Theory and Algorithms for Several Central Problems in Large-Scale Machine Learning

by

Dmitry Storcheus

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

New York University

September, 2021

Professor Mehryar Mohri

DEDICATION

To my Grandfather Dmitry Nikiforovich Storcheus.

ACKNOWLEDGEMENTS

First of all, I would like to thank my advisor Professor Mehryar Mohri. His dedication to fundamental research as well as passion to educate his students, colleagues and engineers in the industry are truly remarkable. Without Professor Mohri's support I would not be able to combine my Ph.D. studies with work at Google and drive research in the right direction.

I would also like to express special thanks to Dr. Corinna Cortes, the VP of our research team at Google. Corinna was my co-advisor during my Ph.D. studies; I highly appreciate her advice and mentorship throughout the entire program.

I want to thank the members of my DQE and thesis defence committee: Professor Fedor Bogomolov, Professor Esteban Tabak. They helped me successfully prepare for exams and thesis defence as well as provided extremely valuable comments that greatly improved the contents of my thesis.

In addition to academics and Professors at NYU, I would like to sincerely thank my colleagues and managers at Google: Xavier Gonzalvo, Afshin Rostamizadeh, Vitaly Kuznetsov, Charles Weill, Guilia DeSalvo, Andres Munoz, Natalia Ponomareva, Thomas Fu. They provided me with great support, advice and understanding, especially in those hard times when I had to balance between multiple conference deadlines and projects at work.

Finally, I would like to thank the two organizations that I have been part of during my Ph.D.

program: the Computer Science Department of NYU and the Research team at Google. In the computer Science Department, I would like to express special thanks to the Department Chair, Professor Denis Zorin, and the Director of Graduate Studies, Professor Oded Regev, for their support and encouragement.

ABSTRACT

This Ph.D. dissertation presents a fundamental analysis of several central problems in large-scale machine learning. We derive novel and scalable algorithms supported by strong theoretical guarantees for the most practically important large-scale learning scenarios. These scenarios include extensions of the standard supervised learning to multiple base hypothesis spaces, multiple objective functions, multiple distributions, multiple classes and high-dimensional feature spaces.

A standard supervised learning scenario consists of fitting a predictor from a fixed hypothesis space that minimizes a certain empirical loss on a sample drawn i.i.d. from a particular distribution. The richness of modern machine learning applications requires the learning scenario to be large-scale by having the ability to learn from many training examples. While scalability in terms of many examples is widely studied, the current state of research in the field overlooks other scenarios and directions for scalability that may be even more important than many training examples. For instance, by allowing the learner to select predictors from multiple hypothesis spaces of varying complexity or fit to multiple objective functions.

While the problems mentioned above may seem to relate to separate aspects of large-scale learning, this thesis provides a unified theoretical analysis framework that brings these central problems together. This framework is based on the Rademacher complexity analysis as well as on the Empirical and Structural Risk Minimization principles.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF APPENDICES

1 | INTRODUCTION

This Ph.D. dissertation proposes novel theoretical and algorithmic solutions to several central problems in large-scale machine learning. In the introduction, we first briefly describe the standard *supervised classification* framework and its limitations, which give rise to the key problems addressed in the core thesis chapters. Next, we discuss an example of one important large-scale application, which is *machine translation*. We analyze this example from multiple angles and explain what research questions arise from it and how they lead to the formulation of each critical learning problem discussed in this dissertation. Finally, having familiarized the reader with important open questions with the assistance of the machine translation example, we formulate five fundamental and interconnected problems that are central to large-scale machine learning:

- Regularized gradient boosting
- Boosting with multiple sources
- Agnostic learning with multiple objectives
- Efficient gradient computation for structured output learning with rational and tropical losses
- Generalization bounds for supervised dimensionality reduction

In sections 1.3 - 1.7 of the introduction, we provide a brief summary of each problem, including a) a rigorous problem definition and the motivation behind it; b) the discussion of related work;

c) a high-level description of the solutions introduced in this dissertation d) the novelty of the techniques and ideas introduced. In the main part of the thesis that consists of Chapters 3 - 7, we provide a complete treatment of each central problem, including theoretical and empirical results.

1.1 SUPERVISED CLASSIFICATION

One of the most widely used frameworks in machine learning is that of *supervised classification* [Mohri et al. 2018a; Friedman et al. 2001]. Given that $\mathcal{X}$ and $\mathcal{Y}$ are the input and output spaces, the learner has access to a training sample $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ drawn i.i.d. according to some distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$. The goal of the learner in this scenario is to find a predictor h from a given hypothesis space $\mathcal{H}$ that minimizes a specific loss function ℓ on sample S. The structure of such a scenario is rather limited in the sense that it contains only one of each of its core concepts: a single hypothesis set $\mathcal{H}$, a single distribution $\mathcal{D}$, a single objective function ℓ. Additionally, such a scenario considers samples of relatively small size m and small input space dimension $d = \dim(\mathcal{X})$. This thesis studies the topic of *large-scale* learning, where the learner has access not only to a large number of training examples and features that can limit computational capacity, but also to a large number of concepts in the learning scenario, which includes: i) multiple hypothesis spaces ii) multiple data distributions iii) multiple objective functions iv) multiple classes. This thesis provides a unified and principled analysis of several central problems in the large-scale machine learning scenarios described above.

Large-scale learning is one of the most important topics in machine learning because computational resources that we have now allow us to process millions of training examples. However, the academic understanding of what is *large-scale* has been limited to problems with many data points. This view overlooks other fundamental aspects of machine learning problems that are often

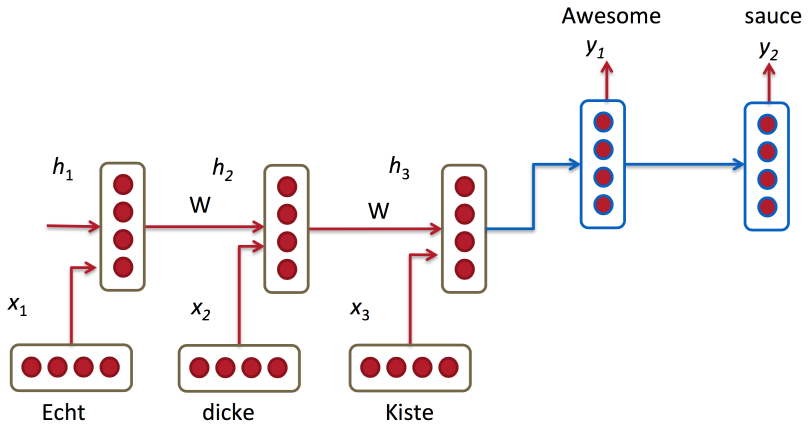


Figure 1.1: Illustration of the encoder-decoder neural machine translation system, where the source language is German, and the target language is English [Britz 2016].

more important than simply scaling to many input examples. Hence, learning problems can also be *large-scale* in terms of hypothesis spaces, objective functions, output classes, and data dimension. Having fundamental theory and practical algorithms that deal with these aspects would be a significant and highly sought-after academic contribution. This thesis closes the missing gap by providing an in-depth analysis of each of the large-scale aspects described above and introducing novel theory and algorithms that efficiently solve problems with large-scale characteristics.

1.2 Machine Translation Example

An important problem in machine learning — *machine translation* — will serve as an example of the large-scale settings that we consider in this thesis. Here, given a fixed pair of source and target languages, the goal is to find the best function $h \in \mathcal{H}$ that maps the sentences in the source language to the sentences in the target language. Clearly, languages contain millions of sentences, which makes the problem large-scale regarding the number of training examples. However, this

is only one aspect of many. For instance, because sentences are represented as high-dimensional vectors (features), it takes significant computational resources to process them. To solve that scalability issue, linear and nonlinear dimensionality reduction methods are used to map the features to a lower-dimensional space. But how does the dimensionality reduction method influence the quality of the resulting translation model? Can we provide the theory that connects the complexity of the reduced feature space to the precision of the translation model itself?

Another central question is: What objective function should we optimize for to obtain a desired machine translation mapping from one language to another? The classic assumption that the learner is interested in optimizing for a specific objective ℓ is too restrictive because there is no single way to evaluate the quality of machine translation. In fact, there are multiple scores that measure the similarity of a source-target pair of sentences. For example, the BLEU score [Papineni et al. 2002] is commonly selected as a single objective for model training and evaluation. However, it *overfits* to short sentences because it is focused on word-based n-gram precision [Duh et al. 2012]. Thus, it is often agreed upon in the research community that machine translation models need to perform well not only in terms of the BLEU score but also for other metrics that measure various aspects of translation beyond the n-gram similarity; for example, METEOR [Lavie and Agarwal 2007], which measures synonym matching, or RIBES [Isozaki et al. 2010], which accounts for deviation in word order. In the end, should the learner optimize for BLEU score, METEOR score, or a weighted combination of these? More generally, can we provide a theoretical framework for learning with multiple losses?

When the correct objective function for the machine translation task is determined, the next challenge is to find $h \in \mathcal{H}$ that minimizes this objective on the training sample; that is, $h^\star = \operatorname{argmin}_{h\in\mathcal{H}} \frac{1}{m}\sum_{i=1}^{m} \ell(h(x_i), y_i)$. This is typically done with gradient descent techniques. However, for many structurally defined objectives, such as BLEU, RIBES, or even the edit-distance, the

gradient computation is intractable. Can we find efficient gradient computation algorithms for such objectives? Can we provide principled theoretical analysis of their convergence properties?

Another assumption of the supervised learning scenario that does not hold for the problem of machine translation is that the data originate from one distribution $\mathcal{D}$. In practice, this is not the case because sentences can be sampled from different people, groups, or dialects of the same language, which calls for an extension of our analysis to multiple distributions $\mathcal{D}_1, \ldots, \mathcal{D}_p$. Given that a learner has access to predictors, each of which is fit to a particular distribution of the sentences in the source language, can we combine such predictors to obtain an even better one? Can we provide theory and learning guarantees for that?

Finally, to build a meaningful machine translation model, one must define the space of sequence - to - sequence mappings $\mathcal{H}$. There is a broad variety of classes of functions to work with: from highly simple ones to extremely rich ones. For example, simple functions such as logistic regression $\mathcal{H}^{\text{reg}}$ or decision trees $\mathcal{H}^{\text{tree}}$ are able to map the most common and easy word patterns from the source language to the target. The richer families such as deep neural networks $\mathcal{H}^{\text{dnn}}$ exhibit such discriminatory power that they can translate rare and complicated sentences. However, they tend to overfit to a particular sample of sentences that they are trained on. Which hypothesis classes $\mathcal{H}^{\text{reg}}$, $\mathcal{H}^{\text{tree}}$ or $\mathcal{H}^{\text{dnn}}$ should be used and when? If the number of such families is large (finite or countably infinite), what is the best and most scalable way to find the right space? Further, can we combine several simple and more complex hypothesis families to have an even better predictor?

In this thesis, a separate chapter is dedicated to each central problem in large-scale learning described above. The solutions provided in each chapter are based on research papers that are published or under review in conferences and journals, namely:

- Chapter 3. Regularized Gradient Boosting [Cortes et al. 2019]
- Chapter 4. Boosting with Multiple Sources (under review)

- Chapter 5. Agnostic Learning with Multiple Objectives [Cortes et al. 2020]
- Chapter 6. Efficient Gradient Computation for Structured Output Learning with Rational and Tropical Losses [Cortes et al. 2018]
- Chapter 7. Generalization Bounds for Supervised Dimensionality Reduction [Mohri et al. 2015]

Each chapter proposes a novel principled theoretical framework that addresses the given large-scale problem. The core part of the theory is the derivation of sample-dependent learning guarantees based on the Rademacher complexity of the underlying hypothesis class. Next, an algorithm is given that is motivated by the learning guarantees. Finally, an empirical study of the algorithm is provided with illustration of its performance on various datasets.

Although the abovementioned problems may seem to relate to separate aspects of large-scale learning, this thesis provides a unified theoretical framework that brings these central problems together. This framework is based on the Rademacher complexity analysis [Koltchinskii and Panchenko 2002a] and the empirical and structural risk minimization principles [Vapnik 1992]. In the several paragraphs below, we provide a brief technical summary of each of the central large-scale learning problems mentioned above, including a) a detailed discussion of previous related work; b) a high-level description of the solutions that we propose; c) the novelty of the techniques and ideas introduced.

1.3 Regularized Gradient Boosting

As mentioned in the machine translation example above, when the learner is provided with a set of hypothesis spaces $\{\mathcal{H}_1, \ldots, \mathcal{H}_p\}$, selecting the space that provides an optimal tradeoff between

the sample fit and complexity is a highly nontrivial task. This issue is particularly relevant for ensemble algorithms, such as ADABOOST or GRADIENTBOOSTING. Ensemble methods form a powerful family of techniques in machine learning that combine multiple base predictors to create more accurate ones. These methods are often highly effective in practice and can achieve a significant performance improvement over the individual base predictors [Quinlan et al. 1996; Caruana et al. 2004; Freund et al. 1996; Dietterich 2000]. ADABOOST [Freund and Schapire 1997b], and its variants are among the most prominent ensemble methods because they are both highly effective in practice and benefit from well-studied theoretical margin guarantees [Freund and Schapire 1997b; Koltchinskii and Panchenko 2002b].

The learning guarantees of boosting critically depend on the complexity of the underlying hypothesis space $\mathcal{H}$, from which the base predictors are selected. The most commonly used families of base predictors are binary decision trees $\mathcal{H}^{\text{trees}}$ or decision stumps $\mathcal{H}^{\text{stumps}}$ (trees of depth one). Using extremely rich hypothesis classes can make the learning guarantees much looser, whereas the algorithm has an increased risk of over-fitting to a particular training sample. *Deep boosting* [Cortes et al. 2014; Kuznetsov et al. 2014] is an alternative framework, where the boosting algorithm has access to p base hypothesis classes of varying complexity $\mathcal{H}_1, \ldots, \mathcal{H}_p$. At every boosting round, the learner chooses a particular base class $\mathcal{H}_k$ and selects a base predictor from that class. Empirically, deep boosting has been demonstrated to produce an optimal balance by selecting more complex hypothesis classes less often. Having hypothesis classes of varying complexity, out of which base predictors are selected, can be understood as *regularization* for the boosting algorithm. A comprehensive theoretical and algorithmic analysis of such regularization techniques for boosting is one of the key contributions of this thesis in Chapter 3.

In addition to the ADABOOST algorithm, another popular and highly successful boosting technique is GRADIENTBOOSTING, which is implemented in several widely used software libraries:

XGBOOST [Chen and Guestrin 2016], MART [Friedman 2002], and DART [Rashmi and Gilad-Bachrach 2015]. Although ADABOOST and GRADIENTBOOSTING may be presented quite differently, both of them are, in fact, instances of *functional gradient descent* [Mason et al. 2000; Grubb and Bagnell 2011] applied to a convex and differentiable objective. Taking the functional gradient descent view provides a thorough understanding of the nature of regularization on the hypothesis class $\mathcal{H}$ in all boosting algorithms. For example, GRADIENTBOOSTING seeks a predictor function h that is closest to the *functional gradient* of the convex objective within some constrained family of base predictors $\mathcal{H}$. Specifying this base predictor family $\mathcal{H}$ such that the selected function does not overfit the gradient, as well as defining an efficient search procedure over $\mathcal{H}$, is crucial for the algorithm's success. In most practical instances, several types of constraints are imposed for that purpose. As an example, for binary regression trees, XGBOOST bounds the number of leaves and the norm of the leaf values vector. This can be viewed as a *regularization*. Further, dividing the original hypothesis space $\mathcal{H}$ that the learning problem is endowed with into subsets of varying complexity, such as the subsets $\mathcal{H}_d^{\text{trees}}$ of decision trees with depth less than d, is nothing but regularization.

Such regularization is widely used by practitioners for ADABOOST, GRADIENTBOOSTING, and other similar algorithms. However, to our knowledge, no theoretical analysis has been provided for these commonly used constraints, which makes the analysis of *Regularized Gradient Boosting* a central problem. A natural question is whether one can derive the learning guarantees that explain how this regularization on $\mathcal{H}$, and, perhaps even more general forms of constraints on functions $h \in \mathcal{H}$, are connected to the generalization performance of GRADIENTBOOSTING. We seek inspiration from the margin-based learning bounds given for ADABOOST [Schapire et al. 1997a; Mohri et al. 2012]. These guarantees, however, do not provide a detailed analysis of the constraints on the families of tree base predictors, nor do they offer guidance on how to conduct an efficient

search of these families to select a predictor during each boosting round. Thus, how to apply regularization in hypothesis spaces used in boosting algorithms becomes a central problem to that area of machine learning. If we had the comprehensive learning guarantees we are seeking, we could significantly benefit from them and use them as a tool to guide the design of regularization and search over $\mathcal{H}$, making GRADIENTBOOSTING methods more robust and less prone to overfitting.

This thesis provides a novel solution to the problem of regularization in boosting algorithms in Chapter 3. We provide a comprehensive analysis of regularization in GRADIENTBOOSTING and derive learning guarantees that explain what type of regularization is appropriate to use under different circumstances. We give data-dependent learning bounds for GRADIENTBOOSTING with regularization, expressed in terms of the Rademacher complexities of the constrained hypotheses' sub-families, from which the base predictors are selected, as well as the ensemble mixture weights. We present a new algorithm, referred to as RGB for regularized gradient boosting, which generalizes the existing gradient boosting methods by introducing a general functional q-norm constraint for the families of the tree base predictors.

Our algorithm and its objective function are directly guided by the theory we develop. Our bound suggests that the *structural risk minimization* principle (SRM) [Vapnik 1992] should be used to break down $\mathcal{H}$ into subsets of varying complexities and, at each round, select a base learner h from a subset that provides the best tradeoff between proximity to the functional gradient and the complexity.

The novelty of the solutions that we present in Chapter 3 includes: a) analyzing the regularization for boosting algorithms in terms of complexities of the nested hypothesis subspaces $\mathcal{H}_1 \subset \mathcal{H}_2 \cdots \subset \mathcal{H}_p$, inspired by deep boosting [Cortes et al. 2014]; b) based on the theory provided, defining a regularized objective function $L(\boldsymbol{\alpha})$ such that the regularized gradient boosting

is equivalent to minimizing that objective function via coordinate descent; c) introducing general q-norm constraints on the families of base predictors, specifically decision trees with bounded leaf values; d) providing a scalable solution for the cases when the number of nested hypotheses p is extremely large, based on randomized coordinate descent [Nesterov 2012].

Finally, Chapter 3 provides the pseudocode of the novel regularized gradient boosting algorithm as well as experimental results on several open-source machine learning datasets. These results demonstrate that our algorithm achieves significantly better out-of-sample performance than the baselines such as XGBOOST. Although the experiments are conducted for specific families of binary decision trees $\mathcal{H}_d^{\text{trees}}$ with depth bounded by a fixed constant d, similar studies can be easily extended to broader families of predictor functions, such as SVMs [Cortes and Vapnik 1995] and deep neural networks [LeCun et al. 2015]. Given favorable experimental results and the generality of the regularization theory that we propose, the results of Chapter 3 have a broad range of applications, including text and image analysis, as well as regression problems.

1.4 BOOSTING WITH MULTIPLE SOURCES

As illustrated by the machine translation example, the common assumption that the source data distribution is the same as the target data distribution does not hold in a vast number of applications. However, such an assumption is still heavily relied upon in the theoretical analysis of many algorithms, including ensemble methods: bagging, AdaBoost, stacking, error-correction techniques, Bayesian averaging, AdaNet, or other adaptive methods for learning neural networks [Breiman 1996; Freund and Schapire 1997a; Smyth and Wolpert 1999; MacKay 1991; Freund et al. 2004; Cortes et al. 2017]. Whereas these algorithms rely on the assumption that the distribution from which the source training data are drawn is the same as the target distribution, in most real-world

applications, the learner actually receives labeled samples from multiple different data distributions $\mathcal{D}_1, \ldots, \mathcal{D}_p$. Given access to these multiple sources, the goal of the learner is to obtain such a predictor that works well on some target distribution that belongs to a convex hull of the sources; that is, $\mathrm{conv}\left(\mathcal{D}_1, \ldots, \mathcal{D}_p\right)$ [Hoffman et al. 2018a; Mansour et al. 2008; Muandet et al. 2013; Xu et al. 2014; Hoffman et al. 2012; Saito et al. 2019; Wang et al. 2019a]. Clearly, the next central problem is: How can we generalize both theory and algorithmic implementation of ensemble methods to the case of multiple data distributions? This thesis proposes several novel solutions to this problem as well as a generalization of boosting to multiple sources in Chapter 4.

Several related problems have been tackled in previous work. In the special case of a single target domain, boosting solutions have been derived [Dai et al. 2007; Yao and Doretto 2010]. These algorithms have been further improved by boosting base predictors jointly with source features (or feature *views*) that are predictive in the target [Yuan et al. 2017; Cheng et al. 2013; Zhang et al. 2014; Xu and Sun 2012, 2011]. Such methods have been widely adopted in various domain adaptation scenarios for different types of data. For example, Huang et al. [2010, 2012] showed that by selecting a base learner jointly with a feature that is predictive across multiple domains at every boosting step, one can achieve higher accuracy than standard transfer learning methods. Moreover, the margin provided by boosting-style algorithms can aid in transfer learning where the target domain is unlabeled. [Habrard et al. 2013] have developed an algorithm that jointly minimizes the source domain error and margin violation proportion on the target domain. [Wang et al. 2019a] have demonstrated that boosting classifiers from different domains can be carried out online. [Taherkhani et al. 2020] and [Becker et al. 2013] have shown that multi-source boosting can be combined with deep neural networks for multi-task learning on large-scale datasets.

Although some solutions to the multi-source boosting exist, they fail to address key questions related to the presence of multiple sources: 1) what should be the form of the ensemble solutions?

2) what should be the form of the objective function? 3) can we derive favorable margin-based guarantees in the presence of multiple sources? In Chapter 4 of this thesis, we address all these questions by developing a novel multi-source boosting framework.

To answer question (1), we illustrate with several examples that standard convex combinations of base predictors, such as those in ADABOOST and GRADIENTBOOSTING may fail in the presence of multiple sources. Instead, our novel technique is to use Q-ensembles, which are convex combinations weighted by a domain classifier Q; that is, $\mathsf{Q}(k|x)$ is the conditional probability of domain k given input point x.

To answer question (2), we provide a general solution that is accurate for any mixture of the source distributions, where the mixture weights may be constrained to belong to a subset of the simplex. For that purpose, we employ the notion of *agnostic* objective [Mohri et al. 2019a]. Further, we present an algorithm that minimizes such an objective using coordinate descent: MULTIBOOST.

For question (3), we provide margin-based and sample-dependent generalization bounds that are expressed in terms of the Rademacher complexity of the underlying hypothesis classes and the empirical margin loss. The bounds presented in Chapter 4 are remarkable in a sense that they hold for any target mixture of the source distributions.

Additionally, Chapter 4 presents an extension of the novel MULTIBOOST algorithm to the *federated learning* scenario. That is an extremely important large-scale learning setting [Kairouz et al. 2021; Brisimi et al. 2018], where a single centralized model is updated using large numbers of independent distributed clients. Chapter 4 concludes with the results of extensive experiments with our MULTIBOOST algorithm, both in its original version and in the FEDMULTIBOOST version.

1.5 AGNOSTIC LEARNING WITH MULTIPLE OBJECTIVES

The machine translation example illustrated that in real-world applications, the learner typically seeks to achieve favorable performance across a number of objectives as opposed to any particular metric. In such cases, the learner has access to a set of *base objective functions* $\{\mathcal{L}_1, \ldots, \mathcal{L}_p\}$, with a certain degree of inter-dependence or correlation between them. Committing to any particular $\mathcal{L}_k$ often fails to capture the full complexity of the underlying problem and causes models to over-fit to that individual objective [Kendall et al. 2018]. The central learning problem considered here is to develop a foundational framework for learning with multiple objectives. Can such a framework have favorable generalization bounds? What combination of the base objectives should the learner optimize for, and how can the complexities of these base losses be incorporated in the learning guarantees?

There is a considerable amount of related work that contains various techniques of learning with multiple objectives. These techniques can be roughly broken down into two types: a) seeking a Pareto-optimal frontier of objectives $\{\mathcal{L}_1, \ldots, \mathcal{L}_p\}$; b) optimizing for a mixture of objectives $\sum_{k=1}^{p} \lambda_k \mathcal{L}_k$ for some fixed mixture weights λ.

The approaches that are directed towards estimating a Pareto-efficient frontier and seeking a solution that lies on that frontier [Jin and Sendhoff 2008; Sener and Koltun 2018; Shah and Ghahramani 2016; Marler and Arora 2004] have a number of drawbacks. In particular, the search for such a frontier is computationally expensive [Duh et al. 2012; Godfrey et al. 2007] and poorly scales with the number of base objectives. Additionally, the Pareto-efficient frontier optimization requires multiple gradient computations, which is a significant challenge for the large-scale problems considered in this dissertation.

The methods that propose using an ensemble $\mathcal{L}_\lambda = \sum_{k=1}^{p} \lambda_k \mathcal{L}_k$ most frequently assume that λ

belongs to some simplex Δ_p, or a strict convex subset of that simplex $\Lambda \subset \Delta_p$ [Jin 2006]. From a computation standpoint, such an approach is favorable because it reduces the learning problem to constrained scalar optimization. However, the fact that the mixture weights λ are fixed raises a number of issues [Van Moffaert et al. 2014]. For instance: a) how can the λ distribution reflect the relative importance of the base objectives? b) can fixed λ account for the cases when the learner's preferences are shifting over time? Moreover, in certain cases, the true learner's preferences may be unobservable or even unknown. This suggests that a better way to describe the multiple objectives problem is to examine the constraint set Λ itself, rather than a particular distribution weights λ.

Given these challenges of working with a fixed mixture weights distribution discussed above, we propose a novel theoretical framework in Chapter 5, termed *Agnostic Learning with Multiple Objectives (ALMO)* backed by strong experimental results. The framework proposed is different from the previous works mentioned above and improves upon them in the following aspects.

First, it performs well for *any* possible combination of mixture weights in the ensemble $\mathcal{L}_\lambda = \sum_{k=1}^{p} \lambda_k \mathcal{L}_k$. Thus, ALMO is also *risk averse* because being robust against any combination of ensemble weights is also robust against their worst (adversarial) combination. Second, our solution is, in fact, guaranteed to be Pareto-optimal, thus lying on the frontier. Our algorithms, hence, provide a theoretical justification for selecting one of the possibly many points on the frontier. Third, the ALMO framework closely matches the learning setting used by most real-world ML practitioners, who often seek to deploy a model that is robust against any mixture of base objectives and does not need to be frequently retrained when the base objective preferences shift. Fourth, we provide a scalable algorithmic implementation of the ALMO framework, based on stochastic gradient descent methods. This solution is implemented in TENSORFLOW [Abadi et al. 2016] and KERAS [Chollet et al. 2015]; thus, it can be made immediately available for applications. Furthermore, we show that our optimization problem has a natural two-player zero-sum game

interpretation.

To achieve such favorable characteristics, we employ the notion of the *agnostic empirical loss*, introduced in earlier works [Mohri et al. 2019a]. Such an objective relies on $\max_{\lambda \in \Lambda} \sum_{k=1}^{p} \lambda_k \mathcal{L}_k$, which is the maximum of the mixture of base objectives $\mathcal{L}_\lambda$ over a convex set of preferences Λ. In Chapter 5, we provide favorable sample-dependent generalization bounds based on the Rademacher complexity of the underlying base loss functions as well as the agnostic empirical loss. We suggest an efficient optimization algorithm for the ALMO setting that is inspired by the generalization bounds derived in this paper. Such an algorithm is based on mirror-prox gradient descent [Juditsky et al. 2011]. Finally, we provide an elaborate experimental study that confirms the superiority of ALMO versus using a fixed mixture distribution for large-scale multi-objective problems. We demonstrate that these results carry over from convex hypothesis spaces to non-convex ones (e.g., Deep Neural Networks).

1.6 Efficient Gradient Computation for Structured Output Learning with Rational and Tropical Losses

Structured prediction is an advanced learning setting that encompasses a significant number of large-scale problems. Whereas in the standard supervised classification, the output space $\mathcal{Y} = \{+1, -1\}$ is assumed to be binary, in structured prediction, the output space takes the form of a sequence, graph, or another combination of *substructures*. Structured prediction problems include the majority of natural language processing tasks, such as pronunciation modeling, part-of-speech tagging, context-free parsing, dependency parsing, machine translation, speech recognition, where

the output labels are sequences of phonemes, part-of-speech tags, words, parse trees, or acyclic graphs, as well as other sequence modeling tasks in computational biology. They also include a variety of problems in computer vision, such as image segmentation, feature detection, object recognition, motion estimation, computational photography and many others.

The key to finding a successful model in the structured prediction setting is to describe how conditional probability densities for each substructure are related with each other. The most successful and widely used algorithms for that have traditionally included: conditional random fields (CRFs) [Lafferty et al. 2001; Gimpel and Smith 2010], StructSVMs [Tsochantaridis et al. 2005], maximum-margin Markov networks (M3N) [Taskar et al. 2003], kernel-regression-based algorithms [Cortes et al. 2007], and search-based methods [Daumé III et al. 2009; Doppa et al. 2014; Lam et al. 2015; Chang et al. 2015; Ross et al. 2011]. A recent development in the field is that *deep learning* methods are becoming increasingly successful for the structured prediction tasks, particularly for part-of-speech tagging [Jurafsky and Martin 2009; Vinyals et al. 2015a], named-entity recognition [Nadeau and Sekine 2007], machine translation [Zhang et al. 2008; Wu et al. 2016], image segmentation [Lucchi et al. 2013], and image annotation [Vinyals et al. 2015b].

Several key problems in structured prediction relate to the objective function and the optimization techniques used to fit a predictor to that objective. Almost every structured prediction problem admits a natural objective that describes sequence similarity in the output space $\mathcal{Y}$. For example, edit-distance or n-gram loss are used for grammar correction tasks, and BLEU score is used for machine translation. The true goal of the learner in the structured prediction setting is to find such a predictor that provides the best similarity measure. However, in real applications, including those that involve deep neural networks, the true sequence similarity measure is often ignored for a number of reasons: a) gradients do not admit closed form; b) gradients can be computationally expensive; c) a key subroutine within the main optimization, such as one requiring to determine

the most violating constraint, may be computationally intractable. Rather than using the true objectives, alternative *surrogate* objectives that are easier to compute and that admit closed form gradients are used, such as the cross-entropy.

The fact that the true sequence similarity measures that ideally should be directly used in structured prediction are replaced by simpler surrogate objectives because of intractable or computationally hard gradients motivates a central problem in that field. Namely, can we provide efficient methods to compute the precise gradients of the true objectives, such as the edit-distance or BLEU score? Are there foundational theory and favorable convergence guarantees behind such methods? In Chapter 6, we provide a novel solution to this central problem that includes a direct and principled way to compute gradients.

Previous work that addresses similar questions consists of a number of methods. The majority of these methods compute the true objective directly either on a small subsample of examples in the input space or sub-structures in the output space. For example, minimum risk training (MRT) [Och 2003; Shen et al. 2016] samples the top-n structured outputs to make the problem computationally tractable. REINFORCE-based methods [Ranzato et al. 2015; Wu et al. 2016] make use of the unbiased stochastic estimate of the true objective, as a result of which the problem becomes computationally tractable. These and similar works [Ranjbar et al. 2013; Eban et al. 2017; McAllester et al. 2010] have demonstrated that even using a restricted (typically quite small) sample to compute the true objective provides much better results than working with a simpler surrogate instead. However, these solutions are incomplete and suffer from a number of drawbacks: a) prohibitively high variance of the gradient estimates; b) gradient bias. Moreover, REINFORCE methods are often required to provide the ground-truth labels of the sequences in the output space while training a predictor, which has a risk of over-fitting to the training sample.

Motivated by the previous work, in Chapter 6, we suggest efficient methods to compute gradi-

ents for the true objectives directly. Compared to previous work, our method does not need to use a small sub-sample of examples or sub-structures to evaluate the true loss and its gradient. Furthermore, the method scales to a large number of examples and complex output space structures. This is achieved by using a number of original techniques: a) using weighted automata and graph operations over appropriate semirings, which allows us to circumvent the computation of exponentially sized sums in the gradient formula; b) using algebras on specific families of semirings: *rational* and *tropical* that allows to execute efficient shortest distance computations. The results of Chapter 6 apply to two broad families of structured prediction objective functions that are based on rational and tropical semirings. These families include, as special cases, the n-gram loss, the edit-distance loss, and many other loss functions commonly used in natural language processing and computational biology tasks that are based on sequence similarity measures.

The results presented in Chapter 6 of this dissertation open doors for efficient learning in large-scale structured prediction problems. Because our solutions are focused on the gradient computation, they are particularly beneficial to deep neural networks. When combined with the recent developments in automatic differentiation (e.g., CNTK [Seide and Agarwal 2016], MXNet [Chen et al. 2015], PyTorch [Paszke et al. 2017], and TensorFlow [Abadi et al. 2016]), they can be used to train structured prediction models such as neural networks with the natural loss of the task. In particular, the use of our techniques for the top layer of neural network models can further accelerate progress in end-to-end training [Amodei et al. 2016; Graves and Jaitly 2014; Wu et al. 2016]. For problems with limited data, such as uncommon languages or some biological problems, our work overcomes the computational bottleneck, uses the exact loss function, and renders the amount of data available the next hurdle for improved performance. For extremely large-scale problems with more data than can be processed, we further present an approximate truncated shortest-path algorithm that can be used for fast approximate gradient computations of the edit-distance.

1.7 Generalization Bounds for Supervised Dimensionality Reduction

Most of the large-scale learning problems employ *dimensionality reduction* techniques one way or another. Such techniques are aimed at projecting the original data onto some low-dimensional subspace that retains most of the information within the data.

The most widely used dimensionality reduction methods include principal component analysis (PCA) [Pearson 1901] and more recent techniques such as isometric feature mapping [Tenenbaum et al. 2000] and locally linear embedding [Roweis and Saul 2000]. Although the goal of these methods is the same—to project the data onto a low-dimensional subspace—each technique provides its specific view on how to define such a subspace and estimate the projection function. More generally, the dimensionality reduction techniques just mentioned and most others have been shown to be specific instances of the kernel PCA (KPCA) algorithm [Ham et al. 2004] for different choices of the kernel function. This equivalence holds simply because the low-dimensional subspace and the projection onto it are defined by the inner product function in that space. A *kernel* function, as long as it satisfies a *positive definite symmetric* property exactly describes the inner product [Mohri et al. 2018a].

There exists a certain degree of inconsistency in the way the methods above construct the low-dimensional subspace and in the way the projections of the original data onto that subspace are used for the subsequent learning algorithm. Specifically, whereas the low-dimensional space is constructed in such a way as to preserve the maximum amount of certain information in the data (e.g., pairwise distances or variance), it is unclear whether this specific information is beneficial for the subsequent learning procedure. Dimensionality reduction used in such a way is typically referred to as *unsupervised* or *decoupled*, which means that the objective of the dimensionality

reduction techniques stage and that of the learning stage are kept separate. However, it is most beneficial for the learner if dimensionality reduction is *supervised* or *coupled*. That is, the low-dimensional subspace and the related projection are estimated in such a way that benefits the most to the subsequent learning task that is performed on the projected data. However, there are no consistent and complete theoretical results in the literature that explain how should the supervised dimensionality reduction happen. The central problem of large-scale learning considered in Chapter 7 is exactly this: can we have principled theory for supervised dimensionality reduction? Can such theory clearly connect the geometric properties of the low-dimensional subspace with the complexity of the subsequent learning problem? Can such theory inspire an algorithm that constructs the subspace jointly with the prediction function within that subspace? Chapter 7 provides complete answers to all these questions and also introduces a novel supervised dimensionality reduction algorithm.

The novel contributions and original techniques in Chapter 7 include the following. First, we introduce the *supervised dimensionality reduction* learning scenario, where the low-dimensional projection step is coupled with the learning step in the projected space. Second, in this learning scenario, we make use of the Kernel PCA and the kernel function, which allows us to describe a variety of projections and generalize many existing dimensionality reduction techniques. Third, we present sample-dependent generalization bounds for this scenario based on the empirical Rademacher complexity of the relevant hypothesis set and the properties of the kernel function. The generalization bounds in our analysis scale as $\widetilde{O}(\sqrt{\Lambda_{(r)}/m})$, where m is the sample size, and $\Lambda_{(r)}$ is the upper bound on the Ky-Fan r-norm of the $r-$dimensional projection operator. Fourth, we derive a novel algorithm, termed *Supervised Kernel PCA (SKPCA)*, that allows to jointly learn the projection function and the discriminatory function in the projected space.

The joint learning algorithm presented in Chapter 7 is largely inspired by the *learning kernels*

topic [Lanckriet et al. 2004; Cortes et al. 2009, 2010; Kloft et al. 2011; Gönen and Alpaydın 2011; Lin et al. 2011; Mosci et al. 2007], where a weighted linear ensemble $\mathcal{K}_\lambda = \sum_{k=1}^{p} \lambda_k \mathcal{K}_k$ of the base kernels $\{\mathcal{K}_1, \ldots, \mathcal{K}_p\}$ is learned. Our work is further motivated by several empirical studies that demonstrate that a coupled dimensionality reduction problem results in much better performance than an uncoupled one [Fukumizu et al. 2004; Gönen 2014].

2 | PRELIMINARIES

In this chapter, we define several fundamental concepts and introduce notations that will be used throughout the thesis. Section 2.1 introduces the key complexity notions, such as *VC-dimension* and *Rademacher complexity*, that will be used throughout the thesis. Section 2.2 describes the *structural risk minimization principle*, which is the foundation for several algorithms in multiple hypothesis space scenarios proposed in the thesis. Section 2.3 introduces boosting, its learning scenario, and learning guarantees based on the properties of convex ensembles of prediction functions. Boosting and its alternative *deep boosting* serve as foundational tools in our analysis of learning with multiple regularized hypothesis spaces in Chapter 3 and learning with multiple distributions (sources) in Chapter 4.

2.1 LEARNING AND COMPLEXITY

We consider the supervised learning setting, in which the learner receives a labeled sample $S = (x_1, y_1), \ldots, (x_m, y_m)$ drawn *i.i.d.* from some distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$ denotes the input space, and $\mathcal{Y} \subset \mathbb{R}$ denotes the output space. Let $\mathcal{H}$ be the hypothesis space, which is a set of functions $h : \mathcal{X} \mapsto \mathcal{Y}$. Let $l : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}_+$ be a loss function that measures the correctness of a hypothesis $h \in \mathcal{H}$. The loss of $h \in \mathcal{H}$ for a labeled instance (x, y) equals $\ell(h(x), y)$, and the

expectation of the loss is

$$\mathcal{L}(h) = \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \left[\ell(h(x), y)\right]. \tag{2.1}$$

In a standard supervised learning scenario, the goal of the learner is to find such a hypothesis $h^\star$ from the given space $\mathcal{H}$ that has the minimal expected loss with respect to the distribution $\mathcal{D}$, that is:

$$h^\star = \operatorname*{argmin}_{h\in\mathcal{H}} \mathcal{L}(h). \tag{2.2}$$

In most cases, the true distribution $\mathcal{D}$ is not available to the learner. Instead, the learner has access to empirical distribution $\mathcal{D}^m = \left(\mathcal{X} \times \mathcal{Y}\right)^m$, which is represented as a labeled sample $S = (x_1, y_1), \ldots, (x_m, y_m)$. The learner will use this sample to evaluate the empirical loss function

$$\hat{\ell} = \frac{1}{m} \sum_{i=1}^{m} \ell(h(x), y) \tag{2.3}$$

and look for a hypothesis that minimizes the empirical loss

$$h^\star = \operatorname*{argmin}_{h\in\mathcal{H}} \hat{\ell}(h). \tag{2.4}$$

A significant part of this thesis is dedicated to the analysis of the complexity of hypothesis sets $\mathcal{H}$ in large-scale learning scenarios; therefore, it is important to introduce several standard measures of the capacity of functional sets, such as the Rademacher complexity and VC dimension. We define the key concepts below, whereas for a more detailed treatment, the reader should refer to [Mohri et al. 2012].

Definition 2.1 (Sample Rademacher Complexity)**.** . Let $\mathcal{G} : \mathcal{Z} \mapsto \mathbb{R}$ be a family of functions mapping from some set $\mathcal{Z}$ to $\mathbb{R}$. Given an i.i.d. sample $S = (z_1, \cdots, z_m) \sim \mathcal{Z}^m$, the empirical

Rademacher complexity of $\mathcal{G}$ is defined as follows:

$$\widehat{\mathfrak{R}}_S(\mathcal{G}) = \frac{1}{m}\mathop{\mathbb{E}}_{\boldsymbol{\sigma}}\left[\sup_{g\in\mathcal{G}}\sum_{i=1}^{m}\sigma_i g(z_i)\right], \tag{2.5}$$

where σ_i-s, $i \in [m]$, are independent uniformly distributed random variables in $\{-1, 1\}$.

Definition 2.2 (Rademacher Complexity)**.** The Rademacher complexity of $\mathcal{G}$ is the expected sample Rademacher complexity of $\mathcal{G}$; that is,

$$\mathfrak{R}_S(\mathcal{G}) = \mathbb{E}\left[\widehat{\mathfrak{R}}_S(\mathcal{G})\right]. \tag{2.6}$$

Definition 2.3 (Growth function)**.** Let $\mathcal{G} : \mathcal{Z} \mapsto \{-1, 1\}$ be a family of functions mapping from $\mathcal{Z}$ to $\{-1, 1\}$. The Growth function $\Pi_{\mathcal{G}} : \mathbb{Z} \mapsto \mathbb{Z}$ of $\mathcal{G}$ is defined as:

$$\Pi_{\mathcal{G}}(m) = \max_{(z_1,\cdots,z_m)\in\mathbb{Z}} |\{(g(z_1), \cdots, g(z_m))|g \in \mathcal{G}\}| \tag{2.7}$$

Growth function is related to Rademacher complexity by Massart's lemma [Massart and Picard 2007].

Theorem 2.4 (Massart's lemma)**.** *Let $\mathcal{G}$ be a family of functions taking values in $\{-1, 1\}$; then, the following inequality holds for any sample S of size m:*

$$\widehat{\mathfrak{R}}_S(\mathcal{G}) \leq \sqrt{\frac{2\log\Pi_{\mathcal{G}}(m)}{m}}. \tag{2.8}$$

Definition 2.5 (VC-dimension)**.** Let $\mathcal{G}$ be a family of functions taking values in $\{-1, 1\}$; then, the VC dimension of $\mathcal{G}$, denoted as $VCdim(\mathcal{G})$, is the largest value of d such that $\Pi_{\mathcal{G}}(d) = 2^d$.

Growth function and VC dimension are related via Sauer's lemma.

Theorem 2.6 (Sauer's lemma)**.** *Let $\mathcal{G}$ be a family of functions with $VCdim(\mathcal{G}) = d$; then, for all $m \geq d$, the following inequality holds:*

$$\Pi_{\mathcal{G}}(m) \leq \sum_{i=1}^{d} \binom{m}{i} \leq \left(\frac{em}{d}\right)^{d}. \tag{2.9}$$

Because the learner is selecting a hypothesis $h^{\star}$ from the function class $\mathcal{H}$, it is natural that the capacity characteristics (or richness) of the space $\mathcal{H}$ plays a key role in how well $h^{\star}$ can fit the training sample and if this result can be generalized to any other sample from the same data distribution. In fact, there are fundamental inequalities (*generalization bounds or learning guarantees*) that connect the richness of $\mathcal{H}$ described in terms of its Rademacher complexity with the error of individual function $h \in \mathcal{H}$ [Koltchinskii and Panchenko 2002b; Ledoux and Talagrand 1991a; Mohri et al. 2012].

Theorem 2.7 (Rademacher complexity bounds for binary classification)**.** *Let $\mathcal{H}$ be a family of functions taking values in $\{-1, 1\}$ and $\mathcal{D}$ be the distribution over the input space $\mathcal{X}$. Then, for any $\delta > 0$ with a probability of at least $1 - \delta$ over a sample S of size m drawn i.i.d. from $\mathcal{D}$, the following inequalities hold for any $h \in \mathcal{H}$:*

$$R(h) \leq \widehat{R}_S(h) + \mathfrak{R}_S(\mathcal{H}) + \sqrt{\frac{\log \frac{1}{\delta}}{2m}}, \tag{2.10}$$

where $\widehat{R}_S(h)$ and $R(h)$ are empirical and expected losses of hypothesis h.

Note that this bound holds uniformly for all $h \in \mathcal{H}$ and consists of two parts (up to probability coefficients): the empirical one $\widehat{R}_S(h)$ and the theoretical one $\mathfrak{R}_S(\mathcal{H})$. The empirical part corresponds to the observed error of a hypotheses h on the training sample S, and the theoretical part describes the richness of the function class $\mathcal{H}$ from which the individual h is taken.

2.2 STRUCTURAL RISK MINIMIZATION

The generalization bound above describes the fundamental *tradeoff* that lies at the core of all machine learning algorithms; that is, the tradeoff between the richness of hypothesis space $\mathcal{H}$ and its generalization properties. In this thesis, we will refer multiple times to this tradeoff; thus, explaining it in more detail is critical. To illustrate, consider a highly rich hypothesis class $\mathcal{H}_{dnn}$ of deep neural networks and a less rich space $\mathcal{H}_{reg}$ of logistic regression. A neural network from $\mathcal{H}_{dnn}$ can often near-perfectly fit a particular sample S, making $\widehat{R}_S(h)$ small, but due to their richness $\mathfrak{R}_S(\mathcal{H}_{dnn})$ is large, which raises the upper bound on the generalization error $R(h)$. On the other hand, a logistic regression from $\mathcal{H}_{reg}$ does not fit a particular sample S as well as a neural network; however, the space $\mathcal{H}_{reg}$ is less rich, making its capacity $\mathfrak{R}_S(\mathcal{H}_{reg})$ much smaller, contributing to a decrease in the generalization bound on $R(h)$. It can often occur that such a tradeoff is in favor of logistic regression, and the upper bound on its generalization error consisting of the sum of two terms is actually smaller than that of the deep neural network class. To be completely rigorous, the opposite situation can also happen when the tradeoff is in favor of neural networks; because the bounds are *sample dependent*, the results can vary for different samples.

The bounds above indicate the importance of understanding the richness and structure of the hypothesis class $\mathcal{H}$ used for a particular problem. This thesis dedicates a significant part to giving novel theoretical analysis of the structure of $\mathcal{H}$ for a variety of learning scenarios. Such an analysis is largely based on the fundamental principle of *SRM* [Vapnik 1992], which we summarize below. This framework assumes that the hypothesis space $\mathcal{H}$ is decomposed into a countable union of smaller subspaces $\mathcal{H} = \cup_{k\geq 1}\mathcal{H}_k$ such that the sequence of subspaces is nested (i.e., $\mathcal{H}_k \subset \mathcal{H}_{k+1}$ for all $k \geq 1$). The SRM consists of choosing the index $k^\star$ and a hypothesis $h^\star \in \mathcal{H}_{k^\star}$ that minimizes sample empirical error. The pair $k^\star, h^\star$ are jointly selected to minimize the upper bound

on generalization error. In this way, the SRM principle serves as a principled guideline to solve the tradeoff that we described above. SRM benefits from strong learning guarantees, which are presented in the following theorem.

Theorem 2.8 (SRM learning guarantees)**.** *For any $\delta > 0$ with a probability of at least $1 - \delta$ over the draw of an i.i.d. training sample S of size m, the generalization error of hypothesis $h^\star$ given by the SRM principle is bounded as follows:*

$$R(h^\star) \leq \inf_{h \in \mathcal{H}} \left(R(h) + 2\mathfrak{R}_S(\mathcal{H}_{k(h)}) + \sqrt{\frac{\log k(h)}{m}} \right) + \sqrt{\frac{2 \log \frac{3}{\delta}}{m}}, \tag{2.11}$$

where $\mathcal{H}_{k(h)}$ is the hypothesis subclass with the smallest complexity among those that contain h.

SRM shows what structure in terms of nesting subclasses needs to be imposed on the space $\mathcal{H}$ and how to select the subclass to minimize the upper bound on generalization error. SRM and its guarantees are key tools for learning with complex and large-scale hypothesis spaces and are called upon multiple times in this thesis. In fact, several novel theoretical results and algorithms presented in the following sections are built up from the SRM foundations.

2.3 BOOSTING

Boosting is a widely used method to combine several base predictors into a more powerful one. Popular boosting algorithms, such as ADABOOSTor GRADIENTBOOSTING, show impressive results in practice and benefit from rich theoretical analysis. This chapter describes the main concepts and results in boosting research, preparing the reader to understand the novel contribution of this thesis.

The learning scenario of boosting is such that the learner has access to N base predictors

$\{h_1, \cdots, h_N\}$, each of which satisfy the *weak learning* property.

Definition 2.9 (Weak Learning)**.** A concept class $\mathcal{C}$ is weakly PAC learnable if there exists algorithm $\mathcal{A}$ and $\gamma > 0$ s.t. $\forall \delta > 0$, for all distributions $\mathcal{D}$ on $\mathcal{X}$ and any target concept $c \in \mathcal{C}$ the following holds for any sample size $m >$ poly$(1/\delta)$:

$$\mathbb{P}_{S \sim \mathcal{D}^m} \left[R(h_S) \leq \frac{1}{2} - \gamma \right] \geq 1 - \delta. \tag{2.12}$$

2.3.1 AdaBoost

The weak learning assumption essentially means that the predictor h is only γ-better than random. Boosting will combine such base predictors into an ensemble $f = \sum_{j=1}^{N} \alpha_j h_j$. In Figure 2.1, we illustrate the pseudocode of the widely used technique termed AdaBoost for binary classification. The algorithm takes a training sample $S = (x_1, y_1), \ldots, (x_m, y_m)$ and maintains a distribution $\mathcal{D}_t$ over that sample for every boosting round $t \in [T]$.

At every round, the algorithm selects a base predictor h_t that minimizes the weighted empirical classification error; that is:

$$h_t = \operatorname*{argmin}_{h \in \mathcal{H}} \mathbb{P}_{i \sim \mathcal{D}_t}(h(x_i) \neq y_i) = \operatorname*{argmin}_{h \in \mathcal{H}} \underbrace{\sum_{i=1}^{m} \mathcal{D}_t(i) \mathbb{1}_{y_i \neq h(x_i)}}_{\epsilon_t}. \tag{2.13}$$

Because $\epsilon_t < \frac{1}{2}$, then the ensemble weights α_t assigned by the AdaBoost to base predictors are nonnegative. Therefore, the next round distribution $\mathcal{D}_{t+1}$ is derived from the previous round distribution $\mathcal{D}_t$ by increasing the weights of incorrectly classified examples and decreasing the weights of correctly classified ones.

ADABOOST$(S = (x_1, y_1), \ldots, (x_m, y_m))$

for $i \leftarrow 1$ **to** m **do**
    $\mathcal{D}_1(i) \leftarrow \frac{1}{m}$
for $t \leftarrow 1$ **to** T **do**
    $h_t \leftarrow \operatorname{argmin}_{h \in \mathcal{H}} \Big[\underbrace{\sum_{i=1}^{m} \mathcal{D}_t(i) \mathbb{1}_{y_i \neq h(x_i)}}_{\epsilon_t} \Big]$
    $\alpha_t \leftarrow \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$
    $Z_t \leftarrow 2\sqrt{\epsilon_t(1-\epsilon_t)}$
    for $i \leftarrow 1$ **to** m **do**
        $\mathcal{D}_{t+1}(i) \leftarrow \frac{\mathcal{D}_t(i) e^{-\alpha_t y_i h_t(x_i)}}{Z_t}$
$f \leftarrow \sum_{t=1}^{T} \alpha_t h_t$
return f

Figure 2.1: Pseudocode of the ADABOOST algorithm.

2.3.2 BOOSTING AS COORDINATE DESCENT

ADABOOST and its extensions, as well as other boosting algorithms, can be represented as instances of the *coordinate descent* algorithm with an appropriately defined convex objective function.

For the purpose of this section, assume that the hypothesis space $\mathcal{H} = \{h_1, \ldots, h_N\}$ is finite, consisting of N base classifiers. The ensemble returned by the ADABOOST algorithm can be written as $f = \sum_{j=1}^{N} \bar{\alpha}_j h_j$, where $\bar{\alpha}_j > 0$. Let $\bar{\boldsymbol{\alpha}} = [\bar{\alpha}_1, \cdots, \bar{\alpha}_N]^\top \in \mathbb{R}^N$ and define the following objective function:

$$F(\bar{\boldsymbol{\alpha}}) = \frac{1}{m} \sum_{i=1}^{m} e^{-y_i f(x_i)} = \frac{1}{m} \sum_{i=1}^{m} e^{-y_i \sum_{j=1}^{N} \bar{\alpha}_j h_j(x_i)}. \tag{2.14}$$

Because the zero-one loss $u \mapsto \mathbb{1}_{u \leq 0}$ can be upper-bounded by the exponential loss $u \mapsto e^{-u}$, the

objective above is an upper bound on the empirical error of the ensemble function f. Observe that $F(\bar{\boldsymbol{\alpha}})$ is convex as a sum of convex functions; thus, its minimum can be found using the maximum derivative coordinate descent algorithm. We will illustrate below that $F(\bar{\boldsymbol{\alpha}})$ corresponds to the objective function of the ADABOOST algorithm and that $\operatorname{argmin}_{\bar{\boldsymbol{\alpha}}} F(\bar{\boldsymbol{\alpha}})$ precisely matches the weights assigned to base classifiers from $\mathcal{H}$ by the ADABOOST algorithm.

The directional derivative of $F(\bar{\boldsymbol{\alpha}})$ along a basis vector $\mathbf{e}_k$ is

$$\begin{aligned} F'(\bar{\boldsymbol{\alpha}}_{t-1}, \mathbf{e}_k) &= \lim_{\eta \mapsto 0} \frac{F(\bar{\boldsymbol{\alpha}}_{t-1} + \eta \mathbf{e}_k) - F(\bar{\boldsymbol{\alpha}}_{t-1})}{\eta} \\ &= -\frac{1}{m} \sum_{i=1}^{m} y_i h_k(x_i) e^{-y_i \sum_{j=1}^{N} [\bar{\boldsymbol{\alpha}}_{t-1}]_j h_j(x_i)} \\ &= \underbrace{\left[2\epsilon_{t,k} - 1\right]}_{<0} \frac{Z_t}{m}, \end{aligned}$$

where $\epsilon_{t,k}$ is the distribution-weighted error of the base classifier h_k. Due to the fact that $\epsilon_{t,k} < \frac{1}{2}$, then $2\epsilon_{t,k} - 1 < 0$, and $\frac{Z_t}{m}$ is nonnegative. Thus, the maximum derivative direction selected by the coordinate descent at step t corresponds to the base classifier h_k with the smallest error selected by the ADABOOST at step t.

The step size η along the direction $\mathbf{e}_k$ is equal to $\eta = \operatorname{argmin}_\eta F(\bar{\boldsymbol{\alpha}}_{t-1} + \eta \mathbf{e}_k)$.

$$\begin{aligned}
\frac{dF(\bar{\boldsymbol{\alpha}}_{t-1} + \eta \mathbf{e}_k)}{d\eta} = 0 &\Leftrightarrow -\frac{1}{m}\sum_{i=1}^{m} y_i h_k(x_i) e^{-y_i \sum_{j=1}^{N} [\bar{\boldsymbol{\alpha}}_{t-1}]_j h_j(x_i)} e^{-\eta y_i h_k(x_i)} = 0 \\
&\Leftrightarrow -\frac{1}{m}\sum_{i=1}^{m} y_i h_k(x_i) \bar{\mathcal{D}}_t(i) \bar{Z}_t e^{-\eta y_i h_k(x_i)} = 0 \\
&\Leftrightarrow -\frac{1}{m}\sum_{i=1}^{m} y_i h_k(x_i) \bar{\mathcal{D}}_t(i) e^{-\eta y_i h_k(x_i)} = 0 \\
&\Leftrightarrow -\left[(1 - \bar{\epsilon}_{t,k}) e^{-\eta} - \bar{\epsilon}_{t,k} e^{\eta}\right] = 0 \\
&\Leftrightarrow \eta = \frac{1}{2} \log \frac{1 - \epsilon_{t,k}}{\epsilon_{t,k}},
\end{aligned}$$

which verifies that the step size selected by the coordinate descent is equal to the weight α_t chosen at the t-th round of ADABOOST.

The coordinate descent view of the ADABOOST algorithm can be extended to other convex and differentiable objective functions of $\bar{\boldsymbol{\alpha}}$. For example, applying coordinate descent to the objective based on the logistic loss $u \mapsto \log_2(1 + e^{-u})$ instead of the exponential loss $u \mapsto e^{-u}$ leads to the LOGITBOOST [Friedman et al. 2000] algorithm.

2.3.3 LEARNING GUARANTEES

The generalization properties of the ADABOOST algorithm can be analyzed using the complexity of families of convex ensembles. Let $\operatorname{conv}(\mathcal{H})$ be the convex hull of a space of real-valued functions $\mathcal{H}$:

$$\operatorname{conv}(\mathcal{H}) = \left\{ \sum_{k=1}^{p} \mu_k h_k : p \geq 1, \forall k \in [p] : \mu_k \geq 0, h_k \in \mathcal{H}, \sum_{k=1}^{p} \mu_k \leq 1 \right\}. \tag{2.15}$$

Theorem 2.10 (Ensemble Rademacher margin bound)**.** *Let $\mathcal{H}$ be the set of real-valued functions. Fix $\rho > 0$. Then, for any $\delta > 0$, with a probability of at least $1 - \delta$ over the draw of a sample S of size m, the following inequalities hold for all $h \in \operatorname{conv}(\mathcal{H})$:*

$$R(h) \leq \widehat{R}_{S,\rho}(h) + \frac{2}{\rho}\mathfrak{R}_m(\mathcal{H}) + \sqrt{\frac{\log \frac{1}{\delta}}{2m}},$$

$$R(h) \leq \widehat{R}_{S,\rho}(h) + \frac{2}{\rho}\widehat{\mathfrak{R}}_S(\mathcal{H}) + 3\sqrt{\frac{\log \frac{2}{\delta}}{2m}}.$$

Note that $f = \sum_{i=1}^{N} \alpha_j h_j$ returned by the ADABOOST algorithm is not in $\operatorname{conv}(\mathcal{H})$. However, its normalized version $\bar{f} = \frac{\sum_{i=1}^{N} \alpha_j h_j}{\|\boldsymbol{\alpha}\|_1} \in \operatorname{conv}(\mathcal{H})$. Thus, we can apply the ensemble Rademacher bound from above to $\bar{f}$ to obtain the following inequality with a probability of at least $1 - \delta$:

$$R(f) \leq \widehat{R}_{S,\rho}(\bar{f}) + \frac{2}{\rho}\mathfrak{R}_m(\mathcal{H}) + \sqrt{\frac{\log \frac{1}{\delta}}{2m}}. \tag{2.16}$$

The bound above guarantees effective generalization for the scenarios where the margin loss $R_\rho(\bar{f})$ is small for a large margin ρ.

2.3.4 REGULARIZATION

It may occur in practice that ADABOOST overfits to particular base predictors h_j, assigning high weights α_j to them in a linear ensemble. Typically, this happens because the algorithm focuses of a small fraction of the training examples that are difficult to classify. Avoiding such situations can be done either by reducing the number of boosting rounds (*early stopping*) or by controlling the mixture coefficients $\alpha_1, \ldots, \alpha_N$. Given $\bar{\boldsymbol{\alpha}} = [\bar{\alpha}_1, \cdots, \bar{\alpha}_N]^\top \in \mathbb{R}^N$; the latter can be done by introducing the norm of $\bar{\boldsymbol{\alpha}}$ into the objective function. For some $\lambda > 0$, let $G(\bar{\boldsymbol{\alpha}})$ be the

L_1-regularized objective function of the ADABOOST algorithm defined by:

$$G(\bar{\boldsymbol{\alpha}}) = \frac{1}{m}\sum_{i=1}^{m} e^{-y_i f(x_i)} + \lambda\|\bar{\boldsymbol{\alpha}}\|_1 = \frac{1}{m}\sum_{i=1}^{m} e^{-y_i \sum_{j=1}^{N} \bar{\alpha}_j h_j(x_i)} + \lambda\|\bar{\boldsymbol{\alpha}}\|_1. \tag{2.17}$$

Because the original ADABOOST objective $\frac{1}{m}\sum_{i=1}^{m} e^{-y_i f(x_i)}$ is convex, and the L_1-norm is a convex function of $\bar{\boldsymbol{\alpha}}$, then $G(\bar{\boldsymbol{\alpha}})$ is convex as well. Therefore, ADABOOST with L_1-regularization is equivalent to applying coordinate descent to the objective $G(\bar{\boldsymbol{\alpha}})$.

2.3.5 DEEP BOOSTING

ADABOOST and its extensions combine functions from a hypothesis set of base classifiers $\mathcal{H}$. In many applications, $\mathcal{H}$ consists of *decision trees*. In particular, decision trees of depth one (also referred to as *boosting stumps*) have been extremely successful in practice for small datasets. However, for large-scale applications (such as speech or image processing), the boosting stumps do not provide sufficient discriminatory power. Thus, it is desirable to use richer hypothesis classes. For example, the families of all decision trees $\mathcal{H}_k^{\text{trees}}$ with depth bounded by a sufficiently large constant k. As has been described in the previous sections, the learning guarantees for boosting depend not only on the empirical error but also on the complexity of the hypothesis class $\mathfrak{R}(\mathcal{H}_k^{\text{trees}})$. If a large-scale problem uses boosting with deep decision trees, then for k sufficiently large, the increased complexity $\mathfrak{R}(\mathcal{H}_k^{\text{trees}})$ can make the learning guarantees much looser, whereas the algorithm has increased risk of over-fitting to a particular training sample.

An alternative boosting framework that solves the problem above is *deep boosting* [Cortes et al. 2014; Kuznetsov et al. 2014]. In this framework, the boosting algorithm has access to p base hypothesis classes of increasing complexity $\mathcal{H}_1, \ldots, \mathcal{H}_p$, such as the $\mathcal{H}_k^{\text{trees}}$ of increasing tree depth. At every boosting round, the learner picks a particular base class $\mathcal{H}_k$ and selects a base

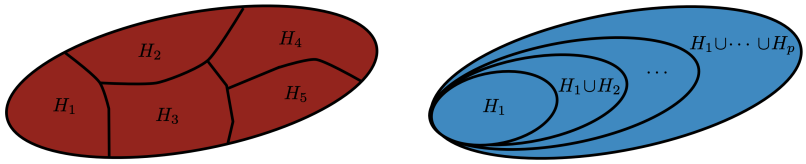


Figure 2.2: Base classifier set $\mathcal{H}$ decomposed into subfamilies $\mathcal{H}_1, \ldots, \mathcal{H}_p$ or their union [Cortes et al. 2014].

predictor from that class. The intuition is that whereas on some boosting rounds, extremely rich classes of functions may be selected, this need not happen at every step. In fact, one can achieve strong performance by using simple $\mathcal{H}_k$ in a significant fraction of the boosting steps and using rich families only on several occasions. This flexibility allows the boosting algorithm to find a balance between the complexity and the sample error of the algorithm at each boosting stage and, in fact, is motivated by the SRM principle.

Theorem 2.11 (Deep Boosting Generalization Bound)**.** *Assume $p > 1$. Fix $\rho > 0$. Then, for any $\delta > 0$, with a probability of at least $1 - \delta$ over the draw of a sample S of size m, the following inequality holds for all $f = \sum_{t=1}^{T} \alpha_t h_t \in \mathcal{F}$, where $\mathcal{F} = \operatorname{conv}\left(\cup_{k=1}^{p} \mathcal{H}_k\right)$*

$$R(h) \leq \widehat{R}_{S,\rho}(h) + \frac{4}{\rho} \sum_{t=1}^{T} \alpha_t \mathfrak{R}_m(\mathcal{H}_{k_t}) + C(m,p),$$

where $C(m,p) = O\left(\sqrt{\frac{\log p}{\rho^2 m}} \log\left(\frac{\rho^2 m}{\log p}\right)\right)$

The theorem above provides data-dependent learning guarantees for convex ensembles with multiple hypothesis sets $\{\mathcal{H}_1, \ldots, \mathcal{H}_p\}$. It is important to note that the complexity term in the bound admits an explicit dependency on the mixture coefficients α_t; that is, the complexity of each base hypothesis class $\mathfrak{R}_m(\mathcal{H}_{k_t})$ is weighted by the mixture weight α_t of the predictor that belongs to $\mathcal{H}_{k_t}$. Thus, the bound in Theorem 2.11 suggests that whereas hypothesis sets of extremely high complexity can be selected by the boosting algorithm, if their total α - ensemble weight is relatively

small, then the algorithm will still achieve favorable margin guarantees.

The learning scenario of deep boosting is extremely powerful while dealing with hypothesis classes of varying complexity. It will be used as a foundation for the analysis of learning with multiple regularized hypothesis spaces in Chapter 3 as well as with multiple data distributions in Chapter 4.

2.4 STRUCTURED PREDICTION

Many important machine learning tasks are instances of *structured prediction* problems. This area is an extension of the standard supervised classification learning scenario to problems beyond those with binary space $\mathcal{Y} = \{+1, -1\}$. These are learning problems where the output labels admit some structure that is important to consider both for statistical and computational reasons. Structured prediction problems include most natural language processing tasks, such as pronunciation modeling, part-of-speech tagging, context-free parsing, dependency parsing, machine translation, speech recognition, where the output labels are sequences of phonemes, part-of-speech tags, words, parse trees, or acyclic graphs, as well as other sequence modeling tasks in computational biology. They also include a variety of problems in computer vision such as image segmentation, feature detection, object recognition, motion estimation, computational photography, and many others.

We consider the supervised learning setting, in which the learner receives a labeled sample $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ drawn i.i.d. from some unknown distribution over $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$ denotes the input space and $\mathcal{Y}$ the output space. In structured prediction, we assume that the elements of the output space $\mathcal{Y}$ can be decomposed into possibly overlapping substructures $y = (y^1, \ldots, y^l)$. We further assume that the loss function $\mathsf{L} \colon \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$ can similarly be decomposed along these substructures. A straightforward subset of the structured prediction problems is *multi-class*

classification, where each substructure in the output space is of unit size and corresponds to a particular class label.

However, the nature of the label space in structured prediction problems is significantly richer than that in multi-class classification. Consider the example of *part-of-speech tagging*, where the task is to assign a part-of-speech (noun, verb, adjective, etc.) to every word in a given sentence. Thus, a sentence with words $w = w_1, \ldots, w_n$ will be associated with a sequence of part-of-speech tags $t_1, \ldots, t_n$, which can be viewed as the label of the sentence w. However, because each word in the sentence w can have any one of the part-of-speech tags, the label space for sentences is exponential in the length of the sentence. More generally, if Σ is the alphabet of part-of-speech tags, and the length of the source sentences is n, then the size of the label space is $|\Sigma|^n$, which corresponds to the number of possible tag sequences.

The critical aspect of most of the structured prediction problems is that the objective L is not simply a zero-one loss over the entire output sequence but a function that is decomposed along different substructures. Moreover, the penalties incurred for incorrect predictions can vary based on which substructure the prediction is taking place.

Some key examples of loss functions that are relevant to our work are the Hamming, n-gram, and edit-distance losses. The *Hamming loss* is defined for all $y = (y^1, \ldots, y^l)$ and $y' = (y'^1, \ldots, y'^l)$ by $\mathsf{L}(y, y') = \frac{1}{l} \sum_{k=1}^{l} 1_{y^k \neq y'^k}$, with $y^k, y'^k \in \mathcal{Y}_k$. The *edit-distance loss* is commonly used in natural language processing (NLP) applications where $\mathcal{Y}$ is a set of sequences defined over a finite alphabet, and the loss function between two sequences y and y' is defined as the minimum cost of a sequence of edit operations, typically insertions, deletions, and substitutions, that transform y into y'. The *n-gram loss* is defined as the negative inner product (or its logarithm) of the vectors of n-gram counts of two sequences. This can serve as an approximation to the BLEU [Papineni et al. 2002] score, which is commonly used in machine translation.

The goal of the learner in the structured prediction setting, similar to that in basic supervised classification, is to find an optimal predictor $h \in \mathcal{H}$ that minimizes the empirical loss on a given sample S; that is, $\widehat{R}_S(h) = \frac{1}{m}\sum_{i=1}^{m} \mathsf{L}(h(x_i), y_i)$.

The features that the prediction algorithm uses to learn from the input data typically depend both on the input and output. Thus, let $\mathbf{\Phi}(x, y)$ be the feature vector associated to the pair $(x, y) \in \mathcal{X} \times \mathcal{Y}$. Given the dependency of the features on both the input and output, the hypothesis space is described as

$$\forall x \in \mathcal{X} : h(x) = \operatorname*{argmax}_{y \in \mathcal{Y}} \mathbf{w} \cdot \mathbf{\Phi}(x, y) \tag{2.18}$$

for some real valued vector $\mathbf{w} \in \mathbb{R}^N$.

Given a labeled sample $S_m = ((x_1, y_1), \ldots, (x_m, y_m)) \in (\mathcal{X} \times \mathcal{Y})^m$, the learner in the structured prediction setting seeks to minimize an empirical objective function that is based on the average loss measure L over S_m. The empirical objective functions commonly used in structured prediction include *Maximum Margin Markov Networks* (M^3N) [Roller et al. 2004]:

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m} \max_{y \neq y_i} \max\Big(0, \mathsf{L}(y, y_i) - \mathbf{w} \cdot \big[\mathbf{\Phi}(x_i, y_i) - \mathbf{\Phi}(x_i, y)\big]\Big), \tag{2.19}$$

as well as *CRFs* [Lafferty et al. 2001]:

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m} \log \sum_{y \in \mathcal{Y}} \exp\Big(\mathsf{L}(y, y_i) - \mathbf{w} \cdot \big[\mathbf{\Phi}(x_i, y_i) - \mathbf{\Phi}(x_i, y)\big]\Big). \tag{2.20}$$

The M^3N objective involves an additive penalty of $\mathsf{L}(y, y_i)$ for margin violation. Such an objective is natural when the output space $\mathcal{Y}$ is endowed with a graph structure that satisfies a Markov property. For instance, $\mathcal{Y}$ can be a chain or a tree. By taking advantage of the Markov structure of the output label space $\mathcal{Y}$, it is possible to decompose the similarity measure $\mathsf{L}(y, y_i)$ along the

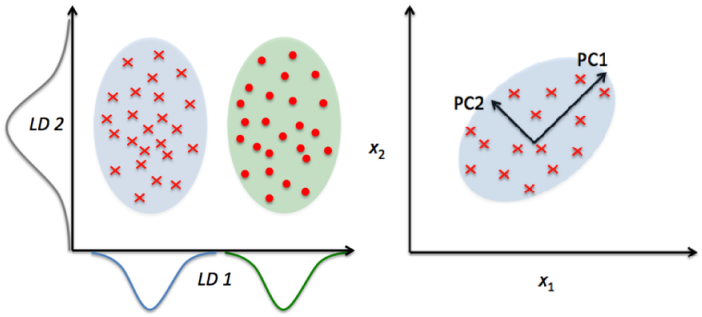


Figure 2.3: The application of principal component analysis and linear discriminant analysis for a sample of two-dimensional data. [Raj 2019].

substructures of $\mathcal{Y}$ and achieve a polynomial-time algorithm to solve the problem in 2.19.

The CRF objective in Equation 2.20 is obtained from the Maximum Margin Markov Networks objective in Equation 2.19 by replacing the $\max$ function with the *soft-max*, which is its smooth approximation, defined as $(x_1, \ldots, x_k) \mapsto \log\left(\sum_{j=1}^{k} e^{x_j}\right)$.

2.5 Dimensionality Reduction

For modern large-scale learning problems, storing a full representation of the input sample S_m of examples from the input space $\mathcal{X}$ is computationally intensive and often infeasible. In these situations, *dimensionality reduction* is often used to find a low-dimensional subspace (manifold) of $\mathcal{X}$ that retains the most important information about the sample S_m for the purpose of a statistical learning procedure.

Each dimensionality reduction method is geared towards constructing a low-dimensional manifold with specific properties. For example, *PCA* [Pearson 1901] defines the low-dimensional subspace as the span of the top eigenvectors of the data covariance matrix. Nonlinear dimensionality reduction methods, such as *Isometric Feature Mapping* [Tenenbaum et al. 2000] and *Locally Linear Embedding* [Roweis and Saul 2000] define a manifold that approximately preserves the

neighborhood properties of the sample points.

Dimensionality reduction is applied for two main reasons [Gönen 2014]: (i) exploratory data analysis and (ii) learning from the reduced data. Dimensionality reduction helps in exploratory analysis by allowing to visualize low-dimensional (typically $\mathbb{R}^2$ or $\mathbb{R}^3$) data representation or perform cluster analysis [Belkin and Niyogi 2003]. Learning, forecasting, and classification can be improved with dimensionality reduction because it enables data's inherent noise to be removed [Zhang 2009]. In this thesis, we are considering the *coupled dimensionality reduction* problem, where the goal of fitting a low-dimensional manifold is to provide the maximum benefit for the subsequent supervised learning on features from that manifold.

When determining the effective subspace $\mathbb{S}$ of data for the purpose of learning, it is shown that an efficient dimensionality reduction method would use not only the structure of features but also the labels of those features (or values of dependent variable). To that means, dimensionality reduction methods can be narrowed down to three types [Lin et al. 2011]: *unsupervised, supervised, and semi-supervised*.

- Unsupervised. Access to the output labels is not provided. Therefore, the learner cannot use them to guide the choice of the low-dimensional subspace $\mathbb{S}$. Rather, one has to rely on the intrinsic structure of explanatory variables. Examples of unsupervised dimensionality reduction include PCA [Pearson 1901], Isometric Feature Mapping [Tenenbaum et al. 2000], and Locally Linear Embedding [Roweis and Saul 2000].

- Supervised. Access to labels/classes or values of dependent variables is provided. Those labels/classes or values are used to guide the fitting of the low-dimensional subspace. For instance, if the labels of each training example are known, fitting a low-dimensional projection will benefit from optimizing the within- and between-class distances, as done in linear discriminant analysis. It is also possible to find an effective subspace $\mathbb{S}_j$ for each different

class j and then combine them, as performed in local discriminant embedding.

- Semi-Supervised. Access to labels or dependent variables is only partially provided. In such cases, one can preserve the intrinsic dimension structure on unlabeled points and combine it with supervised learning on labeled points. An example would be semi-supervised discriminant analysis [Cai et al. 2007].

The large-scale nature of modern machine learning problems is often evident in the dimension of the input space: $d = \dim(\mathcal{X})$. If $\mathcal{X} = \mathbb{R}^d$, then the dimensionality of $\mathcal{X}$ is easily understood. However, for some complex input spaces, it is not straightforward to conceptualize what the dimension of $\mathcal{X}$ is. Thus, one of the critical concepts that exist in machine learning research is the so-called *intrinsic* (sometimes *effective*) dimension of data. Intuitively, the intrinsic dimension is the dimension of the manifold onto which the data can be projected without losing essential information. The common methods of measuring the intrinsic dimension are *PCA*, *Multidimensional Scaling*, and statistical methods based on stochastic processes [Levina and Bickel 2004]. When an estimate of the intrinsic dimension $d^\star$ of the input data is obtained, the learner would project the data onto a $d^\star$-dimensional manifold and then execute a learning algorithm within the projected space.

According to PCA, the intrinsic dimension of the input data is the number of covariance matrix eigenvalues that are larger than some threshold (in fact, that is the dimension of the subspace spanned by top-k eigenvectors). However, the subspace produced by the top-k eigenvectors is linear. Constructing a nonlinear subspace of a specific dimension $d^\star$ provides better discriminatory properties for most dataset types. Such a nonlinear subspace can be constructed using a nonlinear extension of the standard PCA—*Kernel PCA (KPCA)* [Schölkopf et al. 1997]. The key difference of KPCA is that in this procedure, each example x_i is mapped onto a high dimensional Hilbert space $\mathbb{H}$ with a mapping $\Phi(x_i) : \mathcal{X} \to \mathbb{H}$. It is essential that the image space is Hilbert, which

is endowed with an inner product and satisfies the completeness property. Whereas in standard PCA, the orthogonal projection is applied in the input space $\mathcal{X}$, in Kernel PCA, it is carried out in the image space $\mathbb{H}$; that is, the data are projected onto top-k orthonormal eigenvectors of the covariance matrix of images $y_i = \Phi(x_i)$.

2.5.1 KERNEL METHODS AND KERNEL PCA

As articulated in the literature that the dimensionality reduction techniques just mentioned and most others are specific instances of the KPCA algorithm [Ham et al. 2004] for different choices of a kernel function. An even broader view of dimensionality reduction techniques is that they first map input points to the *Reproducing Kernel Hilbert Space (RKHS)* of some *positive semi-definite (PSD)* kernel K and next project vectors onto a low-dimensional subspace of that Hilbert space. Because KPCA is so fundamental to the understanding of dimensionality reduction techniques and, in fact, strictly generalizes most popular methods, we provide a detailed description of the KPCA algorithm as well as summarize the key properties of the Hilbert space that is associated to it.

For simplicity, let $\mathcal{X} = \mathbb{R}^n$, from which a labeled sample $S_m = \{(x_i, y_i), \ldots, (x_m, y_m)\}$ is selected. Assuming that the data are centered, define a covariance matrix $C = \frac{1}{m}\sum_{i=1}^{m} x_i x_i^\top$. The solution of the eigenvalue problem for the covariance matrix $C : \lambda V = CV$ results in eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_m$ and orthonormal eigenvectors $v^1, \ldots, v^m$, corresponding to the ordered eigenvalues. Standard PCA selects top r eigenvalues and projects every x_i onto span $(v^1, \ldots, v^r)$, effectively reducing the dimension of the data to the dimension of the span of the eigenvectors, which is equal to r. Denote such orthogonal projection as Px_i. Given a predictor h, examples are classified using $h(Px_i) \to \{-1, 1\}$.

As described, PCA reduces the data's dimension while keeping most of the variance within the projected subspace. However, although PCA efficiently captures the data's linear structure, it can

fail in cases when the data are embedded in some nonlinear manifold. In such cases, KPCA is applied, where the data are mapped with $\Phi : \mathbb{R}^n \to \mathbb{H}$ into some higher-dimensional space. The space $\mathbb{H}$ is typically referred to as a *feature space*, and Φ is termed a *feature mapping*. The image of the data in $\mathbb{H}$ will have an (approximately) linear structure, which allows to perform PCA in the image space $\mathbb{H}$ instead of $\mathbb{R}^n$. The discriminatory properties of the projection in $\mathbb{H}$ critically depend on the way the inner product is defined in that space, which is carried out using the *kernel* function.

Definition 2.12 (Kernel)**.** A function $K : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ is referred to as a *kernel* over $\mathcal{X}$.

In a particular case, when $\mathcal{X} = \mathbb{R}^n$, the kernel is a functional over the space of real-valued vectors of dimension n.

Definition 2.13 (Positive Definite Symmetric Kernels)**.** A *kernel* $K : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ is said to be positive definite symmetric (PDS) if for any $\{x_1, \ldots, x_m\} \subset \mathcal{X}$, the matrix $\mathbf{K} = [K(x_i, x_j)]_{i,j} \in \mathbb{R}^{m \times m}$ is symmetric positive semidefinite (SPSD).

The kernel matrix $\mathbf{K}$ is SPSD given that it is symmetric and satisfies one of the two equivalent conditions below:

- all eigenvalues of $\mathbf{K}$ are non-negative;
- given any vector $\boldsymbol{c} = [c_1, \ldots, c_m]^\top \in \mathbb{R}^{m \times 1}$, the following inequality holds:

$$\boldsymbol{c}^\top \mathbf{K} \boldsymbol{c} = \sum_{i,j=1}^{n} c_i c_j K(x_i, x_j) \geq 0 \tag{2.21}$$

Widely used examples of PDS kernel functions include *Gaussian kernel*

$$K_{\text{Gaussian}}(x, y) = \exp\bigg(- \frac{\|x - y\|}{2\sigma^2} \bigg), \tag{2.22}$$

Sigmoid kernel

$$K_{\text{Sigmoid}}(x, y) = \tanh\big(a(x \cdot y) + b\big), \tag{2.23}$$

and *Polynomial kernel*

$$K_{\text{poly}}(x, y) = \big(x \cdot y + c\big)^d. \tag{2.24}$$

A fundamental result in functional analysis guarantees that a Hilbert space can be constructed using the kernel function as an inner product [Aronszajn 1950].

Theorem 2.14 (Reproducing Kernel Hilbert Space)**.** *Let $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a PDS Kernel; then, there exists a Hilbert space $\mathbb{H}$ and a mapping $\Phi : \mathcal{X} \mapsto \mathbb{H}$ such that*

$$(\forall x, y \in \mathcal{X})\, K\left(x, y\right) = \left\langle \Phi\left(x\right), \Phi\left(y\right) \right\rangle$$

$\mathbb{H}$ is termed an RKHS associated to kernel K.

The Hilbert space $\mathbb{H}$ constructed in such a way is characterized by the *reproducing property*:

$$\forall h \in \mathbb{H}, \forall x \in \mathcal{X} : h(x) = \langle h, K(x, \cdot) \rangle. \tag{2.25}$$

Theorem 2.15 (Rademacher complexity of kernel-based hypotheses)**.** *Let $K : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ be a PDS kernel and $\Phi : \mathcal{X} \mapsto \mathbb{H}$ be a feature mapping associated to K. Let $S_m \subset \{x : K(x, x) \leq r^2\}$ be a sample of size m and let $\mathcal{H} = \{x \mapsto \langle \mathbf{w}, \Phi(x) \rangle : \|\mathbf{w}\|_{\mathbb{H}} \leq \Lambda\}$ for some $\Lambda \geq 0$; then, the following Rademacher complexity bound holds:*

$$\widehat{\mathfrak{R}}_S(\mathcal{H}) \leq \frac{\Lambda \sqrt{\text{Tr}[\mathbf{K}]}}{m} \leq \sqrt{\frac{r^2 \Lambda^2}{m}} \tag{2.26}$$

Theorem 2.16 (Margin bounds for kernel-based hypotheses)**.** *Let $K : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ be a PDS kernel with $r^2 = \sup_{x \in \mathcal{X}} K(x, x)$. Let $\Phi : \mathcal{X} \mapsto \mathbb{H}$ be the feature mapping associated to K and let $\mathcal{H} = \{x \mapsto \langle \mathbf{w}, \Phi(x) \rangle : \|\mathbf{w}\|_{\mathbb{H}} \leq \Lambda\}$ for some $\Lambda \geq 0$. Fix $\rho > 0$; then, for any $\delta > 0$, the following inequalities hold with a probability of at least $1 - \delta$ for any $h \in \mathcal{H}$:*

$$R(h) \leq \widehat{R}_{S,\rho}(h) + 2\sqrt{\frac{r^2\Lambda^2/\rho^2}{m}} + \sqrt{\frac{\log \frac{1}{\delta}}{2m}}, \tag{2.27}$$

$$R(h) \leq \widehat{R}_{S,\rho}(h) + 2\frac{\sqrt{\mathrm{Tr}[\mathbf{K}]\Lambda^2/\rho^2}}{m} + 3\sqrt{\frac{\log \frac{2}{\delta}}{2m}}. \tag{2.28}$$

3 | REGULARIZED GRADIENT BOOSTING

Gradient Boosting (GB) is a popular and very successful ensemble method for binary trees. While various types of regularization of the base predictors are used with this algorithm, the theory that connects such regularizations with generalization guarantees is poorly understood. We fill this gap by deriving data-dependent learning guarantees for GB used with *regularization*, expressed in terms of the Rademacher complexities of the constrained families of base predictors. We introduce a new algorithm, called RGB, that directly benefits from these generalization bounds and that, at every boosting round, applies the *Structural Risk Minimization* principle to search for a base predictor with the best empirical fit versus complexity trade-off. Inspired by *Randomized Coordinate Descent* we provide a scalable implementation of our algorithm, able to search over large families of base predictors. Finally, we provide experimental results, demonstrating that our algorithm achieves significantly better out-of-sample performance on multiple datasets than the standard GB algorithm used with its regularization.

3.1 BACKGROUND

Ensemble methods form a powerful family of techniques in machine learning that combine multiple base predictors to create more accurate ones. These methods are often very effective in practice and can achieve a significant performance improvement over the individual base predictors [Quin-

lan et al. 1996; Caruana et al. 2004; Freund et al. 1996; Dietterich 2000]. ADABOOST [Freund and Schapire 1997b] and its variants are among the most prominent ensemble methods since they are both very effective in practice and benefit from well-studied theoretical margin guarantees [Freund and Schapire 1997b; Koltchinskii and Panchenko 2002b].

Gradient Boosting (GB) [Friedman 2001] is another popular tree-based ensemble method that has inspired a number of widely-used software libraries (e.g., XGBOOST [Chen and Guestrin 2016], MART [Friedman 2002], and DART [Rashmi and Gilad-Bachrach 2015]) and has frequently ranked among the top in benchmark competitions such as *Kaggle*. But, while it is often introduced and presented differently, GB exactly coincides with AdaBoost, when the objective function used is the exponential function, as shown for example by [Schapire and Freund 2012]. More generally, both of these algorithms are instances of *Functional Gradient Descent* [Mason et al. 2000; Grubb and Bagnell 2011] when non-increasing convex and differentiable upper bounds on the zero-one loss are used. Viewed from the Functional Gradient Descent perspective, at every boosting step, GB seeks a predictor function h that is closest to the *functional gradient* of the objective within some constrained family of base predictors $\mathcal{H}$. Specifying this base predictor family $\mathcal{H}$ such that the selected function does not overfit the gradient, as well as defining an efficient search procedure over $\mathcal{H}$ is crucial for the success of the algorithm. In most practical instances, several types of constraints are imposed to do so. As an example, for binary regression trees, XGBOOST bounds the number of leaves and the norm of the leaf values vector. This can be viewed as a *regularization*. However, to our knowledge, no theoretical analysis has been provided for these commonly-used constraints.

A natural question is whether one can derive learning guarantees that explain how this regularization on $\mathcal{H}$, and, perhaps even more general forms of constraints on functions $h \in \mathcal{H}$, are connected to the generalization performance of GB. We seek inspiration from the margin-based

learning bounds given for ADABOOST [Schapire et al. 1997a; Mohri et al. 2012]. These guarantees, however, do not provide a detailed analysis of the constraints on the families of tree base predictors, nor do they provide guidance on how to conduct an efficient search of these families to select a predictor during each boosting round.

We fill this gap by providing a comprehensive analysis of regularization in GB and derive learning guarantees that explain what type of regularization should be used and how. We give data-dependent learning bounds for GB with regularization, expressed in terms of the Rademacher complexities of the constrained hypotheses' sub-families, from which the base predictors are selected, as well as the ensemble mixture weights. We present a new algorithm, called RGB for Regularized Gradient Boosting, which generalizes the existing gradient boosting methods by introducing a general functional q-norm constraint for the families of the tree base predictors.

Our algorithm and its objective function are directly guided by the theory we develop. Our bound suggests that the *Structural Risk Minimization* principle (SRM) [Vapnik 1992] should be used to break down $\mathcal{H}$ into subsets of varying complexities and, at each round, select a base learner h from a subset that provides the best trade-off between proximity to the functional gradient and the complexity.

Applying SRM to search over subsets of $\mathcal{H}$ is challenging, since often these subsets are extremely rich, possibly infinite. An example is the families of decision trees with bounded depth used in GB. We provide a solution to the problem of expensive search and show how *Randomized Coordinate Descent* [Nesterov 2012] can be used to search over $\mathcal{H}$ efficiently, using our generalization bounds.

Finally, this paper provides experimental results, demonstrating that our algorithm achieves significantly better out-of-sample performance than the baselines such as XGBOOST on multiple datasets. We give specific bounds, as well as the pseudocode and experimental results, for the fam-

ilies of binary regression trees, but our analysis can be extended to broader families of functions, such as SVMs [Cortes and Vapnik 1995] and Deep Neural Networks [LeCun et al. 2015].

The paper is organized as follows. In Section 3.2, we introduce what we name a *Regularized Gradient Boosting* framework. In Section 3.3 we derive a Rademacher complexity bound on the families of regularized regression trees, which allows us to establish learning guarantees for Regularized Gradient Boosting. This bound directly inspire the optimization objective and the RGB algorithm presented in Section 3.4 that benefits from the guarantees following from the SRM principle. A non-uniform randomized search over the families of base predictors provides an efficient solution. In Section 3.5, we present our experimental results, which illustrate the benefits of the RGB algorithm.

3.2 Regularized Gradient Boosting

In this section, we examine the correspondence between gradient descent in functional spaces and coordinate descent in vector spaces. This connection will help us rigorously define a Regularized Gradient Boosting learning scenario and develop a scalable implementation for it.

3.2.1 Gradient Boosting as Functional Gradient Descent

Let $\mathcal{X}$ denote the input space, and let $\mathcal{F}$ be an inner product space of functions from $\mathcal{X}$ to $\mathbb{R}$. We define a restricted family of functions $\mathcal{H} \subseteq \mathcal{F}$ to be a set of base hypotheses. In a standard supervised learning scenario, the training and test points are drawn i.i.d. according to some distribution $\mathcal{D}$ over $\mathcal{X} \times \{-1, 1\}$, and $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ is a training sample of size m drawn from $\mathcal{D}^m$. In this scenario, a general boosting algorithm selects a sequence of functions $h_1, \ldots, h_T$ from $\mathcal{H}$ to minimize a certain empirical loss $\mathcal{L}\colon \mathcal{F} \mapsto \mathbb{R}$. [Friedman 2001; Grubb and Bagnell 2011;

Mason et al. 2000; Schapire 1999; Cortes et al. 2014]. The specification of $\mathcal{H}$ and the method of selecting each $h_t \in \mathcal{H}$ are essential for the success of the boosting algorithms. In fact, different answers to these two questions have resulted in distinct and separately-studied algorithms, such as GB, ADABOOST, and LOGITBOOST [Friedman et al. 1998].

The goal of boosting algorithms is typically to minimize an empirical loss functional:

$$\mathcal{L}(F) = \frac{1}{m}\sum_{i=1}^{m} \Phi\Big(y_i, F(x_i)\Big), \tag{3.1}$$

where $F(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$ such that $\forall t \in [1, T] : h_t \in \mathcal{H}$. Popular ensemble learning algorithms, such as ADABOOST and GB, despite having originated in different research communities at different times, are particular instances of a more general algorithm, Functional Gradient Descent. The objective in Equation 3.1 is viewed by the Functional Gradient Descent as a functional rather than a vector-valued function, with the goal of minimizing $\mathcal{L}$ over $\mathcal{F}$ by taking steps in the direction of the steepest descent $F \leftarrow F - \eta\nabla\mathcal{L}(F)$ for some positive learning rate $\eta \in \mathbb{R}$. In the learning scenario described above, only the trace of F on $x_1, \ldots, x_m$ is observable; therefore, the functional gradient of $\mathcal{L}$ is $\nabla\mathcal{L} = \left[\frac{\partial\mathcal{L}(F)}{\partial F(x_1)}, \ldots, \frac{\partial\mathcal{L}(F)}{\partial F(x_m)}\right]$. This makes the Functional Gradient Descent update equal to $F(x_i) \leftarrow F(x_i) - \eta\frac{\partial\mathcal{L}(F)}{\partial F(x_i)}$. Of course, to make sure this functional update is well defined and to avoid over-fitting, it is natural to restrict $F(x)$ to some hypothesis set $\mathcal{H}$, which implies the following form of the functional update:

$$h = \operatorname*{argmin}_{h\in\mathcal{H}} d(\nabla\mathcal{L}, h), \tag{3.2}$$

where d is some distance measure. This means that $h \in \mathcal{H}$ is chosen to be the closest function $h \in \mathcal{H}$ to the projection of $\nabla\mathcal{L}$ onto $\mathcal{H}$. The update in Equation 3.2 is a fundamental but not well-studied component of virtually all boosting methods. Simply by varying the choice of $\mathcal{H}$ and

d, this single equation recovers most widely-used boosting algorithms.

If we restrict the optimization steps to a set of base hypotheses $\mathcal{H}$, then each step is chosen to be the function closest in the direction to the negative gradient, which means it maximizes

$$-\nabla\mathcal{L}\cdot h = -\sum_{i=1}^{m}\frac{\partial\mathcal{L}(F)}{\partial F(x_i)}h(x_i). \tag{3.3}$$

Particularly, if $\Phi(y_i, f(x_i)) = e^{-y_i f(x_i)}$, then the Functional Gradient Descent recovers the original ADABOOST algorithm, and if $\Phi(y_i, f(x_i)) = \log\left(1 + e^{-y_i f(x_i)}\right)$, then it recovers LOGITBOOST. When, instead of the negative inner product $-\nabla\mathcal{L}\cdot h$, we minimize the distance $\|-\nabla\mathcal{L}(F) - h\|_2^2$, we recover the GB algorithm.

3.2.2 GRADIENT BOOSTING AS VECTOR SPACE COORDINATE DESCENT

There is an equivalence relation between gradient descent in functional spaces and coordinate descent in vector spaces that often helps to obtain efficient algorithms for ensemble learning. At each of the T steps of the Functional Gradient Descent, $\nabla\mathcal{L}$ is projected onto $\mathcal{H}$, hence the final solution F can be expressed as $F_{\boldsymbol{\alpha}} = \sum_{t=1}^{T}\alpha_t h_t$ for some $\boldsymbol{\alpha}\in\mathbb{R}^T$, where $\forall 1\le t\le T : h_t \in \mathcal{H}_{I_t}\subseteq\mathcal{H}$, where $\mathcal{H}_{I_t}$ indicates the subset of $\mathcal{H}$ selected at the t-th step. The subsets $\mathcal{H}_{I_t}$ can be viewed as coordinate blocks in $\mathcal{H}$. In this view, at boosting step t a particular subspace $\mathcal{H}_{I_t}$ out of $\{\mathcal{H}_1,\ldots,\mathcal{H}_K\}$ is selected; then a base predictor $h_t\in\mathcal{H}_{I_t}$ from that subspace is added to the ensemble.

This allows switching from minimizing the loss *functional* $\mathcal{L}(F)$ to minimizing the loss function $L(\boldsymbol{\alpha}) = \mathcal{L}(F_{\boldsymbol{\alpha}})$.

$$L(\boldsymbol{\alpha}) = \frac{1}{m}\sum_{i=1}^{m}\Phi\left(y_i, \sum_{t=1}^{T}\alpha_t h_t(x_i)\right) \tag{3.4}$$

over the ensemble weights vector $\boldsymbol{\alpha}\in\mathbb{R}^T$. Selecting a projection h_t and a step size α_t on the t-th

step of the Functional Gradient Descent on $\mathcal{L}(F_{\boldsymbol{\alpha}})$ or alternatively selecting a coordinate α_t on the t-th step of the vector space coordinate descent on $L(\boldsymbol{\alpha})$ both result in the same form of the update $F_{\boldsymbol{\alpha},t} = F_{\boldsymbol{\alpha},t-1} + \alpha_t h_t$. Additionally, the full sequence of these updates for t from 1 to T is equal since, by the chain rule

$$\forall 1 \leq t \leq T : -\frac{\partial L(\alpha_t)}{\partial \alpha_t} = -\sum_{i=1}^{m} \frac{\partial \mathcal{L}(F_{\boldsymbol{\alpha}})}{\partial F_{\boldsymbol{\alpha}}(x_i)} h_t(x_i) = -\nabla \mathcal{L} \cdot h_t, \tag{3.5}$$

which means that $\min_{1 \leq t \leq T} -\frac{\partial L(\alpha_t)}{\partial \alpha_t}$ selected by the coordinate descent is equal to $\min_{1 \leq t \leq T} -\nabla \mathcal{L} \cdot h_t$ selected by Functional Gradient Descent.

This equivalence illustrates two important points. First, coordinate descent methods can be used to provide efficient numerical solutions for boosting. Second, the proper construction of the subsets $\mathcal{H}_t$ such that $h_t \in \mathcal{H}_{I_t} \subseteq \mathcal{H}$ is crucial for the success of boosting algorithms. We rely on this equivalence when presenting a coordinate-descent-style algorithm for minimizing the regularized boosting objective that scales well to large families of base predictors.

3.2.3 Regularized Gradient Boosting

In this subsection, we describe the main novelty of our work – the analysis of regularization applied to GB. We formulate what we name a *Regularized Gradient Boosting* framework and show the subtle connection between the regularization and the properties of $\mathcal{H}_k \subseteq \mathcal{H}$. As we shall see, the regularization terms are not explicitly introduced in the definition of the objective, but only in the definition of an approximation to the functional gradient.

While the *unregularized* projection step, as in Equation 3.2, has been extensively studied for GB, the fundamental theory of the regularization commonly used is missing. However, a number of empirical studies and software frameworks [Sun et al. 2014; Chen and Guestrin 2016] indicate

that introducing regularization to this step is extremely beneficial. For example, the popular XGBOOST library, dedicated to boosted decision trees, regularizes the norm of the leaf values, as well as the number of leafs. We are filling this gap by providing a theory that links regularization with learning guarantees for GB algorithms.

For a convex function $\Omega\colon \mathcal{F} \mapsto \mathbb{R}$, a closed subspace $\mathcal{H} \subseteq \mathcal{F}$ and $\beta \in \mathbb{R}_+$, let the Regularized Gradient Boosting step be defined by

$$h = \operatorname*{argmin}_{h \in \mathcal{H}} d(\nabla \mathcal{L}, h) + \beta \Omega(h). \tag{3.6}$$

Given the convexity of Ω, this step is equivalent to $h = \operatorname{argmin}_{h \in \widehat{\mathcal{H}}} d(\nabla \mathcal{L}, h)$, where $\widehat{\mathcal{H}} = \mathcal{H} \cap \{h\colon \Omega(h) \leq \beta\}$. Such a reduction illustrates the subtle, yet extremely important, connection between regularization and the definition of hypothesis set $\mathcal{H}$. The equivalence between vector space coordinate descent and Functional Gradient Descent presented in Section 3.2, meaning that both of these methods iteratively select the same sequence of functions $h_t \in \mathcal{H}_{I_t} \subseteq \mathcal{H}$, suggests that a natural way to use regularization for boosting is to define $\mathcal{F} = \operatorname{conv}(\cup_{k=1}^{K} \mathcal{H}_k)$, where $\mathcal{H}_k = \{h\colon \theta_{k-1} < \Omega(h) \leq \theta_k\}$ are disjointed sets of functions for a set of parameters $[\theta_1, \ldots, \theta_K]$. Note that, with this formulation, the regularization is not in the objective function; instead the search for the gradient approximation is constrained by a regularization.

We show, in the following section, that such a definition of $\mathcal{F}$ allows us to obtain margin-based learning guarantees for the Regularized Gradient Boosting that are dependent on the complexities of each individual $\mathcal{H}_k$.

3.3 LEARNING GUARANTEES

As described in the previous section, by projecting the functional gradient onto the subspace $\mathcal{F} = \text{conv}(\cup_{k=1}^{K} \mathcal{H}_k)$ at each step, we are able to learn an ensemble function $f = \sum_{t=1}^{T} \alpha_t f_t \in \mathcal{F}$, where the $\mathcal{H}_k$s are families of functions with varying complexity. Thus, it is natural to seek learning guarantees depending on the properties of each $\mathcal{H}_k$ and the mixture weight vector $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_T]$.

The first margin bound based on the VC-dimension for ensembles $\sum_{t=1}^{T} \alpha_t f_t$ was given by [Freund and Schapire 1997b]. Later, tighter data-dependent bounds in terms of the Rademacher complexity of the underlying function class $\mathcal{H}$ were given by [Koltchinskii and Panchenko 2002b], see also [Mohri et al. 2018b]. For the specific case where $\mathcal{H} = \text{conv}(\cup_{k=1}^{K} \mathcal{H}_k)$, Rademacher complexity-based guarantees were given in [Cortes et al. 2014]. In this section, we will use these theoretical results to derive margin-based guarantees based on the Rademacher complexities of the families of regularized decision trees $\mathcal{H}_k$ and the mixture weights $\boldsymbol{\alpha}$. The bounds that we show, being data-dependent, will not only fill the missing generalization theory for the existing gradient tree boosting frameworks but also motivate a new scalable learning algorithm for the Regularized Gradient Boosting framework, called RGB, in Section 3.4.

Here, we restrict our analysis to the hypothesis families $\mathcal{H}_k$ of regression trees. However, our results can be extended to other families, such as kernel-based hypotheses and neural networks, so long as the sample Rademacher complexities of these families can be bounded.

Each leaf l in a regression tree contains a real-valued number w_l providing the output value of the tree for any sample point allocated to that leaf; thus, we let $\mathbf{w}$ be a vector of stacked leaf values. The function computed by a regression tree can thus be represented by $h(x) = \sum_{l \in \text{leaves(h)}} w_l \mathbb{I}\, x \in \text{leaf}_l$, where $\mathbb{I}\, x \in \text{leaf}_l$ is the indicator function for sample point $x \in \mathbb{R}^d$ being allocated to leaf_l; this value $h(x)$ can be used for classification in a straightforward manner by

thresholding.

The node partition functions in binary regression trees are of the form $[x]_j \leq \theta$ for some feature index $j \in [1, d]$ and $\theta \in \mathbb{R}$, which means that if $[x_i]_j \leq \theta$ for a sample point $x_i \in \mathbb{R}^d$, then x_i is allocated to the left subtree and to the right subtree otherwise. Let $\mathcal{H}_{n,\lambda,q}$ be the set of all *regularized* binary regression trees with the number of internal nodes bounded by n and a leaf values vector $\mathbf{w}$ such that $\|\mathbf{w}\|_q \leq \lambda$, $q \geq 1$. Special instances of these families of trees are widely used in practice. For example, $\mathcal{H}_{n,\lambda,1}$ and $\mathcal{H}_{n,\lambda,2}$ are implemented in XGBOOST and frequently implemented in software libraries.

Theorem 3.1. *For any sample $S = (x_1, \ldots, x_m)$, the empirical Rademacher complexity of a hypothesis set $\mathcal{H}$ is defined by $\widehat{\mathfrak{R}}_S(\mathcal{H}) = \mathbb{E}_{\boldsymbol{\sigma}}\left[\sup_{h \in \mathcal{H}} \sum_{i=1}^m \sigma_i h(x_i)\right]$, where, σ_is, $i \in [m]$, are independent uniformly distributed random variables taking values in $\{-1, 1\}$. Let d be the input data dimension. The following upper bound holds for the empirical Rademacher complexity of $\mathcal{H}_{n,\lambda,q}$:*

$$\widehat{\mathfrak{R}}^S(\mathcal{H}_{n,\lambda,q}) \leq \lambda \sqrt{\frac{(4n+2)\log_2(d+2)\log(m+1)}{m}}.$$

The proof of Theorem 3.1 is given in the Appendix. This bound shows how the empirical Rademacher complexity of the regularized decision trees depends both on on the number of internal nodes n and the upper bound λ on the q-norm of leaf values.

Using this bound, we can now derive our margin-based learning guarantees for the family $\mathcal{F}$. Let $R(f)$ denote the binary classification error of $f \in \mathcal{F}$, $R(f) = \mathbb{E}_{(x,y)\sim\mathcal{D}} \mathbb{I}\, yf(x) \leq 0$, and $R_\rho(f)$ its empirical ρ-margin loss for a sample S, $R_\rho(f) = \mathbb{E}_{(x,y)\sim\mathcal{D}} \mathbb{I}\, yf(x) \leq \rho$. Let $\widehat{R}_\rho(f) = \mathbb{E}_{(x,y)\sim S} \mathbb{I}\, yf(x) \leq \rho$.

Theorem 3.2. *Fix $\rho > 0$. Let $\mathcal{H}_k = \mathcal{H}_{n_k,\lambda_k,q_k}$, where (n_k), (λ_k) are sequences of constraints on the number of internal nodes n and the leaf vector norm $\|\mathbf{w}\|_q$. Define $\mathcal{F} = \operatorname{conv}(\cup_{k=1}^K H_k)$. Then,*

for any $\delta > 0$, with probability at least $1 - \delta$ over the draw of a sample S of size m, the following inequality holds for all $f = \sum_{t=1}^{T} \alpha_t h_t \in \mathcal{F}$:

$$R(f) \leq \widehat{R}_{S,\rho}(f) + \frac{4}{\rho} \sum_{t=1}^{T} \alpha_t \lambda_{I_t} \sqrt{\frac{(4n_{I_t} + 2) \log_2(d+2) \log(m+1)}{m}} + C(m, K),$$

where I_t is the index of the subclass selected at time t and $C(m, K) = O\left(\sqrt{\frac{\log(K)}{\rho^2 m} \log\left[\frac{\rho^2 m}{\log(K)}\right]}\right)$.

The proof of Theorem 3.2 is given in the Appendix. The generalization bound of Theorem 3.2 motivates a specific algorithm for Regularized Gradient Boosting, described and discussed in the next section.

3.4 ALGORITHM

The multiplicative structure of the bound in Theorem 3.2 with respect to the mixture weights $[\alpha_1, \ldots, \alpha_T]$ and the complexities $\mathcal{H}_{I_t}$ suggests the use of these complexities (or upper bounds) in the regularization $\Omega(h)$. Additionally, one may upper-bound the empirical loss function of $u \mapsto \mathbb{I}\, u \leq 0$ in Theorem 3.2, leading to the following objective:

$$L(\boldsymbol{\alpha}) = \frac{1}{m} \sum_{i=1}^{m} \Phi\left(y_i, \sum_{t=1}^{T} \alpha_t h_t(x_i)\right) + \beta \sum_{t=1}^{T} |\alpha_t| \lambda_{I_t} \sqrt{\frac{(4n_{I_t} + 2) \log_2(d+2) \log(m+1)}{m}}. \quad (3.7)$$

Minimizing the function with vector space coordinate descent is equivalent to solving for a projection at each Functional Gradient Descent step of the form

$$h_t = \operatorname*{argmin}_{h \in \mathcal{H}} d(\nabla \mathcal{L}, h) + \beta \sum_{k=1}^{K} \lambda_k \sqrt{\frac{(4n_k + 2) \log_2(d+2) \log(m+1)}{m}}\, \mathbb{I}\, h \in \mathcal{H}_k. \quad (3.8)$$

In this section we will devise an algorithm for minimizing the regularized objective $L(\boldsymbol{\alpha})$, called RGB, that is able to scale to large families of base predictors.

3.4.1 Randomized Coordinate Descent

The practical challenge of building an ensemble of base predictors in the Regularized Gradient Boosting scenario is to both define the hypothesis sets $\mathcal{H}_k$ and implement an efficient search across these sets to select the best update direction h_t, at each optimization step. Applying coordinate descent to the objective in Equation 3.7 may be feasible for finite hypothesis sets; however, we are often required to work with infinite spaces of subfamilies of functions. A typical example would be one where each subfamily is a decision tree with a fixed topology and fixed leaf values. It is common to resort to heuristics or to discretize the search space to define an approximate search.

To solve the problem of an extensive search over $\mathcal{H}_k$, we propose a novel method for boosting updates using randomization applied to the functional space. Random selection of base learners for GB in the context of *Randomized Coordinate Descent* has been shown to be successful in practice. For example, [Lu and Mazumder 2018] demonstrated that uniform sampling helps make the search over base hypothesis classes more scalable, gave favorable convergence guarantees for this method. [Nesterov 2012] introduced probabilistic convergence guarantees for Randomized Coordinate Descent expressed in terms of the local smoothness properties of the objective and suggested a distribution to sample the coordinates.

Inspired by the analysis of [Nesterov 2012], our work is the first one to provide a method of searching over the subspaces $\mathcal{H}_k$ that is fundamentally-justified, an algorithm that is both scalable and admits convergence guarantees. The RGB algorithm picks at each round at random a subset $\{\mathcal{H}_{t_1}, \ldots, \mathcal{H}_{t_S}\}$. Given a meaningful distribution over $\mathcal{H}$ that captures the steepness of the objective $L(\boldsymbol{\alpha})$ within each of these subsets, RGB is able to learn an ensemble of functions

from families $\mathcal{H}_k$ of varying complexity. In the following, we show how to apply the Randomized Coordinate Descent method, as in [Nesterov 2012], to the objective 3.7.

3.4.2 Lipschitz-Continuous Gradients

Consider the problem of minimizing $L(\boldsymbol{\alpha})$ as in Equation 3.7. The following lemma describes the continuity properties of the partial derivatives of $L(\boldsymbol{\alpha})$, which are needed for the application of Randomized Coordinate Descent.

Lemma 3.3. *Assume that $\Phi(y, h)$ is differentiable with respect to the second argument, and that $\frac{\partial \Phi}{\partial h}$ is $C_\Phi(y)$-Lipschitz with respect to the second argument, for any fixed value y of the first argument. For all $k \in [0, K]$, define $L'_k(\boldsymbol{\alpha}) = \frac{\partial L}{\partial \alpha_k}$. Then, $L'_k(\boldsymbol{\alpha})$ is C_k-Lipschitz with C_k bounded as follows:*

$$C_k \leq \frac{1}{m} \sum_{i=1}^{m} h_k^2(x_i) C_\Phi(y_i). \tag{3.9}$$

Randomized Coordinate Descent samples the k-th coordinate with probability

$$p_k = C_k / \sum_{k=1}^{K} C_k. \tag{3.10}$$

The convergence guarantees for this procedure are given in [Nesterov 2012] as a function of the Lipschitz constants C_k.

We can further give upper bounds for the Lipschitz constants above to avoid the computation of the sums $\sum_{i=1}^{m} h_k^2(x_i)$. If we introduce the vectors $\mathbf{h}_k$ and $\mathbf{C}_\Phi$ in $\mathbb{R}^m$ such that $[\mathbf{h}_k]_i = h_k^2(x_i)$ and $[\mathbf{C}_\Phi]_i = C_\Phi(y_i)$, then, by Hölder's inequality,

$$C_k \leq \frac{1}{m} \sum_{i=1}^{m} h_k^2(x_i) C_\Phi(y_i) = \frac{1}{m} \mathbf{h} \cdot \mathbf{C}_\Phi \leq \frac{1}{m} \|\mathbf{h}\|_r \|\mathbf{C}_\Phi\|_q, \tag{3.11}$$

where $\frac{1}{r} + \frac{1}{q} = 1$. Various (r, q)-dual norms can be used, depending on the computational constraints and the complexity of the hypothesis classes for the application of the Randomized Coordinate Descent method. For example, using $\|\mathbf{h}\|_1$ and $\|\mathbf{C}_\Phi\|_\infty$ gives the following upper bound: $C_k \leq \frac{1}{m}\big[\max_{1\leq i\leq m} C_\Phi(y_i)\big] \sum_{i=1}^m h_k^2(x_i)$.

The developed generalization bounds imply the Lipschitz constants and thus define the Randomized Coordinate Descent steps for the minimization of $L(\boldsymbol{\alpha})$, controlling its convergence. To illustrate this point, the convergence rate stated in [Nesterov 2012] is as follows:

$$\mathop{\mathbb{E}}_{t-1}\big[L(\boldsymbol{\alpha}_t)\big] - L(\boldsymbol{\alpha}_\star) \leq \frac{2}{t+1}\bigg(\sum_{j=1}^K C_j\bigg) R_0^2(\boldsymbol{\alpha}_0), \tag{3.12}$$

where $\boldsymbol{\alpha}_0$ is the starting point, $\boldsymbol{\alpha}_\star$ is the global minimizer of $L(\boldsymbol{\alpha})$ and $R_0(\boldsymbol{\alpha}_0)$ is the size of the initial level set of the objective. The conditional expectation is taken over the random choice of the next coordinate. The regularization applied to the base predictor families in our Regularized Gradient Boosting Framework implies the bounds on C_k, thus controlling the convergence of the algorithm.

3.4.3 PSEUDOCODE

The pseudocode of our RGB algorithm is given in Figure 3.1. The algorithm seeks to minimize the objective given in Equation 3.7, using Randomized Coordinate Descent. Let P be a discrete probability distribution over $[1, K]$ with $p_k = C_k / \sum_{j=1}^K C_j$. Equivalently, P is the distribution over the base hypothesis sets $\mathcal{H}_1, \cdots, \mathcal{H}_K$. At each draw from P, we select a sample $\mathcal{H}_{t_1}, \cdots, \mathcal{H}_{t_S}$ of size S and, from this sample, select one function that provides the best trade-off in the decrease in objective $L(\boldsymbol{\alpha})$ and the complexity bound of Theorem 3.1 of the underlying hypotheses family.

The local optimization procedure in Line 6 is an extra step required in the coordinate descent

RGB$(S = (x_1, y_1), \ldots, (x_m, y_m))$

1 **for** $t \in [1, T]$ **do**
2 $\quad [t_1, \cdots, t_S] \leftarrow P$
3 $\quad$ **for** $s \in [1, S]$ **do**
4 $\quad\quad h_s \leftarrow \operatorname{argmin}_{h \in \mathcal{H}_{t_s}} \frac{1}{m} \sum_{i=1}^{m} \Phi\big(y_i, F - \frac{1}{C_{t_s}} L'_{t_s}(\boldsymbol{\alpha}) h\big)$
5 $\quad s^\star \leftarrow \operatorname{argmin}_{s \in [1,S]} \big[\frac{1}{m} \sum_{i=1}^{m} \Phi\big(y_i, F - \frac{1}{C_{t_s}} L'_{t_s}(\boldsymbol{\alpha}) h_s\big) + \beta\Omega(h_{t_s})\big]$
6 $\quad \boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} - \frac{1}{C_{s^\star}} L'_{s^\star}(\boldsymbol{\alpha}) \mathbf{e}_{t_{s^\star}}$
7 $\quad F \leftarrow F - \frac{1}{C_{s^\star}} L'_{s^\star}(\boldsymbol{\alpha}) h_{s^\star}$
8 **return** $\boldsymbol{\alpha}, F$

Figure 3.1: Pseudocode of the RGB algorithm.

procedure to select a single function from $\mathcal{H}_{t_s}$. The step in Line 8 is required to select, out of S sampled directions, the one with the best trade-off between sample fit and complexity bounds. Note that the evaluation of sampled candidates in Line 5 can be done in parallel, making the time of RGB per thread comparable to that of standard GB. More specifically, given a fixed sample of S coordinates, the runtime of one RGB round is equal to the runtime of S rounds of GB when the same subroutine is used for tree splitting.

3.5 EXPERIMENTS

In this section, we present the results of experiments with our RGB algorithm. We restrict our attention to learning an ensemble of the regularized regression trees as defined in the family $\mathcal{H}_{n,\lambda,q}$, and to simplify the presentation, we let $q = 2$, although similar experiments can be easily done for other norms. For the complexity of these base classifiers we use the bound derived in Theorem 3.1.

To define the subfamilies of base learners we impose a grid of size 7 on the maximum number of internal nodes $n \in \{2, 4, 8, 16, 32, 64, 256\}$ and a grid on $\lambda \in \{0.001, 0.01, 0.1, 0.5, 1, 2, 4\}$. For

Table 3.1: Experimental Results

Error %	sonar	cancer	diabetes	ocr17	ocr49	mnist17	mnist49	higgs
				RGB				
Mean	26.94	5.19	28.86	0.90	3.10	0.43	1.53	28.60
(Std)	(2.10)	(0.97)	(4.85)	(0.45)	(0.69)	(0.10)	(0.38)	(0.41)
				GB				
Mean	28.64	6.14	28.39	1.35	3.50	0.55	1.66	29.11
(Std)	(2.13)	(0.94)	(5.08)	(0.52)	(0.65)	(0.11)	(0.32)	(0.37)
				One-tailed, paired sample t-test				
Signif.	5%	5%	-	2.5%	2.5%	2%	5%	2.5%

each element from the Cartesian product of these grids, we assign (n_k, λ_k), thus defining the base families $\mathcal{H}_{n_k,\lambda_k,2}$ and $\mathcal{F} = \text{conv}\left(\cup_{k=1}^{49} \mathcal{H}_{n_k,\lambda_k,2}\right)$.

Given such a decomposition of the functional space, we directly minimize the regularized objective in Equation 3.7 using Randomized Coordinate Descent with the distribution over the coordinate blocks as described above. We use the logistic loss as the per-instance loss Φ. For a given training sample, we normalize the regularization $\Omega(h)$ to be in $[0, 1]$ and tune the RGB parameter β using a grid search over $\beta \in \{0.001, 0.01, 0.1, 0.3, 1\}$.

Section 3.4 describes multiple ways to bound the coordinate-wise Lipschitz constants of the derivative of the objective function, resulting in various coordinate sampling distributions for the Randomized Coordinate Descent. For our experiments, and specifically to the families $\mathcal{H}_{n_k,\lambda_k,2}$ bound the Lipschitz constants by $C_k \leq \lambda_k \left[\max_{1\leq i\leq m} C_\Phi(y_i)\right]$, which implies that the k-th coordinate is sampled with probability $p_k = \lambda_k / \sum_{j=1}^{K} \lambda_j$, since the $\max_{1\leq i\leq m} C_\Phi(y_i)$ terms cancel out (see Lemma A.1 in the Appendix for the derivation of this bound).

As a comparison, we run the standard GB, using the XGBOOST library with ℓ_2 regularization on the vector of leaf scores $\mathbf{w}$. We use grid search to tune the hyperparameters of XGBOOST with a

grid that makes the families of trees explored comparable to the $\mathcal{H}$ defined for RGB above. Specifically, we let the ℓ_2 norm regularization parameter be in $\{0.001, 0.01, 0.1, 0.5, 1, 2, 4\}$, the maximum tree depth in $\{1, 2, 3, 4, 5, 6, 7\}$, and the learning rate parameter in $\{0.001, 0.01, 0.1, 0.5, 1\}$. Both GB and RGB are run for $T = 100$ boosting rounds. The hyperparameters are chosen via 5-fold cross-validation, and the standard errors for the best set of hyperparameters reported.

Table 3.1 shows the classification errors on the test sets for the UCI datasets studied, for both RGB and GB, see Table A.1 in the appendix for details on the dataset. A one-tailed, paired sample t-test on the pairs of results from the different trials demonstrate that these results are in general significant at a 5% level or better. Only for one of the dataset, `diabetes` with an input dimension of just 8, we do not observe an improvement of RGB over GB. One natural hypothesis is that the larger the input dimension, the more the need for proper regularization of the binary regression trees forming the base learner, and the larger the advantage of the RGB algorithm.

In general, the results demonstrate that by randomly taking steps into coordinates that correspond to subspaces $\mathcal{H}_t$ with a theoretically justified distribution, RGB can explore larger hypothesis families more efficiently that the baseline methods. Furthermore, compared to baselines that operate on the same hypotheses space $\mathcal{H}$, by optimizing for the trade-off between sample fit and functional subclass complexity, RGB can reduce over-fitting, thereby achieving greater test accuracy on multiple datasets.

3.6 CONCLUSION

We introduced and analyzed a general framework of Regularized Gradient Boosting, for which we also devised an effective algorithm, RGB. In this framework, regularization is not directly incorporated as a term in the loss function. Instead, its definition affects each boosting step by restricting

the search for the gradient approximation to a constrained subset of base functions. Our analysis is based upon strong margin-based Rademacher complexity learning guarantees. These bounds suggest a natural approach for our optimization solution, which consists of dividing the space of base learners into subfamilies of increasing complexity. For the special case of binary regression trees, we derived explicit Rademacher complexity bounds that we subsequently exploit in the definition of our RGB algorithm. Randomization over the subfamilies of base functions allows us to scale our algorithm to large families of base predictors. Our experimental results suggest improved performance, thanks to a more efficient and theoretically motivated exploration of large function spaces without over-fitting. Also, as already stated, the run-times of the algorithms are comparable, thereby making RGB a strong alternative to XGBOOST. Finally, our analysis can be extended in a similar way to that of boosting with other families of base predictors, such as kernel-based hypothesis sets and Deep Neural Networks.

4 | BOOSTING WITH MULTIPLE SOURCES

We study the problem of learning accurate ensemble predictors, in particular boosting, in the presence of multiple source domains. We show that the standard convex combination ensembles in general cannot succeed in this scenario and adopt instead a domain-weighted combination. We introduce and analyze a new boosting algorithm, MULTIBOOST, for this scenario and show that it benefits from favorable theoretical guarantees. We also report the results of several experiments with our algorithm demonstrating that it outperforms natural baselines on multi-source text-based, image-based and tabular data. Additionally, we present an extension of our algorithm to the federated learning scenario and report favorable experimental results for that setting as well.

4.1 MOTIVATION

Ensemble methods such as Bagging, AdaBoost, Stacking, error-correction techniques, Bayesian averaging, AdaNet or other adaptive methods for learning neural networks are general machine learning techniques used to combine several predictors to devise a more accurate one [Breiman 1996; Freund and Schapire 1997a; Smyth and Wolpert 1999; MacKay 1991; Freund et al. 2004; Cortes et al. 2017]. These techniques are often very effective in practice and benefit from favorable margin-based learning guarantees [Schapire et al. 1997b]. These algorithms assume access to a training sample drawn from the target distribution. But, in many applications, the learner receives

labeled data from multiple source domains that it seeks to use to find a good predictor for target domains, which are typically assumed to be mixtures of source distributions [Hoffman et al. 2018a; Mansour et al. 2008; Muandet et al. 2013; Xu et al. 2014; Hoffman et al. 2012; Saito et al. 2019; Wang et al. 2019a]. How can we generalize ensemble methods such as boosting to this scenario?

Several related problems have been tackled in previous work. In the special case of a single target domain, boosting solutions have been derived [Dai et al. 2007; Yao and Doretto 2010]. These algorithms have been further improved by boosting base predictors jointly with source features (or feature *views*) that are predictive in the target [Yuan et al. 2017; Cheng et al. 2013; Zhang et al. 2014; Xu and Sun 2012, 2011]. Such methods have been widely adopted in various domain adaptation scenarios for different types of data. For example, Huang et al. [2010, 2012] showed that by selecting a base learner jointly with a feature that is predictive across multiple domains at every boosting step, one can achieve higher accuracy than standard transfer learning methods. Moreover, the margin provided by boosting-style algorithms can aid in transfer learning where target domain is unlabelled. [Habrard et al. 2013] have developed an algorithm that jointly minimizes the source domain error and margin violation proportion on the target domain. [Wang et al. 2019a] have demonstrated that boosting classifiers from different domains can be carried out online. [Taherkhani et al. 2020] and [Becker et al. 2013] have shown that multi-source boosting can be combined with Deep Neural Networks for multi-task learning on large scale datasets.

We give a more detailed discussion of this related prior work in Appendix B.1. However, as we demonstrate in this paper, several critical questions related to the presence of multiple domains have not been fully addressed by existing boosting methods from both the algorithmic and the theoretical point of view. First, what should be the form of the ensemble solutions? We show that, in general, the standard convex combinations used in much of previous work may not succeed in this problem (Section 4.2). Instead, we put forward Q-ensembles, which are convex combinations

weighted by a domain classifier Q, that is, $\mathsf{Q}(k|x)$ is the conditional probability of domain k given input point x. This crucially differentiates our work from that of past studies. Our learning scenario strictly generalizes previous algorithms. In the special case of a single domain, the form of our ensemble solutions coincides with that of the familiar ensembles used in AdaBoost.

Second, what should be the form of the objective? Unlike existing boosting methods which optimize for a specific target distribution or domain, we seek a solution that is accurate for any mixture of the source distributions, where the mixture weights may be constrained to be in a subset of the simplex. This conforms with the learning goal in this scenario where the target distribution is typically assumed to be a mixture of the source ones and is further inspired by the formulation of the *agnostic loss* in the related context of federated learning [Mohri et al. 2019b]. Additionally, this formulation can be viewed as more risk-averse and robust, and it also ensures more favorable fairness properties, since it does not favor any distribution possibly biased towards some subset of the source domains. In Section 4.3, we introduce and analyze an algorithm, MULTIBOOST, that is based on such an objective. This further distinguishes our boosting algorithm from related prior work.

Third, in Section 4.4, we present a theoretical analysis of our MULTIBOOST algorithm, including margin-based generalization bounds that hold for any target mixture of the source distributions. In Appendix B.6, we derive finer and more flexible learning bounds that can provide more favorable guarantees, in particular when the family of target mixtures is more constrained.

Fourth, in Section 4.5, we describe FEDMULTIBOOST, an extension of our MULTIBOOST algorithm adapted to the distributed learning scenario of *federated learning*, which is increasingly important in a wide array of applications [Kairouz et al. 2021], including healthcare [Brisimi et al. 2018]. FEDMULTIBOOST allows the server to train ensemble models that benefit from user data, while the data remains with that users. We provide a detailed description of our algorithm, as well

as a brief analysis of the communication costs, which are critical in this scenario. Appendix B.7 contains experimental results comparing FEDMULTIBOOST with several baselines. Finally, in Section 4.6, we report the results of extensive experiments with our MULTIBOOST algorithm. We start with a detailed description of the learning scenario we consider.

4.2 LEARNING SCENARIO

Let $\mathcal{X}$ denote the input space and $\mathcal{Y} = \{-1, +1\}$ be the output space, associated to binary classification. We assume the learner receives labeled samples from p source domains, each defined by a distribution $\mathcal{D}_k$ over $\mathcal{X} \times \mathcal{Y}$, $k \in [p]$. We denote by $S_k = ((x_{k,1}, y_{k,1}), \ldots, (x_{k,m_k}, y_{k,m_k})) \in (\mathcal{X} \times \mathcal{Y})^{m_k}$ the labeled sample of size m_k received from source k, which is drawn i.i.d. from $\mathcal{D}_k$. For any function $f \colon \mathcal{X} \to \mathbb{R}$ and distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$, let $\mathcal{L}(\mathcal{D}, f)$ be the expected loss of f, that is $\mathcal{L}(\mathcal{D}, f) = \mathbb{E}_{(x,y)\sim\mathcal{D}}[\ell(f(x), y)]$, where ℓ is the binary loss $\ell(f(x), y) = \mathbb{I}(yf(x) \leq 0)$.

For any $k \in [p]$, let $\mathcal{H}_k$ denote a hypotheses set of functions mapping from $\mathcal{X}$ to $[-1, +1]$, $|\mathcal{H}_k| = N_k$. The objective of the learner is to find an accurate predictor f for any target distribution $\mathcal{D}_\lambda$ that is a mixture of the source distributions, where λ may be in a subset Λ of the simplex. Specifically, $\mathcal{D}_\lambda = \sum_{k=1}^{p} \lambda_k \mathcal{D}_k$, with $\lambda = (\lambda_1, \ldots, \lambda_p)$, the set of distributions over the p domains. This leads to the following *agnostic loss* of a predictor f [Mohri et al. 2019b]:

$$\mathcal{L}(\mathcal{D}_\Lambda, f) = \max_{\lambda \in \Lambda} \mathcal{L}(\mathcal{D}_\lambda, f). \tag{4.1}$$

To create a predictor f, the learner seeks an ensemble of functions from the base classes $\mathcal{H}_k$. What

should be the form of such ensembles? A natural solution is a convex combinations of the form

$$f = \sum_{l=1}^{p} \sum_{j=1}^{N_l} \alpha_{l,j} h_{l,j}, \tag{4.2}$$

where $h_{l,j} \in \mathcal{H}_l$ and $\alpha_{l,j} \geq 0$, $\sum_{l=1}^{p} \sum_{j=1}^{N_l} \alpha_{l,j} = 1$, as in prominent algorithms such as BAGGING [Breiman 1999] or ADABOOST [Freund and Schapire 1997a]. It is not hard to show, however, that for some distributions, any such convex combination of base predictors would lead to a poor solution.

Proposition 4.1. *There exist distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ and hypotheses h_1 and h_2 with $\mathcal{L}(\mathcal{D}_1, h_1) = 0$ and $\mathcal{L}(\mathcal{D}_2, h_2) = 0$ such that $\mathcal{L}\big(\frac{1}{2}(\mathcal{D}_1 + \mathcal{D}_2), \alpha h_1 + (1-\alpha) h_2\big) \geq \frac{1}{2}$ for any $\alpha \in [0, 1]$,*

see Appendix B.2. The base predictors h_1, h_2 are each perfect for their respective domains, but unlike the common case of a single source domain, standard convex combinations in general may perform poorly. Thus, we will consider instead the following form for the ensembles of base predictors:

$$\forall x \in \mathcal{X}, \quad f(x) = \sum_{l=1}^{p} \sum_{j=1}^{N_l} \alpha_{l,j} \mathsf{Q}(l|x) h_{l,j}(x), \tag{4.3}$$

where $\mathsf{Q}(l|x)$ denotes the conditional probability of domain l given x. This helps us account for that the learner combines base predictors from different domains. For a given point x, experts from the domains where x is more likely to appear are allocated more weight in the voted combination. The following result further substantiates the hope for this method to succeed.

Proposition 4.2. *For the same distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ and hypotheses h_1 and h_2 as in Proposition 4.1, the equality $\mathcal{L}(\mathcal{D}_\lambda, (\alpha \mathsf{Q}(1|\cdot) h_1 + (1-\alpha) \mathsf{Q}(2|\cdot) h_2)) = 0$ holds for any $\lambda \in \Delta$ and any $\alpha \in (0, 1)$,*

see proof in Appendix B.2. Thus, our Q-ensembles can be substantially more effective in the

multiple-source adaptation problem considered. Note, that in the special case of a single domain, the Q-combinations coincide with standard convex combinations, since $\mathsf{Q}(k|x) = 1$ for all x.

As in the standard boosting scenario, we will adopt a weak-learning assumption. However, our assumption here must hold for each source domain: for each domain $k \in [p]$ and any distribution D over S_k, there exists a base classifier $h \in \mathcal{H}_k$ such that the weighted loss of h is γ-better than random: $\frac{1}{2}[1 - \mathbb{E}_{i\sim\mathsf{D}}[y_{k,i}h(x_{k,i})] \leq \frac{1}{2} - \gamma$, for some edge value $\gamma > 0$. This is equivalent to the existence of a good *rule of thumb* for each domain. Unlike the standard scenario, however, here we seek a Q-ensemble and further require it to be accurate for any target mixture $\mathcal{D}_\lambda$. In the next section, we present an algorithm, MULTIBOOST, for deriving an accurate Q-ensemble for any target mixture domains that belongs to the convex combination of the source domains.

4.3 ALGORITHM

Let Φ be a convex, increasing and differentiable function such that $u \mapsto \Phi(-u)$ upper-bounds the binary loss $u \mapsto 1_{u\leq 0}$. Φ could be the exponential function as in ADABOOST or the logistic function, as in logistic regression. Using Φ to upper-bound the agnostic loss leads to the following objective function for an ensemble f defined by (4.3) for any $\alpha = (\alpha_{l,j})_{(l,j)\in[p]\times[N_l]} \geq 0$:

$$F(\alpha) = \max_{\lambda\in\Lambda} \sum_{k=1}^{p} \frac{\lambda_k}{m_k} \sum_{i=1}^{m_k} \Phi\left(-y_{k,i} \sum_{l=1}^{p} \sum_{j=1}^{N_l} \alpha_{l,j}\mathsf{Q}(l|x_{k,i})h_{l,j}(x_{k,i})\right). \tag{4.4}$$

Since a convex function composed with an affine function is convex and a sum of convex functions is convex, F is convex as the maximum of a set of convex functions. In this section, we will consider the case where the set Λ coincides with the full simplex, that is $\Lambda = \Delta$. Our algorithm applies coordinate descent to this objective, as with ADABOOST and other boosting-type algorithms. While convex, F is not differentiable and, in general, coordinate descent may not succeed in such

cases [Tseng 2001; Luo and Tseng 1992]. This is because the algorithm may be *stuck* at a point where no progress is possible along any of the axis directions, while there exists a favorable descent along some other direction. However, we will show that, under the weak-learning assumption we adopted, there always exists a main direction, along which a descent progress is possible. Alternatively, one can replace the maximum with a *soft-max*, that is the $(x_1, \ldots, x_k) \mapsto \frac{1}{\mu} \log(\sum_{k=1}^{p} e^{\mu x_k})$ for $\mu > 0$, which leads to a continuously differentiable function with a close approximation for η sufficiently large.

Description. Let α_{t-1} denote the value of the parameter vector $\alpha = (\alpha_{l,j})$ at the end of the $(t-1)$th iteration and f_{t-1} the corresponding function: $f_{t-1} = \sum_{l=1}^{p} \sum_{j=1}^{N_l} \alpha_{t-1,l,j} \mathsf{Q}(l|\cdot)\, h_{l,j}$. Coordinate descent at iteration t consists of choosing a direction $\mathbf{e}_{q,r}$ corresponding to base classifier $h_{q,r}$ and a step value $\eta > 0$ to minimize $F(\alpha_{t-1} + \eta\, \mathbf{e}_{q,r})$. To select a direction, we consider the subdifferential of F along any $\mathbf{e}_{q,r}$. Observe that, since $\Lambda = \Delta$, for any α, we can write $F(\alpha) = \max_k F_k(\alpha)$, with $F_k(\alpha) = \frac{1}{m_k} \sum_{i=1}^{m_k} \Phi\big(-y_{k,i} \sum_{l=1}^{p} \sum_{j=1}^{N_l} \alpha_{l,j} \mathsf{Q}(l|x_{k,i}) h_{l,j}(x_{k,i})\big)$. Since functions F_k are differentiable, by the subdifferential calculus for the maximum of functions, the subdifferential of F at α_{t-1} along the direction $\mathbf{e}_{q,r}$ is given by:

$$\partial F(\alpha_{t-1}, \mathbf{e}_{q,r}) = \operatorname{conv}\{F'_k(\alpha_{t-1}, \mathbf{e}_{q,r}) \colon k \in \mathcal{K}_t\},$$

where $F'_k(\alpha_{t-1}, \mathbf{e}_{q,r})$ is the directional derivative of F_k at α_{t-1} along the direction $\mathbf{e}_{q,r}$ and where $\mathcal{K}_t = \{k \in [p] \colon F_k(\alpha_{t-1}) = F(\alpha_{t-1})\}$. We will therefore consider the direction $\mathbf{e}_{q,r}$ with the largest absolute directional derivative value $|F'_k(\alpha_{t-1}, \mathbf{e}_{q,r})|$, $k \in \mathcal{K}_t$, but will restrict ourselves to $q = k$ since, as we shall see, that will suffice us to guarantee a non-zero directional gradient. To do so, we will express $F'_k(\alpha_{t-1}, \mathbf{e}_{k,r})$ in terms of the distribution $\mathsf{D}_t(k, \cdot)$ over S_k defined by

$\mathsf{D}_t(k,i) = \frac{\Phi'\left(-y_{k,i} f_{t-1}(x_{k,i})\right)}{Z_{t,k}}$, with $Z_{t,k} = \sum_{i=1}^{m_k} \Phi'(-y_{k,i} f_{t-1}(x_{k,i}))$, for all $i \in [m_k]$:

$$F'_k(\alpha_{t-1}, h_{k,r}) = \frac{1}{m_k} \sum_{i=1}^{m_k} -y_{k,i} \mathsf{Q}(k|x_{k,i}) h_{k,r}(x_{k,i}) \Phi'\left(-y_{k,i} f_{t-1}(x_{k,i})\right) = \frac{Z_{t,k}}{m_k}[2\epsilon_{t,k,r} - 1],$$

where $\epsilon_{t,k,r} = \frac{1}{2}\left[1 - \mathbb{E}_{i \sim \mathsf{D}_t(k,\cdot)}[y_{k,i} \mathsf{Q}(k|x_{k,i}) h_{k,r}(x_{k,i})]\right]$ denotes the *weighted error* of $\mathsf{Q}(k|\cdot)h_{k,r}$. For any $s \in [m_k]$, since $x_{k,s}$ is a sample drawn from $\mathcal{D}_k$, we have $\mathsf{Q}(k|x_{k,s}) > 0$ and therefore we have: $\overline{\mathsf{Q}}_{t,k} = \sum_{s=1}^{m_k} \mathsf{D}_t(k,s)\mathsf{Q}(k|x_{k,s}) > 0$. Thus, we can write $\mathbb{E}_{i \sim \mathsf{D}_t(k,\cdot)}[y_{k,i} \mathsf{Q}(k|x_{k,i}) h_{k,r}(x_{k,i})]$ as

$$\sum_{i=1}^{m_k} \frac{\mathsf{D}_t(k,i)\mathsf{Q}(k|x_{k,i}) y_{k,i} h_{k,r}(x_{k,i})}{\overline{\mathsf{Q}}_{t,k}} \overline{\mathsf{Q}}_{t,k}, = \mathop{\mathbb{E}}_{i \sim \mathsf{D}'_t(k,\cdot)}[y_{k,i} h_{k,r}(x_{k,i})]\, \overline{\mathsf{Q}}_{t,k},$$

where $\mathsf{D}'_t(k,i) = \frac{\mathsf{D}_t(k,i)\mathsf{Q}(k|x_{k,i})}{\overline{\mathsf{Q}}_{t,k}}$. By our weak-learning assumption, there exists $r \in N_k$ such that $\mathbb{E}_{i \sim \mathsf{D}'_t(k,\cdot)}[y_{k,i} h_{k,r}(x_{k,i})] \geq \gamma > 0$. For that choice of r, we have $\epsilon_{t,k,r} < \frac{1}{2} - \overline{\gamma}$, with $\overline{\gamma} = \gamma \overline{\mathsf{Q}}_{t,k} > 0$. In view of that, it suffices for us to search along the directions $h_{k,r}$ and we do not need to consider the directional derivative of F_k along directions $h_{q,r}$ with $q \neq k$. The direction chosen by our coordinate descent algorithm is thus defined by: $\operatorname{argmax}_{k \in \mathcal{K}_t, r \in [N_k]} \frac{Z_{t,k}}{m_k}[1 - 2\epsilon_{t,k,r}]$. Given the direction $\mathbf{e}_{k,r}$, the optimal step value η is $\operatorname{argmin}_{\eta > 0} F(\alpha_{t-1} + \eta \mathbf{e}_{k,r})$. The pseudocode of our algorithm, MULTIBOOST, is provided in Figure 4.1. In the most general case, η can be found via a line search or other numerical methods.

Step size. In some special cases, the line search can be executed using a simpler expression by using an upper bound on $F(\alpha_{t-1} + \eta \mathbf{e}_{k,r})$. Using the convexity of Φ, since $y_{l,i} \mathsf{Q}(k|x_{l,i}) h_{k,r}(x_{l,i}) =$

$\frac{1+y_{l,i}\mathsf{Q}(k|x_{l,i})h_{k,r}(x_{l,i})}{2} \cdot (+1) + \frac{1-y_{l,i}\mathsf{Q}(k|x_{l,i})h_{k,r}(x_{l,i})}{2} \cdot (-1)$, the following holds for all $\eta \in \mathbb{R}$:

$$\Phi(-y_{l,i}f_{t-1}(x_{l,i}) - \eta\, y_{l,i}\mathsf{Q}(k|x_{l,i})h_{k,r}(x_{l,i})) \leq \frac{1+y_{l,i}\mathsf{Q}(k|x_{l,i})h_{k,r}(x_{l,i})}{2}\Phi(-y_{l,i}f_{t-1}(x_{l,i}) - \eta) + \frac{1-y_{l,i}\mathsf{Q}(k|x_{l,i})h_{k,r}(x_{l,i})}{2}\Phi(-y_{l,i}f_{t-1}(x_{l,i}) + \eta).$$

In the case of exponential and logistic functions, the following upper bounds can then be derived.

Lemma 4.3. *For any $k \in [p]$, the following upper bound holds when Φ is the exponential or the logistic function:*

$$F(\alpha_{t-1} + \eta \mathbf{e}_{k,r}) \leq \max_{l\in[p]} \frac{Z_{t,l}}{m_l}\big[(1-\epsilon_{t,l,k,r})e^{-\eta} + \epsilon_{t,l,k,r}e^{\eta}\big],$$

where $\epsilon_{t,l,k,r} = \frac{1}{2}\big[1 - \mathbb{E}_{i\sim \mathsf{D}_t(l,\cdot)}[y_{l,i}\mathsf{Q}(k|x_{l,i})h_{k,r}(x_{l,i})]\big]$.

For any k, function $\eta \mapsto (1-\epsilon_{t,l,k,r})e^{-\eta} + \epsilon_{t,l,k,r}e^{\eta}$ reaches its minimum for $\eta = \frac{1}{2}\log\frac{1-\epsilon_{t,l,k,r}}{\epsilon_{t,l,k,r}}$. When the maximum is achieved for $l = k$, the solution coincides with the familiar expression of the step size $\eta_t = \frac{1}{2}\log\frac{1-\epsilon_{t,k,r}}{\epsilon_{t,k,r}}$ used in AdaBoost.

Q-function. The conditional probability functions $\mathsf{Q}(k|\cdot)$ are crucial to the definition of our algorithm. As pointed out earlier, Q-ensembles can help achieve accurate solutions in some cases that cannot be realized using convex combinations. Furthermore, for any $k \in [p]$, since $\mathsf{D}_t(k,\cdot)$ is a distribution over the sample S_k, it is natural to assume that for any $j \neq k$ we have $\mathbb{E}_{s\sim\mathsf{D}_t(k,\cdot)}[\mathsf{Q}(k|x_{k,s})] \geq \mathbb{E}_{s\sim\mathsf{D}_t(k,\cdot)}[\mathsf{Q}(j|x_{k,s})]$. This implies the following lower bound:

$$\mathop{\mathbb{E}}_{s\sim\mathsf{D}_t(k,\cdot)}[\mathsf{Q}(k|x_{k,s})] \geq \frac{1}{p},$$

which in turn implies $\overline{\gamma} \geq \frac{\gamma}{p}$, since for any $x \in \mathcal{X}$, $\sum_{j=1}^{p}\mathsf{Q}(j|x) = 1$. In the special case where all domains coincide, we have $\mathsf{Q}(k|x_{k,s}) = \frac{1}{p}$ for all s and this lower bound is reached. At another

MULTIBOOST$(S_1, \ldots, S_p)$

$\alpha_0 \leftarrow 0$
for $t \leftarrow 1$ **to** T **do**
  $f_{t-1} \leftarrow \sum_{l=1}^{p} \sum_{j=1}^{N_l} \alpha_{t-1,l,j} \mathsf{Q}(l|\cdot)\, h_{l,j}$
  $\tilde{\Phi}_k \leftarrow \frac{1}{m_k} \sum_{i=1}^{m_k} \Phi(-y_{k,i} f_{t-1}(x_{k,i}))$
  $\mathcal{K}_t \leftarrow \left\{ k \colon k \in \operatorname{argmax}_{k \in [p]} \tilde{\Phi}_k \right\}$
  for $k \in \mathcal{K}_t$ **do**
    $Z_{t,k} \leftarrow \sum_{i=1}^{m_k} \Phi'\big(-y_{k,i} f_{t-1}(x_{k,i})\big)$
    for $i \leftarrow 1$ **to** m_k **do**
      $\mathsf{D}_t(k,i) \leftarrow \frac{\Phi'\left(-y_{k,i} f_{t-1}(x_{k,i})\right)}{Z_{t,k}}$
  $(k, r) \leftarrow \operatorname*{argmax}_{k \in \mathcal{K}_t, r \in [N_k]} \frac{Z_{t,k}}{m_k} [1 - 2\epsilon_{t,k,r}]$
  $\eta_t \leftarrow \operatorname{argmin}_{\eta > 0} F(\alpha_{t-1} + \eta \mathbf{e}_{k,r})$
  $\alpha_t \leftarrow \alpha_{t-1} + \eta_t \mathbf{e}_{k,r}$
$f \leftarrow \sum_{l=1}^{p} \sum_{j=1}^{N_l} \alpha_{T,l,j} \mathsf{Q}(l|\cdot) h_{l,j}$
return f

Figure 4.1: Pseudocode of the MULTIBOOST algorithm. $\epsilon_{t,k,r}$ denotes the weighted error of $\mathsf{Q}(k|\cdot)h_{k,r}$.

extreme, when all domains admit distinct supports, we have $\mathsf{Q}(k|x_{k,s}) = 1$ for all $s \in [m_k]$ and thus $\overline{\gamma} = \gamma$.

In practice, we do not have access to the true conditional probabilities $\mathsf{Q}(k|\cdot)$. Instead, we can derive accurate estimates $\widehat{\mathsf{Q}}(k|\cdot)$ using large unlabeled samples from the source domains, the *label* used for training being simply the domain index. This can be done using algorithms such as conditional maximum entropy models [Berger et al. 1996] (or multinomial logistic regression), which benefit from strong theoretical guarantees [Mohri et al. 2018b, chapter 13], or other rich models based on neural networks.

Other variants of MULTIBOOST. In Appendix B.4, we discuss other variants of our algorithm, including their convergence guarantees.

4.4 THEORETICAL ANALYSIS

In this section, we present a theoretical analysis of our algorithm, including margin-based learning bounds for multiple-source Q-ensembles.

For any $k \in [p]$, define the family $\mathcal{G}_k$ as follows: $\mathcal{G}_k = \{\mathsf{Q}(k|\cdot)\, h \colon h \in \mathcal{H}_k\}$. Then, the family of ensemble functions $\mathcal{F}$ that we consider can be defined as $\mathcal{F} = \operatorname{conv}(\bigcup_{k=1}^{p} \mathcal{G}_k)$. For any $\lambda \in \Delta$, let $\overline{\mathcal{D}}_\lambda$ be the distribution defined by $\mathcal{D}_\lambda = \sum_{k=1}^{p} \lambda_k \widehat{\mathcal{D}}_k$, where $\widehat{\mathcal{D}}_k$ is the empirical distribution associated to an i.i.d. sample S_k drawn from $\mathcal{D}_k$. $\overline{\mathcal{D}}_\lambda$ is distinct from the distribution associated to $\mathcal{D}_\lambda$. We seek to derive margin-based bounds for ensemble functions $f \in \mathcal{F}$ with respect to target mixture distributions $\mathcal{D}_\lambda$, while the empirical data is from multiple source distributions $\mathcal{D}_k$, or from $\overline{\mathcal{D}}_\lambda$. Thus, these guarantees differ from standard learning bounds where the source and target distribution coincide.

For any distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$ and $\rho > 0$, let $\mathcal{L}_\rho(\mathcal{D}, f)$ denote the expected ρ-margin loss of f with respect to $\mathcal{D}$: $\mathcal{L}_\rho(\mathcal{D}, f) = \mathbb{E}_{(x,y)\sim\mathcal{D}_k}[\mathbb{I}(yf(x) \leq \rho)]$. We denote by $\mathfrak{R}_m(\mathcal{F})$ the Rademacher complexity of $\mathcal{F}$ for samples $S = (x_1, \ldots, x_m)$ of size m, defined by $\mathfrak{R}_m(\mathcal{F}) = \mathbb{E}_{S\sim\mathcal{D}^m, \boldsymbol{\sigma}}[\sup_{f\in\mathcal{F}} \sum_{i=1}^{m} \sigma_i f(x_i)]$, with σ_is independent random variables taking values in $\{-1, +1\}$. The following gives a margin-bound for our multiple-source learning with Q-ensembles.

Theorem 4.4. *Fix $\rho > 0$. Then, for any $\delta > 0$, with probability at least $1 - \delta$ over the draw of a sample $S = (S_1, \ldots, S_p) \sim \mathcal{D}_1^{m_1} \otimes \cdots \otimes \mathcal{D}_p^{m_p}$, the following inequality holds for all ensemble functions $f = \sum_{t=1}^{T} \alpha_t \mathsf{Q}(k_t|\cdot) h_t \in \mathcal{F}$ and all $\lambda \in \Delta$:*

$$\mathcal{L}(\mathcal{D}_\lambda, f) \leq \mathcal{L}_\rho(\overline{\mathcal{D}}_\lambda, f) + \sum_{k=1}^{p} \lambda_k \left[\frac{2}{\rho} \max_{l\in[p]} \mathfrak{R}_{m_k}(\mathcal{H}_l) + \sqrt{\frac{\log \frac{p}{\delta}}{2m_k}} \right]. \tag{4.5}$$

Note that the complexity term depends only on the Rademacher complexities of the families of

based predictors $\mathcal{H}_l$ and does not involve the domain classification function Q. Theorem 4.4 improves upon previous known bounds of Mohri et al. [2019b]. In particular, it provides a dimension-independent margin guarantee for the zero-one loss, while the bound of Mohri et al. [2019b], when applied to the zero-one loss, is dimension-dependent [Mohri et al. 2019b, Lemma 3]. Additionally, our bound holds for Q-ensembles. Our margin bound can be straightforwardly generalized to hold uniformly for all $\rho \in (0, 1)$, at the cost of an additional mild term depending on $\log \log_2(2/\rho)$ (see Theorem B.2 in Appendix B.5). For the ensemble functions returned by our algorithm, the bound applies to $x \mapsto \frac{f(x)}{\|\alpha\|_1}$. These learning guarantees suggest choosing $\alpha \geq 0$ and ρ to minimize the following:

$$\max_{\lambda \in \Lambda} \sum_{k=1}^{p} \lambda_k \left\{ \mathop{\mathbb{E}}_{(x,y)\sim S_k} \left[\mathbb{I}\left(y \sum_{t=1}^{T} \frac{\alpha_t}{\|\alpha\|_1} \mathsf{Q}(k_t|\cdot) h_t(x) \leq \rho \right) \right] + \frac{r_k}{\rho} \right\},$$

where $r_k = 2 \max_{l \in [p]} \mathfrak{R}_{m_k}(\mathcal{H}_l)$. Choosing $\frac{1}{\rho} = \|\alpha\|_1$ and upper bounding the binary loss using a convex function Φ, the problem can then be equivalently cast as the following unconstrained minimization, for example in the case of the exponential function:

$$\min_{\alpha \geq 0} \max_{\lambda \in \Lambda} \sum_{k=1}^{p} \lambda_k \left\{ \mathop{\mathbb{E}}_{(x,y)\sim S_k} \left[\exp\left(-y \sum_{t=1}^{T} \alpha_t \mathsf{Q}(k_t|\cdot) h_t(x) \right) \right] + r_k \|\alpha\|_1 \right\}.$$

Thus, the learning guarantees justify a posteriori an ℓ_1-regularized version of our algorithm. It is straightforward to derive that extension of our algorithm and its pseudocode. This is similar to the extension to the ℓ_1-ADABOOST [Rätsch et al. 2001] of the original unregularized ADABOOST [Freund and Schapire 1997a]. Let us add that finer learning guarantees such as those for DEEP BOOSTING [Cortes et al. 2014] ones can be derived similarly here, which would lead to a weighted ℓ_1-regularization, where the weights are the Rademacher complexities of the families $\mathcal{H}_k$. For the sake of simplicity, we do not detail that extension. Let us point out, however, that such an extension would also lead to an algorithm generalizing ADANET [Cortes et al. 2017] to multiple sources.

In some applications, prior knowledge can be used to constrain Λ to a strict subset of the simplex Δ. Finer margin-based learning guarantees can be derived to cover that scenario. Let $\overline{\Lambda}$ be the set of vertices of a subsimplicial cover of Λ, that is a decomposition of a cover of Λ into subsimplices and let $\overline{\Lambda}_\epsilon$ be the set of λs ϵ-close to $\overline{\Lambda}$ in ℓ_1-distance. Then, the following guarantees hold.

Theorem 4.5. *Fix $\rho > 0$ and $\epsilon > 0$. Then, for any $\delta > 0$, with probability at least $1 - \delta$ over the draw of a sample $S = (S_1, \ldots, S_p) \sim \mathcal{D}_1^{m_1} \otimes \cdots \otimes \mathcal{D}_p^{m_p}$, each of the following inequalities holds:*

1. for all ensemble functions $f = \sum_{t=1}^{T} \alpha_t \mathsf{Q}(k_t|\cdot) h_t \in \mathcal{F}$ and all $\lambda \in \overline{\Lambda}_\epsilon$:

$$\mathcal{L}(\mathcal{D}_\lambda, h) \leq \mathcal{L}_\rho(\overline{\mathcal{D}}_\lambda, h) + \frac{2}{\rho} \max_{r \in [p]} \mathbb{R}_{\mathbf{m}}(\mathcal{H}_r, \lambda) + \frac{3\epsilon}{2} + \sqrt{\sum_{k=1}^{p} \frac{\lambda_k^2}{2m_k} \log \frac{|\overline{\Lambda}|}{\delta}}. \tag{4.6}$$

2. for all ensemble functions $f = \sum_{t=1}^{T} \alpha_t \mathsf{Q}(k_t|\cdot) h_t \in \mathcal{F}$ and all $\lambda = \sum_{k=1}^{p} \mu_k \beta_k \in \operatorname{conv}(\overline{\Lambda})$:

$$\mathcal{L}(\mathcal{D}_\lambda, h) \leq \mathcal{L}_\rho(\overline{\mathcal{D}}_\lambda, h) + \frac{2}{\rho} \sum_{l=1}^{p} \mu_k \max_{r \in [p]} \mathbb{R}_{\mathbf{m}}(\mathcal{H}_r, \beta_l) + \sum_{k=1}^{p} \mu_k \sqrt{\sum_{l=1}^{p} \frac{\beta_{l,k}^2}{2m_k} \log \frac{|\overline{\Lambda}|}{\delta}}. \tag{4.7}$$

The proof of the theorem and further discussion are presented in Appendix B.6. In the theorem above, $\mathbb{R}_{\mathbf{m}}(\mathcal{H}_r, \lambda)$ is the λ-weighted Rademacher complexity of $\mathcal{H}_r$ (see Appendix B.6). For Λ a strict subset of the simplex, these learning bounds suggest an algorithm with a regularization term of the form $\sum_{k=1}^{p} \lambda_k^2 / m_k$.

4.5 FEDERATED MULTIBOOST ALGORITHM

In this section, we present an extension of our algorithm to the *federated learning* scenario, called FEDMULTIBOOST. The pseudocode is given in Figure 4.2. Federated learning is a distributed

learning scenario where a global model is trained at the server level, based on data remaining at clients [Konečnỳ et al. 2016b,a; McMahan et al. 2017; Yang et al. 2019]. Clients are typically mobile phones, network sensors, or other IoT devices. While the data remains at the clients, the trained model significantly benefits from user data, as demonstrated, for example, in next word prediction [Hard et al. 2018; Yang et al. 2018] and healthcare applications [Brisimi et al. 2018]. We refer the reader to [Kairouz et al. 2021] for a more detailed survey of federated learning.

A widely used algorithm in this scenario is *federated averaging*, where the server trains the global model via stochastic updates from the clients [McMahan et al. 2017]. It has been argued by [Mohri et al. 2019b], however, that minimizing the expected loss with respect to a particular training distribution, as done by standard federated learning algorithms, may be risk-prone and benefit only a specific subset of clients. This compromises fairness, which is one of the critical questions in federated learning, where clients are often heterogeneous [Bickel et al. 1975; Hardt et al. 2016; Abay et al. 2020; Li et al. 2020]. This issue can be further accentuated when the set of participating clients during inference may significantly differ from those used in training. Such cases may occur, for example, when clients loose access to network connection during training, which results in distinct training and test distributions.

To overcome these problems, we adopt the *agnostic federated learning* approach suggested by [Mohri et al. 2019b] and [Ro et al. 2021]. Following the client-partition approach of [Ro et al. 2021], let each client belong to one of p domains. The domains can be, for example, based on the type of device or their geographical location. In this setting, the global model is optimized for all convex combinations of domain distributions as in the objective of MULTIBOOST (Equation 4.4), by taking the maximum over the mixtures weights λ. This ensures that the solution obtained is risk-averse and reduces the worst-case mismatch between the inference and training distributions. We propose FEDMULTIBOOST, a communication-efficient modification of MULTIBOOST, that

FEDMULTIBOOST$(S_1, S_2, \ldots, S_n)$

$\alpha_0 \leftarrow 0$
for $t \leftarrow 1$ **to** T **do**
  $f_{t-1} \leftarrow \sum_{l=1}^{p} \sum_{j=1}^{N_l} \alpha_{t-1,l,j} \mathsf{Q}(l|\cdot)\, h_{l,j}$
  $C_{t,k} \leftarrow$ SELECTCLIENT(k, r)
  for $k \leftarrow 1$ **to** p **do**
    for $c \in C_{t,k}$ **do**
      $\tilde{\Phi}_{k,c} \leftarrow \frac{1}{m_c} \sum_{i=1}^{m_c} \Phi(-y_{c,i} f_{t-1}(x_{c,i}))$
      $Z_{t,c} \leftarrow \sum_{i=1}^{m_c} \Phi'\big(-y_{c,i} f_{t-1}(x_{c,i})\big)$
  $\mathcal{K}_t \leftarrow \Big\{k \in [p] \colon k \in \operatorname{argmax}_{k \in [p]} \sum_{c \in C_{t,k}} \tilde{\Phi}_{k,c}\Big\}$
  $Z_{t,k} \leftarrow \sum_{c \in C_{t,k}} Z_{t,c}, \forall k \in [p]$
  $\mathcal{H}_{t,k} \leftarrow$ SELECTBASECLASSIFIER(k, s)
  for $k \in \mathcal{K}_t$ **do**
    for $c \in C_{t,k}$ **do**
      for $i \leftarrow 1$ **to** m_c **do**
        $\mathsf{D}_t(c, i) \leftarrow \frac{\Phi'(-y_{c,i} f_{t-1}(x_{c,i}))}{Z_{t,k}}$
      $\beta_{t,c,r} \leftarrow [1 - 2\epsilon_{t,c,r}]$
  $(k, r) \leftarrow \operatorname*{argmax}_{k \in \mathcal{K}_t, r \in \mathcal{H}_{t,k}} \frac{Z_{t,k}}{\sum_{c \in C_{t,k}} m_c} \sum_{c \in C_{t,k}} \beta_{t,c,r}$
  $\eta_t \leftarrow \eta_0 / \sqrt{t}$
  $\alpha_t \leftarrow \alpha_{t-1} + \eta_t \mathbf{e}_{k,r}$
$f \leftarrow \sum_{l=1}^{p} \sum_{j=1}^{N_l} \alpha_{T,l,j} \mathsf{Q}(l|\cdot) h_{l,j}$
return f

Figure 4.2: Pseudocode of the FEDMULTIBOOST algorithm. $\epsilon_{t,c,r}$ denotes the weighted error of $\mathsf{Q}(k|\cdot)h_{k,r}$, $\epsilon_{t,c,r} = \frac{1}{2}\big[1 - \mathbb{E}_{i \sim \mathsf{D}_t(c,\cdot)}[y_{c,i} \mathsf{Q}(k|x_{c,i}) h_{k,r}(x_{c,i})]\big]$, where client c belongs to domain k. cobalt steps are carried out by the server, red on the clients.

can overcome the communication bottleneck in federated learning [Konečnỳ et al. 2016b].

Before presenting our algorithm, we first describe the learning scenario and the notation we adopt. Let n be the number of clients, which in the cross-device setting can be in the order of millions. Let m_c denote the number of samples in client c and $(x_{c,i}, y_{c,i})$ denote the i^{th} sample of client c. The server has access to a set of base-predictor classes $\mathcal{H}_k$ for $k \in [p]$ and a domain

classifier Q that assigns for each sample x, the likelihood of it belonging to each of the p domains. The base classifiers can be obtained by training on public data or training on client data with federated learning. The domain classifier Q can be trained using unlabeled client data.

Additional discussion of the algorithm as well as experimental results can be found in Appendix B.7. At each round, the algorithm randomly selects r clients and s base classifiers from each domain. Next, it chooses the best classifier out of the previously selected subset, based on the data from the subsampled clients. Since FEDMULTIBOOST operates only on a small set of base classifiers at each round, the algorithm is communication-efficient. At round t, the server needs to send Q, f_{t-1}, and s base classifiers to the selected clients. Hence, the server-to-client communication cost at round t is $\mathcal{O}((t+s)K + K')$, where K is the number of parameters in a single base classifier and K' is the number of parameters required to specify Q. Furthermore, the clients communicate only the aggregate statistics $\tilde{\Phi}_c, Z_{t,c}, \beta_{t,c,r}$ to the server and hence the client-to-server communication is $\mathcal{O}(p \cdot s)$ real numbers, which is independent of the number of parameters of the base classifiers.

4.6 EXPERIMENTS

In this section, we present experimental results for the MULTIBOOST algorithm on several datasets with multiple sources. Our study is restricted to learning an ensemble of decision stumps $\mathcal{H}^{\textit{stumps}}$ using the exponential surrogate loss $\Phi(u) = e^{-u}$. To estimate the probabilities $\mathsf{Q}(k|\cdot)$ for $k \in [p]$, we assigned the label k to each sample from domain k and used multinomial logistic regression. This step can be facilitated with unlabeled examples, as only their domain membership information is required.

We compare MULTIBOOST with a set of boosting-style algorithms that operate on the same

hypotheses class $\mathcal{H}^{stumps}$. Those include AdaBoost-k for $k \in [p]$ and AdaBoost-all. The former is a standard AdaBoost algorithm trained only on a single source S_k and the latter is AdaBoost trained on the union of all sources $\cup_{k=1}^{p} S_k$. It is natural to compare MultiBoost against AdaBoost-all, since both of them have access to all sources during training. The difference is that while AdaBoost-all treats $\cup_{k=1}^{p} S_k$ as a single dataset, MultiBoost trains base learners separately for each source and weights examples by domain probabilities $\mathsf{Q}(k|\cdot)$. We used $T = 100$ boosting steps for all benchmarks. T up to 1,000 were explored, but did not change any results significantly.

We report classification errors on the test data for various mixtures of the source distributions $\lambda_1, \ldots, \lambda_p$, including: errors for λ on the simplex Δ edges (errors on each domain separately); errors on the uniform mixture $\forall k : \lambda_k = \frac{1}{p}$; agnostic error, which is the maximum error across all sources. The errors and their standard deviations are reported based on 10-fold cross validation. Each source S_k is independently split into 10 folds $S_k^1, \ldots, S_k^{10}$. For the i-th cross-validation step, the test set is $\{S_1^i, \ldots, S_p^i\}$, while the rest is used for training.

Datasets and preprocessing steps used are described below with additional dataset details provided in Appendix B.8. Note, that all datasets are public and do not contain any personal identifiers or offensive information. The experiments were performed on Linux and Mac workstations with Quad-Core Intel Core i7 2.9 GHz and Intel Xeon 2.20 GHz respectively. The time taken to perform 10-fold cross-validation varies per dataset: from roughly 1 hour on CPU for all benchmarks on tabular data to 8 hours on CPU for all benchmarks on digits data. The run-times of a single MultiBoost iteration is almost identical to that of AdaBoost-all when a closed form solution is used for step size η_t (line 12 in Figure 4.1) for the exponential surrogate loss function. Alternatively, for some experiments we used line search with 100 steps. If training time is critical, one can use $\eta_t = C/\sqrt{t}$ instead, at the cost of slightly worse convergence guarantee. Convergence plots

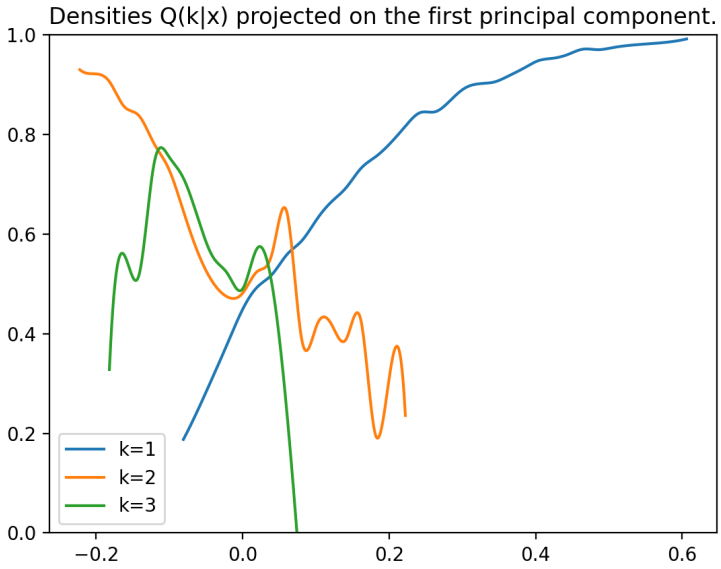


Figure 4.3: Mean values of $\mathsf{Q}(k|\cdot_k)$ for the three domains of the Sentiment data projected on the first principal component of the joint data.

are illustrated in Appendix B.9.

Sentiment analysis. The *Sentiment* dataset [Blitzer et al. 2007] consists of text reviews and rating labels for products sold on Amazon in various categories. We selected $p = 3$ sources from this dataset, where $k = 1$ corresponds to `books`, $k = 2$ means `dvd` and $k = 3$ is `electronics`. We converted the labels to binary values, assigning $y = +1$ for a positive review and $y = -1$ for a negative one. The benchmarks are trained using bigram features generated from the raw text reviews. Since the *Sentiment* dataset contains only 2,000 instances for each domain, we used random train/test splits with 10 different seeds instead of the 10-fold cross-validation. To illustrate the importance of the conditional domain probabilities $\mathsf{Q}(K|x)$, for each domain, we illustrate its values along the projections of x onto the first principal component of the joint *Sentiment* dataset in Figure 4.3. The figure shows that our domain classifiers are able to provide coherent separation between the three domains and narrow the applicability of the base classifiers.

Table 4.1: Test errors for multiple benchmarks.

Algorithm	Error-1	Error-2	Error-3	Error-Uniform	Error-Agnostic
Sentiment Analysis					
AdaBoost-1	**0.326 ± 0.019**	0.300 ± 0.008	0.360 ± 0.017	0.329 ± 0.017	0.360 ± 0.019
AdaBoost-2	0.354 ± 0.020	**0.266 ± 0.009**	0.336 ± 0.023	0.318 ± 0.019	0.357 ± 0.019
AdaBoost-3	0.402 ± 0.015	0.334 ± 0.008	**0.258 ± 0.015**	0.331 ± 0.016	0.402 ± 0.018
AdaBoost-all	0.354 ± 0.020	0.325 ± 0.011	0.313 ± 0.022	0.324 ± 0.021	0.354 ± 0.016
MultiBoost	0.332 ± 0.027	0.288 ± 0.018	0.284 ± 0.027	**0.301 ± 0.027**	**0.332 ± 0.024**
Digits Recognition (4 vs. 9)					
AdaBoost-1	**0.044 ± 0.007**	0.615 ± 0.012	0.476 ± 0.022	0.379 ± 0.008	0.615 ± 0.012
AdaBoost-2	0.455 ± 0.014	0.299 ± 0.011	0.504 ± 0.015	0.420 ± 0.011	0.504 ± 0.015
AdaBoost-3	0.549 ± 0.034	0.488 ± 0.015	0.300 ± 0.013	0.446 ± 0.013	0.549 ± 0.034
AdaBoost-all	0.060 ± 0.009	0.374 ± 0.015	0.353 ± 0.012	0.262 ± 0.009	0.374 ± 0.015
MultiBoost	0.096 ± 0.008	**0.283 ± 0.028**	**0.246 ± 0.014**	**0.209 ± 0.013**	**0.284 ± 0.027**
Digits Recognition (1 vs. 7)					
AdaBoost-1	**0.005 ± 0.002**	0.613 ± 0.007	0.519 ± 0.012	0.379 ± 0.004	0.613 ± 0.007
AdaBoost-2	0.431 ± 0.022	**0.252 ± 0.009**	0.479 ± 0.012	0.387 ± 0.010	0.479 ± 0.012
AdaBoost-3	0.680 ± 0.031	0.490 ± 0.014	**0.244 ± 0.012**	0.474 ± 0.013	0.680 ± 0.031
AdaBoost-all	0.014 ± 0.003	0.286 ± 0.010	0.306 ± 0.012	0.202 ± 0.005	0.306 ± 0.011
MultiBoost	0.026 ± 0.004	0.261 ± 0.013	0.257 ± 0.015	**0.181 ± 0.005**	**0.261 ± 0.011**
Objects Recognition (Fashion-MNIST)					
AdaBoost-1	**0.015 ± 0.003**	0.251 ± 0.026	0.602 ± 0.028	0.288 ± 0.017	0.602 ± 0.028
AdaBoost-2	0.435 ± 0.007	**0.015 ± 0.002**	0.169 ± 0.012	0.173 ± 0.003	0.435 ± 0.007
AdaBoost-3	0.311 ± 0.018	0.097 ± 0.005	**0.014 ± 0.002**	0.140 ± 0.006	0.311 ± 0.018
AdaBoost-all	0.036 ± 0.004	0.020 ± 0.002	0.025 ± 0.003	0.027 ± 0.002	0.036 ± 0.004
MultiBoost	0.028 ± 0.003	0.015 ± 0.003	0.022 ± 0.002	**0.021 ± 0.001**	**0.028 ± 0.003**
Tabular Data (Adult Data)					
AdaBoost-1	0.201 ± 0.012	0.249 ± 0.021	0.215 ± 0.014	0.222 ± 0.012	0.249 ± 0.021
AdaBoost-2	0.298 ± 0.011	**0.131 ± 0.009**	0.142 ± 0.006	0.190 ± 0.007	0.298 ± 0.011
AdaBoost-3	0.225 ± 0.015	0.138 ± 0.010	0.133 ± 0.011	0.165 ± 0.008	0.225 ± 0.015
AdaBoost-all	0.221 ± 0.013	0.134 ± 0.007	0.131 ± 0.010	0.155 ± 0.004	0.221 ± 0.013
MultiBoost	**0.190 ± 0.014**	0.132 ± 0.008	**0.130 ± 0.009**	**0.150 ± 0.005**	**0.190 ± 0.014**

Digits recognition. For this problem, we aggregated 32x32 pixel handwritten digits images from $p = 3$ sources: *MNIST* ($k = 1$), *SVHN* ($k = 2$), *MNIST-M* ($k = 3$). We compared algorithms on two binary classification problems: digits 4 vs. 9 and digits 1 vs. 7.

Object recognition. We divided images of clothes items from *Fashion-MNIST* [Xiao et al. 2017a] dataset into two classes: tops and bottoms from $p = 3$ sources. $k = 1$ consists of t-shirts, pullovers, trousers; $k = 2$ consists of dresses, coats, sandals;

$k = 3$ consists of shirts, bags, sneakers, ankle-boots.

Tabular data. In addition to text and images, we tested MULTIBOOST on tabular data. We used the *Adult* dataset [Kohavi 1996], which consists of numerical and categorical features describing an individual's socioeconomic characteristics, given the task to predict if an person's income exceeds USD $50,000$. We divided this dataset into $p = 3$ sources based on individual's educational background. Source $k = 1$ consisted of individuals with a university degree (BSc, MS or PhD), source $k = 1$ those with only a High School diploma, and source $k = 2$, none of the above.

Discussion As can be seen from Table 4.1, MULTIBOOST provides agnostic and uniform errors that are significantly better than the baselines on all datasets. While ADABOOST-k predictors on digits recognition and object recognition tasks each show the lowest error on their specific domains $\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3$, they fail to generalize to other target distribution in $\text{conv}\{\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3\}$. Moreover, as suggested by the discussion in Section 4.2, the standard convex combination of domain-specific predictors (ADABOOST-all) also performs poorly on several target distributions in $\text{conv}\{\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3\}$, such as the uniform and agnostic targets. As the experimental results confirm, MULTIBOOST, by leveraging domain predictors $\mathsf{Q}(k|\cdot)$ and the agnostic loss, is able to produce an ensemble of domain-specific predictors that generalizes to different targets. Appendix B.9 provides additional illustrations of $\mathsf{Q}(k|\cdot)$ by projecting each domain on the first principal component of the joint data.

Moreover, for tabular data and the digits (4 vs. 9) classification problem, MULTIBOOST benefits from positive knowledge transfer and obtains an error on individual domains that is even smaller than that of the domain-specific predictors. From Figure B.3 in Appendix B.9 it can be seen that the α-weight for the MULTIBOOST classifier in the 4 vs. 9 problem is heavily centered on $\mathcal{D}_2$, but contributions from $\mathcal{D}_1$ and $\mathcal{D}_3$ adds to the stronger performance. The same figure also illustrates the α-mass distribution for the other classification tasks. The standard convex ensemble

ADABOOST-all, however, does not seem to exhibit the positive transfer property. Similar conclusions carry over for other target distributions in $\mathrm{conv}\{\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3\}$, for the sake of simplicity we do not present them here.

4.7 CONCLUSION

We presented a new boosting algorithm, MULTIBOOST, for a multiple-source scenario – a common setting in domain adaptation problems – where the target distribution can be any mixture of the source distributions. We showed that our algorithm benefits from strong theoretical guarantees and exhibits favorable empirical performance. We also highlighted the extension of our work to the federated learning scenario, which is a critical distributed learning setting in modern applications.

5 | AGNOSTIC LEARNING WITH MULTIPLE OBJECTIVES

The majority of machine learning tasks are inherently multi-objective. This means that the learner has to come up with a model that performs well across a number base objectives $\mathcal{L}_1, \ldots, \mathcal{L}_p$ as opposed to a single one. Since optimizing with respect to multiple objectives at the same time is often computationally expensive, the base objectives are often combined in an ensemble $\sum_{k=1}^{p} \lambda_k \mathcal{L}_k$, reducing the problem to scalar optimization. The mixture weights λ_k are set to uniform or some other fixed distribution based on the learner's preferences. We argue that learning with a fixed distribution on the mixture weights has a risk of overfitting to some individual objectives and significantly hurting the others, while still performing well on an entire ensemble. Moreover, in reality the true preferences of a learner across multiple objectives are often unknown or hard to express as a specific distribution. Instead, we propose a new framework of *Agnostic Learning with Multiple Objectives* (ALMO), where a model is optimized for *any* weights in the mixture of base objectives. We present data-dependent Rademacher complexity guarantees for learning in the ALMO framework, which are then used to guide a scalable optimization algorithm and the corresponding regularization. We present convergence guarantees for this algorithm assuming convexity of the loss functions and the underlying hypotheses space. We further implement the algorithm in a popu-

lar symbolic gradient computation framework and empirically demonstrate on a number of datasets the benefits of ALMO framework versus learning with a fixed mixture weights distribution.

5.1 MOTIVATION

Machine learning is inherently a multi-objective task [Jin 2006; Jin and Sendhoff 2008]. In most real-world problems, a learner is required to find a model with strong performance across a set of objectives $\{\mathcal{L}_1, \ldots, \mathcal{L}_p\}$. Committing to a single objective $\mathcal{L}_k$ often fails to capture the full complexity of the underlying problem and causes models to overfit to that individual objective [Kendall et al. 2018].

For example, the BLEU score [Papineni et al. 2002] is commonly selected as a single objective for model training and evaluation in machine translation literature. However, it overfits to short sentences, because it is focused on word-based n-gram precision [Duh et al. 2012]. Thus, it is often agreed in the research community that machine translation models need to perform well not just in the BLEU score, but in other metrics that measure various aspects of translation beyond the n-gram similarity. For example, METEOR [Lavie and Agarwal 2007] that measures synonym matching or RIBES [Isozaki et al. 2010] that accounts for deviation in word order.

A well-studied approach to working with multiple correlated objectives such as the ones described above is to estimate a Pareto-efficient frontier and seek for a solution that lies on that frontier [Jin and Sendhoff 2008; Sener and Koltun 2018; Shah and Ghahramani 2016; Marler and Arora 2004]. While this approach can provide a rigorous description of the trade-offs between multiple correlated objectives, the estimation of Pareto-efficient frontier is computationally expensive [Duh et al. 2012; Godfrey et al. 2007] and poorly scales with the number of base objectives. Moreover, optimization along the Pareto-efficient frontier typically requires to compute multiple

gradients for each step, which is challenging to adapt to large-scale problems. Additionally, even when the Pareto frontier is computed, some selection criterion is needed to choose a point on that frontier. In Section 5.3, we will prove that our solution is actually guaranteed to be Pareto-optimal, thus lying on the frontier. Our algorithms hence provide a theoretical justification for selecting one of the possibly many points on the frontier.

In short, despite the multi-objective nature of machine learning, working with a vector of multiple objectives at the same time turns out to be computationally challenging. For that reason, researchers often combine the set of base objectives $\{\mathcal{L}_1, \ldots, \mathcal{L}_p\}$ into a weighed ensemble $\mathcal{L}_\lambda = \sum_{k=1}^{p} \lambda_k \mathcal{L}_k$ for some λ in a simplex Δ_p, essentially making it a scalar function [Jin 2006]. This allows one to make use of efficient scalar function optimization algorithms on large-scale problems, typically using various stochastic gradient descent techniques [Bottou et al. 2018]. However, working with an ensemble of base objectives raises a natural question: how should we set the mixture weights $\lambda_1, \ldots, \lambda_p$? For example, if we define $\mathcal{L}_\lambda = \lambda_1 \cdot \text{BLEU} + \lambda_2 \cdot \text{METEOR}$, should we assume that BLEU and METEOR are equally important and set the mixture weights to be uniform $\lambda_1 = \lambda_2 = \frac{1}{2}$? Or should we say that METEOR is significantly more important than BLEU for a particular problem and λ_2 should be set higher that λ_1?

Despite the simplicity of the questions above, there is no clear answer to how to determine the mixture weights for multi-objective problems. Particularly, this is because there is no straightforward way to map the requirements of a particular problem that a learner is trying to solve to a corresponding distribution of the mixture weights [Van Moffaert et al. 2014]. Thus, the mixture weights are usually assigned to uniform by default. However, in many cases the uniform ensemble of objectives can do more harm than good. This is simply because fitting models with the uniform mixture weights can significantly hurt some of the individual objectives in the mixture. For many machine learning applications (e.g., vision or speech), a significant decrease in performance on one

objective is intolerable even if the performance on the uniform average of objectives is improved.

One can argue that if the uniform combination of objectives is not the natural target, then we should just set mixture weights for every problem separately based on the relative importance of each of the base losses. However, this is still not the true goal, because the learner's preferences are often shifting over time, unobservable, and in some cases even unknown. It is also often the case in the machine learning industry that when several parties develop a model for a particular multi-objective problem, their preferences for the base objectives are conflicting with each other.

Given these challenges of working with a fixed mixture weights distribution discussed above, we argue that the natural goal of a learner with multiple objectives $\{\mathcal{L}_1, \ldots, \mathcal{L}_p\}$ has *agnostic* nature. That is, a learner wants to obtain a model that performs well for *any* possible combination of mixture weights in the ensemble $\mathcal{L}_\lambda = \sum_{k=1}^{p} \lambda_k \mathcal{L}_k$. Such a goal is also *risk-averse*, since being robust against any combination of ensemble weights is also robust against their worst (adversarial) combination.

We propose a new framework of *Agnostic Learning with Multiple Objectives* (ALMO) inspired by the Agnostic Federated Learning algorithm [Mohri et al. 2019a], where the underlying model is optimized for any possible distribution of the mixture weights in the ensemble $\mathcal{L}_\lambda = \sum_{k=1}^{p} \lambda_k \mathcal{L}_k$. Instead of optimizing the model for a fixed distribution of $\lambda_1, \ldots, \lambda_p$, with the high risk of overfitting to a subset of the base objectives, we define the *agnostic* loss function that ensures the model performs well against any mixture, including the worst-case mixture values. Thus, the ALMO framework closely matches the learning setting used by most real-world ML practitioners, who often seek to deploy a model that is robust against any mixture of base objectives and does not need to be frequently retrained when the base objective preferences shift.

In this paper, we provide sample-dependent generalization bounds based on the Rademacher complexity of the underlying loss functions, which are then used to define the optimization algo-

rithm and the regularization for the problem. We suggest an efficient optimization algorithm for the ALMO setting that is inspired by the generalization bounds derived in this paper. The algorithm can be applied in a straightforward manner across multiple tasks, since it is based on stochastic gradient descent and can be implemented in popular symbolic gradient computation platforms, including TENSORFLOW [Abadi et al. 2016], KERAS [Chollet et al. 2015] or PYTORCH [Paszke et al. 2019].

We conduct a series of experiments on various datasets that demonstrate the benefits of the ALMO framework versus learning with a uniform mixture weights distribution. The experiments also show that the ALMO framework provides a generally robust model that performs better than baselines on metrics that are not even included in the original set of base objectives. Moreover, while the algorithm convergence bound holds for convex hypotheses spaces, we show that the ALMO framework provides robust models even for non-convex hypotheses classes, such as Deep Neural Networks.

The rest of the paper is structured as follows. In Section 5.2 we formally describe the ALMO framework, define the agnostic multi-objective loss function, and discuss the connection of our solution to the Pareto-optimal frontier. In Section 5.3 we derive the sample-dependent Rademacher complexity bounds for the ALMO framework. In Section 5.4 we put forward a stochastic gradient-descent-based algorithm inspired by the generalization bounds. In Section 5.5 we describe our experimental setting and provide empirical results.

5.2 LEARNING SCENARIO

In this section, we introduce the formal learning scenario of the ALMO framework. We define the *agnostic* loss function and argue that by optimizing this loss, the learner obtains a model that is

robust against any mixture weights distribution in the ensemble $\mathcal{L}_\lambda = \sum_{k=1}^{p} \lambda_k \mathcal{L}_k$.

We consider the supervised learning setting, in which the learner receives a labeled sample $S = (x_1, y_1), \ldots, (x_m, y_m)$ drawn *i.i.d.* from some distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$ denotes he input space and $\mathcal{Y}$ denotes the output space. Let $\mathcal{H}$ be the hypotheses space and $l : \mathcal{X} \times \mathcal{Y} \mapsto \mathbb{R}_+$ be a loss function. The loss of $h \in \mathcal{H}$ for a labeled instance (x, y) equals $\ell(h(x), y)$, and the expectation of the loss is $\mathcal{L}(h) = \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\ell(h(x), y)\right]$. In a standard supervised learning scenario, the goal of the learner is to find $h^\star = \operatorname{argmin}_{h\in\mathcal{H}} \mathcal{L}(h)$.

However, as we explained in the previous section, the majority of learning applications require optimizing over a number of objectives (loss functions) at the same time. In such cases, the learner has access to a set of p *base loss functions* $\{l_1, \ldots, l_p\}$. For any $h \in \mathcal{H}$, the expectations of these base loss functions are $\{\mathcal{L}_1(h), \ldots, \mathcal{L}_p(h)\}$, where $\forall k : \mathcal{L}_k(h) = \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\ell_k(h(x), y)\right]$. The goal of the learner is to obtain a hypothesis $h \in \mathcal{H}$ that performs well across any weighted combination of the expected base losses $\mathcal{L}_\lambda(h) = \sum_{k=1}^{p} \lambda_k \mathcal{L}_k(h)$, for some $\lambda \in \Delta_p$.

We assume that the learner has a defined boundary set of preferences over the mixture of objectives that can be described by a convex subset $\Lambda \subset \Delta_p$. However, the true mixture coefficients λ within Λ are unknown. Thus, the learner wants to have a solution that is favorable for any possible $\lambda \in \Lambda$. Therefore, we define the *agnostic multi-objective loss* $\mathcal{L}_\Lambda(h)$ associated to a predictor $h \in \mathcal{H}$ and a subset $\Lambda \subset \Delta_p$ as follows:

$$\mathcal{L}_\Lambda(h) = \max_{\lambda\in\Lambda} \mathcal{L}_\lambda(h). \tag{5.1}$$

The goal of the learner is to solve the constraint optimization and find the hypothesis $h_\Lambda \in \mathcal{H}$:

$$\text{solve:} \quad \min_{h\in\mathcal{H}} \mathcal{L}_\Lambda(h) \quad \text{and find:} \quad h_\Lambda = \operatorname*{argmin}_{h\in\mathcal{H}} \mathcal{L}_\Lambda(h), \tag{5.2}$$

To determine the best hypotheses, a sample estimate of the agnostic multi-objective loss $\mathcal{L}_\Lambda(h)$ is needed. For that we define the *empirical agnostic multi-objective loss* $\widehat{\mathcal{L}}_\Lambda(h)$ as follows:

$$\widehat{\mathcal{L}}_\Lambda(h) = \max_{\lambda \in \Lambda} \sum_{k=1}^{p} \lambda_k \widehat{\mathcal{L}}_k(h) = \max_{\lambda \in \Lambda} \sum_{k=1}^{p} \lambda_k \frac{1}{m} \sum_{i=1}^{m} \ell_k(h(x_i), y_i). \tag{5.3}$$

We will show in Section 5.3 that with high probability $\mathcal{L}_\Lambda(h)$ is bounded by $\widehat{\mathcal{L}}_\Lambda(h)$ and terms dependent on the Rademacher complexity of the underlying function class.

Assume the hypotheses set $\mathcal{H}$ is parameterized by a vector $w \in \mathcal{W}$, thus $h \in \mathcal{H}$ can be denoted as h_w. When the learner has access to empirical distribution S_m, the (unregularized) optimization problem of the learner is formulated as follows:

$$\min_{w \in \mathcal{W}} \max_{\lambda \in \Lambda} \sum_{k=1}^{p} \lambda_k \underbrace{\frac{1}{m} \sum_{i=1}^{m} \ell_k(h_w(x_i), y_i)}_{\widehat{\mathcal{L}}_k(h_w)}. \tag{5.4}$$

Given a number of standard assumptions on $\mathcal{W}, \Lambda$ and the base loss functions, the optimization problem above is convex, and can be solved using gradient-based algorithms, as we show in Section 5.4.

5.2.1 Relationship with the Pareto-Optimal Frontier

In this section, we discuss the connection between the ALMO loss function and the Pareto frontier. In particular, we will show that the solution of the ALMO loss function is a point on the Pareto-optimal frontier.

The Pareto-optimal frontier is the set of feasible loss tuples $(\ell_1(h), \ldots, \ell_p(h))$, $h \in \mathcal{H}$, that are not *strictly dominated* by some other feasible tuple. The function $h \in \mathcal{H}$, or the corresponding tuple $(\ell_1(h), \ldots, \ell_p(h))$, is strictly dominated by $h' \in \mathcal{H}$, if $\ell_k(h') \leq \ell_k(h)$ for all $k \in [p]$ with

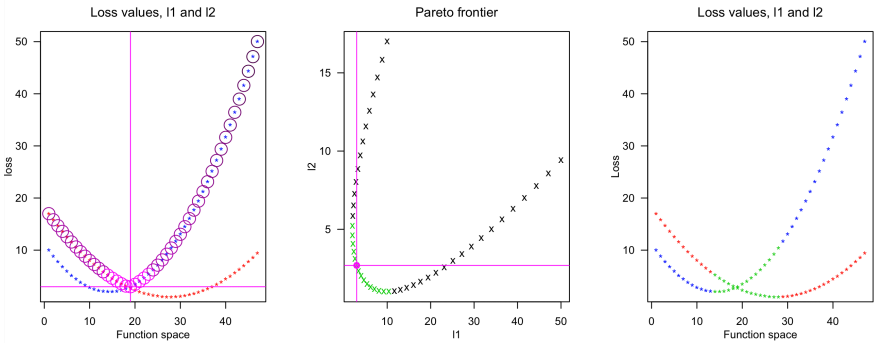


Figure 5.1: Illustration of the connection between the ALMO loss and the Pareto-optimal frontier. Left: two losses in a parameterized function space; middle: the Pareto-optimal frontier; right: the optimal frontier indicated in the function space. See text for further explanation.

$\ell_j(h') < \ell_j(h)$ for at least one index $j \in [p]$.

Proposition 5.1. *The agnostic solution is Pareto-optimal.*

Proof. Assume that $h^* \in \mathcal{H}$ is a minimizer of the ALMO objective for $\{\ell_1, \ldots, \ell_p\}$.[1] That is, for all $h \in \mathcal{H}$,

$$\min_{h \in \mathcal{H}} \mathcal{L}_\Lambda(h) = \sum_{k=1}^{p} \lambda_k^* \ell_k(h^*) \leq \sum_{k=1}^{p} \lambda_k^* \ell_k(h),$$

for some $\lambda^* \in \Lambda$. Let h' be a strictly dominating point for h^*. That is, $\forall k \in [p], \ell_k(h') \leq \ell_k(h^*)$, with $\ell_j(h') < \ell_j(h^*)$ for at least one $j \in [p]$. Then, we must have:

$$\sum_{k=1}^{p} \lambda_k^* \ell_k(h') < \sum_{k=1}^{p} \lambda_k^* \ell_k(h^*) = \min_{h \in \mathcal{H}} \mathcal{L}_\Lambda(h),$$

which, in view of the definition of minimality of h^*, implies that h' is not in $\mathcal{H}$, that the point $(\ell_1(h'), \ldots, \ell_p(h'))$ is not feasible, and that h^* is Pareto-optimal. □

We illustrate the connection between the Pareto-optimal frontier and the optimizer of the ALMO loss with the plots in Figure 5.1 for two losses ℓ_1 and ℓ_2. The leftmost plot in Figure 5.1

[1] Note that the solution is unique for strictly convex losses.

illustrates the two losses, ℓ_1 in blue and ℓ_2 in red as a function of the parameterized function space. For a given function h along the x-axis, the convex combination of the losses $\lambda\ell_1(h)+(1-\lambda)\ell_2(h)$ attains its maximum with respect to λ at the value $\max\{\ell_1(h), \ell_2(h)\}$, circled in violet, with a value of $\lambda \in \{0, 1\}$. The darker the violet color, the larger the value of the ALMO objective. The minimum over $h \in \mathcal{H}$ of the ALMO objective is attained for $\partial\mathcal{L}_\Lambda(h)/\partial\lambda = 0$, which is at the crossing point $\ell_1(h^*) = \ell_2(h^*)$ of the loss curves. At a crossing point, any value of λ attains this minimum, a natural choice is to pick the λ corresponding to uniform weights.

In the middle plot we illustrate the corresponding Pareto-optimal frontier of the points (ℓ_1, ℓ_2) in green and indicate the ALMO solution. Finally in the rightmost plot we tie back the Pareto-optimal frontier to the plot of the losses.

5.3 Learning Guarantees

Here, we derive learning guarantees for the ALMO framework, that rely on the Rademacher complexity bounds on the family of loss functions and the mixture weights λ. The bounds that we show, being data-dependent, will motivate a scalable algorithm for the minimization of the agnostic multi-objective loss. The algorithm is presented in Section 5.4.

Let $\mathcal{G}$ denote the family of the losses associated to a hypotheses set $\mathcal{H}$: $\{\mathcal{G} : (x, y) \mapsto \ell(h(x), y) : \forall h \in \mathcal{H}\}$. Typically, generalization analysis in similar cases involves bounding the sample Rademacher complexity of $\mathcal{G}$, which is

$$\widehat{\mathfrak{R}}_S(\mathcal{G}) = \mathop{\mathbb{E}}_{\boldsymbol{\sigma}}\left[\sup_{h\in\mathcal{H}} \frac{1}{m}\sum_{i=1}^{m} \sigma_i \ell(h(x_i), y_i)\right], \tag{5.5}$$

where σ_i-s, $i \in [m]$, are independent uniformly distributed random variables taking values in $\{-1, 1\}$.

However, the agnostic multi-objective loss $\mathcal{L}_\Lambda(h)$ contains a maximum over $\lambda \in \Lambda$, which causes additional complication. For the proper analysis, we need to extend the definition of the sample Rademacher complexity by including the λ_k terms as follows:

$$\widehat{\mathfrak{R}}_S(\mathcal{G}, \lambda) = \mathop{\mathbb{E}}_{\boldsymbol{\sigma}} \left[\sup_{h \in \mathcal{H}} \sum_{k=1}^{p} \lambda_k \frac{1}{m} \sum_{i=1}^{m} \sigma_i \ell_k(h(x_i), y_i) \right]. \tag{5.6}$$

By using $\widehat{\mathfrak{R}}_S(\mathcal{G}, \lambda)$ in the theorem below, we show the connection between the theoretical and empirical agnostic multi-objective losses $\mathcal{L}_\Lambda(h)$ and $\widehat{\mathcal{L}}_\Lambda(h)$. The theorem below is based on common steps in Rademacher complexity bound proofs of [Koltchinskii and Panchenko 2002b; Hoffman et al. 2018b; Mohri et al. 2012]. Additionally, for the theorem below we assume that $\forall (x, y), (x', y') \in \mathcal{X} \times \mathcal{Y}, \forall h \in \mathcal{H} : ||(h(x'), y') - (h(x), y)|| \leq \mathcal{D}_\mathcal{H}$. This holds for all bounded hypotheses classes, for example for the case of binary classification $\mathcal{D}_\mathcal{H} = O(\sqrt{C})$, where C is the number of classes.

Theorem 5.2. *If the loss functions ℓ_k are M_k-Lipschitz and bounded by M, then for any $\epsilon > 0$ and $\delta > 0$, with probability at least $1 - \delta$, the following inequality holds for $\forall h \in \mathcal{H}$ and $\forall \lambda \in \Lambda$:*

$$\mathcal{L}_\lambda(h) \leq \widehat{\mathcal{L}}_\lambda(h) + 2\widehat{\mathfrak{R}}_S(\mathcal{G}, \lambda) + M\epsilon + \mathcal{D}_\mathcal{H} \sum_{k=1}^{p} \lambda_k M_k \sqrt{\frac{1}{2m} \log \left[\frac{|\Lambda|_\epsilon}{\delta} \right]}, \tag{5.7}$$

where Λ_ϵ is an minimum ϵ-cover of Λ.

The proof of Theorem 5.2 is given in Appendix C.2.1 The bound in Theorem 5.2 is a function of λ, to make it uniform with respect to λ, we first observe that when ℓ_k are M_k-Lipschitz, by direct application of Talagrand's Lemma [Ledoux and Talagrand 1991a]:

$$\widehat{\mathfrak{R}}_S(\mathcal{G}, \lambda) \leq \sum_{k=1}^{p} \lambda_k M_k \widehat{\mathfrak{R}}_S(\mathcal{H}). \tag{5.8}$$

Thus, the only terms that are λ-dependent in the bound are of the form $\sum_{k=1}^{p} \lambda_k M_k$. This naturally leads us to control the complexity by imposing a regularization of the form $\sum_{k=1}^{p} \lambda_k M_k \leq \beta$ and leads to the following generalization bound for the agnostic multi-objective loss:

Theorem 5.3. *Let the loss functions ℓ_k be M_k-Lipschitz and bounded by M. Assume $\forall \lambda \in \Lambda : \sum_{k=1}^{p} \lambda_k M_k \leq \beta$, then for any $\epsilon > 0$ and $\delta > 0$, with probability at least $1 - \delta$, the following inequality holds for $\forall h \in \mathcal{H}$:*

$$\mathcal{L}_{\Lambda}(h) \leq \widehat{\mathcal{L}}_{\Lambda}(h) + 2\beta \widehat{\mathfrak{R}}_{S}(\mathcal{H}) + M\epsilon + \beta \mathcal{D}_{\mathcal{H}} \sqrt{\frac{1}{2m} \log \left[\frac{|\Lambda|_{\epsilon}}{\delta} \right]}. \tag{5.9}$$

The bound in Theorem 5.3 says that with high probability, the agnostic multi-objective loss $\mathcal{L}_{\Lambda}(h)$ is bounded by the empirical agnostic multi-objective loss $\widehat{\mathcal{L}}_{\Lambda}(h)$ and the sample-dependent complexity terms. The bound suggests that the learner in the ALMO framework should seek to find such a hypothesis $h \in \mathcal{H}$ that provides the best trade-off between the empirical loss $\widehat{\mathcal{L}}_{\lambda}(h)$ and the Rademacher complexity. This means that the learner seeking to minimize $\mathcal{L}_{\lambda}(h)$ should minimize its empirical estimate $\widehat{\mathcal{L}}_{\lambda}(h)$ and apply a regularization to control the complexity terms. The bound provides significant and also very clear intuition that the regularization should connect the properties of the constraint set Λ and the Lipshitz properties of each individual base loss function in the form $\sum_{k=1}^{p} \lambda_k M_k \leq \beta$. This means that the ALMO algorithm should control for the M_k-weighted norm of the mixture weights vector $\boldsymbol{\lambda}$. Note, that for the cases where it is nontrivial to estimate the individual Lipschitz constants M_k, our theory allows to use a simplified regularization $\|\boldsymbol{\lambda}\|_q \leq \beta$ for some $q-$norm $\|\cdot\|_q$, which will also control $\sum_{k=1}^{p} \lambda_k M_k$ for bounded loss functions.

5.4 ALGORITHM

In this section, we describe the algorithm for the ALMO framework and the regularization that is inspired by the learning guarantees derived in Section 5.3. Moreover, we derive the convergence guarantees for the algorithm, which demonstrate that it scales well to large problems. To emphasize that for the optimization purpose, $w \in \mathcal{W}$ and $\lambda \in \Lambda$ are real-valued vectors, we will use bold symbols for the pair $(\mathbf{w}, \boldsymbol{\lambda})$.

5.4.1 REGULARIZATION

The generalization bounds in Section 5.3 suggest to minimize the agnostic empirical loss $\widehat{\mathcal{L}}_{\Lambda}(h)$, while controlling for the complexity of hypotheses class $\mathcal{H}$. This naturally translates to the following regularized minimization problem in the ALMO framework:

$$\min_{h \in \mathcal{H}} \max_{\boldsymbol{\lambda} \in \operatorname{conv}(\Lambda)} \widehat{\mathcal{L}}_{\lambda}(h) + \beta_1 \|h\|_{\mathcal{H}} + \beta_2 \sum_{k=1}^{p} \lambda_k M_k, \tag{5.10}$$

where $\beta_1, \beta_2 \geq 0$ are regularization parameters and $\|\cdot\|_{\mathcal{H}}$ is a norm defined on the hypotheses space $\mathcal{H}$. Note that since $\widehat{\mathcal{L}}_{\lambda}(h)$ is linear in λ, we have replaced $\max_{\boldsymbol{\lambda} \in \Lambda}$ by $\max_{\boldsymbol{\lambda} \in \operatorname{conv}(\Lambda)}$.

Given a number of commonly used assumptions, the regularized problem in Equation 5.10 is convex. Assuming that the base loss functions $\{\ell_1, \cdots, \ell_p\}$ are convex in the first argument ensures that $\widehat{\mathcal{L}}_{\lambda}(h)$ is convex as well. Since the norm $\|h\|_{\mathcal{H}}$ is convex, then $\widehat{\mathcal{L}}_{\lambda}(h) + \beta_1 \|h\|_{\mathcal{H}}$ is a convex function of h. The maximum over $\boldsymbol{\lambda}$ of convex functions is a convex function itself. Thus, for a convex hypotheses space $\mathcal{H}$ the problem in Equation 5.10 is convex.

5.4.2 OPTIMIZATION

The optimization problem in Equation 5.10 can be solved using projected gradient descent or more generally with mirror descent algorithms [Nemirovski and Yudin 1983]. For large-scale problems, we suggest to use the Stochastic Mirror-Prox gradient descent algorithm [Juditsky et al. 2011]. For the specific optimization problem presented in this paper, the Mirror-Prox algorithm admits a simplified form and can be implemented in common machine learning software that relies on symbolic gradient computation.

Assuming that the hypotheses set $\mathcal{H}$ is parameterized by a vector $\mathbf{w} \in \mathcal{W}$, we can simplify the notations to emphasize that the optimization is performed over $(\mathbf{w}, \boldsymbol{\lambda})$. Let $L(\mathbf{w}) = \frac{1}{m}\sum_{i=1}^{m} \ell(h_{\mathbf{w}}(x_i), y_i)$, then $L(\mathbf{w}, \boldsymbol{\lambda}) = \sum_{k=1}^{p} \lambda_k L_k(\mathbf{w})$. It is sufficient to present the algorithm for the unregularized problem

$$\min_{\mathbf{w}\in\mathcal{W}} \max_{\boldsymbol{\lambda}\in\Lambda} L(\mathbf{w}, \boldsymbol{\lambda}), \tag{5.11}$$

which can be extended in a straightforward manner to include the regularization term for $\mathbf{w}$ and $\boldsymbol{\lambda}$.

Let $\nabla_{\mathbf{w}} L(\mathbf{w}, \boldsymbol{\lambda})$ be the gradient of the loss function $L(\mathbf{w}, \boldsymbol{\lambda})$ with respect to $\mathbf{w}$ and $\nabla_{\boldsymbol{\lambda}} L(\mathbf{w}, \boldsymbol{\lambda})$ be the gradient with respect to $\boldsymbol{\lambda}$. Let $\delta_{\mathbf{w}} L(\mathbf{w}, \boldsymbol{\lambda})$ and $\delta_{\boldsymbol{\lambda}} L(\mathbf{w}, \boldsymbol{\lambda})$ be the unbiased stochastic estimates of these gradients with respect to $\mathbf{w}$ and $\boldsymbol{\lambda}$. The Mirror-Prox descent adapted to the learning with multiple objectives problem obtains stochastic gradient estimates $\delta_{\mathbf{w}} L(\mathbf{w}, \boldsymbol{\lambda})$ and $\delta_{\boldsymbol{\lambda}} L(\mathbf{w}, \boldsymbol{\lambda})$ at each step $t \in [1, T]$ and then updates $(\mathbf{w}_t, \boldsymbol{\lambda}_t)$. After each update, the projections $\Pi_{\mathcal{W}}$ and Π_{Λ} onto $\mathcal{W}$ and Λ respectively are computed. If $\Lambda = \Delta_p$, then we can efficiently compute these projections [Duchi et al. 2008]. The pseudocode for the ALMO optimization algorithm is given in Algorithm 5.2.

ALMO($\mathbf{w}_0 \in \mathbf{W}, \boldsymbol{\lambda}_0 \in \Lambda$)

1 **for** $t \in [1, T]$ **do**

2 $\quad \mathbf{w}_t \leftarrow \Pi_{\mathcal{W}}\Big[w_{t-1} - \gamma_{\mathbf{w}} \delta_{\mathbf{w}} L(\mathbf{w}_{t-1}, \boldsymbol{\lambda}_{t-1})\Big]$

3 $\quad \boldsymbol{\lambda}_t \leftarrow \Pi_{\Lambda}\Big[\lambda_{t-1} + \gamma_{\boldsymbol{\lambda}} \delta_{\boldsymbol{\lambda}} L(\mathbf{w}_{t-1}, \boldsymbol{\lambda}_{t-1})\Big]$

4 $\mathbf{w}_T \leftarrow \frac{1}{T}\sum_{t=1}^{T} \mathbf{w}_t$

5 $\boldsymbol{\lambda}_T \leftarrow \frac{1}{T}\sum_{t=1}^{T} \boldsymbol{\lambda}_t$

Figure 5.2: Pseudocode for the ALMO algorithm with step sizes $\gamma_{\boldsymbol{\lambda}}, \gamma_{\mathbf{w}}$.

5.4.3 Convergence Guarantees

We can show favorable convergence guarantees for the ALMO algorithm using a number of common assumptions, including of course the convexity of $L(\mathbf{w}, \boldsymbol{\lambda})$ in the first argument.

Theorem 5.4. *[See Appendix C.2.2 for the proof.] For $L(\mathbf{w}, \boldsymbol{\lambda})$ convex in the first argument, assume $\forall \mathbf{w} \in \mathcal{W}, \forall \boldsymbol{\lambda} \in \Lambda : \|\mathbf{w}\|_2 \leq \mathcal{D}_{\mathcal{W}}, \|\boldsymbol{\lambda}\|_2 \leq \mathcal{D}_{\Lambda}, \|\nabla_{\mathbf{w}} L(\mathbf{w}, \boldsymbol{\lambda})\|_2 \leq \mathcal{G}_{\mathbf{w}}, \|\nabla_{\boldsymbol{\lambda}} L(\mathbf{w}, \boldsymbol{\lambda})\|_2 \leq \mathcal{G}_{\boldsymbol{\lambda}}$. Let the variance of unbiased stochastic gradients be bounded by $\sigma_{\mathbf{w}}^2$ and $\sigma_{\boldsymbol{\lambda}}^2$ respectively. For step sizes $\gamma_{\mathbf{w}} = \frac{1}{\sqrt{T}} \frac{2\mathcal{D}_{\mathcal{W}}}{\sqrt{\sigma_{\mathbf{w}}^2 + \mathcal{G}_{\mathbf{w}}^2}}$ and $\gamma_{\boldsymbol{\lambda}} = \frac{1}{\sqrt{T}} \frac{2\mathcal{D}_{\Lambda}}{\sqrt{\sigma_{\boldsymbol{\lambda}}^2 + \mathcal{G}_{\boldsymbol{\lambda}}^2}}$, the following convergence guarantees hold:*

$$\mathbb{E}\left[\max_{\boldsymbol{\lambda} \in \Lambda} L(\mathbf{w}_T, \boldsymbol{\lambda}) - \min_{\mathbf{w} \in \mathcal{W}} \max_{\boldsymbol{\lambda} \in \Lambda} L(\mathbf{w}, \boldsymbol{\lambda})\right] \leq \frac{1}{\sqrt{T}}\left(3\mathcal{D}_{\mathcal{W}}\sqrt{\sigma_{\mathbf{w}}^2 + \mathcal{G}_{\mathbf{w}}^2} + 3\mathcal{D}_{\Lambda}\sqrt{\sigma_{\boldsymbol{\lambda}}^2 + \mathcal{G}_{\boldsymbol{\lambda}}^2}\right). \quad (5.12)$$

5.5 Experiments

The experiments presented in this section illustrate the benefits of the ALMO framework combined with the generalization guarantees derived in Section 5.3 and the stochastic algorithm in Section 5.4. We present results for convex models and deep neural networks to demonstrate that the algorithms performs well even for non-convex hypotheses classes. The ALMO optimization

algorithm is implemented in TENSORFLOW [Abadi et al. 2016] and KERAS [Chollet et al. 2015].

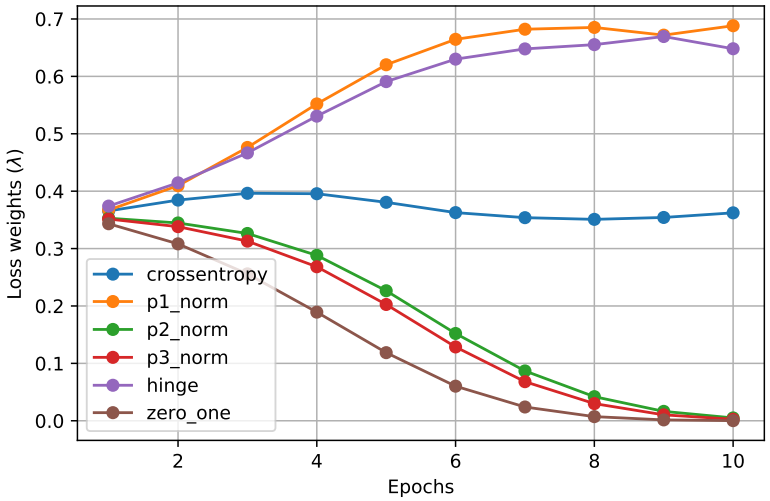


Figure 5.3: DNN training dynamics of mixture weights λ_k on MNIST. Weights are logged at the end of each epoch.

For the base losses $\{\ell_1, \ldots, \ell_p\}$ we use a variety of common training loss functions, each having its own advantages and disadvantages. For example the Zero-One loss function often results in robust models when the data has outliers, but it underperforms when the data is concentrated near decision surface [Zhao et al. 2010]. The opposite usually applies to the hinge loss function. To solve a complex machine learning problem, the learner would want to combine multiple loss functions, making use of their strengths and mitigating weaknesses. For the experiments, each base objective ℓ_k (see Appendix, Table C.1) is normalized so that $M_k \leq 1$ and hence $M = 1$.

For every dataset, we train a model by minimizing the empirical agnostic multi-objective loss $\widehat{\mathcal{L}}_h$ objective and we benchmark it against p models trained with each individual $k-$th base loss function for $k \in [1, p]$. We also include a model trained with fixed uniform mixture weights $\lambda_k = \frac{1}{p}$. Note we don't include techniques based on searching the Pareto-efficient solutions for the mixture of losses, since the end goal of these frameworks is completely different from ALMO. For the connection between the two frameworks, see the discussion in Section 5.2.

The models are compared on MNIST [LeCun and Cortes 2010], Fashion-MNIST [Xiao et al. 2017b] and ADULT [Dua and Graff 2017] datasets with standard feature preprocessing techniques applied. We report results for two model architectures: a logistic regression and a neural network with dimensions 1024-512-128. For both models, we run hyper-parameter tuning with a parameter grid size 50 on a validation set, which is 20% of the training data.

We report the average values of the cross-entropy, zero-one, hinge and 1-norm losses in Tables 5.1 and 5.2. We also provide AUC, which was not included in the training losses. The standard deviations are obtained by retraining every model 3 times.

Experimental results support our claim that the ALMO framework improves the model for the worst performing loss. The hinge loss is often observed with a significant loss value and as illustrated in Figure 5.3 the corresponding λ attains a high value. In the appendix we provide results from training on just one loss at a time and discuss the improvements of ALMO as compared to that baseline.

The resulting model is robust and avoids selecting a specific set of mixture weights for training. It also performs better than the baseline in terms of AUC on all datasets, while AUC was not used for training. Additionally, the ALMO algorithm can be used as a tool for selection of base objectives (e.g., AutoML), as ALMO during training increases the mixture weights for the worst performing base losses, while the others are decreasing (see Figure 5.3). As expected, losses trained with a higher mixture weight show an improvement compared to the Uniform case. Even for cases like cross-entropy in the DNN-MNIST configuration (see Table 5.2), the algorithm balances the 1-norm loss, and AUC shows a more robust model. Also interesting is the significant improvement in AUC for the DNN-Adult dataset, which may be related to the dramatic decrease of the hinge loss. Finally, as shown in both tables, ALMO does not select to optimize the 1-norm loss, assigns it a low mixture weight and does not improve over uniform.

Table 5.1: Comparison of loss functions for logistic regression model.

	MNIST		Fashion MNIST		Adult	
	Uniform	ALMO	Uniform	ALMO	Uniform	ALMO
Cross-entropy	0.1412	**0.0726**	0.1011	**0.0716**	0.680	**0.501**
(std)	(0.0004)	(0.0005)	(0.0002)	(0.0002)	(0.002)	(0.008)
Zero-one	0.143	**0.114**	0.1836	**0.1789**	0.224	**0.214**
(std)	(0.003)	(0.002)	(0.0005)	(0.0008)	(0.002)	(0.002)
Hinge	0.839	**0.711**	**0.633**	0.695	0.967	**0.518**
(std)	(0.003)	(0.005)	(0.009)	(0.011)	(0.004)	(0.021)
1-norm	**0.2349**	0.6587	**0.488**	0.672	**0.250**	0.871
(std)	(0.001)	(0.007)	(0.008)	(0.005)	(0.003)	(0.056)
AUC	0.9801	**0.9877**	0.9776	**0.9794**	0.8291	**0.8321**
(std)	(0.0002)	(0.0002)	(0.0007)	(0.0001)	(0.0018)	(0.0003)

Table 5.2: Comparison of loss functions for DNN model.

	MNIST		Fashion MNIST		Adult	
	Uniform	ALMO	Uniform	ALMO	Uniform	ALMO
Cross-entropy	**0.044**	0.045	0.0700	**0.0614**	0.673	**0.460**
(std)	(0.003)	(0.005)	(0.0005)	(0.0009)	(0.001)	(0.003)
Zero-one	0.0158	**0.0153**	0.114	**0.110**	**0.149**	0.159
(std)	(0.0004)	(0.0002)	(0.003)	(0.004)	(0.002)	(0.002)
Hinge	**0.631**	0.752	0.661	**0.547**	0.914	**0.389**
(std)	(0.002)	(0.041)	(0.004)	(0.008)	(0.001)	(0.006)
1-norm	0.436	**0.313**	**0.435**	0.554	**0.153**	0.845
(std)	(0.002)	(0.038)	(0.006)	(0.0010)	(0.001)	(0.022)
AUC	0.9986	**0.9993**	0.9883	**0.9911**	0.8092	**0.8840**
(std)	(0.0002)	(0.0001)	(0.0002)	(0.0005)	(0.0004)	(0.0031)

5.6 CONCLUSION

We have introduced a new framework (ALMO) for multi-objective learning that is robust against any mixture distribution of the base objectives, hence avoiding the subjective step of selecting mixture coefficients for multi-loss training. We have given a detailed theoretical analysis and a learning algorithm motivated by the theory. The algorithm is based on stochastic gradient descent, therefore applicable to a wide range of large-scale domains and can be easily implemented in popular computational frameworks. The experiments show that the ALMO framework builds more robust

models for a variety of objectives in different machine learning problems. The agnostic framework introduced here is directly applicable and beneficial not only for standard supervised learning settings, but also to other scenarios where a learner is trying to combine multiple objectives, such as transfer learning and domain adaptation.

BROADER IMPACT

This paper presents a novel approach for learning with multiple losses. The algorithm is robust in the sense that it optimizes for the most adversarial mixture of the included losses. It furthermore demonstrates good performance on losses not included in the optimization. This is an important problem from a fairness perspective where multiple losses are often at play and different interest groups may disagree on what loss to optimize for. An illustrative example is the analysis of the COMPAS tool for predicting recidivism by [Angwin et al. 2019] demonstrating how different interest groups would have wanted to optimize for different losses. Not all losses can be optimized for simultaneously, as proven by [Kleinberg et al. 2017], but to the extent that this is possible, our algorithm provides a step in the direction of guarding against the most unfavorable condition.

6 | Efficient Gradient Computation for Structured Output Learning with Rational and Tropical Losses

Many structured prediction problems admit a natural loss function for evaluation such as the edit-distance or n-gram loss. However, existing learning algorithms are typically designed to optimize alternative objectives such as the cross-entropy. This is because a naïve implementation of the natural loss functions often results in intractable gradient computations. In this paper, we design efficient gradient computation algorithms for two broad families of structured prediction loss functions: rational and tropical losses. These families include as special cases the n-gram loss, the edit-distance loss, and many other loss functions commonly used in natural language processing and computational biology tasks that are based on sequence similarity measures. Our algorithms make use of weighted automata and graph operations over appropriate semirings to design efficient solutions. They facilitate efficient gradient computation and hence enable one to train learning models such as neural networks with complex structured losses.

6.1 BACKGROUND

Many important machine learning tasks are instances of structured prediction problems. These are learning problems where the output labels admit some structure that is important to take into account both for statistical and computational reasons. Structured prediction problems include most natural language processing tasks, such as pronunciation modeling, part-of-speech tagging, context-free parsing, dependency parsing, machine translation, speech recognition, where the output labels are sequences of phonemes, part-of-speech tags, words, parse trees, or acyclic graphs, as well as other sequence modeling tasks in computational biology. They also include a variety of problems in computer vision such as image segmentation, feature detection, object recognition, motion estimation, computational photography and many others.

Several algorithms have been designed in the past for structured prediction tasks, including Conditional Random Fields (CRFs) [Lafferty et al. 2001; Gimpel and Smith 2010], StructSVMs [Tsochantaridis et al. 2005], Maximum-Margin Markov Networks (M3N) [Taskar et al. 2003], kernel-regression-based algorithms [Cortes et al. 2007], and search-based methods [Daumé III et al. 2009; Doppa et al. 2014; Lam et al. 2015; Chang et al. 2015; Ross et al. 2011]. More recently, deep learning techniques have been designed for many structured prediction tasks, including part-of-speech tagging [Jurafsky and Martin 2009; Vinyals et al. 2015a], named-entity recognition [Nadeau and Sekine 2007], machine translation [Zhang et al. 2008; Wu et al. 2016], image segmentation [Lucchi et al. 2013], and image annotation [Vinyals et al. 2015b].

Many of these algorithms have been successfully used with specific loss functions such as the Hamming loss. Their use has been also extended to multivariate performance measures such as Precision/Recall or F_1-score [Joachims 2005], which depend on predictions on all training points. However, the natural loss function relevant to a structured prediction task, which may be the n-

gram loss, the edit-distance loss, or some sequence similarity-based loss, is otherwise often ignored. Instead, an alternative measure such as the cross-entropy is used. This is typically due to computational efficiency reasons: a key subroutine within the main optimization such as one requiring to determine the most violating constraint may be computationally intractable, the gradient may not admit a closed-form or may seem difficult to compute, as it may involve sums over a number of terms exponential in the size of the input alphabet, with each term in itself being a large non-trivial computational task.

Several techniques have been suggested in the past to address this issue. They include Minimum Risk Training (MRT) [Och 2003; Shen et al. 2016], which seeks to optimize the natural objective directly but relies on sampling or focusing on only the top-n structured outputs to make the problem computationally tractable. REINFORCE-based methods [Ranzato et al. 2015; Wu et al. 2016] also seek to optimize the natural loss function by defining an unbiased stochastic estimate of the objective, thereby making the problem computationally tractable. While these publications have demonstrated that training directly with the natural loss function yields better results than using a naïve loss function, their solutions naturally suffer from issues such as high variance in the gradient estimate, in the case of sampling, or bias in the case of top-n. Moreover, REINFORCE methods often have to feed the ground-truth at training time, which is inconsistent with the underlying theory.

Another technique has consisted of designing computationally more tractable surrogate loss functions closer to the natural loss function [Ranjbar et al. 2013; Eban et al. 2017]. These publications also report improved performance using an objective closer to the natural loss, while admitting the inherent issue of not optimizing the desired metric. [McAllester et al. 2010] propose a perceptron-like update in the special case of linear models in structured prediction problems, which avoids the use of surrogate losses. However, while they show that direct loss minimization

admits some asymptotic statistical benefits, each update in their work requires solving an argmax problem for which the authors do not give an algorithm and that is known to be computationally hard in general, particularly for non-additive losses.

This paper is strongly motivated by much of this previous work, which reports empirical benefits for using the natural loss associated to the task. We present efficient gradient computation algorithms for two broad families of structured prediction loss functions: rational and tropical losses. These families include as special cases the n-gram loss, the edit-distance loss, and many other loss functions commonly used in natural language processing and computational biology tasks that are based on sequence similarity measures. Our algorithms make use of weighted automata and graph operations over appropriate semirings to design efficient solutions that circumvent the naïve computation of exponentially sized sums in gradient formula.

Our algorithms enable one to train learning models such as neural networks with complex structured losses. When combined with the recent developments in automatic differentiation, e.g. CNTK [Seide and Agarwal 2016], MXNet [Chen et al. 2015], PyTorch [Paszke et al. 2017], and TensorFlow [Abadi et al. 2016], they can be used to train structured prediction models such as neural networks with the natural loss of the task. In particular, the use of our techniques for the top layer of neural network models can further accelerate progress in end-to-end training [Amodei et al. 2016; Graves and Jaitly 2014; Wu et al. 2016].

For problems with limited data, e.g. uncommon languages or some biological problems, our work overcomes the computational bottleneck, uses the exact loss function, and renders the amount of data available the next hurdle for improved performance. For extremely large-scale problems with more data than can be processed, we further present an approximate truncated shortest-path algorithm that can be used for fast approximate gradient computations of the edit-distance.

The rest of the paper is organized as follows. In Section 6.2, we briefly describe structured

prediction problems and algorithms, discuss their learning objectives, and point out the challenge of gradient computation. Section 6.3 defines several weighted automata and transducer operations that we use to design efficient algorithms for gradient-based learning. In Sections 6.4 and 6.5, we give general algorithms for computing the gradient of rational and tropical loss functions, respectively. In Section 6.6, we report the results of experiments verifying the improvement due to using our efficient methods compared to a naïve implementation. Further details regarding weighted automata and transducer operations and training recurrent neural network training with the structured objective are presented in Appendix D.1 and Appendix D.2.

6.2 GRADIENT COMPUTATION IN STRUCTURED PREDICTION

In this section, we introduce the structured prediction learning problem. We start by defining the learning scenario, including the relevant loss functions and features. We then move on to discussing the hypothesis sets and forms of the objection function that are used by many structured prediction algorithms, which leads us to describe the problem of computing their gradients.

6.2.1 STRUCTURED PREDICTION LEARNING SCENARIO

We consider the supervised learning setting, in which the learner receives a labeled sample $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ drawn i.i.d. from some unknown distribution over $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$ denotes the input space and $\mathcal{Y}$ the output space. In structured prediction, we assume that elements of the output space $\mathcal{Y}$ can be decomposed into possibly overlapping substructures $y = (y^1, \ldots, y^l)$. We further assume that the loss function $\mathsf{L} \colon \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$ can similarly be decomposed along these substructures. Some key examples of loss functions that are relevant to our work are the Hamming loss, the n-gram loss and the edit-distance loss.

The *Hamming loss* is defined for all $y = (y^1, \ldots, y^l)$ and $y' = (y'^1, \ldots, y'^l)$ by $\mathsf{L}(y, y') = \frac{1}{l}\sum_{k=1}^{l} 1_{y^k \neq y'^k}$, with $y^k, y'^k \in \mathcal{Y}_k$. The *edit-distance loss* is commonly used in natural language processing (NLP) applications where $\mathcal{Y}$ is a set of sequences defined over a finite alphabet, and the loss function between two sequences y and y' is defined as the minimum cost of a sequence of edit operations, typically insertions, deletions, and substitutions, that transform y into y'. The *n-gram loss* is defined as the negative inner product (or its logarithm) of the vectors of n-gram counts of two sequences. This can serve as an approximation to the BLEU score, which is commonly used in machine translation.

We assume that the learner has access to a feature mapping $\mathbf{\Psi} \colon \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^N$. This mapping can be either a vector of manually designed features, as in the application of the CRF algorithm, or the differentiable output of the penultimate layer of an artificial neural network. In practice, feature mappings that correspond to the inherent structure of the input space $\mathcal{X}$ combined with the structure of $\mathcal{Y}$ can be exploited to derive effective and efficient algorithms. As mentioned previously, a common case in structured prediction is when $\mathcal{Y}$ is a set of sequences of length l over a finite alphabet Δ. This is the setting that we will consider, as other structured prediction problems can often be treated similarly.

We further assume that $\mathbf{\Psi}$ admits a *Markovian property of order* q, that is, for any $(x, y) \in \mathcal{X} \times \mathcal{Y}$, $\mathbf{\Psi}(x, y)$ can be decomposed as $\mathbf{\Psi}(x, y) = \sum_{s=1}^{l} \boldsymbol{\psi}(x, y^{s-q+1:s}, s)$, for some position-dependent feature vector function $\boldsymbol{\psi}$ defined over $\mathcal{X} \times \Delta^q \times [l]$, where the shorthand $y^{s:s'} = (y^s, \ldots, y^{s'})$ stands for the substring of y starting at index s and ending at s'. For convenience, for $s \leq 0$, we define y^s to be the empty string ε. This Markovian assumption is commonly adopted in structured prediction problems such as NLP [Manning and Schütze 1999]. In particular, it holds for feature mappings that are frequently used in conjunction with the CRF, as well as outputs of a recurrent neural network, reset at the begining of each new input (see Appendix D.2).

6.2.2 OBJECTIVE FUNCTION AND GRADIENT COMPUTATION

The hypothesis set we consider is that of linear functions $h\colon (x, y) \mapsto \mathbf{w} \cdot \mathbf{\Psi}(x, y)$ based on the feature mapping $\mathbf{\Psi}$. The empirical loss $\widehat{R}_S(h) = \frac{1}{m}\sum_{i=1}^{m} \mathsf{L}(h(x_i), y_i)$ associated to a hypothesis h is often not differentiable in structured prediction since the loss function admits discrete values. Taking the expectation over the distribution induced by the log-linear model, as in [Gimpel and Smith 2010][Equation 5], does not help resolve this issue, since the method does not result in an upper bound on the empirical loss and does not admit favorable generalization guarantees. Instead, as in the familiar binary classification scenario, one can resort to upper-bounding the loss with a differentiable (convex) surrogate. For instance, by [Cortes et al. 2016][Lemma 4], $\widehat{R}_S(h)$ can be upper-bounded by the following objective function:

$$F(\mathbf{w}) = \frac{1}{m}\sum_{i=1}^{m} \log\left[\sum_{y\in\mathcal{Y}} e^{\mathsf{L}(y,y_i)-\mathbf{w}\cdot(\mathbf{\Psi}(x_i,y_i)-\mathbf{\Psi}(x_i,y))}\right], \tag{6.1}$$

which, modulo a regularization term, coincides with the objective function of CRF. Note that this expression has also been presented as the softmax margin [Gimpel and Smith 2010] and the reward-augmented maximum likelihood [Norouzi et al. 2016]. Both of these references demonstrate strong empirical evidence for this choice of objective function (in addition to the theoretical results presented in [Cortes et al. 2016]).

Our focus in this work is on an efficient computation of the gradient of this objective function. Since the computation of the subgradient of the regularization term often does not pose any issues, we will only consider the unregularized part of the objective. For any $\mathbf{w}$ and $i \in [m]$, let $F_i(\mathbf{w})$ denote the contribution of the i-th training point to the objective function F. A standard gradient descent-based method would sum up all or a subset (mini-batch) of the gradients $\nabla F_i(\mathbf{w})$. As illustrated in [Cortes et al. 2016][Lemma 15], the gradient $\nabla F_i(\mathbf{w})$ can be expressed as follows at

any $\mathbf{w}$:

$$\nabla F_i(\mathbf{w}) = \frac{1}{m} \sum_{s=1}^{l} \sum_{\mathbf{z} \in \Delta^q} \mathsf{Q}_{\mathbf{w}}(\mathbf{z}, s) \boldsymbol{\psi}(x_i, \mathbf{z}, s) - \frac{\boldsymbol{\Psi}(x_i, y_i)}{m},$$

where, for all $\mathbf{z} \in \Delta^q$ and $s \in [l]$, $\mathsf{Q}_{\mathbf{w}}(\mathbf{z}, s)$ is defined by

$$\mathsf{Q}_{\mathbf{w}}(\mathbf{z}, s) = \sum_{y \colon y^{s-q+1:s} = \mathbf{z}} \frac{e^{\mathsf{L}(y, y_i) + \mathbf{w} \cdot \boldsymbol{\Psi}(x_i, y)}}{Z_{\mathbf{w}}} \quad \text{and} \quad Z_{\mathbf{w}} = \sum_{y \in \mathcal{Y}} e^{\mathsf{L}(y, y_i) + \mathbf{w} \cdot \boldsymbol{\Psi}(x_i, y)}.$$

The bottleneck in the gradient computation is the evaluation of $\mathsf{Q}_{\mathbf{w}}(\mathbf{z}, s)$, for all $\mathbf{z} \in \Delta^q$ and $s \in [l]$. There are $l|\Delta|^q$ such terms and each term $\mathsf{Q}_{\mathbf{w}}(\mathbf{z}, s)$ is defined by a sum over the $|\Delta|^{l-q}$ sequences y of length l with a fixed substring $\mathbf{z}$ of length q. A straightforward computation of these terms following their definition would therefore be computationally expensive. To avoid that computational cost, many existing learning algorithms for structured prediction, including most of those mentioned in the introduction, resort to further approximations and omit the loss L from the definition of $\mathsf{Q}_{\mathbf{w}}(\mathbf{z}, s)$. Combining that with the Markovian structure of $\boldsymbol{\Psi}$ can then lead to efficient gradient computations. Of course, the caveat of this approach is that it ignores the key component of the learning problem, namely the loss function.

In what follows, we will present efficient algorithms for the exact computation of the terms $\mathsf{Q}_{\mathbf{w}}(\mathbf{z}, s)$, with their full definition, including the loss function. This leads to an efficient computation of the gradients ∇F_i, which can be used as input to back-propagation algorithms that would enable us to train neural network models with structured prediction losses.

The gradient computation methods we present apply to the Hamming loss, n-gram loss, and edit-distance loss, and more generally to two broad families of losses that can be represented by weighted finite-state transducers (WFSTs). This covers many losses based on sequence similarity measures that are used in NLP and computational biology applications [Cortes et al. 2004; Schölkopf et al. 2004].

We briefly describe the WFST operations relevant to our solutions in the following section and provide an example of how the edit-distance loss can be represented with a WFST in Section 6.5.

6.3 WEIGHTED AUTOMATA AND TRANSDUCERS

Weighted finite automata (WFA) and weighted finite-state transducers (WFST) are fundamental concepts and representations widely used in computer science [Mohri 2009]. We will use WFAs and WFSTs to devise algorithms that efficiently compute gradients of structured prediction objectives. This section introduces some standard concepts and notation for WFAs and WFSTs. We provide additional details in Appendix D.1. For a more comprehensive treatment of these topics, we refer the reader to [Mohri 2009].

Definition. A *weighted finite-state transducer* $\mathcal{T}$ over a semiring $(\mathbb{S}, \oplus, \otimes, \overline{0}, \overline{1})$ is an 8-tuple $(\Sigma, \Delta, Q, I, F, E, \lambda, \rho)$ where Σ is a finite input alphabet, Δ is a finite output alphabet, Q is a finite set of states, $I \subseteq Q$ is the set of initial states, $F \subseteq Q$ is the set of final states, E is a finite multiset of transitions, which are elements of $Q \times (\Sigma \cup \{\epsilon\}) \times (\Delta \cup \{\epsilon\}) \times \mathbb{S} \times Q$, $\lambda \colon I \to \mathbb{S}$ is an initial weight function, and $\rho \colon F \to \mathbb{S}$ is a final weight function. A *weighted finite automaton* is a weighted finite-state transducer where the input and output labels are the same. See Figures 6.1 and 6.3 for some examples.

For many operations to be well defined, the weights of a WFST must belong to a semiring $(\mathbb{S}, \oplus, \otimes, \overline{0}, \overline{1})$. We provide a formal definition of a semiring in Appendix D.1. In this work, we consider two semirings: the probability semiring $(\mathbb{R}_+ \cup \{+\infty\}, +, \times, 0, 1)$ and the tropical semiring $(\mathbb{R} \cup \{-\infty, +\infty\}, \min, +, +\infty, 0)$. The $\otimes$-operation is used to compute the weight of a path by $\otimes$-multiplying the weights of the transitions along that path. The $\oplus$-operation is used to compute the weight of a pair of input and output strings (x, y) by $\oplus$-summing the weights of the

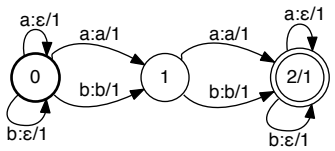


Figure 6.1: Bigram transducer $\mathcal{T}_{\text{bigram}}$ over the semiring $(\mathbb{R}_+ \cup \{+\infty\}, +, \times, 0, 1)$ for the alphabet $\Delta = \{a, b\}$. The weight of each transition (or that of a final state) is indicated after the slash separator. For example, for any string y and bigram $\mathbf{u}$, $\mathcal{T}_{\text{bigram}}(y, \mathbf{u})$ is equal to the number of occurrences of $\mathbf{u}$ in y [Cortes et al. 2015a].

paths labeled with (x, y). We denote this weight by $\mathcal{T}(x, y)$.

As shown in Sections 6.4 and 6.5, in many useful cases, we can reduce the computation of the loss function $\mathsf{L}(y, y')$ between two strings y and y', along with the gradient of the corresponding objective described in (6.1), to that of the $\oplus$-sum of the weights of all paths labeled by $y{:}y'$ in a suitably defined transducer over either the probability or tropical semiring. We will use the following standard WFST operations to construct these transducers: inverse ($\mathcal{T}^{-1}$), projection ($\Pi(\mathcal{T})$), composition ($\mathcal{T}_1 \circ \mathcal{T}_2$), and determinization ($\mathrm{Det}(\mathcal{A})$). The definitions of these operations are given in Appendix D.1.

6.4 An efficient algorithm for the gradient computation of rational losses

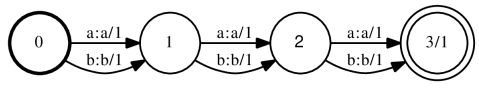
As discussed in Section 6.2, computing $\mathsf{Q}_{\mathbf{w}}(\mathbf{z}, s)$ is the main bottleneck in the gradient computation. In this section, we give an efficient algorithm for computing $\mathsf{Q}_{\mathbf{w}}(\mathbf{z}, s)$ that works for an arbitrary *rational loss*, which includes as a special case the n-gram loss and other sequence similarity-based losses. We first present the definition of a rational loss and show how the n-gram loss can be encoded as a specific rational loss. Then, we present our gradient computation algorithm.

Let $(\mathbb{R}_+ \cup \{+\infty\}, +, \times, 0, 1)$ be the *probability semiring* and let $\mathcal{U}$ be a WFST over the probability semiring admitting Δ as both the input and output alphabet. Then, following [Cortes et al.

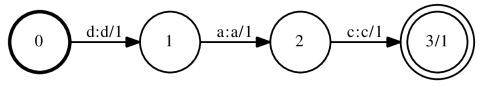
GRAD-RATIONAL$(x_i, y_i, \mathbf{w})$

$\overline{\mathcal{Y}} \leftarrow$ WFA accepting any $y \in \Delta^l$.
$\mathcal{Y}_i \leftarrow$ WFA accepting y_i.
$\mathcal{M} \leftarrow \Pi_1(\overline{\mathcal{Y}} \circ \mathcal{U} \circ \mathcal{Y}_i)$
$\mathcal{M} \leftarrow \mathrm{Det}(\mathcal{M})$
$\mathcal{B} \leftarrow$ INVERSEWEIGHTS$(\mathcal{M})$
$\mathcal{C} \leftarrow \mathcal{A} \circ \mathcal{B}$
$\alpha \leftarrow$ DISTFROMINITIAL$(\mathcal{C}, (+, \times))$
$\beta \leftarrow$ DISTTOFINAL$(\mathcal{C}, (+, \times))$
$Z_{\mathbf{w}} \leftarrow \beta(I_{\mathcal{C}}) \triangleright$ $I_{\mathcal{C}}$ initial state of $\mathcal{C}$
for $(\mathbf{z}, s) \in \Delta^q \times [l]$ **do**
$\mathsf{Q}_{\mathbf{w}}(\mathbf{z}, s) \leftarrow \sum_{e \in E_{\mathbf{z},s}} \alpha(\underline{e}) \times \omega(e) \times \beta(\overline{e})$
$\mathsf{Q}_{\mathbf{w}}(\mathbf{z}, s) \leftarrow \mathsf{Q}_{\mathbf{w}}(\mathbf{z}, s)/Z_{\mathbf{w}}$

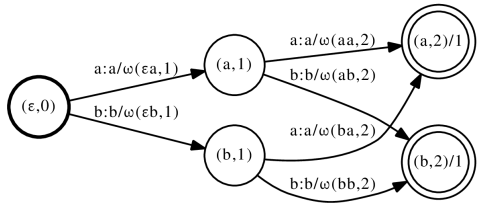
(a)



(b)



(c)



(d)

Figure 6.2: (a) Efficient computation of the key terms of the structured gradient for the rational loss. For each transition $e \in E_{\mathbf{z},s}$, we denote its origin by $\underline{e}$, destination by $\overline{e}$ and weight by $\omega(e)$. (b) Illustration of the WFA $\overline{\mathcal{Y}}$ for $\Delta = \{a, b\}$ and $l = 3$. (c) Illustration of the WFA $\mathcal{Y}_i$ representing string dac. (d) Illustration of WFA $\mathcal{A}$ for $q = 2$, alphabet $\Delta = (a, b)$ and string length $l = 2$. For example, the transition from state $(a, 1)$ to state $(b, 2)$ has the label b and weight $\omega(ab, 2) = e^{\mathbf{w} \cdot \boldsymbol{\psi}(x_i, ab, 2)}$.

2015a], the *rational loss* associated to $\mathcal{U}$ is the function $L_{\mathcal{U}} \colon \Delta^* \times \Delta^* \to \mathbb{R} \cup \{-\infty, +\infty\}$ defined for all $y, y' \in \Delta^*$ by $L_{\mathcal{U}}(y, y') = -\log\big(\mathcal{U}(y, y')\big)$. As an example, the n-gram loss of y and y' is the negative logarithm of the inner product of the vectors of n-gram counts of y and y'. The WFST $\mathcal{U}_{n\text{-gram}}$ of an n-gram loss is obtained by composing a weighted transducer $\mathcal{T}_{n\text{-gram}}$ giving the n-gram counts with its inverse $\mathcal{T}^{-1}_{n\text{-gram}}$, that is the transducer derived from $\mathcal{T}_{n\text{-gram}}$ by swapping input and output labels for each transition. As an example, Figure 6.1 shows the WFST $\mathcal{T}_{\text{bigram}}$ for bigrams.

To compute $\mathsf{Q}_{\mathbf{w}}(\mathbf{z}, s)$ for a rational loss, recall that

$$\mathsf{Q}_{\mathbf{w}}(\mathbf{z}, s) \propto \sum_{y \colon y^{s-q+1:s} = \mathbf{z}} e^{L_{\mathcal{U}}(y, y_i) + \mathbf{w} \cdot \boldsymbol{\Psi}(x_i, y)}.$$

Thus, we will design two WFAs, $\mathcal{A}$ and $\mathcal{B}$, such that $\mathcal{A}(y) = e^{\mathbf{w}\cdot\mathbf{\Psi}(x_i,y)}$, $\mathcal{B}(y) = e^{L_{\mathcal{U}}(y,y_i)}$, and their composition $\mathcal{C}(y) = (\mathcal{A} \circ \mathcal{B})(y) = e^{L_{\mathcal{U}}(y,y_i)+\mathbf{w}\cdot\mathbf{\Psi}(x_i,y)}$. To compute $\mathsf{Q}_{\mathbf{w}}$ from $\mathcal{C}$, we will need to sum up the weights of all paths labeled with some substring $\mathbf{z}$, which we will achieve by treating this as a flow computation problem.

The pseudocode of our algorithm for computing the key terms $\mathsf{Q}_{\mathbf{w}}(\mathbf{z}, s)$ for a rational loss is given in Figure 6.2(a).

DESIGN OF $\mathcal{A}$. We want to design a determnistic WFA $\mathcal{A}$ such that

$$\mathcal{A}(y) = e^{\mathbf{w}\cdot\mathbf{\Psi}(x_i,y)} = \prod_{t=1}^{l} e^{\mathbf{w}\cdot\boldsymbol{\psi}(x_i,y^{t-q+1:t},t)}.$$

To accomplish this task, let $\mathcal{A}$ be a WFA with the following set of states $Q_{\mathcal{A}} = \Big\{(y^{t-q+1:t}, t)\colon y \in \Delta^l, t = 0, \ldots, l\Big\}$, with $I_{\mathcal{A}} = (\varepsilon, 0)$ its single initial state, $F_{\mathcal{A}} = \{(y^{l-q+1:l}, l)\colon y \in \Delta^l\}$ its set of final states, and with a transition from state $(y^{t-q+1:t-1}, t-1)$ to state $(y^{t-q+2:t-1}\, b, t)$ with label b and weight $\omega(y^{t-q+1:t-1}\, b, t) = e^{\mathbf{w}\cdot\boldsymbol{\psi}(x_i,y^{t-q+1:t-1}b,t)}$, that is, the following set of transitions:

$$E_{\mathcal{A}} = \Big\{\Big((y^{t-q+1:t-1}, t-1), b, \omega(y^{t-q+1:t-1}\, b, t), (y^{t-q+2:t-1}\, b, t)\Big)\colon y \in \Delta^l, b \in \Delta, t \in [l]\Big\}.$$

Figure 6.2(d) illustrates this construction in the case $q = 2$. Note that the WFA $\mathcal{A}$ is deterministic by construction. Since the weight of a path in $\mathcal{A}$ is obtained by multiplying the transition weights along the path, $\mathcal{A}(y)$ computes the desired quantity.

DESIGN OF $\mathcal{B}$. We now design a deterministic WFA $\mathcal{B}$ which associates to each sequence $y \in \Delta^l$ the exponential of the loss $e^{L_{\mathcal{U}}(y,y_i)} = 1/\mathcal{U}(y, y_i)$. Let $\overline{\mathcal{Y}}$ denote a WFA over the probability semiring accepting the set of all strings of length l with weight one and let $\mathcal{Y}_i$ denote the WFA

accepting only y_i with weight one. Figures 6.2(b) and 6.2(c) illustrate the constructions of $\overline{\mathcal{Y}}$ and $\mathcal{Y}_i$ in some simple cases.[1] We first use the composition operation for weighted automata and transducers. Then, we use the projection operation on the input, which we denote by Π_1, to compute the following WFA: $\mathcal{M} = \Pi_1(\overline{\mathcal{Y}} \circ \mathcal{U} \circ \mathcal{Y}_i)$. Recalling that $\mathcal{Y}(y) = \mathcal{Y}_i(y_i) = 1$ by construction and applying the definition of WFST composition, we observe that for any $y \in \Delta^l$

$$\mathcal{M}(y) = (\overline{\mathcal{Y}} \circ \mathcal{U} \circ \mathcal{Y}_i)(y, y_i) = \sum_{\mathbf{z}=y, \mathbf{z}'=y_i} \overline{\mathcal{Y}}(\mathbf{z})\mathcal{U}(\mathbf{z}, \mathbf{z}')\mathcal{Y}_i(\mathbf{z}') = \overline{\mathcal{Y}}(y)\mathcal{U}(y, y_i)\mathcal{Y}_i(y_i) = \mathcal{U}(y, y_i). \quad (6.2)$$

Next, we can apply weighted determinization [Mohri 1997] to compute a deterministic WFA equivalent to $\mathcal{M}$, denoted by $\mathrm{Det}(\mathcal{M})$. By [Cortes et al. 2015a][Theorem 3], $\mathrm{Det}(\mathcal{M})$ can be computed in polynomial time. Since $\mathrm{Det}(\mathcal{M})$ is deterministic and by construction accepts precisely the set of strings $y \in \Delta^l$, it admits a unique accepting path labeled with y whose weight is $\mathrm{Det}(\mathcal{M})(y) = \mathcal{M}(y) = \mathcal{U}(y, y_i)$. The weight of that accepting path is obtained by multiplying the weights of its transitions and that of the final state. Let $\mathcal{B}$ be the WFA derived from $\mathrm{Det}(\mathcal{M})$ by replacing each transition weight or final weight u by its inverse $\frac{1}{u}$. Then, by construction, for any $y \in \Delta^l$, we have $\mathcal{B}(y) = \frac{1}{\mathcal{U}(y, y_i)}$.

COMBINING $\mathcal{A}$ AND $\mathcal{B}$. Now consider the WFA $\mathcal{C} = \mathcal{A} \circ \mathcal{B}$, the composition of $\mathcal{A}$ and $\mathcal{B}$. $\mathcal{C}$ is deterministic since both $\mathcal{A}$ and $\mathcal{B}$ are deterministic. Moreover, $\mathcal{C}$ can be computed in time $O(|\mathcal{A}||\mathcal{B}|)$. By definition, for all $y \in \Delta^l$,

$$\mathcal{C}(y) = \mathcal{A}(y) \times \mathcal{B}(y) = \prod_{t=1}^{l} e^{\mathbf{w} \cdot \boldsymbol{\psi}(x_i, y^{t-q+1:t}, t)} \times \frac{1}{\mathcal{U}(y, y_i)} = e^{\mathsf{L}(y, y_i)} \prod_{t=1}^{l} e^{\mathbf{w} \cdot \boldsymbol{\psi}(x_i, y^{t-q+1:t}, t)}. \quad (6.3)$$

[1] Note that we do not need to explicitly construct $\overline{\mathcal{Y}}$, which could be costly when the alphabet size Δ is large. Instead, we can create its transitions on-the-fly as demanded by the composition operation. Thus, for the rational kernels commonly used, at most the transitions labeled with the alphabet symbols appearing in $\mathcal{Y}_i$ need to be created.

To see how $\mathcal{C}$ can be used to compute $\mathsf{Q}_w(\mathbf{z}, s)$, we note first that the states of $\mathcal{C}$ can be identified with pairs $(q_{\mathcal{A}}, q_{\mathcal{B}})$ where $q_{\mathcal{A}}$ is a state of $\mathcal{A}$, $q_{\mathcal{B}}$ is a state of $\mathcal{B}$, and the transitions are obtained by matching a transition in $\mathcal{A}$ with one in $\mathcal{B}$. Thus, for any $\mathbf{z} \in \Delta^q$ and $s \in [l]$, let $E_{\mathbf{z},s}$ be the set of transitions of $\mathcal{C}$ constructed by pairing the transition in $\mathcal{A}$ $((\mathbf{z}^{1:q-1}, s-1), z^q, \omega(\mathbf{z}, s), (\mathbf{z}^{2:q}, s))$ with a transition in $\mathcal{B}$:

$$E_{\mathbf{z},s} = \Big\{ \big((q_{\mathcal{A}}, q_{\mathcal{B}}), z^q, \omega, (q'_{\mathcal{A}}, q'_{\mathcal{B}})\big) \in E_{\mathcal{C}} \colon q_{\mathcal{A}} = (\mathbf{z}^{1:q-1}, s-1) \Big\}. \tag{6.4}$$

Note that, since $\mathcal{C}$ is deterministic, there can be only one transition leaving a state labeled with z^q. Thus, to define $E_{\mathbf{z},s}$, we only needed to specify the origin state of the transitions.

For each transition $e \in E_{\mathbf{z},s}$, we denote its origin by $\underline{e}$, destination by $\overline{e}$ and weight by $\omega(e)$. Then, $\mathsf{Q}_{\mathbf{w}}(\mathbf{z}, s)$ can be computed as $\sum_{e \in E_{\mathbf{z},s}} \alpha(\underline{e}) \times \omega(e) \times \beta(\overline{e})$, where $\alpha(\underline{e})$ is the sum of the weights of all paths from an initial state of $\mathcal{C}$ to $\underline{e}$, and $\beta(\overline{e})$ is the sum of the weights of all paths from $\overline{e}$ to a final state of $\mathcal{C}$. Since $\mathcal{C}$ is acyclic, α and β can be computed for all states in linear time in the size of $\mathcal{C}$ using a single-source shortest-distance algorithm over the $(+, \times)$ semiring [Mohri 2002] or the so-called forward-backward algorithm. We denote these subroutines by DistFromInitial and DistToFinal respectively in the pseudocode. Since $\mathcal{C}$ admits $O(l|\Delta|^q)$ transitions, we can compute all of the quantities $\mathsf{Q}_{\mathbf{w}}(\mathbf{z}, s)$, $s \in [l]$ and $z \in \Delta^q$ and $Z'_{\mathbf{w}}$, in time $O(l|\Delta|^q)$.

Note that a natural alternative to the weighted transducer methods presented in this work is to consider junction tree type methods for graphical methods. However, weighted transducer techniques typically result in more "compact" representations than graphical model methods, and the computational cost of the former can even be exponentially faster than the best one could achieve using the latter [Poon and Domingos 2011].

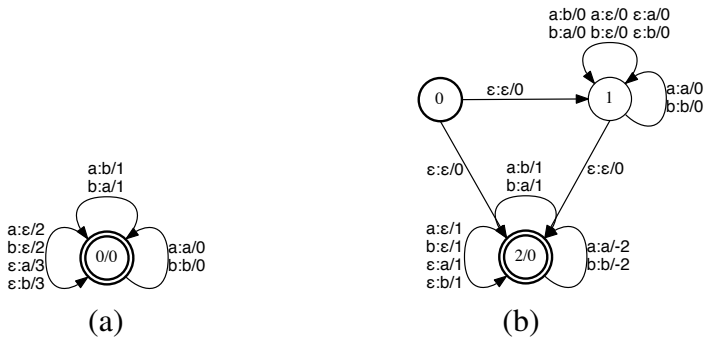


Figure 6.3: (a) Edit-distance transducer $\mathcal{U}_{\text{edit}}$ over the tropical semiring, in the case where the substitution cost is 1, the deletion cost 2, the insertion cost 3, and the alphabet $\Delta = \{a, b\}$. (b) Smith-Waterman transducer $\mathcal{U}_{\text{Smith-Waterman}}$ over the tropical semiring, in the case where the substitution, deletion and insertion costs are 1, and where the matching cost is -2, for the alphabet $\Delta = \{a, b\}$.

6.5 AN EFFICIENT ALGORITHM FOR THE GRADIENT COMPUTATION OF TROPICAL LOSSES

Following the treatment in [Cortes et al. 2015a], the *tropical loss* associated to a weighted transducer $\mathcal{U}$ over the tropical semiring is defined as the function $L_{\mathcal{U}} \colon \Delta^* \times \Delta^* \to \mathbb{R}$ coinciding with $\mathcal{U}$; thus, for all $y, y' \in \Delta^*$, $L_{\mathcal{U}}(y, y') = \mathcal{U}(y, y')$.

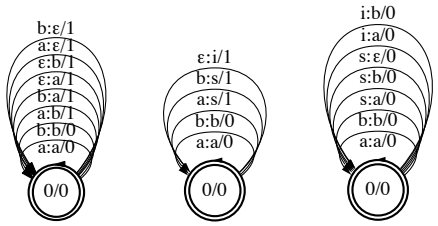
For examples of weighted transducers over the tropical semiring, see Figures 6.3(a) and (b).

Our algorithm for computing $\mathsf{Q}_{\mathbf{w}}(\mathbf{z}, s)$ for a tropical loss, illustrated in Figure 6.4(a), is similar to our algorithm for a rational loss, with the primary difference being that we exponentiate weights instead of invert them in the WFA $\mathcal{B}$. Specifically, we design $\mathcal{A}$ just as in Section 6.4, and we design a deterministic WFA $\mathcal{B}$ by first designing $\mathrm{Det}(\mathcal{M})$ as in Section 6.4 and then deriving $\mathcal{B}$ from $\mathrm{Det}(\mathcal{M})$ by replacing each transition weight or final weight u in $\mathrm{Det}(\mathcal{M})$ by e^u. Then by construction, for any $y \in \Delta^l$, $\mathcal{B}(y) = e^{\mathcal{U}(y, y_i)}$. Moreover, composition of $\mathcal{A}$ with $\mathcal{B}$ yields a WFA

GRAD-TROPICAL$(x_i, y_i, \mathbf{w})$

$\overline{\mathcal{Y}} \leftarrow$ WFA accepting any $y \in \Delta^l$.
$\mathcal{Y}_i \leftarrow$ WFA accepting y_i.
$\mathcal{M} \leftarrow \Pi_1(\overline{\mathcal{Y}} \circ \mathcal{U} \circ \mathcal{Y}_i)$
$\mathcal{M} \leftarrow \mathrm{Det}(\mathcal{M})$
$\mathcal{B} \leftarrow$ EXPONENTIATEWEIGHTS$(\mathcal{M})$
$\mathcal{C} \leftarrow \mathcal{A} \circ \mathcal{B}$
$\alpha \leftarrow$ DISTFROMINITIAL$(\mathcal{C}, (+, \times))$
$\beta \leftarrow$ DISTTOFINAL$(\mathcal{C}, (+, \times))$
$Z_{\mathbf{w}} \leftarrow \beta(I_{\mathcal{C}})$ ▷ $I_{\mathcal{C}}$ initial state of $\mathcal{C}$
for $(\mathbf{z}, s) \in \Delta^q \times [l]$ **do**
$\mathsf{Q}_{\mathbf{w}}(\mathbf{z}, s) \leftarrow \sum_{e \in E_{\mathbf{z},s}} \alpha(\underline{e}) \times \omega(e) \times \beta(\overline{e})$
$\mathsf{Q}_{\mathbf{w}}(\mathbf{z}, s) \leftarrow \mathsf{Q}_{\mathbf{w}}(\mathbf{z}, s)/Z_{\mathbf{w}}$

(a)



(b)

Figure 6.4: (a) Efficient computation of the key terms of the structured gradient for the tropical loss. (b) Factoring of the edit-distance transducer. The leftmost figure is the edit-distance weighted transducer $\mathcal{U}_{\text{edit}}$ over alphabet $\Sigma = \{a, b\}$, the center figure is a weighted transducer $\mathcal{T}_1$, and the rightmost figure is a weighted transducer $\mathcal{T}_2$ such that $\mathcal{U}_{\text{edit}} = \mathcal{T}_1 \circ \mathcal{T}_2$.

$\mathcal{C} = \mathcal{A} \circ \mathcal{B}$ such that for all $y \in \Delta^l$,

$$\mathcal{C}(y) = \mathcal{A}(y) \times \mathcal{B}(y) = \prod_{t=1}^{l} e^{\mathbf{w} \cdot \boldsymbol{\psi}(x_i, y^{t-q+1:t}, t)} \times e^{\mathcal{U}(y, y_i)} = e^{\mathsf{L}(y, y_i)} \prod_{t=1}^{l} e^{\mathbf{w} \cdot \boldsymbol{\psi}(x_i, y^{t-q+1:t}, t)}. \tag{6.5}$$

As an example, the general edit-distance of two sequences y and y' can, as already described, be computed using $\mathcal{U}_{\text{edit}}$ in time $O(|y||y'|)$ [Mohri 2003]. Note that for further computational optimization, $\mathcal{U}_{\text{edit}}$ and $\mathcal{U}_{\text{Smith-Waterman}}$ can be computed on-the-fly as demanded by the composition operation, thereby creating only transitions with alphabet symbols appearing in the strings compared.

In order to achieve optimal dependence on the size of the input alphabet, we can also apply *factoring* to the edit-distance transducer. Figure 6.4(b) illustrates factoring of the edit-distance

transducer over the alphabet $\Sigma = \{a, b\}$, where s is the substitution and deletion symbol and i is the insertion symbol. Note that both T_1 and T_2 are linear in the size of Σ, while $\mathcal{U}_{\text{edit}}$ is quadratic in $|\Sigma|$. Furthermore, using on-the-fly composition, for any $\mathcal{Y}_1$ and $\mathcal{Y}_2$, we can first compute $\mathcal{Y}_1 \circ T_1$ and $T_2 \circ \mathcal{Y}_2$ and then compose the result achieving time and space complexity in $O(|\mathcal{Y}_1||\mathcal{Y}_2|)$.

6.6 Experiments

6.6.1 Runtime Experiments

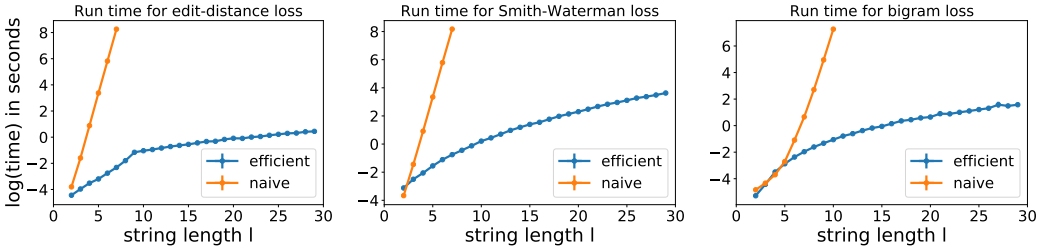


Figure 6.5: Runtime comparison of efficient versus naïve gradient computation methods for edit-distance (a), Smith-Waterman (b) and bigram (c) loss functions. The *naïve* line refers to the average runtime of Grad-Naïve, the *efficient* line refers to Grad-Tropical for edit-distance (a) and Smith-Waterman (b) and Grad-Rational for bigram (c) loss. Naïve computations are shown only up to string length $l = 8$.

In this section, we present experiments validating both the computational efficiency of our gradient computation methods as well as the learning benefits of training with natural loss functions. The experiments in this section should be treated as a proof of concept. We defer an extensive study of training structured prediction models on large-scale datasets for future work.

For the runtime comparison, we randomly generate an input and output data pair (x_i, y_i), both of a given fixed length, as well as a weight vector $\mathbf{w}$, and we compute $\nabla F_i(\mathbf{w})$ using both the naïve and the outlined efficient methods. As shown in Section 6.2, the computationally demanding part in the $\nabla F_i(\mathbf{w})$ calculation is evaluating $\mathsf{Q}_{\mathbf{w}}(\mathbf{z}, s)$ for all $s \in [l]$ and $\mathbf{z} \in \Delta^q$, while the other terms are generally not problematic to compute. We define a procedure Grad-Naïve (see Figure D.1 in

the appendix) and compare the average runtimes of Grad-Naïve with that of Grad-Efficient for both rational and tropical losses. The efficient algorithms suggested in this work improve upon the Grad-Naïve runtime by eliminating the explicit loop over $y \in \mathcal{Y}$ and using the weighted automata and transducer operations instead. All the weighted automata and transducer computations required for Grad-Rational and Grad-Tropical are implemented using `OpenFST` [Allauzen et al. 2007].

More specifically, we define an alphabet $|\Delta| = 10$ and features $\mathbf{\Psi}(x, y)$ as vectors of counts of all 100 possible bigrams. For each string length l from 2 to 30, we draw input pairs $(x_i, y_i) \in \Delta^l \times \Delta^l$ uniformly at random and $\mathbf{w} \in \mathbb{R}^{100}$ according to a standard normal distribution. The average runtimes over 125 random trials are presented in Figure 6.5 for three loss functions: the edit-distance, the Smith-Waterman distance and the bigram loss. The experiments demonstrate a number of crucial benefits of our efficient gradient computation framework. Note that the Grad-Naïve procedure runtime grows exponentially in l, while Grad-Tropical and Grad-Rational exhibit linear dependency on the length of the input strings. In fact, using the threshold pruning as part of determinization can allow one to compute approximate gradient for arbitrarily long input strings. The computational improvement is even more evident for rational losses, in which case the determinization of $\mathcal{M}$ can be achieved in polynomial time [Cortes et al. 2015a], thus pruning is not required.

6.6.2 Learning Experiments

We also provide preliminary learning experiments that illustrate the benefit of learning with a structured loss, compared to training with the cross-entropy loss for two major tasks: sequence alignment and machine translation.

The sequence alignment experiment replicates the artificial genome sequence data in [Joachims et al. 2006], where each example consists of native, homolog, and decoy sequences of length 50

and the task is to predict a sequence that is the closest to native in terms of the Smith-Waterman alignment score. The experiment confirms that a model trained with Smith-Waterman distance as the objective shows significantly higher average Smith-Waterman alignment score (and higher accuracy) on a test set compared to a model trained with cross-entropy objective. The cross-entropy model achieved a Smith-Waterman score of 42.73, while the augmented model achieved a score of 44.65 on a test set with a standard deviation of 0.35 averaged over 10 random folds.

6.7 Conclusion

We presented efficient algorithms for computing the gradients of structured prediction models with rational and tropical losses, reporting experimental results confirming both runtime improvement compared to naïve implementations and learning improvement compared to standard methods that settle for using easier-to-optimize losses. We also showed how our approach can be incorporated into the top layer of a neural network, so that it can be used to train end-to-end models in domains including speech recognition, machine translation, and natural language processing. For future work, we plan to run large-scale experiments with neural networks to further demonstrate the benefit of working directly with rational or tropical losses using our efficient computational methods.

7 | Generalization Bounds for Supervised Dimensionality Reduction

We introduce and study the learning scenario of *supervised dimensionality reduction*, which couples dimensionality reduction and a subsequent supervised learning step. We present new generalization bounds for this scenario based on a careful analysis of the empirical Rademacher complexity of the relevant hypothesis set. In particular, we show an upper bound on the Rademacher complexity that is in $\widetilde{O}(\sqrt{\Lambda_{(r)}/m})$, where m is the sample size and $\Lambda_{(r)}$ the upper bound on the Ky-Fan r-norm of the operator that defines the dimensionality reduction projection. We give both upper and lower bounds in terms of that Ky-Fan r-norm, which strongly justifies the definition of our hypothesis set. To the best of our knowledge, these are the first learning guarantees for the problem of supervised dimensionality reduction with a *learned* kernel-based mapping. Our analysis and learning guarantees further apply to several special cases, such as that of using a fixed kernel with supervised dimensionality reduction or that of unsupervised learning of a kernel for dimensionality reduction followed by a supervised learning algorithm.

7.1 INTRODUCTION

Dimensionality reduction techniques are common methods in machine learning used either to reduce the computational cost of working in higher-dimensional spaces, or to learn or approximate a manifold expected to be more favorable to a subsequent learning task such as classification or regression. They include classical techniques such as Principal Component Analysis (PCA) [Pearson 1901] and more recent techniques such as Isometric Feature Mapping [Tenenbaum et al. 2000] and Locally Linear Embedding [Roweis and Saul 2000]. More generally, the dimensionality reduction techniques just mentioned and most others have been shown to be specific instances of the kernel PCA (KPCA) algorithm [Ham et al. 2004], for different choices of a kernel. An even broader view of dimensionality reduction techniques is that they first map input points to the reproducing kernel Hilbert space (RKHS) of some positive semi-definite (PSD) kernel K, and next project vectors onto a low-dimensional space.

Standard dimensionality reduction techniques seek to determine a lower-dimensional space preserving some geometric properties of the input. However, it is not clear which of these properties would be most beneficial to the later discrimination stage. Since they are typically unsupervised, standard dimensionality reduction techniques also present a risk to the later classification or regression task: the lower-dimensional space found may not be the most helpful one for the second supervised learning stage and, in fact, in some cases could be harmful. Figure 7.1 shows a very simple example where PCA can lead to a projected space that is detrimental to the subsequent learning stage. More complex variants of this example can occur similarly in higher dimension and for the broader case of KPCA, which covers most known techniques. How should we design dimensionality reduction techniques to most benefit the subsequent supervised learning stage?

This paper seeks precisely to create a theoretical foundation guiding the design of dimensional-

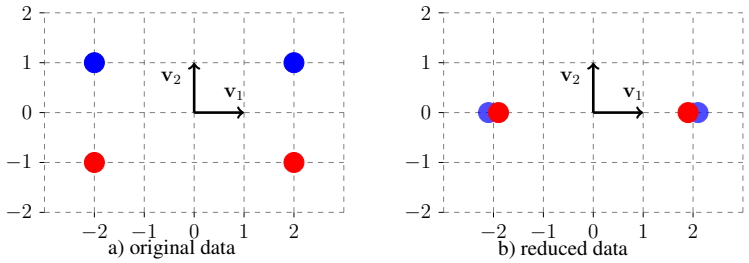


Figure 7.1: A simple example showing that simply preserving some geometric properties can be detrimental to the subsequent learning task. The original data in (a) has four points from the blue and red classes. The eigenvectors of the covariance matrix are $\mathbf{v}_1 = \left(\begin{smallmatrix}1\\0\end{smallmatrix}\right)$ and $\mathbf{v}_2 = \left(\begin{smallmatrix}0\\1\end{smallmatrix}\right)$. Standard rank-one PCA projects both blue and red points onto $\mathbf{v}_1$, thus merging them (as plotted in (b)). Any classification on the reduced data will necessarily incur a classification error of at least $\frac{1}{2}$.

ity reduction with learning guarantees. To do so, we consider a scenario where the dimensionality reduction step is not carried out *blindly* and where, instead, it is coupled with the subsequent supervised learning stage. Since, as already discussed, the key choice defining dimensionality reduction is that of a mapping to an RKHS defined by some PSD kernel K, the learning problem then consists of selecting a PSD kernel K out of a family $\mathcal{K}$ such that the hypothesis learned on a low-dimensional space after projection admits a small generalization error. We call this the *supervised kernel projection* (SKP) setting.

The framework just described bears some similarity with that of *learning kernels* [Lanckriet et al. 2004; Cortes et al. 2009, 2010; Kloft et al. 2011] (see [Gönen and Alpaydın 2011] and references therein for a recent survey). However, while the selection of a kernel is common to both frameworks, the learning problems and analyses are distinct, in particular because of the learner's freedom to select a projection space after mapping to an RKHS in the dimensionality reduction case. Nevertheless, we will adopt the same common choice for the family $\mathcal{K}$ as in much of the literature for learning kernels, that is that of convex combinations of p base PSD kernels. The RKHS we consider is thus associated to a kernel in that family $\mathcal{K}$ and the projection is over the top r eigenspace of an operator that is a function of the covariance operators of the weighted base

kernels. For the scenario of learning kernels, tight generalization bounds are known for this choice of $\mathcal{K}$ [Cortes et al. 2010]. The main contribution of this paper is to similarly derive generalization bounds for the SKP framework. Note that, while we consider a broader framework, our generalization bounds also apply to the special case of algorithms proceeding in two decoupled stages of dimensionality reduction followed by supervised learning with a linear model in an RKHS.

The choice of our learning framework is further justified by some previous empirical studies showing that tuning a dimensionality reduction algorithm in a supervised fashion, i.e. taking into account the subsequent learning algorithm using the reduced features, can result in a considerably better performance [Fukumizu et al. 2004; Gönen 2014]. Some recent work also explores learning kernels in the setting of dimensionality reduction [Lin et al. 2011], though no theoretical analysis or justification is provided for the algorithms considered. The vast majority of existing theoretical analyses of dimensionality reduction techniques, even with a fixed kernel, do not directly take into consideration the subsequent learning task and, instead, focus on the optimization of surrogate metrics such as maximizing the variance of the projected features [Zwald and Blanchard 2005]. One exception is the work of [Mosci et al. 2007], which provides a generalization guarantee for learning with hypotheses defined by KPCA with a fixed kernel followed by a regression algorithm minimizing the squared loss. [Dhillon et al. 2013] also shows that the risk of PCA combined with ordinary least squares regression is at most 4 times that of ridge regression. Recent related work also includes that of [Gottlieb et al. 2013], which derives Rademacher complexity generalization bounds for learning Lipschitz functions in a general (fixed) metric space. They show that the intrinsic dimension of the data can significantly influence learning guarantees by bounding the corresponding Rademacher complexity in terms of dimension of underlying manifold and the distortion of training set relative to that manifold.

The results of this paper are organized as follows. In Section 7.2 and 7.3 , we describe in detail

the learning scenario and the hypothesis set we consider. Section 7.4 presents our main results, which include an upper bound on the empirical Rademacher complexity of the hypothesis set, and our main generalization bound. In Section 7.5, we show a lower bound on the sample Rademacher complexity as well as other quantities, which demonstrates a necessary dependence on several crucial quantities and helps to validate the design of the suggested hypothesis class. Finally, in Section 7.6, we briefly discuss several implications of our results.

7.2 LEARNING SCENARIO

Let $\mathcal{X}$ denote the input space. We assume that the learner receives a labeled sample of size m, $S = ((x_1, y_1), \ldots, (x_m, y_m))$, drawn i.i.d. according to some distribution $\mathcal{D}$ over $\mathcal{X} \times \{-1, +1\}$, as well as an unlabeled sample $U = (x'_1, \ldots, x'_u)$ of size u, typically with $u \gg m$, drawn i.i.d. according to the marginal distribution $\mathcal{D}_\mathcal{X}$ over $\mathcal{X}$.

We assume that the learner has access to p PSD kernels $K_1, \ldots, K_p$. Instead of requiring the learner to commit to a specific kernel K defining an RKHS, we consider the case of an RKHS defined by a kernel $K_{\boldsymbol{\mu}} = \sum_{k=1} \mu_k K_k$ that is a convex combination of $K_1, \ldots, K_p$. The non-negative mixture weights μ_k, $k = 1, \ldots, p$, are parameters that can be selected by the learner to minimize the error of the classifier using the result of the dimensionality reduction (see Figure 7.2). The hypothesis set $\mathcal{H}$ we consider is thus that of linear hypotheses in a space obtained after projection in the RKHS $\mathbb{H}$ defined by $K_{\boldsymbol{\mu}}$:

$$\mathcal{H} = \left\{ x \mapsto \langle \mathbf{w}, \Pi_U \boldsymbol{\Phi}(x) \rangle_\mathbb{H} \colon \|\mathbf{w}\|_\mathbb{H} \leq 1, \boldsymbol{\mu} \in \mathcal{M} \right\}. \tag{7.1}$$

Here, $\langle \cdot, \cdot \rangle_\mathbb{H}$ and $\| \cdot \|_\mathbb{H}$ denote the inner product and norm in $\mathbb{H}$, $\boldsymbol{\Phi} \colon \mathcal{X} \to \mathbb{H}$ is the feature mapping associated to $K_{\boldsymbol{\mu}}$, Π_U a projection using the unlabeled set U, and $\mathcal{M}$ a regularization set out

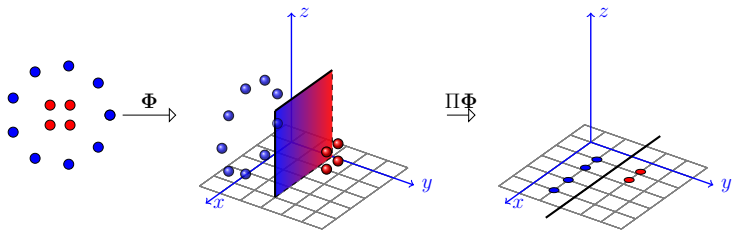


Figure 7.2: Illustration of the supervised learning scenario: (a) raw input points; (b) points mapped to a higher-dimensional space where linear separation is possible but not all dimensions are relevant; (c) projection over a lower-dimensional space preserving linear separability.

of which $\boldsymbol{\mu}$ is selected. Note that to avoid a heavier notation, we do not explicitly indicate the dependency of $\boldsymbol{\Phi}$ on $\boldsymbol{\mu}$ as this should be clear from the context.

We now specify the choices of Π_U and $\boldsymbol{\mu}$. For any $k \in [1, p]$, let $C_{U,k} \colon \mathbb{H}_k \to \mathbb{H}_k$ denote the empirical covariance operator based on the unlabeled sample U associated to the PSD kernel $\mu_k K_k$ with RKHS $\mathbb{H}_k$. Let C_U be the operator defined by $C_U = C_{U,1} + \cdots + C_{U,p}$, which acts on the sum of reproducing spaces $\mathbb{H} = \mathbb{H}_1 + \cdots + \mathbb{H}_p$. For a fixed r, Π_U is the rank-r projection over the eigenspace of C_U that corresponds to the top-r eigenvalues of C_U denoted by $\lambda_1(C_U) \geq \ldots \geq \lambda_r(C_U)$.[1] We define similarly the operators $C_{S,k}$ and C_S for the sample S, as well as the projection Π_S.

Note that the dimensionality reduction method just described is in general somewhat different from the standard KPCA with kernel K_μ. Of course, in both cases, the projection is onto the top-r eigenspace of an operator. But, while for KPCA that operator is the empirical covariance operator associated to kernel K_μ, in the setting just described, the operator C_U is the sum of the empirical covariance operators associated to each base kernel. In the special case $p = 1$, the two methods coincide. Also, both methods fall into the general SKP framework. Here, we consider the operator

[1] To simplify the presentation, we assume that the selected dimension r satisfies $\lambda_r(C_U) \neq \lambda_{r+1}(C_U)$, but this assumption is not necessary and our results can be straightforwardly extended to more general cases. Note that this assumption is satisfied in particular when the eigenvalues are simple.

C_U or C_S as they admit a favorable structure further discussed in Section 7.3.

We define the set $\mathcal{M}$ of admissible weight vectors $\boldsymbol{\mu}$ as follows:

$$\mathcal{M} = \left\{ \boldsymbol{\mu} \in \mathbb{R}^p \colon \|\boldsymbol{\mu}\|_{(r)} \leq \Lambda_{(r)}, \;\; \|\boldsymbol{\mu}\|_1 \leq 1, \;\; \sum_{k=1}^{p} \frac{1}{\mu_k} \leq \nu, \;\; \boldsymbol{\mu} \geq 0 \right\}, \tag{7.2}$$

where $\Lambda_{(r)} \geq 0$ and $\nu \geq 0$ are hyperparameters and where $\|\boldsymbol{\mu}\|_{(r)}$ is the Ky-Fan r-norm of C_U [Bhatia 1997]:[2]

$$\|\boldsymbol{\mu}\|_{(r)} = \|C_U\|_{(r)} = \sum_{i=1}^{r} \lambda_i(C_U)\,. \tag{7.3}$$

We will later show that this choice of regularization is key as it appears as a crucial term in generalization guarantees and in lower bounds. The vector $\boldsymbol{\mu}$ is further upper bounded by an L_1-norm inequality $\|\boldsymbol{\mu}\|_1 \leq 1$ as is standard in the learning kernel literature with similar kernel combinations. The lower bound constraint on $\boldsymbol{\mu}$, $\sum_{k=1}^{p} \mu_k^{-1} \leq \nu$, implies an upper bound on the eigengap of the induced covariance operator, which is a fundamental quantity that influences the concentration of eigenspaces. In Section 7.5, we give a simple example demonstrating that the dependency on the eigengap is tight, which implies the necessity of this lower bound regularization.

7.3 KERNEL PROPERTIES

In this section, we discuss the properties assumed about the base kernels, which turn out to be rather mild assumptions. We will assume that the base kernels K_k, $k \in [1, p]$, satisfy the condition $K_k(x, x) \leq 1$ for all $x \in \mathcal{X}$, which is guaranteed to hold for all normalized kernels, such as Gaussian kernels. We also assume that C_U admits at least r non-zero eigenvalues and that at least one kernel matrix among those associated to kernel K_k on sample S admits rank at least r, and

[2]The Ky-Fan r-norm is in fact a semi-norm.

similarly for the kernel matrices defined over the sample U.

We denote by $\mathbf{K}$ the kernel matrix of a kernel K associated to the sample S, $[\mathbf{K}]_{i,j} = K(x_i, x_j)$ and by $\overline{\mathbf{K}}$ the normalized kernel matrix defined by $\overline{\mathbf{K}} = \frac{\mathbf{K}}{m}$. Note that a kernel matrix thereby normalized admits the same eigenvalues as the corresponding sample covariance operator (see for example [Rosasco et al. 2010] Proposition 9.2). In particular, for any $k \in [1, p]$ and $i \in [1, m]$, we have $\lambda_i(\overline{\mathbf{K}}_k) = \lambda_i(C_{S,k})$.

We will assume the base kernels are *linearly independent with respect to the union of the samples S and U*.

Definition 7.1 (Linearly Independent Kernels)**.** Let $K_1, \ldots, K_p$ be p PDS kernels and let $S = (x_1, \ldots, x_n)$ be a sample of size m. For any $k \in [1, p]$, let $\mathbb{H}_k$ denote the RKHS associated to K_k and $\overline{\mathbb{H}}_k$ the subspace of $\mathbb{H}_k$ spanned by the set of functions $\{\mathbf{\Phi}_{K_k}(x_i) \colon i = 1, \ldots, m\}$. Then, $K_1, \ldots, K_p$ are said to be *linearly independent with respect to the sample S* if, for any $k \in [1, p]$, no non-zero function in $\overline{\mathbb{H}}_k$ can be expressed as a linear combination of the functions in $\cup_{l \neq k} \overline{\mathbb{H}}_l$.

This condition typically holds in practice, e.g., for polynomial and Gaussian kernels on $\mathbb{R}^N$. As an example, let $\mathcal{X} = \mathbb{R}^N$ and define the sample $S = \{x_1, \ldots, x_m\}$. Define two base kernels: Gaussian $K_1(x, y) = e^{-\|x-y\|^2}$ and linear $K_2(x, y) = \langle x, y \rangle$. Then $\mathbf{\Phi}_{K_1}$ is defined by $\mathbf{\Phi}_{K_1}(x) \colon t \mapsto e^{-\|x-t\|^2}$, that is $\mathbf{\Phi}_{K_1}(x)$ is an exponential function $e^{-\|x-t\|^2}$ with parameter x and argument t. Similarly, $\mathbf{\Phi}_{K_2}$ is defined by $\mathbf{\Phi}_{K_2}(x) \colon t \mapsto \langle x, t \rangle$. Thus, $\overline{\mathbb{H}}_1$ is the span of exponential functions $\{e^{-\|x_1-t\|^2}, \ldots, e^{-\|x_m-t\|^2}\}$ and $\overline{\mathbb{H}}_2$ is the span of linear functions $\{\langle x_1, t \rangle, \ldots, \langle x_m, t \rangle\}$. Clearly, no exponential function can be represented as a linear combination of linear functions and likewise, in general, no linear function is represented as a (finite) linear combination of exponential functions. Thus, the base kernels K_1 and K_2 are linearly independent with respect to a finite sample S as in Definition 7.1. More generally, the support of the base kernels can be straightforwardly modified to ensure that this condition is satisfied.

By definition of $\mathbb{H} = \mathbb{H}_1 + \cdots + \mathbb{H}_p$ and by the results of [Aronszajn 1950, Section 6], when the base kernels K_k are linearly independent with respect to sample S, then $\overline{\mathbb{H}}_k$ are orthogonal subspaces of $\mathbb{H}$, thus we can define $\overline{\mathbb{H}} = \bigoplus_{k=1}^{p} \overline{\mathbb{H}}_k$, which will be extremely useful in decomposing the spectra of operators C_S. Linearly independent base kernels imply that C_S admits at most pm nonzero eigenvalues of the form $\mu_k \lambda_j(C_{S,k})$.

7.4 GENERALIZATION BOUND

In this section, we present our generalization bound for learning with the hypothesis set $\mathcal{H}$ we introduced in Section 7.2. To obtain our generalization bound, we derive an upper bound on the empirical Rademacher complexity of $\mathcal{H}$ for a sample $S = (x_1, \ldots, x_m)$, which is defined by

$$\widehat{\mathbb{R}}_S(\mathcal{H}) = \frac{1}{m} \mathop{\mathbb{E}}_{\boldsymbol{\sigma}} \left[\sup_{h \in \mathcal{H}} \sum_{i=1}^{m} \sigma_i h(x_i) \right].$$

Here, σ_is are i.i.d. random variables taking values $+1$ and -1 with equal probabilities. The hypothesis set $\mathcal{H}$ we consider is parametrized by $\mathbf{w}$ and $\boldsymbol{\mu}$, thus $\widehat{\mathbb{R}}_S(\mathcal{H})$ can be rewritten as follows:

$$\widehat{\mathbb{R}}_S(\mathcal{H}) = \frac{1}{m} \mathop{\mathbb{E}}_{\boldsymbol{\sigma}} \left[\sup_{\substack{\|\mathbf{w}\| \leq 1 \\ \boldsymbol{\mu} \in \mathcal{M}}} \left\langle \mathbf{w}, \Pi_U \sum_{n=1}^{m} \sigma_i \boldsymbol{\Phi}(x_i) \right\rangle \right] = \frac{1}{m} \mathop{\mathbb{E}}_{\boldsymbol{\sigma}} \left[\sup_{\boldsymbol{\mu} \in \mathcal{M}} \left\| \Pi_U \sum_{i=1}^{m} \sigma_i \boldsymbol{\Phi}(x_i) \right\| \right],$$

where we used the equality case of the Cauchy-Schwarz inequality. To bound the resulting expression, it will be more convenient to work with Π_S instead of Π_U, since we are projecting instances from sample S, and similarly control the Ky-Fan r-norm $\|C_S\|_{(r)}$ rather than $\|C_U\|_{(r)}$. Both of these issues can be addressed by using concentration inequalities to bound the difference of the projections Π_U and Π_S [Zwald and Blanchard 2005] as well as the difference of the operators C_U

and C_S [Shawe-Taylor and Cristianini 2003]. To that end, we first extend the constraint set $\mathcal{M}$ to a larger one $\mathcal{N}$ defined by

$$\mathcal{N} = \left\{ \boldsymbol{\mu} \in \mathbb{R}^p \colon \|C_S\|_{(r)} \leq \Lambda_{(r)} + \kappa, \;\; \|\boldsymbol{\mu}\|_1 \leq 1, \;\; \sum_{k=1}^{p} \frac{1}{\mu_k} \leq \nu, \;\; \boldsymbol{\mu} \geq 0 \right\}, \tag{7.4}$$

where $\kappa = 4\Big(1 + \sqrt{\frac{\log\left(\frac{2p}{\delta}\right)}{2}}\Big)$. Then, the following lemma provides an upper bound in terms of Π_S.

Lemma 7.2. *For any $\delta > 0$, with probability at least $1 - \delta$, the following inequality holds for any $u \in \mathbb{H} = \mathbb{H}_1 + \cdots + \mathbb{H}_p$:*

$$\sup_{\boldsymbol{\mu} \in \mathcal{M}} \|\Pi_U u\| \leq \sup_{\boldsymbol{\mu} \in \mathcal{N}} \Big(\|\Pi_S u\| + \frac{8\kappa\nu \|u\|}{\Delta_r \sqrt{m}} \Big), \tag{7.5}$$

where $\Delta_r = \min_{k \in [1,p]} \big(\lambda_r(C_k) - \lambda_{r+1}(C_k)\big)$, C_k is the population covariance operator of kernel K_k and $\kappa = 4\Big(1 + \sqrt{\frac{\log\left(\frac{2p}{\delta}\right)}{2}}\Big)$.

The proof of this lemma is given in Appendix E.1. In view of this lemma, with probability at least $1 - \delta$, $\widehat{\mathbb{R}}_S(\mathcal{H})$ can be bounded as follows

$$\begin{aligned}
\widehat{\mathbb{R}}_S(\mathcal{H}) &\leq \frac{1}{m} \mathop{\mathbb{E}}_{\boldsymbol{\sigma}} \left[\sup_{\boldsymbol{\mu} \in \mathcal{N}} \left(\Big\| \Pi_S \sum_{i=1}^{m} \sigma_i \boldsymbol{\Phi}(x_i) \Big\| + \frac{8\kappa\nu \| \sum_{i=1}^{m} \sigma_i \boldsymbol{\Phi}(x_i) \|}{\Delta_r \sqrt{m}} \right) \right] \\
&\leq \frac{1}{m} \mathop{\mathbb{E}}_{\boldsymbol{\sigma}} \left[\sup_{\boldsymbol{\mu} \in \mathcal{N}} \Big\| \Pi_S \sum_{i=1}^{m} \sigma_i \boldsymbol{\Phi}(x_i) \Big\| \right] + \Big(\frac{8\kappa\nu}{\Delta_r \sqrt{m}} \Big) \frac{1}{m} \mathop{\mathbb{E}}_{\boldsymbol{\sigma}} \left[\sup_{\boldsymbol{\mu} \in \mathcal{N}} \Big\| \sum_{i=1}^{m} \sigma_i \boldsymbol{\Phi}(x_i) \Big\| \right],
\end{aligned}$$

using the sub-additivity of the supremum operator and the linearity of expectation. The second term can be bounded as follows:

$$\frac{1}{m} \mathop{\mathbb{E}}_{\boldsymbol{\sigma}} \left[\sup_{\boldsymbol{\mu} \in \mathcal{N}} \Big\| \sum_{i=1}^{m} \sigma_i \boldsymbol{\Phi}(x_i) \Big\| \right] \leq \frac{1}{m} \mathop{\mathbb{E}}_{\boldsymbol{\sigma}} \left[\sup_{\|\boldsymbol{\mu}\|_1 \leq 1} \Big\| \sum_{i=1}^{m} \sigma_i \boldsymbol{\Phi}(x_i) \Big\| \right] \leq \sqrt{\frac{\eta_0 e \lceil \log p \rceil}{m}}, \tag{7.6}$$

where $\eta_0 = \frac{23}{22}$, using the bound on the Rademacher complexity of learning kernels given by Theorem 2 of [Cortes et al. 2010]. The following lemma helps us bound the first term.

Lemma 7.3. *For any $\delta > 0$, with probability at least $1 - \delta$, the following inequality holds:*

$$\frac{1}{m}\mathop{\mathbb{E}}_{\boldsymbol{\sigma}}\left[\sup_{\boldsymbol{\mu}\in\mathcal{N}}\left\|\Pi_S\sum_{i=1}^{m}\sigma_i\boldsymbol{\Phi}(x_i)\right\|\right] \leq \sqrt{\frac{2\big(\Lambda_{(r)}+\kappa\big)\log\left(2pm\right)}{m}}, \tag{7.7}$$

where $\kappa = 4\Big(1+\sqrt{\frac{\log\frac{2p}{\delta}}{2}}\Big)$.

The proof of the lemma is given in Appendix E.2. Combining Lemmas 7.2 and 7.3 yields directly the following result.

Theorem 7.4. *Let $\mathcal{H}$ be the hypothesis set defined in* (7.1)*. Then, for any $\delta > 0$, with probability at least $1 - \delta$ over the draw of an i.i.d. labeled sample S of size $m < u$, the empirical Rademacher complexity of the hypothesis set $\mathcal{H}$ can be bounded as follows:*

$$\widehat{\mathbb{R}}_S(\mathcal{H}) \leq \frac{1}{\sqrt{m}}\left[\sqrt{2\big(\Lambda_{(r)}+\kappa\big)\log\left(2pm\right)} + \frac{8\kappa\nu}{\Delta_r}\sqrt{\frac{\eta_0 e\lceil\log p\rceil}{m}}\right], \tag{7.8}$$

where $\Delta_r = \min_{k\in[1,p]}\big(\lambda_r(C_k) - \lambda_{r+1}(C_k)\big)$, C_k is the population covariance operator of kernel K_k, $\kappa = 4\Big(1+\sqrt{\frac{\log\left(2p/\delta\right)}{2}}\Big)$ and $\eta_0 = \frac{23}{22}$.

Note that Δ_r is not a random variable and does not depend on the choice of S or U. It only depends on the spectral properties of the covariance operator for the distribution $\mathcal{D}_{\mathcal{X}}$ and the choice of the projection dimension r.

We now compare this bound to the one known for the Rademacher complexity of a similar hypothesis set in the scenario of learning kernels where a convex combination kernel $K_{\boldsymbol{\mu}}$ is also used [Cortes et al. 2010]. This will help us measure the additional complexity cost due to the

dimensionality reduction step. Of course, the learning kernel scenario and regularization differ from ours. But, we can make them comparable by considering the case $U = S$, that is the case where U is an unlabeled version of S and can express $\Lambda_{(r)}$ in terms of unscaled sample kernel matrices as follows:

$$\Lambda_{(r)} = \frac{1}{m} \sup_{|I|=r} \sum_{(k,j)\in I} \mu_k \lambda_j(\mathbf{K}_k) \leq \frac{1}{m} \sup_{|I|=r} \sum_{(k,j)\in I} \lambda_j(\mathbf{K}_k)\,. \tag{7.9}$$

If we define $s_r = \sup_{|I|=r} \sum_{(k,j)\in I} \lambda_j(\mathbf{K}_k)$ as the largest r-sum of eigenvalues selected from all base kernel matrices, and $s'_m = \sup_{k\in[1,p]} \operatorname{Tr}[\mathbf{K}_k]$, which is the largest m-sum of eigenvalues selected from a single base kernel matrix, the Rademacher complexity of our hypothesis class is in $\widetilde{O}\big(\sqrt{s_r/m}\big)$, while that of the hypothesis used in the learning kernel setting is in $\widetilde{O}\big(\sqrt{s'_m/m}\big)$. Thus, for $r = m$, the upper bound on the Rademacher complexity in our supervised dimensionality case is higher. The difference is due precisely to the extra freedom that the learner has to define a projection space by selecting eigenvectors from different kernel matrices, while in the learning kernel case he needs to commit instead to a single kernel matrix. For r sufficiently smaller than m, the complexity term in the supervised dimensionality case could of course be more favorable ($s_r \leq s'_m$).

The following is our main generalization bound for supervised dimensionality reduction. We denote by $R(h)$ the generalization error with respect to the zero-one loss and by $\widehat{R}_{S,\rho}(h)$ the empirical margin loss of $h \in \mathcal{H}$, that is the fraction of points in S classified with margin less than ρ by h.

Theorem 7.5. *Let $\mathcal{H}$ be the hypothesis set defined in* (7.1)*. Then, with probability at least $1 - \delta$*

over the draw of a sample S of size m, the following holds for all $h \in \mathcal{H}$:

$$R(h) \leq \widehat{R}_{S,\rho}(h) + \frac{2}{\rho\sqrt{m}}\left(\sqrt{2\big(\Lambda_{(r)} + \kappa\big)\log\left(2pm\right)} + \frac{8\kappa\nu}{\Delta_r}\sqrt{\frac{\eta_0 e\lceil \log p\rceil}{m}}\right) + 3\sqrt{\frac{\log\frac{4p}{\delta}}{2m}}\,.$$

Proof. The theorem follows directly by combining the high-probability upper bound on the Rademacher complexity given by Theorem 7.4 and the standard high-probability Rademacher-based generalization bound of [Koltchinskii and Panchenko 2002a] (see also [Bartlett and Mendelson 2003]). □

To our knowledge, this is the first learning guarantee given for the scenario of supervised dimensionality reduction. The bound of the theorem is in $O\big(\sqrt{\Lambda_{(r)}\log(pm)/m}\big)$. Thus, it suggests that the Ky-Fan r-norm of the covariance operator plays a key role in the generalization ability of hypotheses in this scenario. This is further supported by the presence of that term in a lower bound proven in the next section. Note that the dependency of the bound on the number of base kernels p is only logarithmic, which suggests using a rather large number of base kernels. The presence of the term in the bound depending on ν and Δ_r is due to the concentration bound for projections. The parameter ν controls the eigengap of the learned operator C_U, while, as already pointed out, Δ_r is a quantity that does not depend on the sample or on $\boldsymbol{\mu}$, it is entirely defined by the choice of the base kernel functions. We further elaborate on this in Section 7.5.

We note that [Mosci et al. 2007] and [Gottlieb et al. 2013] also give generalization bounds for a supervised scenario of dimensionality reduction, however, they do not learn a mapping and projection for dimensionality reduction jointly with a hypothesis learned on the projected space using a discriminative algorithm. Nevertheless, their generalization bounds are comparable to the special case of our bound where $p = 1$.

The analysis of [Gottlieb et al. 2013] is presented for general metric spaces, which is more gen-

eral than what we consider here. In the case of the Euclidean space, their generalization bound is in $O\big(\sqrt{d/m} + \sqrt{\eta/m}\big)$, where d is the dimension of underlying data manifold and η is the average distance of the training set to that manifold. While both bounds admit a similar dependence on m, our bound relies on the Ky-Fan norm of the projection rather than the intrinsic dimension of the dataset. We note that the existence of an approximate low-dimensional manifold is a distributional assumption which, depending on the task, may not hold. Furthermore, even when it does, the estimation of the intrinsic dimension is typically a difficult task. The Ky-Fan norm, on the other hand, can be directly controlled by the choice of the regularization parameter in the definition of the hypothesis set.

The generalization bound of [Mosci et al. 2007] is in $O\big(1/\sqrt{m}\big)$. However, while we fix the number of eigenvalues for dimensionality reduction to r, their bound requires selecting all eigenvalues above a threshold $\lambda_m = O\big(1/\sqrt{m}\big)$. Furthermore, as already mentioned, their analysis holds for the specific setting of KPCA with a fixed kernel ($p = 1$) followed by Ridge regression.

7.5 Lower bounds

In this section, we show a lower bound on the Rademacher complexity of the hypothesis class $\mathcal{H}$ defined by (7.1). Furthermore, we give a simple example demonstrating the necessity of the eigengap term appearing in Lemma 7.2 and also motivate the additional regularization term ν.

Theorem 7.6. *For any m and r there exist samples S and U, a setting of the regularization parameter $\Lambda_{(r)}$, as well as a choice of base kernels $K_1, \ldots, K_p$ such that the following inequality holds:*

$$\widehat{\mathbb{R}}_S(\mathcal{H}) \geq \sqrt{\frac{\Lambda_{(r)}}{2m}}\,.$$

The proof is given in Appendix E.3. The result proves the tightness of the upper bound we

derived in terms of m, up to logarithmic factors. It further shows the key role of the regularization parameter $\Lambda_{(r)}$ and justifies the presence of Ky-Fan r-norm constraint in the definition of the hypothesis set.

We now also give a simple example showing that the projections must necessary depend on an eigengap quantity. This in turn motivates the dependency of Lemma 7.2 on the quantity Δ_r as well as the regularization $\sum_{k=1}^{p} \mu_k^{-1} \leq \nu$ which is used to bound the eigengap of the learned operator C_S (see equation (E.3) in the proof of Lemma 7.2). The fact that the eigengap is essential for the concentration of projections has been known in the matrix perturbation theory literature [Stewart and Sun 1990]. The following proposition gives an example which shows that the dependence on the eigengap is tight.

Proposition 7.7. *There exist operators A and B such that*

$$\|P_r(A) - P_r(B)\| = \frac{2\|A - B\|}{\lambda_r(A) - \lambda_{r+1}(A)},$$

where $P_r(A)$ (resp. $P_r(B)$) is the orthogonal projection onto the top r eigenspace of A (resp. B).

Proof. Let $r = 1$ and consider A and B defined as follows: $A = \left(\begin{smallmatrix} 1+\epsilon & 0 \\ 0 & 1 \end{smallmatrix}\right)$ and $B = \left(\begin{smallmatrix} 1 & 0 \\ 0 & 1+\epsilon \end{smallmatrix}\right)$, thus $A - B = \left(\begin{smallmatrix} \epsilon & 0 \\ 0 & -\epsilon \end{smallmatrix}\right)$, which implies that $\|A - B\| = \epsilon$. Also, the eigengap is equal to $\lambda_1(A) - \lambda_2(A) = \epsilon$. Now, note that $P_1(A)$ is the projection onto $e_1 = (1, 0)^\top$ and $P_1(B)$ is the projection onto $e_2 = (0, 1)^\top$. Since e_1 and e_2 are orthogonal, this implies $\|P_1(A) - P_1(B)\| = \|P_1(A)\| + \|P_1(B)\| = 2$. On the other hand, $\frac{2\|A-B\|}{\lambda_1(A)-\lambda_2(A)} = \frac{2\epsilon}{\epsilon} = 2$, which completes the proof. □

7.6 DISCUSSION

We now briefly discuss the results presented in the previous sections. Let us first emphasize that our choice of the hypothesis class $\mathcal{H}$ (Section 7.2) is strongly justified a posteriori by the learning guarantees we presented: both our upper and lower bounds on the Rademacher complexity (Sections 7.4 and 7.5) suggest that the quantities present in the definition of $\mathcal{H}$ play an important role. The regularization parameters we provide can be tuned to directly bound each of these crucial quantities and thereby limit the risk of over fitting.

Second, the hypothesis set suggested in this paper provides a unified framework for choosing an optimal dimensionality reduction method. It suggests to specify a set of potential methods (equivalent to a set of base kernels) and then learn their combination jointly with a projection. Moreover, the generalization bound is logarithmic in the number of base kernels, which encourages the use of a very large base set.

Third, we observe that the hypothesis class $\mathcal{H}$ clearly motivates the design of a single-stage coupled algorithm. Such an algorithm would be based on structural risk minimization (SRM) and seek to minimize the empirical error over increasingly complex hypothesis sets, by varying the parameters $\Lambda_{(r)}$ and ν, to trade-off empirical error and model complexity. It is worthwhile to note that our hypothesis set is constructed in such a way that the search over the choices of parameters $\boldsymbol{\mu}$ does not incur the bottleneck of recomputing the eigendecomposition of operator C_U at every iteration. Instead, we require the computation of the eigendecomposition of the (unweighted) base kernel matrices once as a preprocessing step. The key to that is the assumption of linearly independent kernels, which is typically satisfied in practice.

We note that the existing literature has empirically evaluated both learning kernels with KPCA in an unsupervised (two-stage) fashion [Zhuang et al. 2011; Lin et al. 2011] and applied supervised

KPCA (single-stage training) with a fixed kernel function [Fukumizu et al. 2004; Gönen 2014]. While these existing algorithms do not directly consider the hypothesis class we motivated, they can, in certain cases, still select a hypothesis function that is found in our class. In particular, our learning guarantees are applicable to hypotheses chosen in a two-stage manner, as long as the regularization constraints are satisfied and the same family of projections are used. Similarly, the case $p = 1$, which corresponds to the standard fixed-kernel supervised learning scenario, is covered by our analysis. Even in such cases, the bounds that we provide would be the first to guarantee the generalization ability of the algorithm via bounding the sample Rademacher complexity.

7.7 Conclusion

We presented a new analysis and generalization guarantees for the scenario of supervised dimensionality reduction with a learned kernel. The hypothesis class is designed with regularization constraints that are directly motivated by the upper and lower bounds on its Rademacher complexity. Our analysis suggests the design of learning algorithms for selecting hypotheses from this specifically tailored class, either in a two-stage or a single-stage manner. Our analysis can also benefit the study of other similar hypothesis sets within the SKP framework.

8 | CONCLUSION

We presented a unified in-depth analysis of several central problems in large-scale machine learning. The analysis provided includes principled learning guarantees that connect the generalization error to the complexity of the underlying learning scenario. For each of the large-scale problems considered, we have given an algorithm motivated by the learning guarantees and illustrated its performance on a variety of datasets.

The experiments showed that algorithms based on our theory outperform the baselines. Thus, this dissertation opens doors for a wide range of applications. For instance, the boosting-related solutions given in Chapter 3 and Chapter 4 can be generalized to different hypotheses classes beyond decision trees (e.g. Kernels and Deep Neural Networks). Thus, they can significantly improve speech, image and video recognition solutions.

The solutions to learning with multiple objectives, presented as the ALMO algorithm in Chapter 5 are proven to build more robust models for a variety of objectives in different machine learning problems. This framework is directly beneficial to complex learning scenarios that go beyond supervised learning, where a learner is seeking to combine multiple objectives, such as transfer learning and domain adaptation. Moreover, ALMO is implemented in state-of-the-art software libraries, which makes it publicly available to broader communities.

The solution to efficient gradient computation for large-scale structured prediction problems

presented in Chapter 6 will directly improve sequence-to-sequence machine learning applications, such as machine translation, grammar correction and speech recognition. Moreover, it's gradient computation is implemented in popular gradient-based machine learning libraries. Thus, researchers and practitioners in the relevant fields can quickly benefit from deploying our gradient computation framework in their applications.

The supervised dimensionality reduction theory in Chapter 7 should serve as a principled and general instrumental framework to any application with a large number of input features. Moreover, it suggests a coupled dimensionality reduction algorithm based on the Structural Risk Minimization principle, where the kernel family regularization is varied to achieve an optimal trade-off between the empirical error and model complexity. The applications of our solution include, but not limited to high-dimensional tabular data, time series analysis, text and image classification.

A | APPENDIX TO CHAPTER 3.

A.0.1 PROOF OF THEOREM 3.1

Theorem 1. For any sample $S = (x_1, \ldots, x_m)$, the empirical Rademacher complexity of a hypothesis set $\mathcal{H}$ is defined by $\widehat{\mathfrak{R}}_S(\mathcal{H}) = \mathbb{E}_{\boldsymbol{\sigma}}\left[\sup_{h\in\mathcal{H}} \sum_{i=1}^{m} \sigma_i h(x_i)\right]$, where, σ_is, $i \in [m]$, are independent uniformly distributed random variables taking values in $\{-1, 1\}$. The following upper bound holds for the empirical Rademacher complexity of $\mathcal{H}_{n,\lambda,q}$:

$$\widehat{\mathfrak{R}}^S(\mathcal{H}_{n,\lambda,q}) \leq \lambda\sqrt{\frac{(4n+2)\log_2(d+2)\log(m+1)}{m}},$$

where d is input data dimension.

Proof. For the purpose of this proof, let $\mathcal{H}_n$ be the family of binary decision trees with leaf values $w_j \in \{-1, +1\}$. We use the regularization in the family $\mathcal{H}_{n,\lambda,q}$ and the connection to the family $\mathcal{H}_n$ in the proof below. Additionally, let $r \geq 1$ such that $\frac{1}{r} + \frac{1}{q} = 1$, meaning that the $r-$norm is the dual to the $q-$norm. To aid the presentation in the proof, we are going to define a vector $\widehat{\boldsymbol{\sigma}}$ s.t. $[\widehat{\boldsymbol{\sigma}}]_j = \sum_{x_i\in\text{leaf}_j} \sigma_i$, the j-th coordinate of which contains the sum of the Rademacher variables that correspond to the sample points that fall within j-th leaf of a tree h.

$$\widehat{\mathfrak{R}}^S(\mathcal{H}_{n,\lambda,q}) = \frac{1}{m}\,\mathop{\mathbb{E}}_{\sigma}\left[\sup_{h\in\mathcal{H}_{n,\lambda,q}}\left[\sum_{n=1}^{m}\sigma_n h(x_n)\right]\right] \tag{A.1}$$

$$= \frac{1}{m}\,\mathop{\mathbb{E}}_{\sigma}\left[\sup_{h\in\mathcal{H}_{n,\lambda,q}}\left[\widehat{\boldsymbol{\sigma}}\cdot\mathbf{w}\right]\right] \tag{A.2}$$

$$\leq \frac{1}{m}\,\mathop{\mathbb{E}}_{\sigma}\left[\sup_{h\in\mathcal{H}_{n,\lambda,q}}\|\widehat{\boldsymbol{\sigma}}\|_r\|\mathbf{w}\|_q\right] \tag{A.3}$$

$$\leq \frac{\lambda}{m}\,\mathop{\mathbb{E}}_{\sigma}\left[\sup_{h\in\mathcal{H}_{n}}\|\widehat{\boldsymbol{\sigma}}\|_r\right] \tag{A.4}$$

$$\leq \frac{\lambda}{m}\,\mathop{\mathbb{E}}_{\sigma}\left[\sup_{h\in\mathcal{H}_{n}}\|\widehat{\boldsymbol{\sigma}}\|_1\right] \tag{A.5}$$

$$= \frac{\lambda}{m}\,\mathop{\mathbb{E}}_{\sigma}\left[\sup_{h\in\mathcal{H}_{n}}\sum_{i=1}^{n}|[\widehat{\boldsymbol{\sigma}}]_i|\right] \tag{A.6}$$

$$= \frac{\lambda}{m}\,\mathop{\mathbb{E}}_{\sigma}\left[\sup_{h\in\mathcal{H}_{n}}\sum_{l\in\text{leaves(h)}}\Big|\sum_{i=1}^{m}\sigma_i 1_{\{x_i\in l\}}\Big|\right] \tag{A.7}$$

$$\leq \frac{\lambda}{m}\,\mathop{\mathbb{E}}_{\sigma}\left[\sup_{h\in\mathcal{H}_{n},s_l\in\{+1,-1\}}\sum_{l\in\text{leaves(h)}}s_l\sum_{i=1}^{m}\sigma_i 1_{\{x_i\in l\}}\right] \tag{A.8}$$

$$= \frac{\lambda}{m}\,\mathop{\mathbb{E}}_{\sigma}\left[\sup_{h\in\mathcal{H}_{n},s_l\in\{+1,-1\}}\sum_{i=1}^{m}\sigma_i\sum_{l\in\text{leaves(h)}}s_l 1_{\{x_i\in l\}}\right] \tag{A.9}$$

$$\leq \lambda\sqrt{\frac{(4n+2)\log_2(d+2)\log(m+1)}{m}} \tag{A.10}$$

Where n is the number of internal nodes, and d is the input data dimension. The inequality (14) is a direct application of the Hölder's inequality for dual norms. The inequality (16) uses $\|\cdot\|_r \leq \|\cdot\|_1$. The equality (18) directly follows from the definition of $\widehat{\boldsymbol{\sigma}}$. The last inequality (21) follows from the fact that the VC-dimension of binary classification trees can be bounded by $(2n+1)\log_2(d+2)$ [Mohri et al. 2012] and a direct application of Massart's lemma [Massart and Picard 2007]. □

A.0.2 PROOF OF THEOREM 3.2

Theorem 2. Fix $\rho > 0$. Let $\mathcal{H}_k = \mathcal{H}_{n_k,\lambda_k,q_k}$, where (n_k), (λ_k) are sequences of constraints on the number of internal nodes n and the leaf vector norm $\|\mathbf{w}\|_q$. Define $\mathcal{F} = \operatorname{conv}(\cup_{k=1}^{K}\mathcal{H}_k)$. Then, for any $\delta > 0$, with probability at least $1-\delta$ over the draw of a sample S of size m, the following inequality holds for all $f = \sum_{t=1}^{T}\alpha_t h_t \in \mathcal{F}$:

$$R(f) \leq \widehat{R}_{S,\rho}(f) + \frac{4}{\rho}\sum_{t=1}^{T}\alpha_t\lambda_{I_t}\sqrt{\frac{(4n_{I_t}+2)\log_2(d+2)\log(m+1)}{m}} + C(m,K),$$

where I_t is the index of the subclass selected at time t and $C(m,K) = O\left(\sqrt{\frac{\log(K)}{\rho^2 m}\log\left[\frac{\rho^2 m}{\log(K)}\right]}\right)$.

Proof. For this proof we are going to make use of the generalization bounds for broad families of real-valued functions given in Theorem 1 of [Cortes et al. 2014]. Adapted to our notation, it states that for any f from a family of real-valued functions $\mathcal{F}$ that is equal to the convex hull of $\cup_{k=1}^{K}\mathcal{H}_k$, for any $\delta > 0$ with probability at least $1-\delta$ over the choice of sample $S \sim \mathcal{D}^m$, the following generalization bound holds:

$$R(f) \leq \widehat{R}_{S,\rho}(f) + \frac{4}{\rho}\sum_{t=1}^{T}\alpha_t\mathfrak{R}_m(\mathcal{H}_t) + \frac{2}{\rho}\sqrt{\frac{\log K}{m}} + \sqrt{\left\lceil\frac{4}{\rho^2}\log\left(\frac{\rho m^2}{\log K}\right)\right\rceil\frac{\log K}{m} + \frac{\log(\frac{2}{\delta})}{2m}}.$$

where α_t is are the weights that represent f in the convex hull of $\cup_{k=1}^{K}\mathcal{H}_k$, that is $f = \sum_{t=1}^{T}\alpha_t h_t$ s.t. $\boldsymbol{\alpha} = [\alpha_1,\ldots,\alpha_T]$ is in the simplex Δ. This bound is directly applicable to the Regularized Gradient Boosting that we define, since at each boosting round, the algorithm selects a base predictor $h_t \in \mathcal{H}_t$, and multiplies it by a coefficient α_t. Thus, after T boosting rounds, we will have obtained an ensemble f such that $f = \sum_{t=1}^{T}\alpha_t h_t \in \operatorname{conv}(\cup_{k=1}^{K}\mathcal{H}_k)$ and $\boldsymbol{\alpha}$ directly in the simplex

Δ.

Applying the Rademacher complexity bound on the regularized families of regression trees $\mathcal{H}_{n,\lambda,q}$ that we derived in Theorem 3.1 and noting that

$$\frac{2}{\rho}\sqrt{\frac{\log K}{m}} + \sqrt{\left\lceil \frac{4}{\rho^2}\log\left(\frac{\rho m^2}{\log K}\right)\right\rceil \frac{\log K}{m} + \frac{\log(\frac{2}{\delta})}{2m}} = O\left(\sqrt{\frac{\log(K)}{\rho^2 m}\log\left[\frac{\rho^2 m}{\log(K)}\right]}\right) \quad \text{(A.11)}$$

We obtain the expression for the bound in Theorem 3.2. □

A.0.3 PROOF OF LEMMA 3.3

Lemma 3. Assume that $\Phi(y, h)$ is differentiable with respect to the second argument, and that $\frac{\partial \Phi}{\partial h}$ $C_\Phi(y)$-Lipschitz with respect to the second argument, for any fixed value y of the first argument. for all $k \in [0, K]$, define $L'_k(\boldsymbol{\alpha}) = \frac{\partial L}{\partial \alpha_k}$. Then, $L'_k(\boldsymbol{\alpha})$ is Lipschitz-continuous with the corresponding Lipschitz constants C_k bounded as follows:

$$C_k \leq \frac{1}{m}\sum_{i=1}^{m} h_k^2(x_i) C_\Phi(y_i). \quad \text{(A.12)}$$

Proof. The k-th derivative of $L(\boldsymbol{\alpha})$ is equal to (except $\alpha_k = 0$):

$$L'_k(\boldsymbol{\alpha}) = \frac{1}{m}\sum_{i=1}^{m} \frac{\partial \Phi}{\partial h}\left(y_i, \sum_{t=1}^{T} \alpha_t h_t(x_i)\right) h_k(x_i) + c_k, \quad \text{(A.13)}$$

where $c_k = \beta\lambda_k\sqrt{\frac{(4n_k+2)\log_2(d+2)\log(m+1)}{m}}$. Let $\mathbf{e}_k$ be the k-th standard basis vector and define the following: $\Delta L'_k(\boldsymbol{\alpha}) = L'_k(\boldsymbol{\alpha}) - L'_k(\boldsymbol{\alpha} + \delta\mathbf{e}_k)$.

$$
\begin{aligned}
\left|\Delta L'_k(\boldsymbol{\alpha})\right| &= \left|\frac{1}{m}\sum_{i=1}^{m} h_k(x_i)\left[\frac{\partial\Phi}{\partial h}\left(y_i, \sum_{t=1}^{T}\alpha_t h_t(x_i)\right) - \frac{\partial\Phi}{\partial h}\left(y_i, \sum_{t=1}^{T}\alpha_t h_t(x_i) + \delta h_k(x_i)\right)\right]\right| \\
&\leq \frac{1}{m}\sum_{i=1}^{m} |h_k(x_i)|\left|\frac{\partial\Phi}{\partial h}\left(y_i, \sum_{t=1}^{T}\alpha_t h_t(x_i)\right) - \frac{\partial\Phi}{\partial h}\left(y_i, \sum_{t=1}^{T}\alpha_t h_t(x_i) + \delta h_k(x_i)\right)\right| \\
&= \frac{1}{m}\sum_{i=1}^{m} |h_k(x_i)|\left|\frac{\partial\Phi}{\partial h}\left(y_i, f\right) - \frac{\partial\Phi}{\partial h}\left(y_i, f + \delta h_k(x_i)\right)\right| \\
&\leq \frac{1}{m}\sum_{i=1}^{m} |h_k(x_i)| C_\Phi(y_i)\big|h_k(x_i)\big||\delta| \\
&= \frac{1}{m}\sum_{i=1}^{m} h_k^2(x_i) C_\Phi(y_i)|\delta|
\end{aligned}
$$

Thus, $L'_k(\boldsymbol{\alpha})$ is Lipschitz-continuous with the corresponding Lipschitz constant bounded by $\frac{1}{m}\sum_{i=1}^{m} h_k^2(x_i) C_\Phi(y_i)$.

□

A.0.4 PROOF OF LEMMA A.1

Lemma A.1. *For each $k \in [0, K]$ let $\mathcal{H}_{n_k,\lambda_k,2}$ be the family of regularized regression trees with $\|\mathbf{w}\|_2 \leq \lambda_k$ and the number of internal nodes bounded by n_k. The regularized objective $L(\boldsymbol{\alpha})$ as in Equation 3.7 has Lipschitz-continuous derivatives with the coordinate-wise Lipschitz constants C_k bounded as follows:*

$$
C_k \leq \lambda_k\left[\max_{1\leq i\leq m} C_\Phi(y_i)\right]. \tag{A.14}
$$

Proof. For a sample S and a fixed tree h let η_l be the number of sample points falling within the

leaf l.

$$
\begin{aligned}
C_k &\leq \frac{1}{m}\left[\max_{1\leq i\leq m} C_\Phi(y_i)\right]\sum_{i=1}^{m} h_k^2(x_i) \\
&\leq \frac{1}{m}\left[\max_{1\leq i\leq m} C_\Phi(y_i)\right]\sum_{l\in\text{leaves}(h_k)} \eta_l w_l^2 \\
&\leq \frac{1}{m}\left[\max_{1\leq i\leq m} C_\Phi(y_i)\right]\|\mathbf{w}\|_2 \max_{l\in\text{leaves}(h_k)} \eta_l \\
&\leq \|\mathbf{w}\|_2\left[\max_{1\leq i\leq m} C_\Phi(y_i)\right] \\
&\leq \lambda_k\left[\max_{1\leq i\leq m} C_\Phi(y_i)\right]
\end{aligned}
$$

This results in the coordinate sampling distribution for the Randomized Coordinate Descent.

$$p_k = \frac{\lambda_k}{\sum_{j=1}^{K}\lambda_j} \tag{A.15}$$

□

A.0.5 DESCRIPTIVE STATISTICS OF THE UCI DATASETS

Table A.1: Dataset statistics

	sonar	**cancer**	**diabetes**	**ocr17**	**ocr49**	**mnist17**	**mnist49**	**higgs**
Examples	208	699	768	2000	2000	15170	13782	98050
Features	60	9	8	196	196	400	400	29

Note that mnist17 and mnist49 refer to the original 20-by-20 pixel datasets, where only two digits (1,7 and 4,9 respectively) were sampled. The *cancer* dataset refers to the *breastcancer* dataset in the UCI repository. The *higgs* dataset refers to the preprocessed Higgs boson dataset

from the *OpenML* challenge, 2016.

B | APPENDIX TO CHAPTER 4.

B.1 PREVIOUS WORK: MORE DETAILED DISCUSSION

Domain adaptation with a single source domain and a target domain has been widely studied [Ben-David et al. 2007; Blitzer et al. 2008; Mansour et al. 2009a; Ben-David et al. 2010; Cortes and Mohri 2014; Cortes et al. 2015b; Wang et al. 2019b] and has applications to several fields ranging from acoustic modelling [Liao 2013] to object recognition [Torralba and Efros 2011]. It has been studied in unsupervised settings with unlabeled target domain examples [Gong et al. 2012; Long et al. 2015; Ganin and Lempitsky 2015], in supervised settings with the aid of labeled target domain examples [Yang et al. 2007; Hoffman et al. 2013; Girshick et al. 2014; Motiian et al. 2017b], and in semi-supervised settings where both labeled and unlabeled target examples are available [Tzeng et al. 2015; Saito et al. 2019].

In a wide variety of applications, the learner has access to information from multiple sources. Such problems are often referred to as *multiple-source adaptation*. Multiple-source adaptation problems, where the learner has access to unlabeled source data together with predictors that are trained for each particular domain has been formalized in [Mansour et al. 2008, 2009b; Hoffman et al. 2018a]. There are other multiple-source adaptation scenarios, where labeled examples are available from multiple sources and unlabeled or labeled examples are available from the target

domain. This includes adversarial training, which has been studied by [Motiian et al. 2017a; Pei et al. 2018; Zhao et al. 2018; Xu et al. 2018]. Algorithms for learning from multiple untrusted sources have been proposed by [Konstantinov and Lampert 2019]. Another related problem is *domain generalization* [Pan and Yang 2009; Muandet et al. 2013; Xu et al. 2014], where information from multiple sources is used to obtain a predictor that generalizes to a previously unseen domain.

There are various algorithms, successfully applying boosting with multiple sources to domain adaptation and transfer learning problems, that have inspired our analysis. The TRADABOOST [Dai et al. 2007] algorithm, having a set of weak learners trained on the source domain, at every boosting round selects those that minimize the error on the target domain. In case of multiple sources and a single target, [Yao and Doretto 2010] developed MULTISOURCETRADABOOST algorithm that trains weak learners on the union of each of the sources and the target, thus reducing the negative knowledge transfer effect. These algorithms have been further improved and widely adopted in practice [Yuan et al. 2017; Cheng et al. 2013; Zhang et al. 2014]. Another approach that uses multi-view ADABOOST for single and multi-source domain adaptation was proposed by [Xu and Sun 2012, 2011]. They divide the feature space into two *views* based on the source and target; at each boosting step, two weak learners are trained on these views and the sample distribution is updated according to the errors on the target domain.

A number of experimental studies have shown the benefits of having an ensemble of weak learners for multi-task learning and domain adaptation problems. Moreover, in certain cases the boosting approach can outperform traditional methods. For example, [Huang et al. 2010, 2012] showed that by selecting a weak learner jointly with a feature that is predictive across multiple domains at every boosting step, one can achieve higher accuracy than standard transfer learning methods. Moreover, the margin provided by boosting-style algorithms can aid in transfer learning where target domain is unlabelled. [Habrard et al. 2013] have developed an algorithm that jointly

minimizes the the source domain error and margin violation proportion on the target domain.

[Wang et al. 2019a] have demonstrated that boosting classifiers from different domains can be done online and showed efficient algorithm for the ADABOOST-style sample distribution updates. For certain types of high-dimensional data, such as images and text, boosting may be not as efficient as other multi-task learning methods. However, a number of works such as [Taherkhani et al. 2020] and [Becker et al. 2013] have shown that multi-source boosting can be combined with Deep Neural Networks for multi-task learning on large scale datasets.

In the context of neural networks, the idea of using domain probabilities when combining experts, also termed *gating networks*, goes back to [Hampshire and Waibel 1992] and [Jacobs et al. 1991].

Agnostic loss has been used in several machine learning problems. [Namkoong and Duchi 2016; Levy et al. 2020] proposed efficient algorithms to minimize agnostic loss in the context of distributionally robust optimization. Agnostic loss in federated learning has been studied by [Mohri et al. 2019b; Hamer et al. 2020; Ro et al. 2021], who provided theoretical guarantees and algorithms. [Lahoti et al. 2020] used agnostic loss to achieve fairness in machine learning models.

B.2 PROOF OF THE PROPOSITIONS 4.1 AND 4.2

This section contains the proofs for Proposition 4.1 and Proposition 4.2 discussed in Section 4.2.

Proposition 4.1. *There exist distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ and hypotheses h_1 and h_2 with $\mathcal{L}(\mathcal{D}_1, h_1) = 0$ and $\mathcal{L}(\mathcal{D}_2, h_2) = 0$ such that $\mathcal{L}\big(\frac{1}{2}(\mathcal{D}_1 + \mathcal{D}_2), \alpha h_1 + (1-\alpha)h_2\big) \geq \frac{1}{2}$ for any $\alpha \in [0, 1]$,*

Proof. Consider the case where $\mathcal{X}$ is reduced to two elements, $\mathcal{X} = \{a_1, a_2\}$, where $\mathcal{D}_1$ is the point mass on a_1, $\mathcal{D}_2$ the point mass on a_2, and where the target labeling function is f defined

by $f(a_1) = +1$, $f(a_2) = -1$. Let h_1 be defined by $h_1(a_1) = h_1(a_2) = +1$ and and h_2 by $h_2(a_1) = h_2(a_2) = -1$.

Then, h_1 is a perfect predictor for the first domain since $\mathcal{L}(\mathcal{D}_1, h_1) = \mathbb{I}(h_1(a_1)f(a_1) \leq 0) = 0$, and h_2 is a perfect predictor for the second domain since $\mathcal{L}(\mathcal{D}_2, h_2) = \mathbb{I}(h_2(a_2)f(a_2) \leq 0) = 0$. However, for any $\alpha \in [0, 1]$, we have

$$\mathcal{L}\left(\frac{1}{2}(\mathcal{D}_1 + \mathcal{D}_2), \alpha h_1 + (1-\alpha)h_2\right) = \frac{1}{2}\,\mathbb{I}(2\alpha - 1 \leq 0) + \frac{1}{2}\,\mathbb{I}(1 - 2\alpha \leq 0) \geq \frac{1}{2}.$$

This concludes the proof of Proposition 4.1. □

Proposition 4.2. *For the same distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ and hypotheses h_1 and h_2 as in Proposition 4.1, the equality $\mathcal{L}(\mathcal{D}_\lambda, (\alpha \mathsf{Q}(1|\cdot)h_1 + (1-\alpha)\mathsf{Q}(2|\cdot)h_2)) = 0$ holds for any $\lambda \in \Delta$ and any $\alpha \in (0, 1)$,*

Proof. For the counterexample of Proposition 4.1, for any $\alpha \in (0, 1)$, the Q-ensemble $f(x) = (\alpha \mathsf{Q}(1|x)h_1(x) + (1-\alpha)\mathsf{Q}(2|x)h_2(x))$ admits no loss with respect to any target distribution $\mathcal{D}_\lambda$:

$$\begin{aligned}\mathcal{L}(\mathcal{D}_\lambda, f) &= (\lambda\,\mathbb{I}(f(a_1) \leq 0) + (1-\lambda)\,\mathbb{I}(-f(a_2) \leq 0)) \\ &= \lambda(\mathbb{I}(\alpha \leq 0) + (1-\lambda)\,\mathbb{I}((1-\alpha) \leq 0)) = 0,\end{aligned}$$

since $\mathsf{Q}(1|a_1) = \mathsf{Q}(2|a_2) = 1$ and $\mathsf{Q}(2|a_1) = \mathsf{Q}(1|a_2) = 0$.

This concludes the proof of Proposition 4.2. □

B.3 Proof of Lemma 4.3

Lemma 4.3. *For any $k \in [p]$, the following upper bound holds when Φ is the exponential or the logistic function:*

$$F(\alpha_{t-1} + \eta \mathbf{e}_{k,r}) \leq \max_{l \in [p]} \frac{Z_{t,l}}{m_l} \big[(1 - \epsilon_{t,l,k,r})e^{-\eta} + \epsilon_{t,l,k,r} e^{\eta}\big],$$

where $\epsilon_{t,l,k,r} = \frac{1}{2}\big[1 - \mathbb{E}_{i \sim \mathsf{D}_t(l,\cdot)}[y_{l,i}\mathsf{Q}(k|x_{l,i})h_{k,r}(x_{l,i})]\big]$.

Proof. In the special case where $\Phi = \exp$, we have:

$$\begin{aligned}
&\Phi(-y_{l,i} f_{t-1}(x_{l,i}) - \eta\, y_{l,i}\mathsf{Q}(k|x_{l,i})h_{k,r}(x_{l,i})) \\
&\leq \frac{1 + y_{l,i}\mathsf{Q}(k|x_{l,i})h_{k,r}(x_{l,i})}{2} e^{-y_{l,i} f_{t-1}(x_{l,i})} e^{-\eta} + \frac{1 - y_{l,i}\mathsf{Q}(k|x_{l,i})h_{k,r}(x_{l,i})}{2} e^{-y_{l,i} f_{t-1}(x_{l,i})} e^{\eta} \\
&= \frac{1 + y_{l,i}\mathsf{Q}(k|x_{l,i})h_{k,r}(x_{l,i})}{2} \mathsf{D}_t(l,i) Z_{t,l} e^{-\eta} + \frac{1 - y_{l,i}\mathsf{Q}(k|x_{l,i})h_{k,r}(x_{l,i})}{2} \mathsf{D}_t(l,i) Z_{t,l} e^{\eta}.
\end{aligned}$$

Thus, we have:

$$\begin{aligned}
F(\alpha_{t-1} + \eta \mathbf{e}_{k,r}) &= \max_{l \in [p]} \frac{1}{m_l} \sum_{i=1}^{m_l} \Phi(-y_{l,i} f_{t-1}(x_{l,i}) - \eta\, y_{l,i}\mathsf{Q}(k|x_{l,i})h_{k,r}(x_{l,i})) \\
&\leq \max_{l \in [p]} \frac{Z_{t,l}}{m_l} \sum_{i=1}^{m_l} \frac{1}{m_l} \sum_{i=1}^{m_l} \frac{1 + y_{l,i}\mathsf{Q}(k|x_{l,i})h_{k,r}(x_{l,i})}{2} \mathsf{D}_t(l,i) e^{-\eta} \\
&\quad + \frac{1 - y_{l,i}\mathsf{Q}(k|x_{l,i})h_{k,r}(x_{l,i})}{2} \mathsf{D}_t(l,i) e^{\eta} \\
&= \max_{l \in [p]} \frac{Z_{t,l}}{m_l} \big[(1 - \epsilon_{t,l,k,r})e^{-\eta} + \epsilon_{t,l,k,r} e^{\eta}\big].
\end{aligned}$$

The proof is similar in the case of the logistic function. □

B.4 OTHER VARIANTS OF MULTIBOOST

Other variants of MULTIBOOST. As already mentioned, instead of the maximum, the *softmax* function $(x_1, \ldots, x_k) \mapsto \frac{1}{\mu} \log(\sum_{k=1}^{p} e^{\mu x_k})$ can be used in the definition of the algorithm, modulo an approximation that can be controlled via the parameter $\mu > 0$. Using the *softmax* not only

leads to a differentiable objective, but also makes the algorithm focus on several top most difficult domains instead of the single most difficult one, thereby offering a useful trade-off in some applications. Another variant of the algorithm with also a differentiable objective function consists of simply upper-bounding the maximum by a sum:

$$F_{\text{sum}}(\alpha) = \sum_{k=1}^{p} \frac{1}{m_k} \sum_{i=1}^{m_k} \Phi\left(-y_{k,i} \sum_{l=1}^{p} \sum_{j=1}^{N_l} \alpha_{l,j} \mathsf{Q}(l|x_{k,i}) h_{l,j}(x_{k,i})\right). \tag{B.1}$$

In the following, we present convergence guarantees for the F_{sum} objective. A similar guarantee with the same proof holds for the softmax variant of MULTIBOOST.

Theorem B.1. *Assume that Φ is twice differentiable and $\Phi''(u) \geq 0$ for all $u \in \mathbb{R}$. Let $F = F_{sum}$, then, projected coordinate descent applied to $F(\boldsymbol{\alpha})$ converges to the optimal solution $\boldsymbol{\alpha}^*$ of $\min_{\alpha \geq 0} F(\boldsymbol{\alpha})$. If further Φ is strongly convex on the path of the iterates α_t, then there exist $\tau > 0$ and $\gamma > 0$ such that for all $t > \tau$:*

$$F(\boldsymbol{\alpha}_{t+1}) - F(\boldsymbol{\alpha}^*) \leq \left(1 - \tfrac{1}{\gamma}\right)(F(\boldsymbol{\alpha}_t) - F(\boldsymbol{\alpha}^*)). \tag{B.2}$$

Proof. We show that F_{sum} can be represented as $F_{\text{sum}}(\boldsymbol{\alpha}) = G(\mathbf{H}\boldsymbol{\alpha})$, such that $\nabla^2 G(\mathbf{H}\boldsymbol{\alpha})$ is positive definite for all $\boldsymbol{\alpha}$ and apply Theorem 2.1 in [Luo and Tseng 1992] to obtain the convergence guarantees. Let $\mathbf{H}$ be the matrix whose row indexes are $\{(k,i)\colon k \in [p], i \in [m_k]\}$ and whose column indexes are $\{(l,j)\colon l \in [p], j \in [N_k]\}$. Define matrix $\mathbf{H}$ by $\mathbf{H}_{(k,i),(l,j)} = -y_{k,i}\mathsf{Q}(l|x_{k,i})h_{l,j}(x_{k,i})$. Let $\mathbf{e}_{(k,i)}$ be the (k,i)-th unit vector, then for any $\boldsymbol{\alpha}$:

$$\mathbf{e}_{(k,i)}^{\top} \mathbf{H}\boldsymbol{\alpha} = -y_{k,i} \sum_{l=1}^{p} \sum_{j=1}^{N_l} \alpha_{l,j} \mathsf{Q}(l|x_{k,i}) h_{l,j}(x_{k,i}). \tag{B.3}$$

Define the function G as follows for all $\mathbf{u}$:

$$G(\mathbf{u}) = \sum_{k=1}^{p} \frac{1}{m_k} \sum_{i=1}^{m_k} \Phi\big(-\mathbf{e}_{(k,i)}^{\top}\mathbf{u}\big). \tag{B.4}$$

By definition, we have $F_{\text{sum}}(\boldsymbol{\alpha}) = G(\mathbf{H}\boldsymbol{\alpha})$. Moreover, G is twice differentiable and $\nabla^2 G(\mathbf{u})$ is a diagonal matrix with diagonal entries $\frac{1}{m_k}\Phi''(-\mathbf{e}_{(k,i)}^{\top}\mathbf{u}) \geq 0$. Thus, $\nabla^2 G(\mathbf{u})$ is positive definite for all $\alpha \geq 0$. Thus, Theorem 2.1 in [Luo and Tseng 1992] holds for the optimization problem

$$\min_{\boldsymbol{\alpha}\geq 0} G(\mathbf{H}\boldsymbol{\alpha}), \tag{B.5}$$

which guarantees the convergence rate of the coordinate descent for F_{sum}. If further F is is strongly convex over the sequence of α_ts, then, by [Luo and Tseng 1992][page 26], the inequality:

$$F(\boldsymbol{\alpha}_{t+1}) - F(\boldsymbol{\alpha}^*) \leq \Big(1 - \tfrac{1}{\gamma}\Big)(F(\boldsymbol{\alpha}_t) - F(\boldsymbol{\alpha}^*))$$

holds for the projected coordinate method based on the best direction at each round, as with the Gauss-Southwell method. □

Note, that the proof can be extended straightforwardly to a regularized F_{sum} objective, simply by considering $F_{\text{sum}}(\boldsymbol{\alpha}) = G(\mathbf{H}\boldsymbol{\alpha}) + \beta^{\top}\boldsymbol{\alpha}$ in the proof for some $\beta \geq 0$.

B.5 Proofs of Theorem 4.4 and Theorem B.2

Theorem 4.4. *Fix $\rho > 0$. Then, for any $\delta > 0$, with probability at least $1 - \delta$ over the draw of a sample $S = (S_1, \ldots, S_p) \sim \mathcal{D}_1^{m_1} \otimes \cdots \otimes \mathcal{D}_p^{m_p}$, the following inequality holds for all ensemble functions $f = \sum_{t=1}^{T} \alpha_t \mathsf{Q}(k_t|\cdot)h_t \in \mathcal{F}$ and all $\lambda \in \Delta$:*

$$\mathcal{L}(\mathcal{D}_\lambda, f) \leq \mathcal{L}_\rho(\overline{\mathcal{D}}_\lambda, f) + \sum_{k=1}^{p} \lambda_k \left[\frac{2}{\rho} \max_{l \in [p]} \mathfrak{R}_{m_k}(\mathcal{H}_l) + \sqrt{\frac{\log \frac{p}{\delta}}{2m_k}} \right]. \tag{4.5}$$

Proof. Fix $\lambda \in \Delta$ and $\delta > 0$. For any $k \in [p]$, by the standard Rademacher complexity margin bound for $\mathcal{F}$ [Mohri et al. 2018b][Theorem 5.8], with probability at least $1 - \delta$, the following inequality holds for all $f \in \mathcal{F}$:

$$\mathcal{L}(\mathcal{D}_k, f) \leq \mathcal{L}_\rho(\widehat{\mathcal{D}}_k, f) + \frac{2}{\rho}\mathbb{R}_{m_k}(\mathcal{F}) + \sqrt{\frac{\log \frac{1}{\delta}}{2m_k}}.$$

By the union bound, the following inequalities hold simultaneously for all $k \in [p]$:

$$\mathcal{L}(\mathcal{D}_k, f) \leq \mathcal{L}_\rho(\widehat{\mathcal{D}}_k, f) + \frac{2}{\rho}\mathbb{R}_{m_k}(\mathcal{F}) + \sqrt{\frac{\log \frac{p}{\delta}}{2m_k}}.$$

Multiplying each by λ_k and summing them up yields:

$$\mathcal{L}(\mathcal{D}_\lambda, f) \leq \mathcal{L}_\rho(\overline{\mathcal{D}}_\lambda, f) + \sum_{k=1}^{p} \lambda_k \left[\frac{2}{\rho}\mathbb{R}_{m_k}(\mathcal{F}) + \sqrt{\frac{\log \frac{p}{\delta}}{2m_k}} \right].$$

Since the Rademacher of a family coincides with that of its convex hull [Mohri et al. 2018b], we have $\mathbb{R}_{m_k}(\mathcal{F}) = \mathbb{R}_{m_k}(\bigcup_{k=1}^{p} \mathcal{G}_k) \leq \max_{l \in [p]} \mathbb{R}_{m_k}(\mathcal{G}_l)$. We will show that the following inequality holds for any $k \in p$: $\mathbb{R}_{m_k}(\mathcal{G}_l) \leq 2\mathbb{R}_{m_k}(\mathcal{H}_l)$. Note that we can write for any $h \in \mathcal{H}_k$: $\mathsf{Q}(l|\cdot)h = \frac{1}{4}[(\mathsf{Q}(l|\cdot) + h)^2 - (\mathsf{Q}(l|\cdot) - h)^2]$. Thus, since the Rademacher complexity of a sum can be bounded by the sum of the Rademacher complexities, we have:

$$\mathbb{R}_{m_k}(\mathcal{G}_l) \leq \frac{1}{4}\mathbb{R}_{m_k}\big(\{(\mathsf{Q}(l|\cdot) + h)^2 \colon h \in \mathcal{H}_l\}\big) + \frac{1}{4}\mathbb{R}_{m_k}\big(\{(\mathsf{Q}(l|\cdot) - h)^2 \colon h \in \mathcal{H}_l\}\big).$$

Now, functions $\mathsf{Q}(l|\cdot) + h$ and $\mathsf{Q}(l|\cdot) - h$ both take values in $[-1, 2]$ and the function $x \mapsto \frac{1}{4}x^2$ is 1-Lipschitz on $[-1, 2]$ since the absolute value of its derivative $|x|/2$ reaches it maximum at $x = 2$. Thus, by Talagrand's contraction lemma [Ledoux and Talagrand 1991b], we have

$$\mathbb{R}_{m_k}(\mathcal{G}_l) \leq \mathbb{R}_{m_k}(\{(\mathsf{Q}(l|\cdot) + h) \colon h \in \mathcal{H}_l\}) + \mathbb{R}_{m_k}(\{(\mathsf{Q}(l|\cdot) - h) \colon h \in \mathcal{H}_l\}).$$

Now, these Rademacher complexities can be straightforwardly analyzed as follows:

$$\begin{aligned}
\mathbb{R}_{m_k}(\{(\mathsf{Q}(l|\cdot) + h) \colon h \in \mathcal{H}_l\}) &= \mathop{\mathbb{E}}_{\substack{S \sim \mathcal{D}^m \\ \boldsymbol{\sigma}}} \left[\sup_{h \in \mathcal{H}_l} \sum_{i=1}^{m_k} \sigma_i [h(x_i) + \mathsf{Q}(l|x_i)] \right] \\
&= \mathop{\mathbb{E}}_{\substack{S \sim \mathcal{D}^m \\ \boldsymbol{\sigma}}} \left[\sup_{h \in \mathcal{H}_l} \sum_{i=1}^{m_k} \sigma_i h(x_i) \right] + \mathop{\mathbb{E}}_{\substack{S \sim \mathcal{D}^m \\ \boldsymbol{\sigma}}} \left[\sum_{i=1}^{m_k} \sigma_i \mathsf{Q}(l|x_i) \right] \\
&= \mathop{\mathbb{E}}_{\substack{S \sim \mathcal{D}^m \\ \boldsymbol{\sigma}}} \left[\sup_{h \in \mathcal{H}_l} \sum_{i=1}^{m_k} \sigma_i h(x_i) \right] + \mathop{\mathbb{E}}_{S \sim \mathcal{D}^m} \left[\sum_{i=1}^{m_k} \mathop{\mathbb{E}}_{\boldsymbol{\sigma}}[\sigma_i] \mathsf{Q}(l|x_i) \right] \\
&= \mathop{\mathbb{E}}_{\substack{S \sim \mathcal{D}^m \\ \boldsymbol{\sigma}}} \left[\sup_{h \in \mathcal{H}_l} \sum_{i=1}^{m_k} \sigma_i h(x_i) \right] = \mathbb{R}_{m_k}(\mathcal{H}_l).
\end{aligned}$$

Similarly, we have $\mathbb{R}_{m_k}(\{(\mathsf{Q}(l|\cdot) - h) \colon h \in \mathcal{H}_l\}) = \mathbb{R}_{m_k}(\mathcal{H}_l)$. This completes the proof. □

Theorem B.2. *For any $\delta > 0$, with probability at least $1 - \delta$ over the draw of a sample $S = (S_1, \ldots, S_p) \sim \mathcal{D}_1^{m_1} \otimes \cdots \otimes \mathcal{D}_p^{m_p}$, the following inequality holds for all ensemble functions $f = \sum_{t=1}^{T} \alpha_t \mathsf{Q}(k_t|\cdot) h_t \in \mathcal{F}$ and all $\rho \in (0, 1)$ and $\lambda \in \Delta$:*

$$\mathcal{L}(\mathcal{D}_\lambda, f) \leq \mathcal{L}_\rho(\overline{\mathcal{D}}_\lambda, f) + \sum_{k=1}^{p} \lambda_k \left[\frac{2}{\rho} \max_{l \in [p]} \mathfrak{R}_{m_k}(\mathcal{H}_l) + \sqrt{\frac{\log \log_2 \frac{2}{\rho}}{m_k}} + \sqrt{\frac{\log \frac{p}{\delta}}{2m_k}} \right]. \tag{B.6}$$

Proof. By the uniform margin bound [Mohri et al. 2018b, Theorem 5.9], for any $k \in [p]$, with

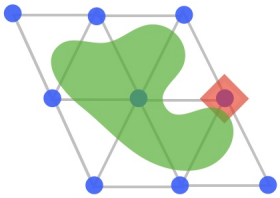


Figure B.1: Illustration of the set $\overline{\Lambda}$, vertices of simplices (in blue). The area in green represents the set Λ. The small area in pink shows an ϵ-ball in l_1-distance (and hence a lozenge) around a vertex.

probability at least $1-\delta$ the following inequality holds for all $f \in \mathcal{F}$ and $\rho \in (0,1]$:

$$\mathcal{L}(\mathcal{D}_k, f) \leq \mathcal{L}_\rho(\widehat{\mathcal{D}}_k, f) + \frac{2}{\rho}\mathbb{R}_{m_k}(\mathcal{F}) + \sqrt{\frac{\log \log_2 \frac{2}{\rho}}{m_k}} + \sqrt{\frac{\log \frac{1}{\delta}}{2m_k}}.$$

The rest of the proof is similar to that of Theorem 4.4. □

B.6 FINER MARGIN-BASED LEARNING GUARANTEES

In this section, we give finer margin-based generalization bounds for the family of ensemble $\mathcal{F}$. These learning bounds are particularly more relevant in the case where Λ is a strict subset of the simplex Δ. Λ may be in fact a much smaller set, motivated by prior knowledge about the task and thus possible target mixtures. In some instances, it may even be a finite subset, which corresponds to only a finite set of mixtures.

For any family of real-valued functions $\mathcal{G}$, define the *weighted Rademacher complexity of* $\mathcal{G}$ for the vector of samples $S_k = (z_{k,1}, \ldots, z_{k,m_k})$ of sizes $\mathbf{m} = (m_1, \ldots, m_p)$ as follows:

$$\mathbb{R}_{\mathbf{m}}(\mathcal{G}, \lambda) = \mathop{\mathbb{E}}_{S_k \sim \mathcal{D}_k^{m_k}, \boldsymbol{\sigma}} \left[\sup_{g \in \mathcal{G}} \sum_{k=1}^{p} \frac{\lambda_k}{m_k} \sum_{i=1}^{m_k} \sigma_{k,i}\, g(z_{k,i}) \right]. \tag{B.7}$$

Fix $\lambda \in \Lambda$ and define $\Psi(S)$ for the vector of training samples $S = (S_1, \ldots, S_p)$ as follows:

$$\Psi(S) = \sup_{h \in \mathcal{G}} \left\{ \mathop{\mathbb{E}}_{z \sim \mathcal{D}_\lambda} [g(z)] - \mathop{\mathbb{E}}_{z \sim \overline{\mathcal{D}}_\lambda} [g(z)] \right\},$$

where $\overline{\mathcal{D}}_\lambda = \sum_{k=1}^{p} \lambda_k \widehat{\mathcal{D}}_k$, with $\widehat{\mathcal{D}}_k$ the empirical distribution associated with the sample S_k. Assume that functions in $\mathcal{G}$ take values in $[0, 1]$. For any vector of samples S' differing from S only by point $z'_{k,i}$ in S'_k and $z_{k,i}$ in S_k, we have

$$\begin{aligned}
\Psi(S') - \Psi(S) &\leq \sup_{g \in \mathcal{G}} \left\{ \left\{ \mathop{\mathbb{E}}_{z \sim \mathcal{D}_\lambda} [g(z)] - \mathop{\mathbb{E}}_{z \sim \overline{\mathcal{D}}'_\lambda} [g(z)] \right\} - \left\{ \mathop{\mathbb{E}}_{z \sim \mathcal{D}_\lambda} [g(z)] - \mathop{\mathbb{E}}_{z \sim \overline{\mathcal{D}}_\lambda} [g(z)] \right\} \right\} \\
&= \sup_{g \in \mathcal{G}} \left\{ \mathop{\mathbb{E}}_{z \sim \overline{\mathcal{D}}_\lambda} [g(z)] - \mathop{\mathbb{E}}_{z \sim \overline{\mathcal{D}}'_\lambda} [g(z)] \right\} = \sup_{g \in \mathcal{G}} \frac{\lambda_k}{m_k} \left[g(z_{k,i}) - g(z'_{k,i}) \right] \leq \frac{\lambda_k}{m_k}.
\end{aligned}$$

Furthermore, as with the standard Rademacher complexity [Mohri et al. 2018b], the expectation can be upper bounded in terms of the weighted Rademacher complexity:

$$\mathop{\mathbb{E}}_{S_k \sim \mathcal{D}_k^{m_k}} \left[\sup_{g \in \mathcal{G}} \mathop{\mathbb{E}}_{z \sim \mathcal{D}_\lambda} [g(z)] - \mathop{\mathbb{E}}_{z \sim \overline{\mathcal{D}}_\lambda} [g(z)] \right] \leq 2\mathbb{R}_{\mathbf{m}}(\mathcal{G}, \lambda).$$

Thus, by McDiarmid's inequality, for any $\delta > 0$, with probability at least $1 - \delta$,

$$\mathop{\mathbb{E}}_{z \sim \mathcal{D}_\lambda} [g(z)] \leq \mathop{\mathbb{E}}_{z \sim \overline{\mathcal{D}}_\lambda} [g(z)] + 2\mathbb{R}_{\mathbf{m}}(\mathcal{G}, \lambda) + \sqrt{\sum_{k=1}^{p} \frac{\lambda_k^2}{2m_k} \log \frac{1}{\delta}}.$$

Let $\overline{\Lambda}$ be the set of vertices of a subsimplicial cover of Λ, that is a decomposition of a cover of Λ into subsimplices. When the subsimples are formed by vertices that are ϵ-close in ℓ_1-distance, then $\overline{\Lambda}$ is an ϵ-cover of Λ for the ℓ_1-distance. Figure B.1 illustrates the sets λ and $\overline{\Lambda}$. By the union

bound, with probability at least $1 - \delta$, the following holds for all $\lambda \in \overline{\Lambda}$:

$$\mathop{\mathbb{E}}_{z \sim \mathcal{D}_\lambda}[g(z)] \leq \mathop{\mathbb{E}}_{z \sim \overline{\mathcal{D}}_\lambda}[g(z)] + 2\mathbb{R}_{\mathbf{m}}(\mathcal{G}, \lambda) + \sqrt{\sum_{k=1}^{p} \frac{\lambda_k^2}{2m_k} \log \frac{|\overline{\Lambda}|}{\delta}}.$$

Now, fix $\rho > 0$. Let $\mathcal{H}$ be a hypothesis set of real-valued functions and let ϕ_ρ denote the ρ-ramp loss. Let $\mathcal{G}$ be the family of ρ-ramp losses of functions in $\mathcal{H}$:

$$\mathcal{G} = \{z = (x, y) \mapsto \phi_\rho(yh(x)) \colon h \in \mathcal{H}\}.$$

Then, proceeding as with the derivation of margin-based Rademacher complexity bounds in the standard case and using the $\frac{1}{\rho}$-Lipschitzness of the ρ-ramp loss [Mohri et al. 2018b], we obtained that, with probability at least $1 - \delta$, the following holds for all $\lambda \in \overline{\Lambda}$:

$$\mathcal{L}(\mathcal{D}_\lambda, h) \leq \mathcal{L}_\rho(\overline{\mathcal{D}}_\lambda, h) + \frac{2}{\rho}\mathbb{R}_{\mathbf{m}}(\mathcal{H}, \lambda) + \sqrt{\sum_{k=1}^{p} \frac{\lambda_k^2}{2m_k} \log \frac{|\overline{\Lambda}|}{\delta}}.$$

Now, for any $\lambda, \lambda' \in \Delta$ with $\|\lambda - \lambda'\|_1 \leq \epsilon$, we have:

$$\begin{aligned}
\sum_{k=1}^{p} \frac{\lambda_k'^2}{2m_k} &= \sum_{k=1}^{p} \frac{(\lambda_k' - \lambda_k + \lambda_k)^2}{2m_k} \\
&= \sum_{k=1}^{p} \frac{\lambda_k^2 + 2(\lambda_k' - \lambda_k)\lambda_k + (\lambda_k' - \lambda_k)^2}{2m_k} \\
&= \sum_{k=1}^{p} \frac{\lambda_k^2}{2m_k} + \sum_{k=1}^{p} \frac{2|\lambda_k' - \lambda_k|\lambda_k + (\lambda_k' - \lambda_k)^2}{2m_k} \\
&\leq \sum_{k=1}^{p} \frac{\lambda_k^2}{2m_k} + \epsilon \max_{k \in [p]} \frac{\lambda_k}{m_k} + \frac{\epsilon^2}{2} \max_{k \in [p]} \frac{1}{m_k} && \text{(Hölder's inequality)} \\
&\leq \sum_{k=1}^{p} \frac{\lambda_k^2}{2m_k} + \frac{3\epsilon}{2}.
\end{aligned}$$

Let $\overline{\Lambda}_\epsilon$ denote the family of λs that are ϵ-close to $\overline{\Lambda}$ in ℓ_1-distance, then, for any $\lambda \in \overline{\Lambda}_\epsilon$ we have:

$$\mathcal{L}(\mathcal{D}_\lambda, h) \leq \mathcal{L}_\rho(\overline{\mathcal{D}}_\lambda, h) + \frac{2}{\rho}\mathbb{R}_{\mathbf{m}}(\mathcal{H}, \lambda) + \frac{3\epsilon}{2} + \sqrt{\sum_{k=1}^{p} \frac{\lambda_k^2}{2m_k} \log \frac{|\overline{\Lambda}|}{\delta}}.$$

Also, for any λ in a subsimplex formed by elements of $\overline{\Lambda}$, there exist $\mu = (\mu_1, \ldots, \mu_p)$ and $\beta_1, \ldots, \beta_p$ in $\overline{\Lambda}$ such that $\lambda = \sum_{k=1}^{p} \mu_k \beta_k$. Thus, for any such λ, we have

$$\mathcal{L}(\mathcal{D}_\lambda, h) \leq \mathcal{L}_\rho(\overline{\mathcal{D}}_\lambda, h) + \frac{2}{\rho} \sum_{l=1}^{p} \mu_k \mathbb{R}_{\mathbf{m}}(\mathcal{H}, \beta_l) + \sum_{k=1}^{p} \mu_k \sqrt{\sum_{l=1}^{p} \frac{\beta_{l,k}^2}{2m_k} \log \frac{|\overline{\Lambda}|}{\delta}}.$$

Applying these results to the analysis of the Q-ensembles we are interested yields the following margin-based generalization bounds.

Theorem 4.5. *Fix $\rho > 0$ and $\epsilon > 0$. Then, for any $\delta > 0$, with probability at least $1 - \delta$ over the draw of a sample $S = (S_1, \ldots, S_p) \sim \mathcal{D}_1^{m_1} \otimes \cdots \otimes \mathcal{D}_p^{m_p}$, each of the following inequalities holds:*

1. for all ensemble functions $f = \sum_{t=1}^{T} \alpha_t \mathsf{Q}(k_t|\cdot)h_t \in \mathcal{F}$ *and all* $\lambda \in \overline{\Lambda}_\epsilon$*:*

$$\mathcal{L}(\mathcal{D}_\lambda, h) \leq \mathcal{L}_\rho(\overline{\mathcal{D}}_\lambda, h) + \frac{2}{\rho} \max_{r \in [p]} \mathbb{R}_{\mathbf{m}}(\mathcal{H}_r, \lambda) + \frac{3\epsilon}{2} + \sqrt{\sum_{k=1}^{p} \frac{\lambda_k^2}{2m_k} \log \frac{|\overline{\Lambda}|}{\delta}}. \tag{4.6}$$

2. for all ensemble functions $f = \sum_{t=1}^{T} \alpha_t \mathsf{Q}(k_t|\cdot)h_t \in \mathcal{F}$ *and all* $\lambda = \sum_{k=1}^{p} \mu_k \beta_k \in \operatorname{conv}(\overline{\Lambda})$*:*

$$\mathcal{L}(\mathcal{D}_\lambda, h) \leq \mathcal{L}_\rho(\overline{\mathcal{D}}_\lambda, h) + \frac{2}{\rho} \sum_{l=1}^{p} \mu_k \max_{r \in [p]} \mathbb{R}_{\mathbf{m}}(\mathcal{H}_r, \beta_l) + \sum_{k=1}^{p} \mu_k \sqrt{\sum_{l=1}^{p} \frac{\beta_{l,k}^2}{2m_k} \log \frac{|\overline{\Lambda}|}{\delta}}. \tag{4.7}$$

Note that, for a given $\lambda \in \Lambda$, the most favorable of the two statements of the theorem can be used. Observe also that the second learning bound coincides with that of Theorem 4.4 when $\overline{\Lambda}$ is chosen to be the vertices of the simplex Δ since in that case $|\overline{\Lambda}| = p$, $\operatorname{conv}(\overline{\Lambda}) = \Delta$, and since $\mathbb{R}_{\mathbf{m}}(\mathcal{H}_r, \beta_l) = \mathfrak{R}_{\mathbf{m}}(\mathcal{H}_r, \beta_l)$ then coincides with $\mathfrak{R}_{m_l}(\mathcal{H}_r)$. Choosing the best statement of the theorem therefore always provides a finer guarantee than that of Theorem 4.4.

When Λ is a small set, for example the set of λs ϵ-close to a finite set of discrete elements $\overline{\Lambda}$, then the last term of the learning bound of the first statement can be more favorable that of Theorem 4.4 since $|\overline{\Lambda}|$ can then be in the same order as p or smaller while, by the sub-additivity of the square-root, the following inequality holds: $\sqrt{\sum_{k=1}^{p} \frac{\lambda_k^2}{m_k}} \leq \sum_{k=1}^{p} \sqrt{\frac{\lambda_k^2}{m_k}} = \sum_{k=1}^{p} \lambda_k \sqrt{\frac{1}{m_k}}$.

The theorem suggests a regularization term of the form $\sum_{k=1}^{p} \frac{\lambda_k^2}{m_k}$, especially in the case where Λ is a small subset of the simplex, which can lead to better algorithms in that case.

B.7 FEDMULTIBOOST: RELATED WORK AND EXPERIMENTS

Following MULTIBOOST, we propose a boosting-style approach with the agnostic loss. Boosting in federated learning was first studied by [Hamer et al. 2020]. The authors proposed an efficient

algorithm for minimizing the standard empirical risk and agnostic loss, based on mirror descent. However, their algorithm is optimal only for density estimation [Hamer et al. 2020, Section 3.2] and is sub-optimal for general classification tasks such as in Proposition 4.1. Furthermore, their mirror descent solution is inadequate for the boosting framework in this paper, where a (block) coordinate descent approach for learning sparser solutions is critical. Recently, [Shen et al. 2021] proposed a functional gradient boosting algorithm for federated learning. Their algorithm iteratively determines base classifiers and mixing weights to compute a convex combination in a distributed manner. However, their algorithm minimizes the uniform loss over all samples. In contrast, we propose to minimize the agnostic loss over multiple domains, which is more risk-averse, and seek Q-ensembles which are more adequate than convex combinations in a multiple-source scenario.

Federated learning experiments. We used the *EMNIST* dataset [Caldas et al. 2018; Bonawitz et al. 2019], which contains 32x32 pixel handwritten digits images annotated by users. The images are divided into $p = 2$ sources based on the group of writers that provided the annotation: high school ($k = 1$) and census ($k = 2$). We compared algorithms on two binary classification problems: digits 4 vs. 9 and digits 1 vs. 7. The results are presented in Table B.1. The error bars are obtained from breaking the set of clients into 10 random folds.

We compared FEDMULTIBOOST with three benchmarks, FEDADABOOST-k for $k \in [p]$ and FEDADABOOST-all. The former is a federated version of ADABOOST algorithm trained only on a single source k and the latter is federated ADABOOST trained on both the sources. In the federated ADABOOST versions, at each boosting step we randomly select 20 clients and train weak learners on each of those clients, next, the server selects the weak learner with the smallest weighted error and adds it to the ensemble. For the FEDMULTIBOOST algorithm, we randomly select 20 clients per round. Since the number of clients sampled at each round is small, we run each algorithm for 500 boosting steps.

Table B.1: Test errors for multiple benchmarks in the federated setting.

Algorithm	Error-1	Error-2	Error-Uniform	Error-Agnostic
		EMNIST (4 vs. 9)		
FEDADABOOST-1	0.075 ± 0.008	0.133 ± 0.014	0.104 ± 0.009	0.133 ± 0.014
FEDADABOOST-2	0.095 ± 0.009	0.096 ± 0.014	0.095 ± 0.012	0.096 ± 0.014
FEDADABOOST-all	0.076 ± 0.006	0.125 ± 0.016	0.101 ± 0.011	0.125 ± 0.007
FEDMULTIBOOST	$\mathbf{0.064 \pm 0.013}$	$\mathbf{0.076 \pm 0.008}$	$\mathbf{0.070 \pm 0.009}$	$\mathbf{0.076 \pm 0.016}$
		EMNIST (1 vs. 7)		
FEDADABOOST-1	$\mathbf{0.029 \pm 0.010}$	0.050 ± 0.009	0.039 ± 0.011	0.050 ± 0.009
FEDADABOOST-2	0.062 ± 0.014	$\mathbf{0.030 \pm 0.007}$	0.046 ± 0.014	0.062 ± 0.007
FEDADABOOST-all	0.032 ± 0.007	0.043 ± 0.006	0.037 ± 0.007	0.043 ± 0.014
FEDMULTIBOOST	0.030 ± 0.008	0.035 ± 0.006	$\mathbf{0.032 \pm 0.008}$	$\mathbf{0.035 \pm 0.010}$

As can be seen from Table B.1, FEDMULTIBOOST provides agnostic and uniform errors that are significantly better than the baselines on both the datasets.

B.8 DATASET REFERENCES AND DETAILS

- MNIST: `http://yann.lecun.com/exdb/mnist/`
- SVHN: `http://ufldl.stanford.edu/housenumbers/`
- MNIST-M: `http://yaroslav.ganin.net/`
- SENTIMENT: `https://www.cs.jhu.edu/~mdredze/datasets/sentiment`
- FASHION-MNIST: `https://github.com/zalandoresearch/fashion-mnist`
- ADULT: `https://archive.ics.uci.edu/ml/datasets/adult`

Table B.2: Number of examples per domain for each classification problem.

Problem	Source k=1	Source k=2	Source k=3	Total
Sentiment Analysis	2,000	2,000	2,000	6,000
Digit Classification (1 vs 7)	15,170	26,574	14,728	56,472
Digit Classification (4 vs 9)	13,782	16,235	13,406	43,423
Object Recognition	21,000	21,000	28,000	70,000
Tabular Data (Adult Data)	10,628	14,783	19,811	45,222

B.9 SUPPLEMENTARY PLOTS

This section contains additional plots illustrating the performance of the proposed MULTIBOOST algorithm. We illustrate convergence characteristics, Figure B.4, $\mathsf{Q}(k|\cdot_k)$ functions, Figure B.2, and α-mass distributions over the domains and $\mathsf{Q}(k|\cdot_k)$ functions, Figure B.3.

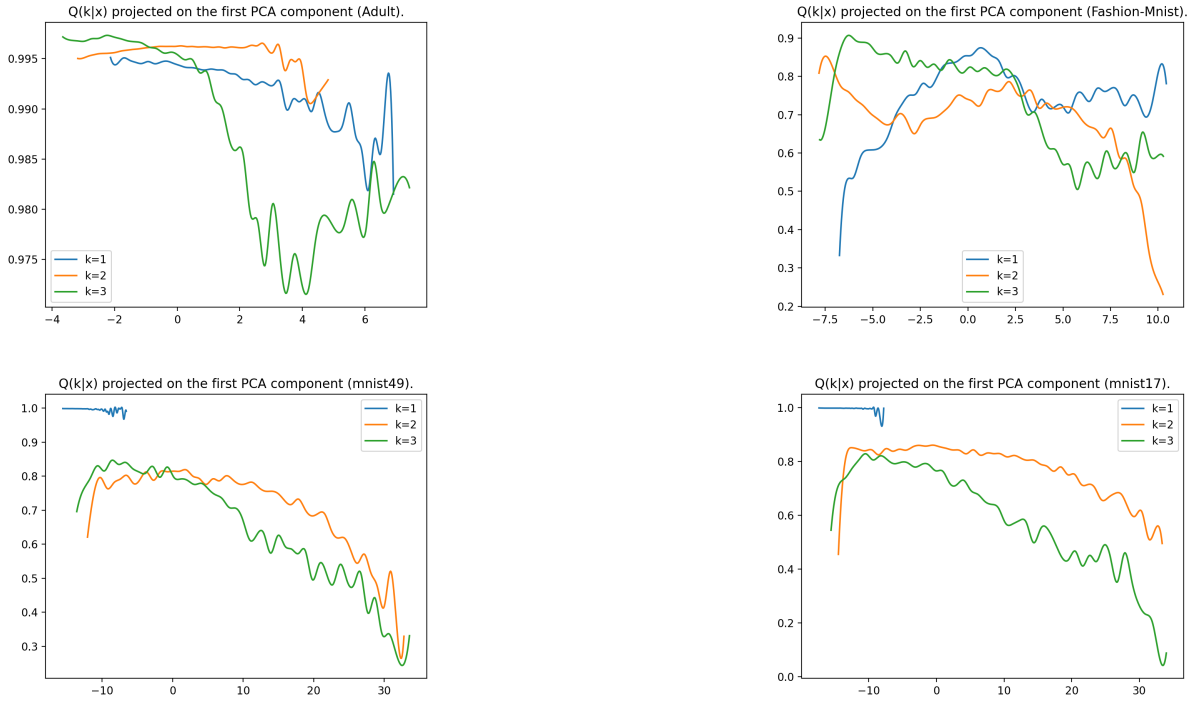


Figure B.2: Mean values of $\mathrm{Q}(k|\cdot_k)$ for the three domains of the data projected on the first principal component of the joint data.
Top, left: *Adult* data. Source $k = 1$ consists of individuals with a university degree (BSc, MS or PhD), source $k = 2$ those with only a High School diploma, and source $k = 3$, none of the above.
Top, right: *Fashion-MNIST* data. Source $k = 1$ consists of `t-shirts, pullovers, trousers`; $k = 2$ consists of `dresses, coats, sandals`; $k = 3$ consists of `shirts, bags, sneakers, ankle boots`.
Bottom, left: Digits (4 vs. 9), where pixel handwritten digits images are taken from sources: *MNIST* ($k = 1$), *SVHN* ($k = 2$), *MNIST-M* ($k = 3$).
Bottom, right: Digits (1 vs. 7), where pixel handwritten digits images are taken from sources: *MNIST* ($k = 1$), *SVHN* ($k = 2$), *MNIST-M* ($k = 3$).
Note that for the two bottom plots domain $k = 1$ is significantly further separated from the other two domains, since the pixels for $k = 1$ are black and white and those for $k = 2, 3$ are grayscale.

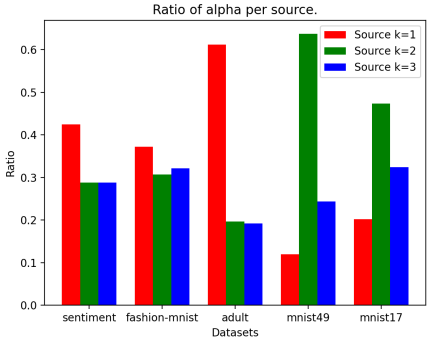


Figure B.3: The ratio of ensemble weights $\alpha_{k,j}$ after training that corresponds to each source $k = 1, 2, 3$. For each dataset and fixed k, the bar corresponds to the ratio of $\sum_{j=1}^{N_k} \alpha_{k,j}$ to $\sum_{k=1}^{p} \sum_{j=1}^{N_k} \alpha_{k,j}$.

Figure B.4: Left: The evolution of surrogate losses Φ for sources $k = 1, 2, 3$ during the training of MULTIBOOST on object recognition problem (*Fashion-MNIST*). Line k corresponds to $\frac{1}{m_k} \sum_{i=1}^{m_k} \Phi(y_i, f_t(x_i))$ as a function of t, where f_t is the MULTIBOOST ensemble of weak learners obtained at round t. Right: The evolution of the step size η_t during MULTIBOOST training on object recognition problem (*Fashion-MNIST*).

C | APPENDIX TO CHAPTER 5.

C.1 LOSSES

Table C.1 lists the losses used for training.

Table C.1: Base loss functions used for experiments.

Base loss function	$\ell(h(x), y)$
Zero-One	$l_1 = \begin{cases} 0, & \text{if } yh(x) \geq 0 \\ 1, & \text{if } yh(x) < 0 \end{cases}$
Hinge	$l_2 = \begin{cases} 0, & \text{if } yh(x) \geq 1 \\ \max(0, 1 - yh(x)), & \text{if } yh(x) > 1 \end{cases}$
Norm q	$l_3 = \|h(x) - y\|_q^q$
Cross-Entropy	$l_5 = \log(1 + \exp(-yh(x)))$

C.2 PROOF OF THEOREMS

C.2.1 PROOF OF THEOREM 5.2

If the loss functions ℓ_k are M_k-Lipschitz and bounded by M, then for any $\epsilon > 0$ and $\delta > 0$, with probability at least $1 - \delta$, the following inequality holds for $\forall h \in \mathcal{H}$ and $\forall \lambda \in \Lambda$:

$$\mathcal{L}_\lambda(h) \leq \widehat{\mathcal{L}}_\lambda(h) + 2\widehat{\mathfrak{R}}_S(\mathcal{G}, \lambda) + M\epsilon + \mathcal{D}_{\mathcal{H}} \sum_{k=1}^{p} \lambda_k M_k \sqrt{\frac{1}{2m} \log \left[\frac{|\Lambda|_\epsilon}{\delta} \right]}, \tag{C.1}$$

where Λ_ϵ is an minimum ϵ-cover of Λ.

Proof. For any $\lambda \in \Lambda$ and sample $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$, let $\Psi(S) = \sup_{h \in \mathcal{H}} \mathcal{L}_\lambda(h) -$

$\widehat{\mathcal{L}}_\lambda(h)$. Let S' be a sample different from S by only one point (x', y'), then

$$\begin{aligned}
\Psi(S') - \Psi(S) &= \sup_{h\in\mathcal{H}} \big[\mathcal{L}_\lambda(h) - \widehat{\mathcal{L}}'_\lambda(h)\big] - \sup_{h\in\mathcal{H}} \big[\mathcal{L}_\lambda(h) - \widehat{\mathcal{L}}_\lambda(h)\big] \\
&\leq \sup_{h\in\mathcal{H}} \big[\mathcal{L}_\lambda(h) - \widehat{\mathcal{L}}'_\lambda(h) - \mathcal{L}_\lambda(h) + \widehat{\mathcal{L}}_\lambda(h)\big] \\
&= \sup_{h\in\mathcal{H}} \big[\widehat{\mathcal{L}}_\lambda(h) - \widehat{\mathcal{L}}'_\lambda(h)\big] \\
&= \sup_{h\in\mathcal{H}} \big[\sum_{k=1}^{p} \lambda_k \frac{1}{m} \sum_{i=1}^{m} \ell_k(h(x'_i), y'_i) - \sum_{k=1}^{p} \lambda_k \frac{1}{m} \sum_{i=1}^{m} \ell_k(h(x_i), y_i)\big] \\
&= \frac{1}{m} \sup_{h\in\mathcal{H}} \sum_{k=1}^{p} \lambda_k \big(\ell_k(h(x'_i, y'_i)) - \ell_k(h(x_i), y_i)\big) \\
&\leq \frac{1}{m} \sup_{h\in\mathcal{H}} \sum_{k=1}^{p} \lambda_k M_k ||[h(x'_i), y'_i] - [h(x_i), y_i]|| \\
&\leq \frac{1}{m} \sup_{h\in\mathcal{H}} \sum_{k=1}^{p} \lambda_k M_k \mathcal{D}_\mathcal{H} \\
&\leq \frac{\mathcal{D}_\mathcal{H}}{m} \sum_{k=1}^{p} \lambda_k M_k.
\end{aligned}$$

By McDiarmid's inequality, for any $\delta > 0$ with probability at least $1 - \delta$ for any $h \in \mathcal{H}$:

$$\mathcal{L}_\lambda(h) \leq \widehat{\mathcal{L}}_\lambda(h) + \mathbb{E}\left[\sup_{h\in\mathcal{H}} \mathcal{L}_\lambda(h) - \widehat{\mathcal{L}}_\lambda(h)\right] + \mathcal{D}_\mathcal{H} \sum_{k=1}^{p} \lambda_k M_k \sqrt{\frac{1}{2m} \log \frac{1}{\delta}}.$$

The inequality above holds for a particular value of λ fixed in advance. Using the union over the minimum ϵ-cover Λ_ϵ, with probability at least $1 - \delta$ for any $\lambda \in \Lambda_\epsilon$ and $h \in \mathcal{H}$:

$$\mathcal{L}_\lambda(h) \leq \widehat{\mathcal{L}}_\lambda(h) + \mathbb{E}\left[\sup_{h\in\mathcal{H}} \mathcal{L}_\lambda(h) - \widehat{\mathcal{L}}_\lambda(h)\right] + \mathcal{D}_\mathcal{H} \sum_{k=1}^{p} \lambda_k M_k \sqrt{\frac{1}{2m} \log \frac{|\Lambda_\epsilon|}{\delta}}.$$

Using the definition of the minimum ϵ-cover, and also recalling that $\mathbb{E}\left[\sup_{h\in\mathcal{H}} \mathcal{L}_\lambda(h) - \widehat{\mathcal{L}}_\lambda(h)\right] \leq$

$\widehat{\mathfrak{R}}_S(\mathcal{G}, \lambda)$, with probability at least $1 - \delta$ for any $\lambda \in \Lambda$ and $h \in \mathcal{H}$:

$$\begin{aligned}\mathcal{L}_\lambda(h) &\leq \widehat{\mathcal{L}}_\lambda(h) + \mathbb{E}\left[\sup_{h \in \mathcal{H}} \mathcal{L}_\lambda(h) - \widehat{\mathcal{L}}_\lambda(h)\right] + M\epsilon + \mathcal{D}_\mathcal{H} \sum_{k=1}^{p} \lambda_k M_k \sqrt{\frac{1}{2m} \log \frac{|\Lambda_\epsilon|}{\delta}} \\ &\leq \widehat{\mathfrak{R}}_S(\mathcal{G}, \lambda) + M\epsilon + \mathcal{D}_\mathcal{H} \sum_{k=1}^{p} \lambda_k M_k \sqrt{\frac{1}{2m} \log \frac{|\Lambda_\epsilon|}{\delta}}.\end{aligned}$$

□

C.2.2 PROOF OF THEOREM 5.4

For $L(\mathbf{w}, \boldsymbol{\lambda})$ convex in the first argument, assume $\forall \mathbf{w} \in \mathcal{W}, \forall \boldsymbol{\lambda} \in \Lambda : \|\mathbf{w}\|_2 \leq \mathcal{D}_\mathcal{W}, \|\boldsymbol{\lambda}\|_2 \leq \mathcal{D}_\Lambda, \|\nabla_\mathbf{w} L(\mathbf{w}, \boldsymbol{\lambda})\|_2 \leq \mathcal{G}_\mathbf{w}, \|\nabla_{\boldsymbol{\lambda}} L(\mathbf{w}, \boldsymbol{\lambda})\|_2 \leq \mathcal{G}_{\boldsymbol{\lambda}}$. Let the variance of unbiased stochastic gradients be bounded by $\sigma_\mathbf{w}^2$ and $\sigma_{\boldsymbol{\lambda}}^2$ respectively. If the step sizes are $\gamma_\mathbf{w} = \frac{1}{\sqrt{T}} \frac{2\mathcal{D}_\mathcal{W}}{\sqrt{\sigma_\mathbf{w}^2 + \mathcal{G}_\mathbf{w}^2}}$ and $\gamma_{\boldsymbol{\lambda}} = \frac{1}{\sqrt{T}} \frac{2\mathcal{D}_\Lambda}{\sqrt{\sigma_{\boldsymbol{\lambda}}^2 + \mathcal{G}_{\boldsymbol{\lambda}}^2}}$, then the following convergence guarantees apply for the ALMO algorithm:

$$\mathbb{E}\left[\max_{\boldsymbol{\lambda} \in \Lambda} L(\mathbf{w}_T, \lambda) - \min_{\mathbf{w} \in \mathcal{W}} \max_{\boldsymbol{\lambda} \in \Lambda} L(\mathbf{w}, \boldsymbol{\lambda})\right] \leq \frac{1}{\sqrt{T}} \left(3\mathcal{D}_\mathcal{W} \sqrt{\sigma_\mathbf{w}^2 + \mathcal{G}_\mathbf{w}^2} + 3\mathcal{D}_\Lambda \sqrt{\sigma_{\boldsymbol{\lambda}}^2 + \mathcal{G}_{\boldsymbol{\lambda}}^2}\right). \quad \text{(C.2)}$$

Proof.

$$\begin{aligned}\max_{\boldsymbol{\lambda} \in \Lambda} L(\mathbf{w}_T, \lambda) - \min_{\mathbf{w} \in \mathcal{W}} \max_{\boldsymbol{\lambda} \in \Lambda} L(\mathbf{w}, \boldsymbol{\lambda}) &= \max_{\boldsymbol{\lambda} \in \Lambda} L(\mathbf{w}_T, \lambda) - \max_{\boldsymbol{\lambda} \in \Lambda} \min_{\mathbf{w} \in \mathcal{W}} L(\mathbf{w}, \boldsymbol{\lambda}) \\ &\leq \max_{\boldsymbol{\lambda} \in \Lambda} \left[L(\mathbf{w}_T, \lambda) - \min_{\mathbf{w} \in \mathcal{W}} L(\mathbf{w}, \boldsymbol{\lambda}_T)\right] \\ &= \max_{\boldsymbol{\lambda} \in \Lambda, \mathbf{w} \in \mathcal{W}} \left[L(\mathbf{w}_T, \lambda) - L(\mathbf{w}, \boldsymbol{\lambda}_T)\right] \\ &\leq \frac{1}{T} \max_{\boldsymbol{\lambda} \in \Lambda, \mathbf{w} \in \mathcal{W}} \left[\sum_{t=1}^{T} L(\mathbf{w}_t, \lambda) - L(\mathbf{w}, \boldsymbol{\lambda}_t)\right]\end{aligned}$$

The last inequality follows from the convexity in $\mathbf{w}$. Given the resulting inequality above, the next step is to bound the difference $L(\mathbf{w}_t, \lambda) - L(\mathbf{w}, \boldsymbol{\lambda}_t)$ for each $t \in [1, T]$, using the standard techniques in convex optimization proofs.

$$\begin{aligned}
L(\mathbf{w}_t, \boldsymbol{\lambda}) - L(\mathbf{w}, \boldsymbol{\lambda}_t) &= L(\mathbf{w}_t, \boldsymbol{\lambda}) - L(\mathbf{w}_t, \boldsymbol{\lambda}_t) + L(\mathbf{w}_t, \boldsymbol{\lambda}_t) - L(\mathbf{w}, \boldsymbol{\lambda}_t) \\
&\leq (\boldsymbol{\lambda} - \boldsymbol{\lambda}_t)\nabla_{\boldsymbol{\lambda}} L(\mathbf{w}_t, \boldsymbol{\lambda}_t) + (\mathbf{w}_t - \mathbf{w})\nabla_{\mathbf{w}} L(\mathbf{w}_t, \boldsymbol{\lambda}_t) \\
&\leq (\boldsymbol{\lambda} - \boldsymbol{\lambda}_t)\delta_{\boldsymbol{\lambda}} L(\mathbf{w}_t, \boldsymbol{\lambda}_t) + (\mathbf{w}_t - \mathbf{w})\delta_{\mathbf{w}} L(\mathbf{w}_t, \boldsymbol{\lambda}_t) \\
&+ (\boldsymbol{\lambda} - \boldsymbol{\lambda}_t)(\nabla_{\lambda} L(\mathbf{w}_t, \boldsymbol{\lambda}_t) - \delta_{\lambda} L(\mathbf{w}_t, \boldsymbol{\lambda}_t)) \\
&+ (\mathbf{w}_t - \mathbf{w})(\nabla_{\mathbf{w}} L(\mathbf{w}_t, \boldsymbol{\lambda}_t) - \delta_{\mathbf{w}} L(\mathbf{w}_t, \boldsymbol{\lambda}_t))
\end{aligned}$$

Given the bound on $L(\mathbf{w}_t, \boldsymbol{\lambda}) - L(\mathbf{w}, \boldsymbol{\lambda}_t)$ we can obtain the following series of inequalities:

$$\begin{aligned}
&\max_{\boldsymbol{\lambda} \in \Lambda, \mathbf{w} \in \mathcal{W}} \left[\sum_{t=1}^{T} L(\mathbf{w}_t, \lambda) - L(\mathbf{w}, \boldsymbol{\lambda}_t) \right] \\
&\leq \underbrace{\max_{\boldsymbol{\lambda} \in \Lambda, \mathbf{w} \in \mathcal{W}} \sum_{t=1}^{T} (\boldsymbol{\lambda} - \boldsymbol{\lambda}_t)\delta_{\boldsymbol{\lambda}} L(\mathbf{w}_t, \boldsymbol{\lambda}_t) + (\mathbf{w}_t - \mathbf{w})\delta_{\mathbf{w}} L(\mathbf{w}_t, \boldsymbol{\lambda}_t)}_{A} \\
&+ \underbrace{\max_{\boldsymbol{\lambda} \in \Lambda, \mathbf{w} \in \mathcal{W}} \sum_{t=1}^{T} \boldsymbol{\lambda}(\nabla_{\lambda} L(\mathbf{w}_t, \boldsymbol{\lambda}_t) - \delta_{\lambda} L(\mathbf{w}_t, \boldsymbol{\lambda}_t)) - \mathbf{w}(\nabla_{\mathbf{w}} L(\mathbf{w}_t, \boldsymbol{\lambda}_t) - \delta_{\mathbf{w}} L(\mathbf{w}_t, \boldsymbol{\lambda}_t))}_{B} \\
&+ \underbrace{\sum_{t=1}^{T} \lambda_t(\nabla_{\boldsymbol{\lambda}} L(\mathbf{w}_t, \boldsymbol{\lambda}_t) - \delta_{\boldsymbol{\lambda}} L(\mathbf{w}_t, \boldsymbol{\lambda}_t)) - \mathbf{w}_t(\nabla_{\mathbf{w}} L(\mathbf{w}_t, \boldsymbol{\lambda}_t) - \delta_{\mathbf{w}} L(\mathbf{w}_t, \boldsymbol{\lambda}_t))}_{C}
\end{aligned}$$

To complete the proof, we need to bound each of the terms A, B, C in the sum above and take expectation. First, we show the bounds on A as follows:

$$\mathbb{E}\left[\max_{\boldsymbol{\lambda}\in\Lambda,\mathbf{w}\in\mathcal{W}}\sum_{t=1}^{T}(\boldsymbol{\lambda}-\boldsymbol{\lambda}_t)\delta_{\boldsymbol{\lambda}}L(\mathbf{w}_t,\boldsymbol{\lambda}_t)\right]$$

and

$$\mathbb{E}\left[\max_{\boldsymbol{\lambda}\in\Lambda,\mathbf{w}\in\mathcal{W}}\sum_{t=1}^{T}(\mathbf{w}_t-\mathbf{w})\delta_{\mathbf{w}}L(\mathbf{w}_t,\boldsymbol{\lambda}_t)\right],$$

which can be obtained in a similar way. Consider the following series of inequalities:

$$\begin{aligned}
&(\mathbf{w}_t-\mathbf{w})\delta_{\mathbf{w}}L(\mathbf{w}_t,\boldsymbol{\lambda}_t)\\
&=\frac{1}{2\gamma_{\mathbf{w}}}\sum_{t=1}^{T}\|\mathbf{w}-\mathbf{w}_t\|_2^2+\gamma_{\mathbf{w}}^2\|\delta_{\mathbf{w}}L(\mathbf{w}_t,\boldsymbol{\lambda}_t)\|_2^2-\|\mathbf{w}_t-\gamma_{\mathbf{w}}\delta_{\mathbf{w}}L(\mathbf{w}_t,\boldsymbol{\lambda}_t)-\mathbf{w}\|_2^2\\
&\leq\frac{1}{2\gamma_{\mathbf{w}}}\sum_{t=1}^{T}\|\mathbf{w}-\mathbf{w}_t\|_2^2+\gamma_{\mathbf{w}}^2\|\delta_{\mathbf{w}}L(\mathbf{w}_t,\boldsymbol{\lambda}_t)\|_2^2-\|\mathbf{w}_{t+1}-\mathbf{w}\|_2^2\\
&=\frac{1}{2\gamma_{\mathbf{w}}}\|\mathbf{w}_1-\mathbf{w}\|_2^2-\|\mathbf{w}_{T+1}-\mathbf{w}\|_2^2+\frac{\gamma_{\mathbf{w}}}{2}\sum_{t=1}^{T}\|\delta_{\mathbf{w}}L(\mathbf{w}_t,\boldsymbol{\lambda}_t)\|_2^2\\
&\leq\frac{1}{2\gamma_{\mathbf{w}}}\|\mathbf{w}_1-\mathbf{w}\|_2^2+\frac{\gamma_{\mathbf{w}}}{2}\sum_{t=1}^{T}\|\delta_{\mathbf{w}}L(\mathbf{w}_t,\boldsymbol{\lambda}_t)\|_2^2\\
&\leq\frac{2R_{\mathcal{W}}}{\gamma_{\mathbf{w}}}+\frac{\gamma_{\mathbf{w}}}{2}\sum_{t=1}^{T}\|\delta_{\mathbf{w}}L(\mathbf{w}_t,\boldsymbol{\lambda}_t)\|_2^2\\
&\leq\frac{2R_{\mathcal{W}}}{\gamma_{\mathbf{w}}}+\frac{\gamma_{\mathbf{w}}}{2}\sum_{t=1}^{T}\|\delta_{\mathbf{w}}L(\mathbf{w}_t,\boldsymbol{\lambda}_t)-\nabla_{\mathbf{w}}L(\mathbf{w}_t,\boldsymbol{\lambda}_t)+\nabla_{\mathbf{w}}L(\mathbf{w}_t,\boldsymbol{\lambda}_t)\|_2^2
\end{aligned}$$

Taking the maximum of both sides with respect to $\mathbf{w}$ and the expectation yields

$$\mathbb{E}\left[\max_{\boldsymbol{\lambda}\in\Lambda,\mathbf{w}\in\mathcal{W}}\sum_{t=1}^{T}(\mathbf{w}_t-\mathbf{w})\delta_{\mathbf{w}}L(\mathbf{w}_t,\boldsymbol{\lambda}_t)\right]\leq\frac{1}{2}\left(\frac{4\mathcal{D}_{\mathcal{W}}^2}{\gamma_{\mathbf{w}}}+\gamma_{\mathbf{w}}T\sigma_{\mathbf{w}}^2+\gamma_{\mathbf{w}}T\mathcal{G}_{\mathbf{w}}^2\right)$$

and a repeating the same steps for $\boldsymbol{\lambda}$ we obtain

$$\mathbb{E}\left[\max_{\boldsymbol{\lambda}\in\Lambda,\mathbf{w}\in\mathcal{W}}\sum_{t=1}^{T}(\boldsymbol{\lambda}-\boldsymbol{\lambda}_t)\delta_{\boldsymbol{\lambda}}L(\mathbf{w}_t,\boldsymbol{\lambda}_t)\right]\leq\frac{1}{2}\left(\frac{4\mathcal{D}_{\Lambda}^2}{\gamma_{\boldsymbol{\lambda}}}+\gamma_{\boldsymbol{\lambda}}T\sigma_{\boldsymbol{\lambda}}^2+\gamma_{\boldsymbol{\lambda}}T\mathcal{G}_{\boldsymbol{\lambda}}^2\right)$$

Next, we bound B in the following way:

$$\max_{\boldsymbol{\lambda}\in\Lambda,\mathbf{w}\in\mathcal{W}}\sum_{t=1}^{T}\boldsymbol{\lambda}(\nabla_{\lambda}L(\mathbf{w}_t,\boldsymbol{\lambda}_t)-\delta_{\lambda}L(\mathbf{w}_t,\boldsymbol{\lambda}_t))\leq R_{\Lambda}\|\sum_{t=1}^{T}\boldsymbol{\lambda}(\nabla_{\lambda}L(\mathbf{w}_t,\boldsymbol{\lambda}_t)-\delta_{\lambda}L(\mathbf{w}_t,\boldsymbol{\lambda}_t)\|_2$$

After we take expectation of both sides, we get

$$\mathbb{E}\left[\max_{\boldsymbol{\lambda}\in\Lambda,\mathbf{w}\in\mathcal{W}}\sum_{t=1}^{T}\boldsymbol{\lambda}(\nabla_{\lambda}L(\mathbf{w}_t,\boldsymbol{\lambda}_t)-\delta_{\lambda}L(\mathbf{w}_t,\boldsymbol{\lambda}_t))\right]\leq\mathcal{D}_{\Lambda}\sqrt{T}\sigma_{\boldsymbol{\lambda}}$$

and in a completely similar way we can derive that

$$\mathbb{E}\left[\max_{\boldsymbol{\lambda}\in\Lambda,\mathbf{w}\in\mathcal{W}}\sum_{t=1}^{T}\mathbf{w}(\nabla_{\mathbf{w}}L(\mathbf{w}_t,\boldsymbol{\lambda}_t)-\delta_{\mathbf{w}}L(\mathbf{w}_t,\boldsymbol{\lambda}_t))\right]\leq\mathcal{D}_{\mathcal{W}}\sqrt{T}\sigma_{\mathbf{w}}\tag{C.3}$$

For the term C, it directly follows from the unbiased stochastic gradients that $\mathbb{E}[C]=0$. If we combine the bounds on A,B,C that we derived above and let the step sizes be $\gamma_{\mathbf{w}}=\frac{1}{\sqrt{T}}\frac{2\mathcal{D}_{\mathcal{W}}}{\sqrt{\sigma_{\mathbf{w}}^2+\mathcal{G}_{\mathbf{w}}^2}}$ and $\gamma_{\boldsymbol{\lambda}}=\frac{1}{\sqrt{T}}\frac{2\mathcal{D}_{\Lambda}}{\sqrt{\sigma_{\boldsymbol{\lambda}}^2+\mathcal{G}_{\boldsymbol{\lambda}}^2}}$, we immediately obtain the final result. □

Table C.2: Comparison of logistic regression models trained with individual losses for the MNIST dataset.

Model / Metric	Zero-one	Hinge	Cross-entropy	AUC
One Loss	**0.0747**	-	-	-
(std)	(0.0001)	-	-	-
Hinge	-	**0.0656**	-	-
(std)	-	(0.0002)	-	-
Uniform	0.1187	0.5550	0.0911	0.9859
(std)	(0.0020)	(0.0005)	(0.0005)	(0.0004)
ALMO	0.1030	0.9228	0.0489	0.9905
(std)	(0.0020)	(0.0050)	(0.0005)	(0.0002)
λ	0.0000	0.0478	0.8042	-

C.3 Additional Experiments

In this section, we present additional baseline studies to further highlight the benefits of the proposed ALMO algorithm. As baselines, we train with just one loss at a time and compare the ALMO performance to this per-loss optimal performance. This experimental setup is the same as the one detailed in the main section, but the realizations of the split of the data differs, which accounts for the small performance differences as compared to the tables in the main section.

In these tables, the boldfaced numbers indicate the performance of a classifier trained just for that loss. The ALMO algorithm often achieves a performance close to these values, without sacrificing any loss significantly. The mean values of the corresponding (un-normalized) λ-s are also reported to illustrate the weight ALMO assigns to the given loss. In a few cases, especially on the Adult dataset, the ALMO algorithm appears to be performing even slightly better than the baseline. We attribute this discrepancy to the non-convexity of the optimization problem and the small size of the Adult dataset.

Table C.3: Comparison of logistic regression models trained with individual losses for the Fashion-MNIST dataset.

Model / Metric	Zero-one	Hinge	Cross-entropy	AUC
Zero-one	**0.1603**	-	-	-
(std)	(0.0005)	-	-	-
Hinge	-	**0.0958**	-	-
(std)	-	(0.0006)	-	-
Uniform	0.1814	0.5431	0.0932	0.9786
(std)	(0.0005)	(0.0035)	(0.0002)	(0.0003)
ALMO	0.1774	0.4078	0.0683	0.9800
(std)	(0.0008)	(0.0011)	(0.0002)	(0.0001)
λ	0.0000	0.1043	0.6410	-

Table C.4: Comparison of logistic regression models trained with individual losses for the Adult dataset.

Model / Metric	Zero-one	Hinge	Cross-entropy	AUC
Zero-one	**0.1849**	-	-	-
(std)	(0.0007)	-	-	-
Hinge	-	**0.4146**	-	-
(std)	-	(0.0001)	-	-
Uniform	0.2001	0.8624	0.6320	0.8378
(std)	(0.0008)	(0.0067)	(0.0033)	(0.0004)
ALMO	0.1938	0.4085	0.4232	0.8409
(std)	(0.0020)	(0.0021)	(0.0080)	(0.0003)
λ	0.0000	0.1317	0.0000	-

Table C.5: Comparison of DNN models trained with individual losses for the MNIST dataset.

Model / Metric	Zero-one	Hinge	Cross-entropy	AUC
Zero-one	**0.0215**	-	-	-
(std)	(0.0022)	-	-	-
Hinge	-	**0.0132**	-	-
(std)	-	(0.0002)	-	-
Uniform	0.0168	0.6255	0.0437	0.9984
(std)	(0.0013)	(0.0031)	(0.0008)	(0.0004)
ALMO	0.0143	0.0092	0.0397	0.9996
(std)	(0.0001)	(0.0001)	(0.0040)	(0.0001)
λ	0.0001	0.3893	0.1905	-

Table C.6: Comparison of DNN models trained with individual losses for the Fashion-MNIST dataset.

Model / Metric	Zero-one	Hinge	Cross-entropy	AUC
Zero-one	**0.1137**	-	-	-
(std)	(0.0010)	-	-	-
Hinge	-	**0.0595**	-	-
(std)	-	(0.0016)	-	-
Uniform	0.1111	0.6603	0.0695	0.9889
(std)	(0.0025)	(0.0030)	(0.0005)	(0.0006)
ALMO	0.1077	0.1085	0.0374	0.9908
(std)	(0.0035)	(0.0030)	(0.0008)	(0.0005)
λ	0.0000	0.1779	0.5199	-

Table C.7: Comparison of DNN models trained with individual losses for the Adult dataset.

Model / Metric	Zero-one	Hinge	Cross-entropy	AUC
Zero-one	**0.1564**	-	-	-
(std)	(0.0018)	-	-	-
Hinge	-	**0.5349**	-	-
(std)	-	(0.0176)	-	-
Uniform	0.1483	0.9123	0.6707	0.8093
(std)	0.0003	0.0001	0.0005	(0.0108)
ALMO	0.1450	0.3043	0.4437	0.8716
(std)	0.0020	0.0000	0.0000	(0.0021)
λ	0.0000	0.0000	0.0000	-

D | APPENDIX TO CHAPTER 6.

D.1 WEIGHTED AUTOMATA AND TRANSDUCERS OPERATIONS

This section provides further details on the concepts of weighted automata and transducers that were introduced in Section 6.3.

Recall that we have defined a *weighted finite-state transducer* $\mathcal{T}$ over a semiring $(\mathbb{S}, \oplus, \otimes, \overline{0}, \overline{1})$ as an 8-tuple $(\Sigma, \Delta, Q, I, F, E, \lambda, \rho)$, where Σ is a finite input alphabet, Δ is a finite output alphabet, Q is a finite set of states, $I \subseteq Q$ is the set of initial states, $F \subseteq Q$ is the set of final states, E is a finite multiset of transitions, which are elements of $Q \times (\Sigma \cup \{\epsilon\}) \times (\Delta \cup \{\epsilon\}) \times \mathbb{S} \times Q$, $\lambda \colon I \to \mathbb{S}$ is an initial weight function, and $\rho \colon F \to \mathbb{S}$ is a final weight function. Moreover, we defined a *weighted finite automaton* to be a weighted finite-state transducer where the input and output labels are the same.

A tuple $(\mathbb{S}, \oplus, \otimes, \overline{0}, \overline{1})$ is a *semiring* if $(\mathbb{S}, \oplus, \overline{0})$ is a commutative monoid with identity element $\overline{0}$, $(\mathbb{S}, \otimes, \overline{1})$ is a monoid with identity element $\overline{1}$, $\otimes$ distributes over $\oplus$, and $\overline{0}$ is an annihilator for $\otimes$. In other words, a semiring is a ring that may lack negation.

The construction of weighted transducers and automata used in Sections 6.4 and 6.5 required the following operations: inverse ($\mathcal{T}^{-1}$), projection ($\Pi(\mathcal{T})$), composition ($\mathcal{T}_1 \circ \mathcal{T}_2$), and determinization ($\mathrm{Det}(\mathcal{A})$). We provide precise definitions of these operations below.

The *inverse* of a WFST $\mathcal{T}$ is denoted by $\mathcal{T}^{-1}$ and defined as the transducer obtained by swapping the input and output labels of every transition of $\mathcal{T}$, that is, $\mathcal{T}^{-1}(x, y) = \mathcal{T}(y, x)$ for all (x, y).

The *projection* of a WFST $\mathcal{T}$ is the weighted automaton denoted by $\Pi(\mathcal{T})$ obtained from $\mathcal{T}$ by omitting the input label of each transition and keeping only the output label.

The *composition* of two WFSTs $\mathcal{T}_1$ with output alphabet Δ and $\mathcal{T}_2$ with a matching input alphabet Δ is a weighted transducer defined for all x, y by:

$$(\mathcal{T}_1 \circ \mathcal{T}_2)(x, y) = \bigoplus_{z \in \Delta^*} \Big(\mathcal{T}_1(x, z) \otimes \mathcal{T}_2(z, y) \Big), \tag{D.1}$$

where the sum runs over all strings z labeling a path of $\mathcal{T}_1$ on the output side and a path of $\mathcal{T}_2$ on the input side. The worst case complexity of computing $(\mathcal{T}_1 \circ \mathcal{T}_2)$ is quadratic, that is $O(|\mathcal{T}_1||\mathcal{T}_2|)$, assuming that the $\otimes$-operation can be computed in constant time. The composition operation can also be used with WFAs by viewing a WFA as a WFST with equal input and output labels at every transition. Thus, for two WFAs $\mathcal{A}_1$ and $\mathcal{A}_2$, $(\mathcal{A}_1 \circ \mathcal{A}_2)$ is a WFA defined for all x by $(\mathcal{A}_1 \circ \mathcal{A}_2)(x) = \mathcal{A}_1(x) \otimes \mathcal{A}_2(x)$.

A weighted automaton is said to be *deterministic* iff it has a unique initial state and if no two transitions leaving any state share the same input label. As for (unweighted) finite automata, there exists a *determinization* algorithm for WFAs. The algorithm returns a deterministic WFA equivalent to its input WFA [Mohri 1997]. Unlike the unweighted case, weighted determinization is not defined for all input WFAs but it can be applied to any acyclic WFA, which is the case of interest for us. When it can be applied to $\mathcal{A}$, we will denote by $\mathrm{Det}(\mathcal{A})$ the deterministic WFA returned by determinization.

D.2 SEQUENCE-TO-SEQUENCE MODEL TRAINING WITH RATIONAL AND TROPICAL LOSSES

In this section, we describe how our algorithms can be incorporated into standard procedures for training modern neural network architectures for structured prediction tasks, particularly sequence-to-sequence models [Sutskever et al. 2014]. Sequence-to-sequence models for structured prediction, such as RNNs and LSTMs, typically consist of an *encoder* network, which maps input data from $\mathcal{X}$ to abstract representations and a *decoder* network, which models a conditional distribution over the output space $\mathcal{Y}$. The decoder returns a $l|\Delta|$-dimensional vector of scores or *logits* $\mathbf{w}(x) = (w_{y,s}(x))$. We define $\boldsymbol{\psi}(x, y^s, s)$ to be a vector of dimension $l|\Delta|$ such that the coordinate corresponding to (y^s, s) is equal to one and zero otherwise. Then, setting $\boldsymbol{\Psi}(x, y) = \sum_{s=1}^{l} \boldsymbol{\psi}(x, y^s, s)$ defines Markovian features of order $q = 0$ as in Section 6.2. This allows us to use $\mathbf{w}$ and $\boldsymbol{\Psi}$ to compute $F(\mathbf{w})$ in (6.1) along wtih its gradient $\nabla_{\mathbf{w}} F(\mathbf{w})$ in both the forward and backward pass, using techniques presented in Sections 6.4 and 6.5. In particular, $\nabla F(\mathbf{w})$ can be propagated down to lower layers of the neural network model using the chain rule.

Note that, in practice, the generation of scores from the decoder to construct these features for each $y \in \mathcal{Y}$ is expensive, and the common solution [Ranzato et al. 2015; Prabhavalkar et al. 2017] is to restrict the output vocabulary of Δ to a subset Δ_s of size k at each position s. This is often accomplished via the beam search algorithm. In our framework, we run the beam search to construct the features automata $\mathcal{A}$, the topology of which is equal to the topology of the beam search tree.

D.3 PSEUDOCODE FOR GRAD-NAÏVE

GRAD-NAÏVE$(x_i, y_i, \mathbf{w})$

1 $Z_{\mathbf{w}} \leftarrow \sum_{y \in \mathcal{Y}} e^{\mathsf{L}(y, y_i) + \mathbf{w} \cdot \mathbf{\Psi}(x_i, y)}$

2 **for** $(\mathbf{z}, s) \in \Delta^q \times [l]$ **do**

3 $\quad \mathsf{Q}_{\mathbf{w}}(\mathbf{z}, s) \leftarrow \sum_{y \colon y^{s-q+1:s} = \mathbf{z}} e^{\mathsf{L}(y, y_i) + \mathbf{w} \cdot \mathbf{\Psi}(x_i, y)}$

4 $\quad \mathsf{Q}_{\mathbf{w}}(\mathbf{z}, s) \leftarrow \mathsf{Q}_{\mathbf{w}}(\mathbf{z}, s) / Z_{\mathbf{w}}$

Figure D.1: Computation of the key term of the gradient using the naïve direct method.

E | APPENDIX TO CHAPTER 7.

E.1 PROOF OF LEMMA 7.2

Proof. For the first part of the proof, let $E_k = \mathbb{E}\left[C_{S,k}\right] = \mathbb{E}\left[C_{U,k}\right]$ be the population covariance operator of kernel $\mu_k K_k$. We will show a concentration bound on $C_{S,k}$ and $C_{U,k}$ that holds uniformly over $k \in [1, p]$. Using the union bound, Lemma 1 from [Zwald and Blanchard 2005] (equivalently Corollary 5 from [Shawe-Taylor and Cristianini 2003]), and assuming $u > m$, with probability at least $1 - \delta$, the following holds for all $k \in [1, p]$,

$$\max\left[\|E_k - C_{S,k}\|_{\overline{\mathbb{H}}_k}, \|E_k - C_{U,k}\|_{\overline{\mathbb{H}}_k}\right] \leq \frac{\mu_k \kappa}{2\sqrt{m}}\,, \tag{E.1}$$

where $\kappa = 4\left(1 + \sqrt{\frac{\log\left(\frac{2p}{\delta}\right)}{2}}\right)$.

Let Π_k be the orthogonal projection onto the top r eigenfunctions of E_k. By decomposing over orthogonal subspaces of $\overline{\mathbb{H}} = \bigoplus_{k=1}^{p} \overline{\mathbb{H}}_k$ as well as adding and subtracting Π_k, we can bound $\|\Pi_S - \Pi_U\|$ by $\sum_{k=1}^{p} \|\Pi_k - \Pi_S\|_{\overline{\mathbb{H}}_k} + \sum_{k=1}^{p} \|\Pi_k - \Pi_U\|_{\overline{\mathbb{H}}_k}$. Now, since $C_{S,k}$ is the restriction of C_S to $\overline{\mathbb{H}}_k$, the following inequality holds for all $k \in [1, p]$:

$$\|\Pi_k - \Pi_S\|_{\overline{\mathbb{H}}_k} \leq \frac{8\|E_k - C_S\|_{\overline{\mathbb{H}}_k}}{\lambda_r(E_k) - \lambda_{r+1}(E_k)} = \frac{8\|E_k - C_{S,k}\|_{\overline{\mathbb{H}}_k}}{\lambda_r(E_k) - \lambda_{r+1}(E_k)}\,. \tag{E.2}$$

A similar statement holds for the projection with respect to sample U. To obtain the bound above we consider two cases, either $8\|E_k - C_S\|_{\overline{\mathbb{H}}_k}/(\lambda_r(E_k) - \lambda_{r+1}(E_k)) \leq 1/4$, which is a sufficient condition for Theorem 3 of [Zwald and Blanchard 2005] that directly implies (E.2). Otherwise, if the condition does not hold, then the right-hand side of (E.2) will be larger than 2, which is a trivial bound on the difference of two projections.

Next, we use the constraint $\|\boldsymbol{\mu}\|_1 \leq 1$ to upper bound (E.1) by $\kappa/(2\sqrt{m})$ and lower bound $\lambda_r(E_k) - \lambda_{r+1}(E_k) = \mu_k\big(\lambda_r(C_k) - \lambda_{r+1}(C_k)\big) \geq \mu_k\Delta_r$, where C_k is the population covariance operator of kernel K_k and $\Delta_r = \min_{k\in[1,p]}\big(\lambda_r(C_k) - \lambda_{r+1}(C_k)\big)$. Now $4\kappa/(\sqrt{m}\mu_k\Delta_r)$ is the uniform bound on the norm of projections in (E.2). Summing up $\|\Pi_k - \Pi_S\|_{\overline{\mathbb{H}}_k} + \|\Pi_k - \Pi_U\|_{\overline{\mathbb{H}}_k}$ over k and applying the uniform bound $4\kappa/(\sqrt{m}\mu_k\Delta_r)$, which holds for both samples U and S, we conclude that the following holds:

$$\|\Pi_S - \Pi_U\| \leq \sum_{k=1}^{p} \frac{1}{\mu_k}\frac{8\kappa}{\Delta_r\sqrt{m}} \leq \frac{8\kappa\nu}{\Delta_r\sqrt{m}}\,. \tag{E.3}$$

For the second portion of the proof, we use a series of inequalities to show

$$|\|C_U\|_{(r)} - \|C_S\|_{(r)}| \leq \sum_{i=1}^{r} |\lambda_i(C_U) - \lambda_i(C_S)| \leq \sqrt{r}\bigg(\sum_{i=1}^{r} |\lambda_i(C_U) - \lambda_i(C_S)|^2\bigg)^{1/2},$$

which is in turn bounded by $\sqrt{r}\|C_U - C_S\|$ using the Hoffman-Wielandt inequality. Next, by decomposing over orthogonal subspaces of $\overline{\mathbb{H}} = \bigoplus_{k=1}^{p}\overline{\mathbb{H}}_k$ together with adding and subtracting E_k, we bound $\|C_U - C_S\|$ by $\sum_{k=1}^{p}\|E_k - C_{S,k}\|_{\overline{\mathbb{H}}_k} + \sum_{k=1}^{p}\|E_k - C_{U,k}\|_{\overline{\mathbb{H}}_k}$. If we again apply the uniform bound from (E.1) in the form $\mu_k\kappa/2\sqrt{m}$, we obtain that with probability at least $1-\delta$, the following holds:

$$|\|C_U\|_{(r)} - \|C_S\|_{(r)}| \leq \sum_{k=1}^{p} \frac{\sqrt{r}\mu_k\kappa}{\sqrt{m}} \leq \kappa\,. \tag{E.4}$$

Combining (E.3) and (E.4) yields

$$\sup_{\boldsymbol{\mu}\in\mathcal{M}} \|\Pi_U u\| \leq \sup_{\boldsymbol{\mu}\in\mathcal{N}} \|\Pi_U u\| \leq \sup_{\boldsymbol{\mu}\in\mathcal{N}} \left(\|\Pi_S u\| + \frac{8\kappa\nu}{\Delta_r\sqrt{m}} \|u\| \right), \tag{E.5}$$

with probability $1-\delta$. □

E.2 Proof of Lemma 7.3

We will use the following lemma to give the proof of Lemma 7.3.

Lemma E.1. *For each $k \in [1,p]$ let $\lambda_1(\overline{\mathbf{K}}_k) \geq \cdots \geq \lambda_m(\overline{\mathbf{K}}_k)$ be the eigenvalues of $\overline{\mathbf{K}}_k$ with corresponding orthonormal eigenvectors $\mathbf{v}_{k,1}, \cdots, \mathbf{v}_{k,m}$. For $\mu \in \mathcal{N}$, let I_μ denote the set of indices (k,j) corresponding to the largest r elements of the set $\{\mu_k \lambda_j(\overline{\mathbf{K}}_k)\}_{k,j}$, then the following equality holds:*

$$\left\| \Pi_S \sum_{i=1}^{m} \sigma_i \boldsymbol{\Phi}(x_i) \right\| = \sqrt{m \sum_{(k,j)\in I_{\boldsymbol{\mu}}} \mu_k \lambda_j(\overline{\mathbf{K}}_k)(\mathbf{v}_{k,j}^\top \boldsymbol{\sigma})^2}\,, \tag{E.6}$$

where $\boldsymbol{\sigma} = (\sigma_1, \cdots, \sigma_m)^\top$.

Proof. Recall from Section 7.3 that when $\overline{\mathbb{H}} = \bigoplus_{k=1}^{p} \overline{\mathbb{H}}_k$, the eigenvalues of C_S take the form $\mu_k \lambda_j(\overline{\mathbf{K}}_k)$ with orthonormal eigenfunctions $u_{k,j}$, where, for each $k \in [1,p]$, the given functions $u_{k,1}, \ldots, u_{k,m}$ belong to the orthogonal component $\overline{\mathbb{H}}_k$. Thus, we can write $\|\Pi_S f\|^2 = \sum_{(k,j)\in I_\mu} \langle u_{k,j}, f\rangle^2$, for any $f \in \overline{\mathbb{H}}$. When $f = \boldsymbol{\Phi}(x_i)$, it suffices to take the inner product of $u_{k,j}$ and f in $\overline{\mathbb{H}}_k$, which, by the reproducing property, is equal to $u_{k,j}(x_i)$. By [Blanchard and Zwald 2008, Equation (8)], $u_{k,j}(x_i)$ takes the form

$$u_{k,j}(x_i) = \sqrt{\frac{\mu_k}{\lambda_j(\overline{\mathbf{K}}_k) m}} \sum_{n=1}^{m} K_k(x_i, x_n)[\mathbf{v}_{k,j}]_n\,. \tag{E.7}$$

Since $\mathbf{v}_{k,j}$ is an eigenvector of $\overline{\mathbf{K}}_k$, we see that

$$\sum_{n=1}^{m} K_k(x_i, x_n)[\mathbf{v}_{k,j}]_n = m \sum_{n=1}^{m} [\overline{\mathbf{K}}_k]_{i,n}[\mathbf{v}_{k,j}]_n = m\lambda_j(\overline{\mathbf{K}}_k)[\mathbf{v}_{k,j}]_i \,, \tag{E.8}$$

which means that $u_{k,j}(x_i) = \sqrt{m\mu_k\lambda_j(\overline{\mathbf{K}}_k)}[\mathbf{v}_{k,j}]_i$. In view of this expression, we can write

$$\langle u_{k,j}, \sum_{i=1}^{m} \sigma_i \mathbf{\Phi}(x_i)\rangle = \sum_{i=1}^{m} \sigma_i u_{k,j}(x_i) = \sqrt{m\mu_k\lambda_j(\overline{\mathbf{K}}_k)}\mathbf{v}_{k,j}^\top \boldsymbol{\sigma}. \tag{E.9}$$

Squaring the terms above and summing them over the set of indices I_μ completes the proof. □

The following gives the proof of Lemma 7.3.

Lemma 7.3. The term $\|\Pi_S \sum_{i=1}^m \sigma_i \mathbf{\Phi}(x_i)\|$ can be directly bounded using only the Ky-Fan norm constraint on $\|C_S\|_{(r)}$, since it controls the spectrum of the projection. Thus, we will simplify the problem to analyze the supremum over choices of $\boldsymbol{\mu}$ that satisfy $\|C_S\|_{(r)} \leq \epsilon$, where $\epsilon = \Lambda_{(r)} + \kappa$. This clearly includes all elements in $\mathcal{N}$ as well.

For this proof, we will use the representation of $\|\Pi_S \sum_{i=1}^m \sigma_i \mathbf{\Phi}(x_i)\|$ from Lemma E.1, which will be upper bounded by the supremum over the choice of all size-r sets in order to remove the dependence of the set of indices on the identity of the top eigenvalues:

$$\sup_{\|C_S\|_{(r)} \leq \epsilon} \sum_{(k,j)\in I_{\boldsymbol{\mu}}} \mu_k\lambda_j(\overline{\mathbf{K}}_k)(\mathbf{v}_{k,j}^\top \boldsymbol{\sigma})^2 \leq \sup_{|I|=r} \sup_{\|C_S\|_{(r)} \leq \epsilon} \sum_{(k,j)\in I} \mu_k\lambda_j(\overline{\mathbf{K}}_k)(\mathbf{v}_{k,j}^\top \boldsymbol{\sigma})^2 \,, \tag{E.10}$$

where $\sup_{|I|=r}$ indicates the supremum over all indexing sets. We can express the sum above as an inner product $\mathbf{u}_\mu \cdot \mathbf{u}_\sigma$, where $\mathbf{u}_\mu$ is an $r-$dimensional vector with entries $\mu_k\lambda_j(\overline{\mathbf{K}}_k)$ and $\mathbf{u}_\sigma$ has entries $(\mathbf{v}_{k,j}^\top \boldsymbol{\sigma})^2$ such that $(k,j) \in I$. By construction, we have $\|C_S\|_{(r)} = \|\mathbf{u}_\mu\|_1$, thus, we will reduce the problem to that of analyzing $\sup_{\|\mathbf{u}_\mu\|_1 \leq \epsilon}$. Then, by definition of the dual norm, we can

write:

$$\sup_{|I|=r} \sup_{\|\mathbf{u}_\mu\|_1 \leq \epsilon} \mathbf{u}_\mu \cdot \mathbf{u}_\sigma = \sup_{|I|=r} \epsilon \|\mathbf{u}_\sigma\|_\infty = \epsilon \max_{k,j} (\mathbf{v}_{k,j}^\top \boldsymbol{\sigma})^2 \,. \tag{E.11}$$

Thus, $\|\Pi_S \sum_{i=1}^m \sigma_i \boldsymbol{\Phi}(x_i)\|$ is bounded by the following:

$$\max_{k,j} \sqrt{m\epsilon (\mathbf{v}_{k,j}^\top \boldsymbol{\sigma})^2} = \sqrt{m\epsilon} \max_{k,j} |\mathbf{v}_{k,j}^\top \boldsymbol{\sigma}| = \sqrt{m\epsilon} \max_{k,j} \max_{s_t \in \{-1,1\}} s_t \mathbf{v}_{k,j}^\top \boldsymbol{\sigma} \,.$$

By Massart's lemma [Massart 2000], we can write

$$\mathop{\mathbb{E}}_{\boldsymbol{\sigma}} \left[\max_{k,j} \max_{s_t \in \{-1,1\}} s_t \mathbf{v}_{k,j}^\top \boldsymbol{\sigma} \right] \leq \sqrt{2 \log (2pm)} \,. \tag{E.12}$$

This follows since the norm of $s_t \mathbf{v}_{k,j}$ is bounded by 1 and since the cardinality of the set over which the maximum is taken is bounded by $2pm$. Combining all the intermediate results leads to the following:

$$\frac{1}{m} \mathop{\mathbb{E}}_{\boldsymbol{\sigma}} \left[\sup_{\|C_S\|_{(r)} \leq \epsilon} \|\Pi_S \sum_{i=1}^m \sigma_i \boldsymbol{\Phi}(x_i)\| \right] \leq \sqrt{\frac{2\epsilon \log (2pm)}{m}}. \tag{E.13}$$

The final result is obtained by setting $\epsilon = \Lambda_{(r)} + \kappa$. □

E.3 Proof of Theorem 7.6

Proof. First we let S and U be any two samples, both of size m, such that U is simply an unlabeled version of S. Now, assume that $p = 1$ and that the sample kernel matrix $\overline{\mathbf{K}}_1$ of kernel K_1 admits exactly r distinct non-zero simple eigenvalues. Finally, select $\Lambda_{(r)}$ such that $\Lambda_{(r)} / \lambda_1(\overline{\mathbf{K}}_1) \leq 1$.

As calculated in Section 7.4, $\sup_{\|\mathbf{w}\| \leq 1} \sum_{i=1}^m \sigma_i h(x_i) = \|\Pi_U \sum_{i=1}^m \sigma_i \boldsymbol{\Phi}(x_i)\|$ and in this particular scenario $\|C_U\|_{(r)} = \|C_S\|_{(r)}$. Thus, the empirical Rademacher complexity simplifies to $\widehat{\mathbb{R}}_S(H) = \frac{1}{m} \mathbb{E}_{\boldsymbol{\sigma}} \left[\sup_{\|C_S\|_{(r)} \leq \Lambda_{(r)}} \|\Pi_S \sum_{i=1}^m \sigma_i \boldsymbol{\Phi}(x_i)\| \right]$, where the projection can be written di-

rectly in terms of the sample S. Here, the L_1 constraint on $\boldsymbol{\mu}$ is not needed, since it is satisfied by the Ky-Fan r-norm constraint when $\Lambda_{(r)} \leq \lambda_1(\overline{\mathbf{K}}_1)$.

Now, following the steps from Lemma E.1, we can express the norm of the projection as follows:

$$\left\| \Pi_S \sum_{i=1}^{m} \sigma_i \boldsymbol{\Phi}(x_i) \right\| = \sqrt{m \sum_{j=1}^{r} \mu_1 \lambda_j(\overline{\mathbf{K}}_1)(\boldsymbol{\sigma}^\top \mathbf{v}_{1,j})^2}\,. \tag{E.14}$$

Note that here, unlike the general statement of Lemma E.1, the choice of the r entries that appear in the sum is not effected by the value of $\boldsymbol{\mu}$, since there are in fact only r non-zero eigenvalues in total, by construction (i.e. there is one base kernel of rank r). The choice of $\boldsymbol{\mu}$, however, still affects the scale of the r eigenvalues.

The expression is furthermore simplified by introducing the vectors $\mathbf{u}_\mu$ with entries $\mu_1 \lambda_j(\overline{\mathbf{K}}_1)$ and $\mathbf{u}_\sigma$ with entries $(\mathbf{v}_{1,j}^\top \boldsymbol{\sigma})^2$, which is similar to the proof of Lemma 7.3. By the monotonicity of the square-root function and using the definition of $\mathbf{u}_{\boldsymbol{\mu}}$ as well as the dual norm we have

$$\sup_{\|C_S\|_{(r)} \leq \Lambda_{(r)}} \sqrt{\mathbf{u}_\mu \cdot \mathbf{u}_{\boldsymbol{\sigma}}} = \sqrt{\sup_{\|\mathbf{u}_{\boldsymbol{\mu}}\|_1 \leq \Lambda_{(r)}} \mathbf{u}_\mu \cdot \mathbf{u}_{\boldsymbol{\sigma}}} = \sqrt{\Lambda_{(r)} \|\mathbf{u}_\sigma\|_\infty}\,. \tag{E.15}$$

Thus, the Rademacher complexity is reduced to

$$\hat{\mathfrak{R}}_S(\mathcal{H}) = \sqrt{\frac{\Lambda_{(r)}}{m}}\, \mathop{\mathbb{E}}_{\boldsymbol{\sigma}} \left[\sqrt{\max_{j \in [1,r]} (\mathbf{v}_{1,j}^\top \boldsymbol{\sigma})^2} \right] = \sqrt{\frac{\Lambda_{(r)}}{m}}\, \mathop{\mathbb{E}}_{\boldsymbol{\sigma}} \left[\max_{j \in [1,r]} |\mathbf{v}_{1,j}^\top \boldsymbol{\sigma}| \right]. \tag{E.16}$$

Finally, we use Jensen's inequality and Khintchine's inequality to show

$$\mathop{\mathbb{E}}_{\boldsymbol{\sigma}} \left[\max_{j \in [1,r]} |\mathbf{v}_{1,j}^\top \boldsymbol{\sigma}| \right] \geq \max_{j \in [1,r]} \mathop{\mathbb{E}}_{\boldsymbol{\sigma}} \left[|\mathbf{v}_{1,j}^\top \boldsymbol{\sigma}| \right] \geq \max_{j \in [1,r]} 2^{-1/2} \|\mathbf{v}_{1,j}\| = 2^{-1/2}, \tag{E.17}$$

where the tight constant $2^{-1/2}$ used in Khintchine's inequality can be found in [Nazarov and Pod-

korytov 2000][Chapter II]. Plugging this constant back into equation (E.16) completes the proof of the theorem. □

BIBLIOGRAPHY

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., and Zheng, X. (2016). Tensorflow: a system for large-scale machine learning. In *Proceedings of USENIX*.

Abay, A., Zhou, Y., Baracaldo, N., Rajamoni, S., Chuba, E., and Ludwig, H. (2020). Mitigating bias in federated learning.

Allauzen, C., Riley, M., Schalkwyk, J., Skut, W., and Mohri, M. (2007). OpenFst: a general and efficient weighted finite-state transducer library. In *Proceedings of CIAA*. Springer.

Amodei, D., Anubhai, R., Battenberg, E., Case, C., Casper, J., Catanzaro, B., Chen, J., Chrzanowski, M., Coates, A., Diamos, G., Elsen, E., Engel, J., Fan, L., Fougner, C., Hannun, A. Y., Jun, B., Han, T., LeGresley, P., Li, X., Lin, L., Narang, S., Ng, A. Y., Ozair, S., Prenger, R., Qian, S., Raiman, J., Satheesh, S., Seetapun, D., Sengupta, S., Wang, C., Wang, Y., Wang, Z., Xiao, B., Xie, Y., Yogatama, D., Zhan, J., and Zhu, Z. (2016). Deep speech 2: End-to-end speech recognition in english and mandarin. In *Proceedings of ICML*.

Angwin, J., Larson, J., Mattu, S., and Kirchner, L. (2019). Machine bias: There's software

used across the country to predict future criminals. and it's biased against blacks. 2016. *URL https://www. propublica. org/article/machine-bias-risk-assessments-in-criminal-sentencing*.

Aronszajn, N. (1950). Theory of reproducing kernels. *Transactions of the American mathematical society*, pages 337–404.

Bartlett, P. L. and Mendelson, S. (2003). Rademacher and Gaussian complexities: Risk bounds and structural results. *The Journal of Machine Learning Research*, 3:463–482.

Becker, C. J., Christoudias, C. M., and Fua, P. (2013). Non-linear domain adaptation with boosting. In *Proceedings of Advances in Neural Information Processing Systems 26: 27th Annual Conference on NIPS*, pages 485–493.

Belkin, M. and Niyogi, P. (2003). Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396.

Ben-David, S., Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., and Vaughan, J. W. (2010). A theory of learning from different domains. *Machine learning*, 79(1-2):151–175.

Ben-David, S., Blitzer, J., Crammer, K., and Pereira, F. (2007). Analysis of representations for domain adaptation. In *Advances in neural information processing systems*, pages 137–144.

Berger, A. L., Pietra, S. D., and Pietra, V. J. D. (1996). A maximum entropy approach to natural language processing. *Comp. Linguistics*, 22(1).

Bhatia, R. (1997). *Matrix Analysis*. Springer.

Bickel, P. J., Hammel, E. A., and O'Connell, J. W. (1975). Sex bias in graduate admissions: Data from Berkeley. *Science*, 187(4175):398–404.

Blanchard, G. and Zwald, L. (2008). Finite-dimensional projection for classification and statistical learning. *Information Theory, IEEE Transactions on*, 54(9):4169–4182.

Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., and Wortman, J. (2008). Learning bounds for domain adaptation. In *Advances in neural information processing systems*, pages 129–136.

Blitzer, J., Dredze, M., and Pereira, F. (2007). Biographies, Bollywood, Boom-boxes and Blenders: Domain Adaptation for Sentiment Classification. In *Proceedings of ACL 2007*, Prague, Czech Republic.

Bonawitz, K., Eichner, H., Grieskamp, W., Huba, D., Ingerman, A., Ivanov, V., Kiddon, C., Konečnỳ, J., Mazzocchi, S., McMahan, H. B., et al. (2019). Towards federated learning at scale: System design. *arXiv preprint arXiv:1902.01046*.

Bottou, L., Curtis, F. E., and Nocedal, J. (2018). Optimization methods for large-scale machine learning. *Siam Review*, 60(2):223–311.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24(2):123–140.

Breiman, L. (1999). Prediction games and arcing algorithms. *Neural Computation*, 11:1493–1517.

Brisimi, T. S., Chen, R., Mela, T., Olshevsky, A., Paschalidis, I. C., and Shi, W. (2018). Federated learning of predictive models from federated electronic health records. *International journal of medical informatics*, 112:59–67.

Britz, D. (2016). Attention and memory in deep learning and nlp. `http://www.wildml.com/2016/01/attention-and-memory-in-deep-learning-and-nlp/`.

Cai, D., He, X., and Han, J. (2007). Semi-supervised discriminant analysis. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–7. IEEE.

Caldas, S., Wu, P., Li, T., Konečnỳ, J., McMahan, H. B., Smith, V., and Talwalkar, A. (2018). Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097*.

Caruana, R., Niculescu-Mizil, A., Crew, G., and Ksikes, A. (2004). Ensemble selection from libraries of models. In *Proceedings of ICML*, page 18. ACM.

Chang, K., Krishnamurthy, A., Agarwal, A., Daumé III, H., and Langford, J. (2015). Learning to search better than your teacher. In *Proceedings of ICML*.

Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. ACM.

Chen, T., Li, M., Li, Y., Lin, M., Wang, N., Wang, M., Xiao, T., Xu, B., Zhang, C., and Zhang, Z. (2015). MXNet: A flexible and efficient machine learning library for heterogeneous distributed systems. *CoRR*, abs/1512.01274.

Cheng, Y., Cao, G., Wang, X., and Pan, J. (2013). Weighted multi-source tradaboost. *Chinese Journal of Electronics*, 22(3):505–510.

Chollet, F. et al. (2015). Keras. `https://keras.io`.

Cortes, C., Gonzalvo, X., Kuznetsov, V., Mohri, M., and Yang, S. (2017). Adanet: Adaptive structural learning of artificial neural networks. In *Proceedings of ICML*, pages 874–883.

Cortes, C., Haffner, P., and Mohri, M. (2004). Rational kernels: Theory and algorithms. *JMLR*, 5:1035–1062.

Cortes, C., Kuznetsov, V., Mohri, M., Storcheus, D., and Yang, S. (2018). Efficient gradient

computation for structured output learning with rational and tropical losses. *Advances in Neural Information Processing Systems*, 31:6810–6821.

Cortes, C., Kuznetsov, V., Mohri, M., and Warmuth, M. K. (2015a). On-line learning algorithms for path experts with non-additive losses. In *Proceedings of COLT*.

Cortes, C., Kuznetsov, V., Mohri, M., and Yang, S. (2016). Structured prediction theory based on factor graph complexity. In *Proceedings of NIPS*.

Cortes, C. and Mohri, M. (2014). Domain adaptation and sample bias correction theory and algorithm for regression. *Theoretical Computer Science*, 519:103–126.

Cortes, C., Mohri, M., Gonzalvo, J., and Storcheus, D. (2020). Agnostic learning with multiple objectives. *Advances in Neural Information Processing Systems*, 33.

Cortes, C., Mohri, M., and Muñoz Medina, A. (2015b). Adaptation algorithm and theory based on generalized discrepancy. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 169–178.

Cortes, C., Mohri, M., and Rostamizadeh, A. (2009). L_2 regularization for learning kernels. In *Proceedings of UAI*, pages 109–116.

Cortes, C., Mohri, M., and Rostamizadeh, A. (2010). Generalization bounds for learning kernels. In *Proceedings of ICML*, pages 247–254.

Cortes, C., Mohri, M., and Storcheus, D. (2019). Regularized gradient boosting. *Advances in Neural Information Processing Systems*, 32:5449–5458.

Cortes, C., Mohri, M., and Syed, U. (2014). Deep boosting. In *Proceedings of ICML*.

Cortes, C., Mohri, M., and Weston, J. (2007). A General Regression Framework for Learning String-to-String Mappings. In *Predicting Structured Data*. MIT Press.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273–297.

Dai, W., Yang, Q., Xue, G.-R., and Yu, Y. (2007). Boosting for transfer learning. In *Proceedings of ICML*, pages 874–883.

Daumé III, H., Langford, J., and Marcu, D. (2009). Search-based structured prediction. *Machine Learning*, 75(3):297–325.

Dhillon, P. S., Foster, D. P., Kakade, S. M., and Ungar, L. H. (2013). A risk comparison of ordinary least squares vs ridge regression. *The Journal of Machine Learning Research*, 14(1):1505–1511.

Dietterich, T. G. (2000). An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine learning*, 40(2):139–157.

Doppa, J. R., Fern, A., and Tadepalli, P. (2014). Structured prediction via output space search. *JMLR*, 15(1):1317–1350.

Dua, D. and Graff, C. (2017). UCI machine learning repository.

Duchi, J., Shalev-Shwartz, S., Singer, Y., and Chandra, T. (2008). Efficient projections onto the l 1-ball for learning in high dimensions. In *Proceedings of the 25th international conference on Machine learning*, pages 272–279.

Duh, K., Sudoh, K., Wu, X., Tsukada, H., and Nagata, M. (2012). Learning to translate with multiple objectives. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1–10.

Eban, E., Schain, M., Mackey, A., Gordon, A., Rifkin, R., and Elidan, G. (2017). Scalable learning of non-decomposable objectives. In *Artificial Intelligence and Statistics*, pages 832–840.

Freund, Y., Mansour, Y., and Schapire, R. E. (2004). Generalization bounds for averaged classifiers. *The Annals of Statistics*, 32:1698–1722.

Freund, Y. and Schapire, R. E. (1997a). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer System Sciences*, 55(1):119–139.

Freund, Y. and Schapire, R. E. (1997b). A decision-theoretic generalization of on-line learning and application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139.

Freund, Y., Schapire, R. E., et al. (1996). Experiments with a new boosting algorithm. In *icml*, volume 96, pages 148–156. Citeseer.

Friedman, J., Hastie, T., and Tibshirani, R. (1998). Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28:2000.

Friedman, J., Hastie, T., Tibshirani, R., et al. (2000). Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *Annals of statistics*, 28(2):337–407.

Friedman, J., Hastie, T., Tibshirani, R., et al. (2001). *The elements of statistical learning*, volume 1. Springer series in statistics New York.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232.

Friedman, J. H. (2002). Stochastic gradient boosting. *Computational statistics & data analysis*, 38(4):367–378.

Fukumizu, K., Bach, F. R., and Jordan, M. I. (2004). Dimensionality reduction for supervised learning with reproducing kernel Hilbert spaces. *The Journal of Machine Learning Research*, 5:73–99.

Ganin, Y. and Lempitsky, V. (2015). Unsupervised domain adaptation by backpropagation. In *International conference on machine learning*, pages 1180–1189. PMLR.

Gimpel, K. and Smith, N. A. (2010). Softmax-margin CRFs: Training log-linear models with cost functions. In *Proceedings of ACL*.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587.

Godfrey, P., Shipley, R., and Gryz, J. (2007). Algorithms and analyses for maximal vector computation. *The VLDB Journal*, 16(1):5–28.

Gönen, M. (2014). Coupled dimensionality reduction and classification for supervised and semi-supervised multilabel learning. *Pattern recognition letters*, 38:132–141.

Gönen, M. and Alpaydın, E. (2011). Multiple kernel learning algorithms. *The Journal of Machine Learning Research*, 12:2211–2268.

Gong, B., Shi, Y., Sha, F., and Grauman, K. (2012). Geodesic flow kernel for unsupervised domain adaptation. In *2012 IEEE conference on computer vision and pattern recognition*, pages 2066–2073. IEEE.

Gottlieb, L.-A., Kontorovich, A., and Krauthgamer, R. (2013). Adaptive metric dimensionality reduction. In *Proceedings of ALT*, pages 279–293. Springer.

Graves, A. and Jaitly, N. (2014). Towards end-to-end speech recognition with recurrent neural networks. In *Proceedings of ICML*.

Grubb, A. and Bagnell, J. A. (2011). Generalized boosting algorithms for convex optimization. *arXiv preprint arXiv:1105.2054*.

Habrard, A., Peyrache, J.-P., and Sebban, M. (2013). Boosting for unsupervised domain adaptation. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 433–448. Springer.

Ham, J., Lee, D. D., Mika, S., and Schölkopf, B. (2004). A kernel view of the dimensionality reduction of manifolds. In *Proceedings of ICML*, page 47. ACM.

Hamer, J., Mohri, M., and Suresh, A. T. (2020). Fedboost: A communication-efficient algorithm for federated learning. In *International Conference on Machine Learning*, pages 3973–3983. PMLR.

Hampshire, J. and Waibel, A. (1992). The meta-pi network: building distributed knowledge representations for robust multisource pattern recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(7):751–769.

Hard, A., Rao, K., Mathews, R., Beaufays, F., Augenstein, S., Eichner, H., Kiddon, C., and Ramage, D. (2018). Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604*.

Hardt, M., Price, E., Price, E., and Srebro, N. (2016). Equality of opportunity in supervised learning. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

Hoffman, J., Kulis, B., Darrell, T., and Saenko, K. (2012). Discovering latent domains for multi-source domain adaptation. In *ECCV*, volume 7573, pages 702–715.

Hoffman, J., Mohri, M., and Zhang, N. (2018a). Algorithms and theory for multiple-source adaptation. In *Proceedings of NeurIPS*, pages 8256–8266.

Hoffman, J., Mohri, M., and Zhang, N. (2018b). Algorithms and theory for multiple-source adaptation. In *Advances in Neural Information Processing Systems*, pages 8246–8256.

Hoffman, J., Rodner, E., Donahue, J., Darrell, T., and Saenko, K. (2013). Efficient learning of domain-invariant image representations. *arXiv preprint arXiv:1301.3224*.

Huang, P., Wang, G., and Qin, S. (2010). A novel learning approach to multiple tasks based on boosting methodology. *Pattern recognition letters*, 31(12):1693–1700.

Huang, P., Wang, G., and Qin, S. (2012). Boosting for transfer learning from multiple data sources. *Pattern Recognition Letters*, 33(5):568–579.

Isozaki, H., Hirao, T., Duh, K., Sudoh, K., and Tsukada, H. (2010). Automatic evaluation of translation quality for distant language pairs. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 944–952. Association for Computational Linguistics.

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., and Hinton, G. E. (1991). Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87.

Jin, Y. (2006). *Multi-objective machine learning*, volume 16. Springer Science & Business Media.

Jin, Y. and Sendhoff, B. (2008). Pareto-based multiobjective machine learning: An overview and

case studies. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(3):397–415.

Joachims, T. (2005). A support vector method for multivariate performance measures. In *Proceedings of ICML*.

Joachims, T., Galor, T., and Elber, R. (2006). Learning to align sequences: A maximum-margin approach. In *New algorithms for macromolecular simulation*, pages 57–69. Springer.

Juditsky, A., Nemirovski, A., and Tauvel, C. (2011). Solving variational inequalities with stochastic mirror-prox algorithm. *Stochastic Systems*, 1(1):17–58.

Jurafsky, D. and Martin, J. H. (2009). *Speech and Language Processing (2nd Edition)*. Prentice-Hall, Inc.

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., D'Oliveira, R. G. L., Eichner, H., Rouayheb, S. E., Evans, D., Gardner, J., Garrett, Z., Gascón, A., Ghazi, B., Gibbons, P. B., Gruteser, M., Harchaoui, Z., He, C., He, L., Huo, Z., Hutchinson, B., Hsu, J., Jaggi, M., Javidi, T., Joshi, G., Khodak, M., Konečný, J., Korolova, A., Koushanfar, F., Koyejo, S., Lepoint, T., Liu, Y., Mittal, P., Mohri, M., Nock, R., Özgür, A., Pagh, R., Raykova, M., Qi, H., Ramage, D., Raskar, R., Song, D., Song, W., Stich, S. U., Sun, Z., Suresh, A. T., Tramèr, F., Vepakomma, P., Wang, J., Xiong, L., Xu, Z., Yang, Q., Yu, F. X., Yu, H., and Zhao, S. (2021). Advances and open problems in federated learning.

Kendall, A., Gal, Y., and Cipolla, R. (2018). Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7482–7491.

Kleinberg, J., Mullainathan, S., and Raghavan, M. (2017). Inherent trade-offs in the fair determination of risk scores. In *Innovations in Theoretical Computer Science Conference (ITCS)*.

Kloft, M., Brefeld, U., Sonnenburg, S., and Zien, A. (2011). Lp-norm multiple kernel learning. *The Journal of Machine Learning Research*, 12:953–997.

Kohavi, R. (1996). Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid. In *Kdd*, volume 96, pages 202–207.

Koltchinskii, V. and Panchenko, D. (2002a). Empirical margin distributions and bounding the generalization error of combined classifiers. *Annals of Statistics*, pages 1–50.

Koltchinskii, V. and Panchenko, D. (2002b). Empirical margin distributions and bounding the generalization error of combined classifiers. *Annals of Statistics*, 30.

Konečnỳ, J., McMahan, H. B., Ramage, D., and Richtárik, P. (2016a). Federated optimization: Distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527*.

Konečnỳ, J., McMahan, H. B., Yu, F. X., Richtárik, P., Suresh, A. T., and Bacon, D. (2016b). Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*.

Konstantinov, N. and Lampert, C. (2019). Robust learning from untrusted sources. In *International Conference on Machine Learning*, pages 3488–3498.

Kuznetsov, V., Mohri, M., and Syed, U. (2014). Multi-class deep boosting. In *NIPS*.

Lafferty, J., McCallum, A., and Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*.

Lahoti, P., Beutel, A., Chen, J., Lee, K., Prost, F., Thain, N., Wang, X., and Chi, E. H. (2020). Fairness without demographics through adversarially reweighted learning. *arXiv e-prints*, pages arXiv–2006.

Lam, M., Doppa, J. R., Todorovic, S., and Dietterich, T. G. (2015). C-search for structured prediction in computer vision. In *CVPR*.

Lanckriet, G. R., Cristianini, N., Bartlett, P., Ghaoui, L. E., and Jordan, M. I. (2004). Learning the kernel matrix with semidefinite programming. *The Journal of Machine Learning Research*, 5:27–72.

Lavie, A. and Agarwal, A. (2007). Meteor: An automatic metric for mt evaluation with high levels of correlation with human judgments. In *Proceedings of the second workshop on statistical machine translation*, pages 228–231.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436.

LeCun, Y. and Cortes, C. (2010). MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/.

Ledoux, M. and Talagrand, M. (1991a). *Probability in Banach Spaces: Isoperimetry and Processes*. Springer.

Ledoux, M. and Talagrand, M. (1991b). *Probability in Banach Spaces: Isoperimetry and Processes*. Springer, New York.

Levina, E. and Bickel, P. J. (2004). Maximum likelihood estimation of intrinsic dimension. In *Advances in neural information processing systems*, pages 777–784.

Levy, D., Carmon, Y., Duchi, J. C., and Sidford, A. (2020). Large-scale methods for distributionally robust optimization. *Advances in Neural Information Processing Systems*, 33.

Li, T., Sanjabi, M., Beirami, A., and Smith, V. (2020). Fair resource allocation in federated learning. In *International Conference on Learning Representations*.

Liao, H. (2013). Speaker adaptation of context dependent deep neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 7947–7951. IEEE.

Lin, Y.-Y., Liu, T.-L., and Fuh, C.-S. (2011). Multiple kernel learning for dimensionality reduction. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(6):1147–1160.

Long, M., Cao, Y., Wang, J., and Jordan, M. (2015). Learning transferable features with deep adaptation networks. In *International conference on machine learning*, pages 97–105. PMLR.

Lu, H. and Mazumder, R. (2018). Randomized gradient boosting machine. *arXiv preprint arXiv:1810.10158*.

Lucchi, A., Yunpeng, L., and Fua, P. (2013). Learning for structured prediction using approximate subgradient descent with working sets. In *Proceedings of CVPR*.

Luo, Z.-Q. and Tseng, P. (1992). On the convergence of the coordinate descent method for convex differentiable minimization. *Journal of Optimization Theory and Applications*, 72(1):7–35.

MacKay, D. J. C. (1991). *Bayesian methods for adaptive models*. PhD thesis, California Institute of Technology.

Manning, C. D. and Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts.

Mansour, Y., Mohri, M., and Rostamizadeh, A. (2008). Domain adaptation with multiple sources. *Advances in neural information processing systems*, 21:1041–1048.

Mansour, Y., Mohri, M., and Rostamizadeh, A. (2009a). Domain adaptation: Learning bounds and algorithms. In *COLT*.

Mansour, Y., Mohri, M., and Rostamizadeh, A. (2009b). Multiple source adaptation and the Rényi divergence. In *UAI*, pages 367–374.

Marler, R. T. and Arora, J. S. (2004). Survey of multi-objective optimization methods for engineering. *Structural and multidisciplinary optimization*, 26(6):369–395.

Mason, L., Baxter, J., Bartlett, P. L., and Frean, M. R. (2000). Boosting algorithms as gradient descent. In *Advances in neural information processing systems*, pages 512–518.

Massart, P. (2000). Some applications of concentration inequalities to statistics. *Annales de la Faculté des Sciences de Toulouse*, 9(2):245–303.

Massart, P. and Picard, J. (2007). *Concentration inequalities and model selection: Ecole d'Eté de Probabilités de Saint-Flour XXXIII - 2003*. Number no. 1896 in Ecole d'Eté de Probabilités de Saint-Flour. Springer-Verlag.

McAllester, D. A., Hazan, T., and Keshet, J. (2010). Direct loss minimization for structured prediction. In *Proceedings of NIPS*.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. (2017). Communication-efficient learning of deep networks from decentralized data. In *Proceedings of AISTATS*, pages 1273–1282.

Mohri, M. (1997). Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311.

Mohri, M. (2002). Semiring Frameworks and Algorithms for Shortest-Distance Problems. *Journal of Automata, Languages and Combinatorics*, 7(3):321–350.

Mohri, M. (2003). Edit-distance of weighted automata: General definitions and algorithms. *International Journal of Foundations of Computer Science*, 14(6):957–982.

Mohri, M. (2009). Weighted automata algorithms. In *Handbook of Weighted Automata*, pages 213–254. Springer.

Mohri, M., Rostamizadeh, A., and Storcheus, D. (2015). Generalization bounds for supervised dimensionality reduction. In *Feature Extraction: Modern Questions and Challenges*, pages 226–241. PMLR.

Mohri, M., Rostamizadeh, A., and Talwalkar, A. (2012). *Foundations of machine learning*. MIT press.

Mohri, M., Rostamizadeh, A., and Talwalkar, A. (2018a). *Foundations of machine learning*. MIT press.

Mohri, M., Rostamizadeh, A., and Talwalkar, A. (2018b). *Foundations of Machine Learning*. The MIT Press, second edition.

Mohri, M., Sivek, G., and Suresh, A. T. (2019a). Agnostic federated learning. *CoRR*, abs/1902.00146.

Mohri, M., Sivek, G., and Suresh, A. T. (2019b). Agnostic federated learning. In *International Conference on Machine Learning*, pages 4615–4625. PMLR.

Mosci, S., Rosasco, L., and Verri, A. (2007). Dimensionality reduction and generalization. In *Proceedings of ICML*, pages 657–664. ACM.

Motiian, S., Jones, Q., Iranmanesh, S., and Doretto, G. (2017a). Few-shot adversarial domain adaptation. In *Advances in Neural Information Processing Systems*, pages 6670–6680.

Motiian, S., Piccirilli, M., Adjeroh, D. A., and Doretto, G. (2017b). Unified deep supervised domain adaptation and generalization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5715–5725.

Muandet, K., Balduzzi, D., and Schölkopf, B. (2013). Domain generalization via invariant feature representation. In *International Conference on Machine Learning*, pages 10–18. PMLR.

Nadeau, D. and Sekine, S. (2007). A survey of named entity recognition and classification. *Linguisticae Investigationes*, 30(1):3–26.

Namkoong, H. and Duchi, J. C. (2016). Stochastic gradient methods for distributionally robust optimization with f-divergences. In *Advances in Neural Information Processing Systems*, pages 2208–2216.

Nazarov, F. L. and Podkorytov, A. N. (2000). Ball, haagerup, and distribution functions. In *Complex analysis, operators, and related topics*, pages 247–267. Springer.

Nemirovski, A. S. and Yudin, D. B. (1983). *Problem complexity and Method Efficiency in Optimization*. Wiley.

Nesterov, Y. (2012). Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362.

Norouzi, M., Bengio, S., Jaitly, N., Schuster, M., Wu, Y., and Schuurmans, D. (2016). Reward augmented maximum likelihood for neural structured prediction. In *Proceedings of NIPS*.

Och, F. J. (2003). Minimum error rate training in statistical machine translation. In *Proceedings of ACL*, volume 1.

Pan, S. J. and Yang, Q. (2009). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in PyTorch. In *Proceedings of NIPS*.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., dÁlché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Pearson, K. (1901). Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572.

Pei, Z., Cao, Z., Long, M., and Wang, J. (2018). Multi-adversarial domain adaptation. In *AAAI*, pages 3934–3941.

Poon, H. and Domingos, P. (2011). Sum-product networks: A new deep architecture. In *ICCV Workshops*, pages 689–690.

Prabhavalkar, R., Sainath, T. N., Wu, Y., Nguyen, P., Chen, Z., Chiu, C.-C., and Kannan, A. (2017). Minimum word error rate training for attention-based sequence-to-sequence models. *arXiv preprint arXiv:1712.01818*.

Quinlan, J. R. et al. (1996). Bagging, boosting, and c4. 5. In *AAAI/IAAI, Vol. 1*, pages 725–730.

Raj, J. T. (2019). A beginner's guide to dimensionality reduction in machine learning. `https://towardsdatascience.com/dimensionality-reduction-for-machine-learning-80a46c2ebb7e`.

Ranjbar, M., Lan, T., Wang, Y., Robinovitch, S. N., Li, Z.-N., and Mori, G. (2013). Optimizing nondecomposable loss functions in structured prediction. *IEEE transactions on pattern analysis and machine intelligence*, 35(4):911–924.

Ranzato, M., Chopra, S., Auli, M., and Zaremba, W. (2015). Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*.

Rashmi, K. V. and Gilad-Bachrach, R. (2015). Dart: Dropouts meet multiple additive regression trees. In *AISTATS*, pages 489–497.

Rätsch, G., Mika, S., and Warmuth, M. K. (2001). On the convergence of leveraging. In *NIPS*, pages 487–494.

Ro, J., Chen, M., Mathews, R., Mohri, M., and Suresh, A. T. (2021). Communication-efficient agnostic federated averaging. *arXiv preprint arXiv:2104.02748*.

Roller, B., Taskar, C., and Guestrin, D. (2004). Max-margin markov networks. *Advances in neural information processing systems*, 16:25.

Rosasco, L., Belkin, M., and Vito, E. D. (2010). On learning with integral operators. *The Journal of Machine Learning Research*, 11:905–934.

Ross, S., Gordon, G. J., and Bagnell, D. (2011). A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of AISTATS*.

Roweis, S. T. and Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326.

Saito, K., Kim, D., Sclaroff, S., Darrell, T., and Saenko, K. (2019). Semi-supervised domain adaptation via minimax entropy. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 8050–8058.

Schapire, R. E. (1999). A brief introduction to boosting. In *Ijcai*, volume 99, pages 1401–1406.

Schapire, R. E. and Freund, Y. (2012). *Boosting: Foundations and algorithms*. MIT press.

Schapire, R. E., Freund, Y., Barlett, P., and Lee, W. S. (1997a). Boosting the margin: A new explanation for the effectiveness of voting methods. In *Proceedings of ICML*, pages 322–330.

Schapire, R. E., Freund, Y., Bartlett, P., and Lee, W. S. (1997b). Boosting the margin: A new explanation for the effectiveness of voting methods. In *ICML*, pages 322–330.

Schölkopf, B., Smola, A., and Müller, K.-R. (1997). Kernel principal component analysis. In *Artificial Neural Networks—ICANN'97*, pages 583–588. Springer.

Schölkopf, B., Tsuda, K., and Vert, J.-P. (2004). *Kernel methods in computational biology*. MIT Press, Cambridge, Mass.

Seide, F. and Agarwal, A. (2016). CNTK: Microsoft's open-source deep-learning toolkit. In *Proceedings of KDD*. ACM.

Sener, O. and Koltun, V. (2018). Multi-task learning as multi-objective optimization. In *Advances in Neural Information Processing Systems*, pages 527–538.

Shah, A. and Ghahramani, Z. (2016). Pareto frontier learning with expensive correlated objectives. In *International Conference on Machine Learning*, pages 1919–1927.

Shawe-Taylor, J. and Cristianini, N. (2003). Estimating the moments of a random vector with applications. In *Proceedings of GRETSI*, pages 47–52.

Shen, S., Cheng, Y., He, Z., He, W., Wu, H., Sun, M., and Liu, Y. (2016). Minimum risk training for neural machine translation. In *Proceedings of ACL*, volume 1.

Shen, Z., Hassani, H., Kale, S., and Karbasi, A. (2021). Federated functional gradient boosting. *arXiv preprint arXiv:2103.06972*.

Smyth, P. and Wolpert, D. (1999). Linearly combining density estimators via stacking. *Machine Learning*, 36:59–83.

Stewart, G. and Sun, J. (1990). *Matrix Perturbation Theory*. Academic Press.

Sun, P., Zhang, T., and Zhou, J. (2014). A convergence rate analysis for logitboost, mart and their variant. In *ICML*, pages 1251–1259.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Proceedings of NIPS*.

Taherkhani, A., Cosma, G., and McGinnity, T. M. (2020). Adaboost-cnn: An adaptive boosting algorithm for convolutional neural networks to classify multi-class imbalanced datasets using transfer learning. *Neurocomputing*, 404:351–366.

Taskar, B., Guestrin, C., and Koller, D. (2003). Max-margin Markov networks. In *NIPS*.

Tenenbaum, J. B., De Silva, V., and Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323.

Torralba, A. and Efros, A. A. (2011). Unbiased look at dataset bias. In *CVPR 2011*, pages 1521–1528. IEEE.

Tseng, P. (2001). Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal Optimization Theory and Applications*, pages 475–494.

Tsochantaridis, I., Joachims, T., Hofmann, T., and Altun, Y. (2005). Large margin methods for structured and interdependent output variables. *JMLR*, 6:1453–1484.

Tzeng, E., Hoffman, J., Darrell, T., and Saenko, K. (2015). Simultaneous deep transfer across domains and tasks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4068–4076.

Van Moffaert, K., Brys, T., Chandra, A., Esterle, L., Lewis, P. R., and Nowé, A. (2014). A novel adaptive weight selection algorithm for multi-objective multi-agent reinforcement learning. In *2014 International joint conference on neural networks (IJCNN)*, pages 2306–2314. IEEE.

Vapnik, V. (1992). Principles of risk minimization for learning theory. In *Advances in neural information processing systems*, pages 831–838.

Vinyals, O., Kaiser, L., Koo, T., Petrov, S., Sutskever, I., and Hinton, G. (2015a). Grammar as a foreign language. In *Proceedings of NIPS*.

Vinyals, O., Toshev, A., Bengio, S., and Erhan, D. (2015b). Show and tell: A neural image caption generator. In *Proceedings of CVPR*.

Wang, B., Mendez, J., Cai, M., and Eaton, E. (2019a). Transfer learning via minimizing the performance gap between domains. In *Advances in Neural Information Processing Systems*, pages 10645–10655.

Wang, T., Zhang, X., Yuan, L., and Feng, J. (2019b). Few-shot adaptive faster r-cnn. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7173–7182.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Łukasz Kaiser, Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M., and Dean, J. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

Xiao, H., Rasul, K., and Vollgraf, R. (2017a). Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747.

Xiao, H., Rasul, K., and Vollgraf, R. (2017b). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747.

Xu, R., Chen, Z., Zuo, W., Yan, J., and Lin, L. (2018). Deep cocktail network: Multi-source unsupervised domain adaptation with category shift. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3964–3973.

Xu, Z., Li, W., Niu, L., and Xu, D. (2014). Exploiting low-rank structure from latent domains for domain generalization. In *European Conference on Computer Vision*, pages 628–643. Springer.

Xu, Z. and Sun, S. (2011). Multi-view transfer learning with adaboost. In *2011 IEEE 23rd International Conference on Tools with Artificial Intelligence*, pages 399–402. IEEE.

Xu, Z. and Sun, S. (2012). Multi-source transfer learning with multi-view adaboost. In *International conference on neural information processing*, pages 332–339. Springer.

Yang, J., Yan, R., and Hauptmann, A. G. (2007). Cross-domain video concept detection using adaptive svms. In *Proceedings of the 15th ACM international conference on Multimedia*, pages 188–197.

Yang, Q., Liu, Y., Chen, T., and Tong, Y. (2019). Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–19.

Yang, T., Andrew, G., Eichner, H., Sun, H., Li, W., Kong, N., Ramage, D., and Beaufays, F. (2018). Applied federated learning: Improving google keyboard query suggestions. *arXiv preprint arXiv:1812.02903*.

Yao, Y. and Doretto, G. (2010). Boosting for transfer learning with multiple sources. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 1855–1862. IEEE.

Yuan, Z., Bao, D., Chen, Z., and Liu, M. (2017). Integrated transfer learning algorithm using multi-source tradaboost for unbalanced samples classification. In *2017 international conference on computing intelligence and information system (CIIS)*, pages 188–195. IEEE.

Zhang, D., Sun, L., and Li, W. (2008). A structured prediction approach for statistical machine translation. In *Proceedings of IJCNLP*.

Zhang, H. (2009). Orthogonality from disjoint support in reproducing kernel Hilbert spaces. *Journal of Mathematical Analysis and Applications*, 349(1):201–210.

Zhang, Q., Li, H., Zhang, Y., and Li, M. (2014). Instance transfer learning with multisource dynamic tradaboost. *The scientific world journal*, 2014.

Zhao, H., Zhang, S., Wu, G., Moura, J. M., Costeira, J. P., and Gordon, G. J. (2018). Adversarial multiple source domain adaptation. In *Advances in neural information processing systems*, pages 8559–8570.

Zhao, L., Mammadov, M., and Yearwood, J. (2010). From convex to nonconvex: a loss function analysis for binary classification. In *2010 IEEE International Conference on Data Mining Workshops*, pages 1281–1288. IEEE.

Zhuang, J., Wang, J., Hoi, C. H., and Lan, X. (2011). Unsupervised multiple kernel learning. *Journal of Machine Learning Research (JMLR)*, 20:129–144.

Zwald, L. and Blanchard, G. (2005). On the convergence of eigenspaces in kernel principal component analysis. In *Proceedings of NIPS*, pages 1649–1656.