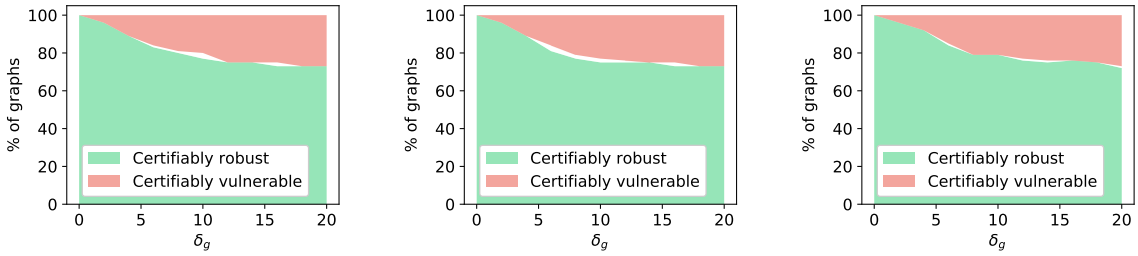
Appendix D (Continued)

D.5.2 More results on Enzymes



(a) Linear activation (b) ReLU activation

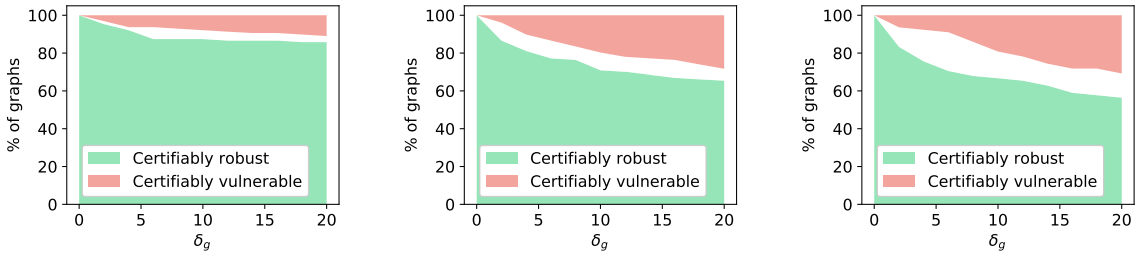
Figure 21: Fraction of graphs certified robust with $s \in \{2, 3, 4\}$, under robust training (**R**) and non-robust training (**NR**). Dataset: Enzymes.



(a) $s = 2$ (b) $s = 3$ (c) $s = 4$

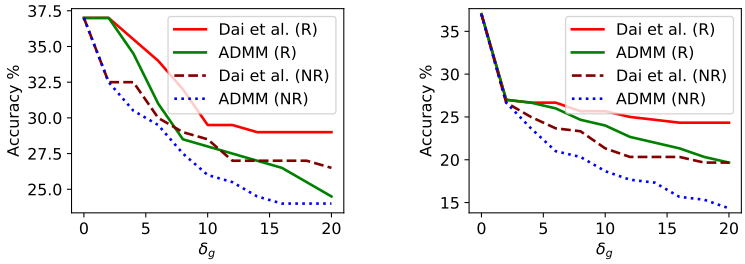
Figure 22: Fraction of graphs that are certified as robust (lower green area) and vulnerable (upper red area, percentage = 100 − y-axis. **Linear activation.** All are under robust training. Dataset: Enzymes.

Appendix D (Continued)



(a) $s = 2$ (b) $s = 3$ (c) $s = 4$

Figure 23: Same as Figure 22, but using **ReLU activation.**

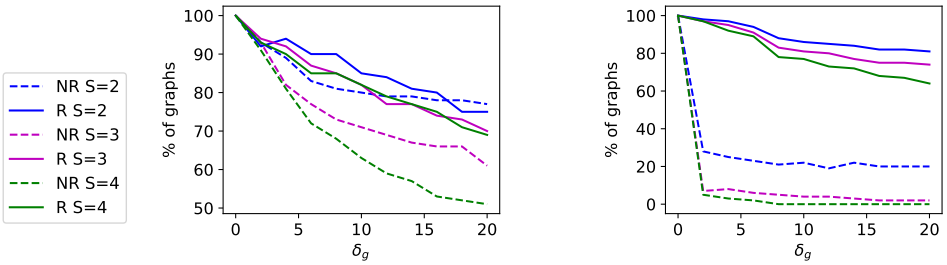


(a) $s = 3$ (b) $s = 6$

Figure 24: Test accuracy under various attacks, and robust training (**R**) or non-robust training (**NR**). ReLU activation. Dataset: Enzymes.

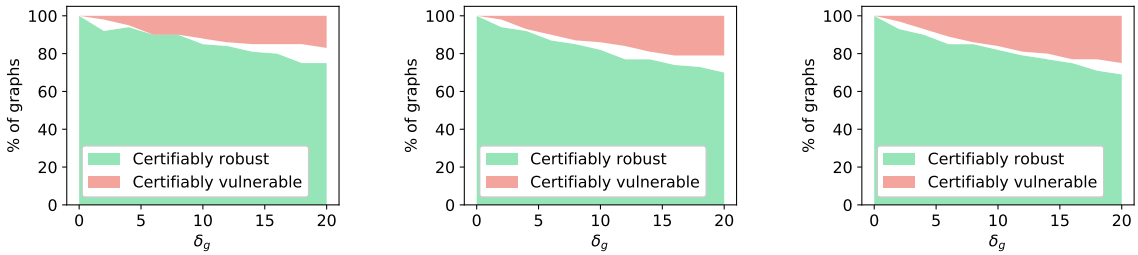
Appendix D (Continued)

D.5.3 More results on NCI1



(a) Linear activation (b) ReLU activation

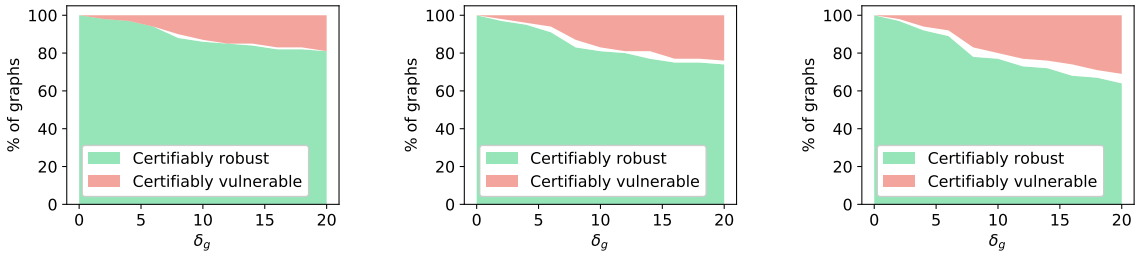
Figure 25: Fraction of graphs certified robust with $s \in \{2, 3, 4\}$, under robust training (**R**) and non-robust training (**NR**). Dataset: NCI1.



(a) $s = 2$ (b) $s = 3$ (c) $s = 4$

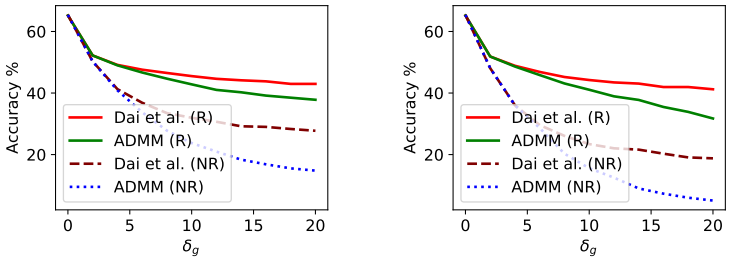
Figure 26: Fraction of graphs that are certified as robust (lower green area) and vulnerable (upper red area, percentage = 100 − y-axis. **Linear activation.** All are under robust training. Dataset: NCI1.

Appendix D (Continued)



(a) $s = 2$ (b) $s = 3$ (c) $s = 4$

Figure 27: Same as Figure 26, but using **ReLU activation.**



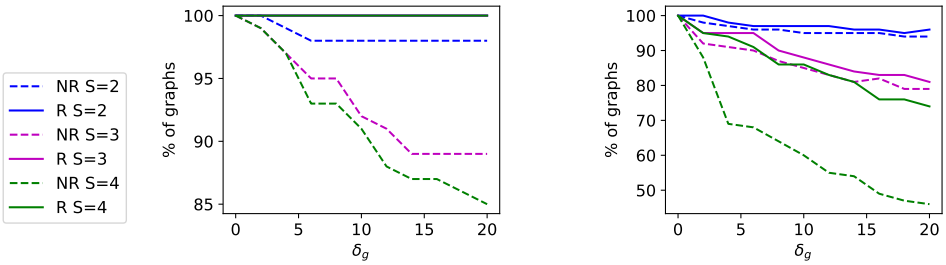
(a) $s = 3$ (b) $s = 6$

Figure 28: Test accuracy under various attacks, and robust training (**R**) or non-robust training (**NR**). ReLU activation. Dataset: NCI1.

In general, the gap for linear activation should be smaller than that for ReLU activation. This has been the case for all datasets, except when $s = 2$ and 3 for NCI1 (Figure Figure 26 and Figure 27). Since the convex envelop is still a lower bound of the true objective $F_c(A)$, there could be exceptions in some cases for some datasets.

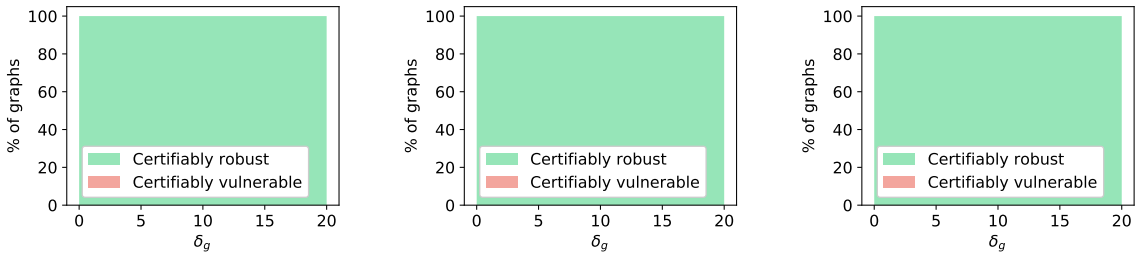
Appendix D (Continued)

D.5.4 More results on PROTEINS



(a) Linear activation (b) ReLU activation

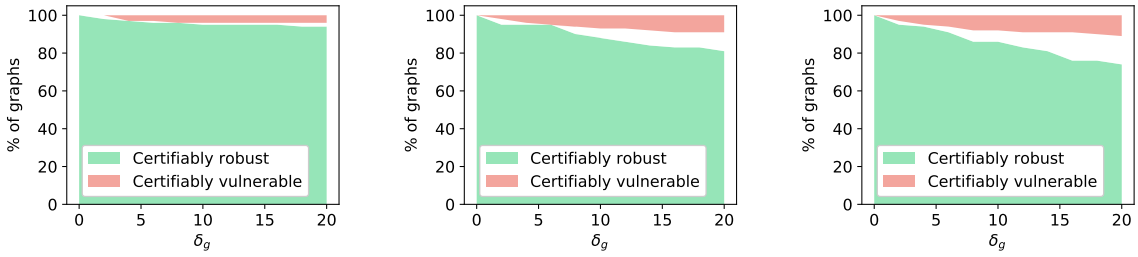
Figure 29: Fraction of graphs certified robust with $s \in \{2, 3, 4\}$, under robust training (**R**) and non-robust training (**NR**). Dataset: PROTEINS. In (a), all the three lines for NR certified 100% graphs as robust for all δ_g.



(a) $s = 2$ (b) $s = 3$ (c) $s = 4$

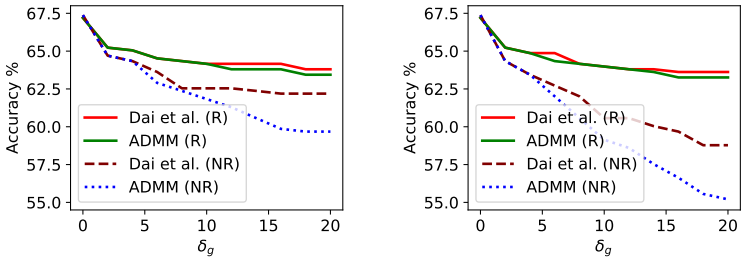
Figure 30: Fraction of graphs that are certified as robust (lower green area) and vulnerable (upper red area, percentage = 100 − y-axis. **Linear activation.** All are under robust training. Dataset: PROTEINS.

Appendix D (Continued)



(a) $s = 2$ (b) $s = 3$ (c) $s = 4$

Figure 31: Same as Figure 30, but using **ReLU activation.**

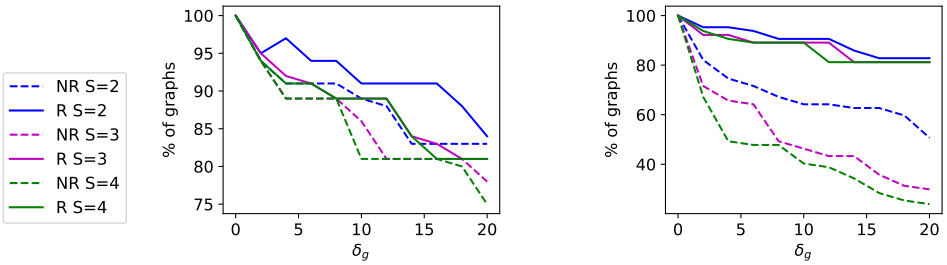


(a) $s = 3$ (b) $s = 6$

Figure 32: Test accuracy under various attacks, and robust training (**R**) or non-robust training (**NR**). ReLU activation. Dataset: PROTEINS.

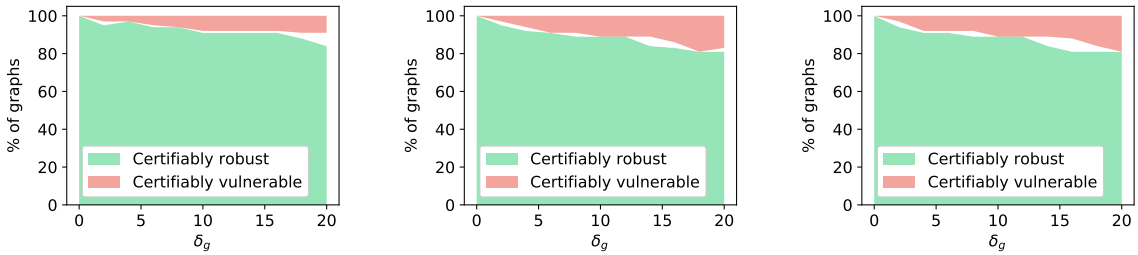
Appendix D (Continued)

D.5.5 More results on MUTAG



(a) Linear activation (b) ReLU activation

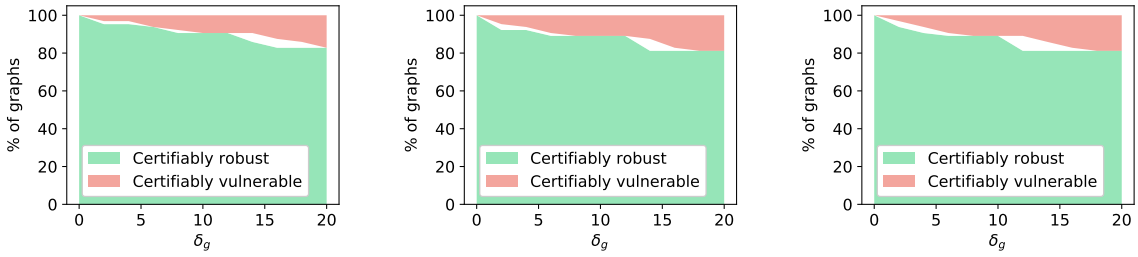
Figure 33: Fraction of graphs certified robust with $s \in \{2, 3, 4\}$, under robust training (**R**) and non-robust training (**NR**). Dataset: MUTAG.



(a) $s = 2$ (b) $s = 3$ (c) $s = 4$

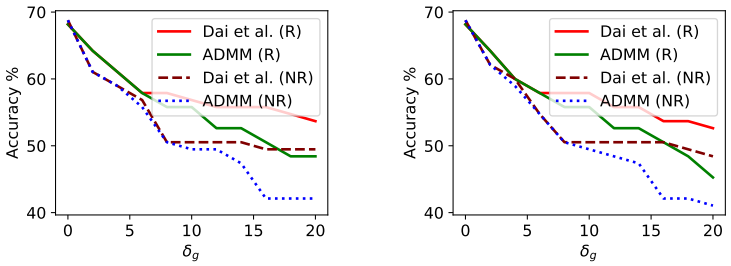
Figure 34: Fraction of graphs that are certified as robust (lower green area) and vulnerable (upper red area, percentage = 100 − y-axis. **Linear activation.** All are under robust training. Dataset: MUTAG.

Appendix D (Continued)



(a) $s = 2$ (b) $s = 3$ (c) $s = 4$

Figure 35: Same as Figure 34, but using **ReLU activation.**



(a) $s = 3$ (b) $s = 6$

Figure 36: Test accuracy under various attacks, and robust training (**R**) or non-robust training (**NR**). ReLU activation. Dataset: MUTAG.

CITED LITERATURE

1. Shi, Z., Zhang, X., and Yu, Y.: Bregman divergence for stochastic variance reduction: saddle-point and adversarial prediction. In *Advances in Neural Information Processing Systems* , pages 6031–6041, 2017.

2. Cranko, Z., Shi, Z., Zhang, X., Nock, R., and Kornblith, S.: Generalised lipschitz regularisation equals distributional robustness. In *International Conference on Machine Learning* , pages 2178–2188. PMLR, 2021.

3. Jin, H., Shi, Z., Peruri, V. J. S. A., and Zhang, X.: Certified robustness of graph convolution networks for graph classification under topological attacks. *Advances in Neural Information Processing Systems* , 33, 2020.

4. Cortes, C. and Vapnik, V.: Support-vector networks. *Machine learning* , 20(3):273–297, 1995.

5. Vapnik, V. and Chervonenkis, A. Y.: On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability & Its Applications* , 16(2):264–280, 1971.

6. Kearns, M. J., Schapire, R. E., and Sellie, L. M.: Toward efficient agnostic learning. *Machine Learning* , 17(2-3):115–141, 1994.

7. Alon, N., Ben-David, S., Cesa-Bianchi, N., and Haussler, D.: Scale-sensitive dimensions, uniform convergence, and learnability. *Journal of the ACM (JACM)* , 44(4):615–631, 1997.

8. Bartlett, P. L., Long, P. M., and Williamson, R. C.: Fat-shattering and the learnability of real-valued functions. *journal of computer and system sciences* , 52(3):434–452, 1996.

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

10. Bartlett, P. L. and Mendelson, S.: Rademacher and gaussian complexities: Risk bounds and structural results. In *International Conference on Computational Learning Theory* , pages 224–240. Springer, 2001.

11. Tibshirani, R.: Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)* , 58(1):267–288, 1996.

12. Zhou, Z.-H. and Liu, X.-Y.: On multi-class cost-sensitive learning. *Computational Intelligence* , 26(3):232–257, 2010.

13. Brefeld, U., Geibel, P., and Wysotzki, F.: Support vector machines with example dependent costs. In *European Conference on Machine Learning* , pages 23–34. Springer, 2003.

14. Lomax, S. and Vadera, S.: A survey of cost-sensitive decision tree induction algorithms. *ACM Computing Surveys (CSUR)* , 45(2):16, 2013.

15. Krizhevsky, A., Sutskever, I., and Hinton, G. E.: Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* , pages 1097–1105, 2012.

16. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al.: Gradient-based learning applied to document recognition. *Proceedings of the IEEE* , 86(11):2278–2324, 1998.

17. Andor, D., Alberti, C., Weiss, D., Severyn, A., Presta, A., Ganchev, K., Petrov, S., and Collins, M.: Globally normalized transition-based neural networks. *arXiv preprint arXiv:1603.06042* , 2016.

18. Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Kingsbury, B., et al.: Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal processing magazine* , 29, 2012.

19. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R.: Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199* , 2013.

20. Dai, H., Li, H., Tian, T., Huang, X., Wang, L., Zhu, J., and Song, L.: Adversarial attack on graph structured data. In *International Conference on Machine Learning (ICML)* , 2018.

21. Zügner, D., Akbarnejad, A., and Günnemann, S.: Adversarial attacks on neural networks for graph data. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)* , pages 2847–2856. ACM, 2018.

22. Zügner, D. and Günnemann, S.: Certifiable robustness and robust training for graph convolutional networks. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)* , pages 246–256. ACM, 2019.

23. Feng, F., He, X., Tang, J., and Chua, T.-S.: Graph adversarial training: Dynamically regularizing based on graph structure. *arXiv preprint arXiv:1902.08226* , 2019.

24. Deng, Z., Dong, Y., and Zhu, J.: Batch virtual adversarial training for graph convolutional networks. *arXiv preprint arXiv:1902.09192* , 2019.

25. Xu, K., Chen, H., Liu, S., Chen, P.-Y., Weng, T.-W., Hong, M., and Lin, X.: Topology attack and defense for graph neural networks: An optimization perspective. In *International Joint Conference on Artificial Intelligence (IJCAI)* , 2019.

26. Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W. L., and Leskovec, J.: Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* , pages 974–983. ACM, 2018.

27. Castillo, C. and Davison, B. D.: Adversarial web search. *Found. Trends Inf. Retr.* , 4(5):377–486, May 2011.

28. Hooi, B., Shah, N., Beutel, A., Gunnemann, S., Akoglu, L., Kumar, M., Makhija, D., and Faloutsos, C.: BIRDNEST: Bayesian inference for ratings-fraud detection. In *International Conference on Data Mining (ICDM)* , 2016.

29. Grünwald, P. D., Dawid, A. P., et al.: Game theory, maximum entropy, minimum discrepancy and robust bayesian decision theory. *the Annals of Statistics* , 32(4):1367–1433, 2004.

30. Asif, K., Xing, W., Behpour, S., and Ziebart, B. D.: Adversarial cost-sensitive classification. In *UAI* , pages 92–101, 2015.

31. Wang, H., Xing, W., Asif, K., and Ziebart, B.: Adversarial prediction games for multivariate losses. In *Advances in Neural Information Processing Systems* , pages 2728–2736, 2015.

32. Esfahani, P. M. and Kuhn, D.: Data-driven distributionally robust optimization using the wasserstein metric: Performance guarantees and tractable reformulations. *Mathematical Programming* , 171(1-2):115–166, 2018.

33. Zhao, C. and Guan, Y.: Data-driven risk-averse stochastic optimization with wasserstein metric. *Operations Research Letters* , 46(2):262–267, 2018.

34. Blanchet, J. and Murthy, K. R.: Quantifying distributional model risk via optimal transport. *arXiv preprint arXiv:1604.01446* , 2016.

35. Gao, R. and Kleywegt, A. J.: Distributionally robust stochastic optimization with wasserstein distance. *arXiv preprint arXiv:1604.02199* , 2016.

36. Sinha, A., Namkoong, H., and Duchi, J.: Certifiable distributional robustness with principled adversarial training. *stat* , 1050:29, 2017.

37. Shafieezadeh-Abadeh, S., Kuhn, D., and Esfahani, P. M.: Regularization via mass transportation. *arXiv preprint arXiv:1710.10016* , 2017.

38. Xu, H., Caramanis, C., and Mannor, S.: Robustness and regularization of support vector machines. *Journal of Machine Learning Research* , 10(Jul):1485–1510, 2009.

39. McShane, E. J.: Extension of range of functions. *Bulletin of the American Mathematical Society* , 40(12):837–842, 1934.

40. Whitney, H.: Analytic extensions of differentiable functions defined in closed sets. *Transactions of the American Mathematical Society* , 36(1):63–89, 1934.

41. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A.: Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083* , 2017.

42. Tsuzuku, Y., Sato, I., and Sugiyama, M.: Lipschitz-margin training: Scalable certification of perturbation invariance for deep neural networks. In *Advances in Neural Information Processing Systems* , pages 6542–6551, 2018.

43. Yoshida, Y. and Miyato, T.: Spectral norm regularization for improving the generalizability of deep learning. *arXiv preprint arXiv:1705.10941* , 2017.

44. Gouk, H., Frank, E., Pfahringer, B., and Cree, M.: Regularisation of neural networks by enforcing lipschitz continuity. *arXiv preprint arXiv:1804.04368* , 2018.

45. Cisse, M., Bojanowski, P., Grave, E., Dauphin, Y., and Usunier, N.: Parseval networks: Improving robustness to adversarial examples. In *Proceedings of the 34th Interna-*

tional Conference on Machine Learning-Volume 70 , pages 854–863. JMLR. org, 2017.

46. Huster, T., Chiang, C.-Y. J., and Chadha, R.: Limitations of the lipschitz constant as a defense against adversarial examples. *arXiv preprint arXiv:1807.09705* , 2018.

47. Zhang, Y., Lee, J. D., and Jordan, M. I.: l1-regularized neural networks are improperly learnable in polynomial time. In *International Conference on Machine Learning* , pages 993–1001, 2016.

48. Zhang, Y., Liang, P., and Wainwright, M. J.: Convexified convolutional neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70* , pages 4044–4053. JMLR. org, 2017.

49. Shalev-Shwartz, S., Shamir, O., and Sridharan, K.: Learning kernel-based halfspaces with the 0-1 loss. *SIAM Journal on Computing* , 40(6):1623–1646, 2011.

50. Williams, C. K. and Seeger, M.: Using the nyström method to speed up kernel machines. In *Advances in neural information processing systems* , pages 682–688, 2001.

51. Drineas, P. and Mahoney, M. W.: On the nyström method for approximating a gram matrix for improved kernel-based learning. *journal of machine learning research* , 6(Dec):2153–2175, 2005.

52. Bojchevski, A. and Günnemann, S.: Certifiable robustness to graph perturbations. In *Advances in Neural Information Processing Systems (NeurIPS)* , 2019.

53. Hoffgen, K.-U., Simon, H.-U., and Vanhorn, K. S.: Robust trainability of single neurons. *Journal of Computer and System Sciences* , 50(1):114–125, 1995.

54. Weston, J., Watkins, C., et al.: Support vector machines for multi-class pattern recognition. In *Esann* , volume 99, pages 219–224, 1999.

55. Crammer, K. and Singer, Y.: On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of machine learning research* , 2(Dec):265–292, 2001.

56. Lee, Y., Lin, Y., and Wahba, G.: Multicategory support vector machines: Theory and application to the classification of microarray data and satellite radiance data. *Journal of the American Statistical Association* , 99(465):67–81, 2004.

57. Morgenstern, O. and Von Neumann, J.: *Theory of games and economic behavior* . Princeton university press, 1953.

58. Sion, M. et al.: On general minimax theorems. *Pacific Journal of mathematics* , 8(1):171–176, 1958.

59. Beck, A. and Teboulle, M.: Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters* , 31(3):167–175, 2003.

60. Zhang, Y. and Xiao, L.: Stochastic primal-dual coordinate method for regularized empirical risk minimization. In *International Conference on Machine Learning (ICML)* , 2015.

61. Zhu, Z. and Storkey, A. J.: Adaptive stochastic primal-dual coordinate descent for separable saddle point problems. In *Machine Learning and Knowledge Discovery in Databases* , pages 645–658, 2015.

62. Nemirovski, A., Juditsky, A., Lan, G., and Shapiro, A.: Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization* , 19(4):1574–1609, 2009.

63. Balamurugan, P. and Bach, F.: Stochastic variance reduction methods for saddle-point problems. In *Advances in Neural Information Processing Systems (NeurIPS)* , 2016.

64. Duchi, J. C., Shalev-Shwartz, S., Singer, Y., and Tewari, A.: Composite objective mirror descent. In *Conference on Computational Learning Theory (COLT)* , 2010.

65. Tseng, P.: On accelerated proximal gradient methods for convex-concave optimization. *submitted to SIAM Journal on Optimization* , 2009.

66. Johnson, R. and Zhang, T.: Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems (NeurIPS)* , 2013.

67. Xiao, L. and Zhang, T.: A proximal stochastic gradient method with progressive variance reduction. *SIAM Journal on Optimization* , 24(4):2057–2075, 2014.

68. Lin, H., Mairal, J., and Harchaoui, Z.: A universal catalyst for first-order optimization. In *Advances in Neural Information Processing Systems (NeurIPS)* , 2015.

69. Warmuth, M. K., Glocer, K. A., and Vishwanathan, S. V. N.: Entropy regularized LPBoost. In *International Conference on Algorithmic Learning Theory (ALT)* , eds. Y. Freund, Y. L. Györfi, and G. Turàn, number 5254 in Lecture Notes in Artificial Intelligence, pages 256 – 271, Budapest, October 2008. Springer-Verlag.

70. Farnia, F. and Tse, D.: A minimax approach to supervised learning. In *Advances in Neural Information Processing Systems* , pages 4240–4248, 2016.

71. Qin, T. and Liu, T.-Y.: Introducing letor 4.0 datasets. *arXiv preprint arXiv:1306.2597* , 2013.

72. Lanckriet, G. R., Ghaoui, L. E., Bhattacharyya, C., and Jordan, M. I.: A robust minimax approach to classification. *Journal of Machine Learning Research* , 3(Dec):555–582, 2002.

73. Wang, Z., Glynn, P. W., and Ye, Y.: Likelihood robust optimization for data-driven problems. *Computational Management Science* , 13(2):241–261, 2016.

74. Abadeh, S. S., Esfahani, P. M. M., and Kuhn, D.: Distributionally robust logistic regression. In *Advances in Neural Information Processing Systems* , pages 1576–1584, 2015.

75. Ben-Tal, A., Den Hertog, D., De Waegenaere, A., Melenberg, B., and Rennen, G.: Robust solutions of optimization problems affected by uncertain probabilities. *Management Science* , 59(2):341–357, 2013.

76. Müller, A.: Integral probability metrics and their generating classes of functions. *Advances in Applied Probability* , 29(2):429–443, 1997.

77. Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Smola, A.: A kernel two-sample test. *Journal of Machine Learning Research* , 13(Mar):723–773, 2012.

78. Lopez-Paz, D., Muandet, K., Schölkopf, B., and Tolstikhin, I.: Towards a learning theory of cause-effect inference. In *International Conference on Machine Learning* , pages 1452–1461, 2015.

79. Staib, M. and Jegelka, S.: Distributionally robust optimization and generalization in kernel methods. *arXiv preprint arXiv:1905.10943* , 2019.

80. Fournier, N. and Guillin, A.: On the rate of convergence in wasserstein distance of the empirical measure. *Probability Theory and Related Fields* , 162(3-4):707–738, 2015.

81. Clément, P. and Desch, W.: An elementary proof of the triangle inequality for the wasserstein metric. 2008.

82. Villani, C.: *Optimal transport: old and new* , volume 338. Springer Science & Business Media, 2008.

83. Blanchet, J., Kang, Y., and Murthy, K.: Robust wasserstein profile inference and applications to machine learning. *arXiv preprint arXiv:1610.05627* , 2016.

84. Blanchet, J., Kang, Y., Zhang, F., and Murthy, K.: Data-driven optimal transport cost selection for distributionally robust optimizatio. *arXiv preprint arXiv:1705.07152* , 2017.

85. Duchi, J., Glynn, P., and Namkoong, H.: Statistics of robust optimization: A generalized empirical likelihood approach. *arXiv preprint arXiv:1610.03425* , 2016.

86. Petrakis, I.: Mcshane-whitney extensions in constructive analysis. *arXiv preprint arXiv:1804.06757* , 2018.

87. Carlini, N. and Wagner, D.: Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)* , pages 39–57. IEEE, 2017.

88. Goodfellow, I. J., Shlens, J., and Szegedy, C.: Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* , 2014.

89. Kurakin, A., Goodfellow, I., and Bengio, S.: Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533* , 2016.

90. Dong, Y., Liao, F., Pang, T., Su, H., Zhu, J., Hu, X., and Li, J.: Boosting adversarial attacks with momentum. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* , pages 9185–9193, 2018.

91. Guo, C., Rana, M., Cisse, M., and van der Maaten, L.: Countering adversarial images using input transformations. *arXiv preprint arXiv:1711.00117* , 2017.

92. Li, X. and Li, F.: Adversarial examples detection in deep networks with convolutional filter statistics. In *Proceedings of the IEEE International Conference on Computer Vision* , pages 5764–5772, 2017.

93. Feinman, R., Curtin, R. R., Shintre, S., and Gardner, A. B.: Detecting adversarial samples from artifacts. *arXiv preprint arXiv:1703.00410* , 2017.

94. Metzen, J. H., Genewein, T., Fischer, V., and Bischoff, B.: On detecting adversarial perturbations. *arXiv preprint arXiv:1702.04267* , 2017.

95. Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z. B., and Swami, A.: Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security* , pages 506–519. ACM, 2017.

96. Athalye, A., Carlini, N., and Wagner, D.: Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *Proceedings of the 35th International Conference on Machine Learning* , pages 274–283, 2018.

97. Buckman, J., Roy, A., Raffel, C. A., and Goodfellow, I. J.: Thermometer encoding: One hot way to resist adversarial examples. In *ICLR 2018* , 2018.

98. Ma, X., Li, B., Wang, Y., Erfani, S. M., Wijewickrema, S. N. R., Houle, M. E., Schoenebeck, G., Song, D. X., and Bailey, J.: Characterizing adversarial subspaces using local intrinsic dimensionality. *CoRR* , abs/1801.02613, 2018.

99. Song, Y., Kim, T., Nowozin, S., Ermon, S., and Kushman, N.: Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. *CoRR* , abs/1710.10766, 2018.

100. Samangouei, P., Kabkab, M., and Chellappa, R.: Defense-gan: Protecting classifiers against adversarial attacks using generative models. *CoRR* , abs/1805.06605, 2017.

101. Kantorovich, L. V.: On a problem of monge. *Uspekhi Mat. Nauk. 3* , page 225–226, 1948.

102. Pratelli, A.: On the equality between monge's infimum and kantorovich's minimum in optimal mass transportation. In *Annales de l'Institut Henri Poincare (B) Probability and Statistics* , volume 43, pages 1–13. Elsevier, 2007.

103. Giner, E.: Necessary and sufficient conditions for the interchange between infimum and the symbol of integration. *Set-Valued and Variational Analysis* , 17(4):321, 2009.

104. Cranko, Z., Kornblith, S., Shi, Z., and Nock, R.: Lipschitz networks and distributional robustness. *arXiv preprint arXiv:1809.01129* , 2018.

105. Suggala, A. S., Prasad, A., Nagarajan, V., and Ravikumar, P.: On adversarial risk and training. *arXiv preprint arXiv:1806.02924* , 2018.

106. Bartlett, P. L., Foster, D. J., and Telgarsky, M. J.: Spectrally-normalized margin bounds for neural networks. In *Advances in Neural Information Processing Systems* , pages 6240–6249, 2017.

107. Raghunathan, A., Steinhardt, J., and Liang, P.: Certified defenses against adversarial examples. *arXiv preprint arXiv:1801.09344* , 2018.

108. Anil, C., Lucas, J., and Grosse, R.: Sorting out lipschitz function approximation. *icml* , 2018.

109. Scholkopf, B. and Smola, A. J.: *Learning with kernels: support vector machines, regularization, optimization, and beyond* . MIT press, 2001.

110. Steinwart, I. and Christmann, A.: *Support vector machines* . Springer Science & Business Media, 2008.

111. Bietti, A. and Mairal, J.: Group invariance, stability to deformations, and complexity of deep convolutional representations. *The Journal of Machine Learning Research* , 20(1):876–924, 2019.

112. Bietti, A., Mialon, G., Chen, D., and Mairal, J.: A kernel perspective for regularizing deep neural networks. *preprint* , 2019.

113. Gittens, A. and Mahoney, M. W.: Revisiting the nyström method for improved large-scale machine learning. *Journal of Machine Learning Research (JMLR)* , 17(117):1–65, 2016.

114. Shawe-Taylor, J. and Williams, C. K. I.: The stability of kernel principal components analysis and its relation to the process eigenspectrum. In *Advances in Neural Information Processing Systems (NeurIPS)* , 2002.

115. Rahimi, A. and Recht, B.: Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. In *Advances in Neural Information Processing Systems (NeurIPS)* , eds. D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, pages 1313–1320. Curran Associates, Inc., 2009.

116. Rahimi, A. and Recht, B.: Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems (NeurIPS)* , eds. J. Platt, D. Koller, Y. Singer, and S. Roweis. Cambridge, MA, MIT Press, 2008.

117. Williamson, R. C., Smola, A. J., and Scholkopf, B.: Generalization performance of regularization networks and support vector machines via entropy numbers of compact operators. *IEEE Transactions on Information Theory* , 47(6):2516–2532, 2001.

118. J.C. MacKay, D.: Introduction to gaussian processes. *NATO Adv Stud Inst Ser F Comput Syst Sci* , 168, 01 1998.

119. Rasmussen, C. E. and Williams, C. K. I.: *Gaussian Processes for Machine Learning* . Cambridge, MA, MIT Press, 2006.

120. Zhu, H., Williams, C. K., Rohwer, R. J., and Morciniec, M.: Gaussian regression and optimal finite dimensional linear models. In *Neural Networks and Machine Learning* , ed. C. M. Bishop. Berlin, Springer-Verlag, 1998.

121. E Fasshauer, G.: Positive definite kernels: Past, present and future. *Dolomite Res. Notes Approx.* , 4, 01 2011.

122. Zhou, D.-X.: The covering number in learning theory. *Journal of Complexity* , 18(3):739–767, 2002.

123. D. Lafferty, J. and Lebanon, G.: Diffusion kernels on statistical manifolds. *Journal of Machine Learning Research (JMLR)* , 6:129–163, 01 2005.

124. König, H.: *Eigenvalue Distribution of Compact Operators* . Basel, Birkhäuser, 1986.

125. Lin, S.-B., Guo, X., and Zhou, D.-X.: Distributed learning with regularized least squares. *Journal of Machine Learning Research (JMLR)* , 18(92):1–31, 2017.

126. Minh, H. Q., Niyogi, P., and Yao, Y.: Mercer's theorem, feature maps, and smoothing. In *Conference on Computational Learning Theory (COLT)* , eds. G. Lugosi and H. U. Simon, pages 154–168, 2006.

127. Kipf, T. N. and Welling, M.: Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)* , 2017.

128. Hamilton, W., Ying, Z., and Leskovec, J.: Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems (NeurIPS)* , pages 1024–1034, 2017.

129. Levie, R., Monti, F., Bresson, X., and Bronstein, M. M.: Cayleynets: Graph convolutional neural networks with complex rational spectral filters. *IEEE Trans. Signal Processing* , 67(1):97–109, 2019.

130. Shervashidze, N., Schweitzer, P., van Leeuwen, E. J., Mehlhorn, K., and Borgwardt, K. M.: Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research (JMLR)* , 12:2539–2561, 2011.

131. Vishwanathan, S. V. N., Schraudolph, N. N., Kondor, I. R., and Borgwardt, K. M.: Graph kernels. *Journal of Machine Learning Research (JMLR)* , 11:1201–1242, 2010.

132. Ma, Y., Wang, S., Aggarwal, C. C., and Tang, J.: Graph convolutional networks with eigenpooling. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)* , 2019.

133. Ying, R., You, J., Morris, C., Ren, X., Hamilton, W. L., and Leskovec, J.: Hierarchical graph representation learning with differentiable pooling. In *Advances in Neural Information Processing Systems (NeurIPS)* , 2018.

134. Dwivedi, V. P., Joshi, C. K., Laurent, T., Bengio, Y., and Bresson, X.: Benchmarking graph neural networks. *arXiv:2003.00982* , 2020.

135. Zhang, M., Cui, Z., Neumann, M., and Chen, Y.: An end-to-end deep learning architecture for graph classification. In *National Conference of Artificial Intelligence (AAAI)* , 2018.

136. Jia, J., Wang, B., Cao, X., and Gong, N. Z.: Certified robustness of community detection against adversarial structural perturbation via randomized smoothing. In *International Conference on the World Wide Web (WWW)* , 2020.

137. Bojchevski, A., Klicpera, J., and Günnemann, S.: Efficient robustness certificates for discrete data: Sparsity-aware randomized smoothing for graphs, images and more. In *International Conference on Machine Learning (ICML)* , 2020.

138. Zügner, D. and Günnemann, S.: Certifiable robustness of graph convolutional networks under structure perturbations. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)* , 2020.

139. Wong, E. and Kolter, J. Z.: Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning (ICML)* , 2018.

140. Zhang, H., Weng, T.-W., Chen, P.-Y., Hsieh, C.-J., and Daniel, L.: Efficient neural network robustness certification with general activation functions. In *Advances in Neural Information Processing Systems (NeurIPS)* , 2018.

141. Weng, T.-W., Zhang, H., Chen, H., Song, Z., Hsieh, C.-J., Boning, D., Dhillon, I. S., and Daniel, L.: Towards fast computation of certified robustness for relu networks. In *International Conference on Machine Learning (ICML)* , 2018.

142. (Dj)Dvijotham, K., Stanforth, R., Gowal, S., Mann, T., and Kohli, P.: A dual approach to scalable verification of deep networks. In *Conference on Uncertainty in Artificial Intelligence (UAI)* , 2018.

143. Katz, G., Barrett, C., Dill, D., Julian, K., and Kochenderfer, M.: Reluplex: An efficient smtsolver for verifying deep neural networks. In *International Conference on Computer Aided Verification (CAV)* , 2017.

144. Singla, S. and Feizi, S.: Robustness certificates against adversarial examples for ReLU networks. *arXiv:1902.01235* , 2019.

145. Cohen, J. M., Rosenfeld, E., and Kolter, J. Z.: Certified adversarial robustness via randomized smoothing. In *International Conference on Machine Learning (ICML)* , 2019.

146. Li, B., Chen, C., Wang, W., and Carin, L.: Certified adversarial robustnesswith additive noise. In *Advances in Neural Information Processing Systems (NeurIPS)* , 2019.

147. Lécuyer, M., Atlidakis, V., Geambasu, R., Hsu, D., and Jana, S.: Certified robustness to adversarial examples with differential privacy. In *IEEE Symposium on Security and Privacy (S&P)* , 2019.

148. Hiriart-Urruty, J. and Lemaréchal, C.: *Convex Analysis and Minimization Algorithms, I and II* , volume 305 and 306. Springer-Verlag, 1996.

149. Jaggi, M.: Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In *International Conference on Machine Learning (ICML)* , 2013.

150. Anstee, R. P.: A polynomial algorithm for b-matchings: An alternative approach. *Information Processing Letters* , 24(3):153–157, 1987.

151. TUD dataset. `https://chrsmrrs.github.io/datasets/docs/datasets`.

152. DD dataset. `https://ls11-www.cs.tu-dortmund.de/staff/morris/graphkerneldatasets`.

153. Defazio, A., Bach, F., and Lacoste-Julien, S.: Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in neural information processing systems* , pages 1646–1654, 2014.

154. Schmidt, M., Le Roux, N., and Bach, F.: Minimizing finite sums with the stochastic average gradient. *Mathematical Programming* , 162(1-2):83–112, 2017.

155. Mairal, J.: Incremental majorization-minimization optimization with application to large-scale machine learning. *SIAM Journal on Optimization* , 25(2):829–855, 2015.

156. Papernot, N., McDaniel, P., Wu, X., Jha, S., and Swami, A.: Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE Symposium on Security and Privacy (SP)* , pages 582–597. IEEE, 2016.

157. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y.: Generative adversarial nets. In *Advances in neural information processing systems* , pages 2672–2680, 2014.

158. Arjovsky, M., Chintala, S., and Bottou, L.: Wasserstein generative adversarial networks. In *International Conference on Machine Learning* , pages 214–223, 2017.

159. Li, C.-L., Chang, W.-C., Cheng, Y., Yang, Y., and Póczos, B.: Mmd gan: Towards deeper understanding of moment matching network. In *Advances in Neural Information Processing Systems* , pages 2203–2213, 2017.

160. Bińkowski, M., Sutherland, D. J., Arbel, M., and Gretton, A.: Demystifying mmd gans. *arXiv preprint arXiv:1801.01401* , 2018.

161. Combettes, P. L. and Pesquet, J.-C.: Proximal splitting methods in signal processing. *Fixed-Point Algorithms for Inverse Problems in Science and Engineering* , 49:185–212, 2011.

162. Penot, J.-P.: *Calculus without derivatives* , volume 266. Springer Science & Business Media, 2012.

163. Tropp, J. A.: An introduction to matrix concentration inequalities. *Foundations and Trends in Machine Learning* , 8(1-2):1–230, 2015.

164. Minh, H. Q.: Some properties of Gaussian reproducing kernel Hilbert spaces and their implications for function approximation and learning theory. *Constructive Approximation* , 32(2):307–338, 2010.

165. Schneider, H.: An inequality for latent roots applied to determinants with dominant principal diagonal. *Journal of the London Mathematical Society* , s1-28(1):8–20, 1953.

166. Shi, Z.-C. and Wang, B.-Y.: Bounds for the determinant, characteristic roots and condition number of certain types of matrices. *Acta Math. Sinica* , 15(3):326–341, 1965.

167. Fasshauer, G. and McCourt, M.: Stable evaluation of Gaussian radial basis function interpolants. *SIAM Journal on Scientific Computing* , 34(2):A737–A762, 2012.

168. Mestre, J.: Greedy in approximation algorithms. In *Algorithms – ESA 2006* , eds. Y. Azar and T. Erlebach, pages 528–539, 2006.

169. Heller, I. and Tompkins, C.: An extension of a theorem of dantzig's. In *Linear Inequalities and Related Systems* , eds. H. Kuhn and A. Tucker, volume 38 of *Annals of Mathematics Studies* . AMS, 1956.

170. Li, Y., Tarlow, D., Brockschmidt, M., and Zemel, R.: Gated graph sequence neural networks. In *International Conference on Learning Representations (ICLR)* , 2016.

VITA

EDUCATION

University of Illinois at Chicago 2015 - 2021

Ph.D. in Computer Science *Chicago, Illinois, United States*

University of Electronic Science and Technology of China 2011 - 2015

B.Eng. in Software Engineering *Chengdu, Sichuan, China*

ACADEMIC EXPERIENCE

Research Assistant, Department of Computer Science, University of Illinois at Chicago.

Teaching Assistant, Department of Computer Science, University of Illinois at Chicago.

- Introduction to Machine Learning, Spring 2018 and Fall 2019
- Artificial Intelligence I, Spring 2018
- Computer Systems, Fall 2015 and Spring 2016

PUBLICATIONS

- Zac Cranko*, Zhan Shi*, Xinhua Zhang, Richard Nock, Simon Kornblith. Generalised Lipschitz Regularisation Equals Distributional Robustness. International Conference on Machine Learning, 2021.
- Hongwei Jin*, Zhan Shi*, Ashish Peruri, Xinhua Zhang. Certified Robustness of Graph Convolution Networks for Graph Classification under Topological Attacks. Advances in Neural Information Processing Systems, 2020.

- Zac Cranko, Aditya Krishna Menon, Richard Nock, Cheng-Soon Ong, Zhan Shi, Christian Walder. Monge blunts Bayes: Hardness Results for Adversarial Training. International Conference on Machine Learning, 2019.
- Vignesh Ganapathiraman, Zhan Shi, Xinhua Zhang, Yaoliang Yu. Inductive Two-layer Modeling with Parametric Bregman Transfer. International Conference on Machine Learning, 2018.
- Zhan Shi, Xinhua Zhang, Yaoliang Yu. Bregman Divergence for Stochastic Variance Reduction: Saddle-Point and Adversarial Prediction. Advances in Neural Information Processing Systems, 2017.