Online Learning: Theory, Algorithms, and Applications

Thesis submitted for the degree of "Doctor of Philosophy"

by

Shai Shalev-Shwartz

Submitted to the Senate of the Hebrew University

July 2007

This work was carried out under the supervision of

Prof. Yoram Singer

To Moriah and Meitav

קְנֹה חָכְמָה מַה טּוֹב מֵחָרוּץ;

וּקְנוֹת בִּינָה, נִבְחָר מִכָּסֶף.

[משלי ט"ז, ט"ז]

"How much better to get wisdom than gold,
to choose understanding rather than silver"

[Proverbs 16:16]

Acknowledgments

First, I would like to express my deep gratitude to my advisor, Yoram Singer. Yoram has had a profound influence on my research. He exposed me to the fascinating world of online learning. I have learned a variety of research skills from Yoram including scientific writing, basic analysis tools, and (hopefully) good taste in choosing problems. Besides being a good teacher, Yoram is also a great friend.

I would also like to thank the other members of my research committee: Jeff Rosenschein, Mike Werman, and Mira Balaban.

I collaborated and wrote papers with great people. Thank you Yonatan Amit, Koby Crammer, Ofer Dekel, Michael Fink, Joseph Keshet, Andrew Ng, Sivan Sabato, and Nati Srebro.

I am fortunate to have been a member of the machine learning lab at the Hebrew university. I would like to thank former and current members of the lab for being a supportive community and for your friendship. Thank you Ran, Gal E., Gal C., Amir G., Amir N., Adi, Noam, Koby, Lavi, Michael, Eyal, Yossi, Naama, Yonatan R., Yonatan A., Tal, Ofer, Menahem, Tamir, Chen, Talya, Efrat, Kobi, Yoseph, Ariel, Tommy, Naomi, Moran, Matan, Yevgeni, Ohad, and forgive me if I forgot someone.

Being a Ph.D. student at the computer science department at the Hebrew university has been a fantastic experience. I had the pleasure of attending fascinating courses and getting advice from brilliant researchers such as Tali Tishby, Nati Linial, Irit Dinur, Nir Friedman, Yair Weiss, Daphna Weinshall, and Amnon Shashua. I would also like to thank the administrative staff of the computer science department for totally easing the burden of bureaucracy. Special thanks goes to Esther Singer for making a great effort to correct my English at very short notice.

Yair Censor from Haifa University guided me through the world of optimization and row action methods. He has been a major influence on this thesis.

During the summer of 2006 I interned at IBM research labs. I would like to thank Shai Fine, Michal Rosen-Zvi, Sivan Sabato, Hani Neuvirth, Elad Yom-Tov, and the rest of the great people at the machine learning and verification group.

Last but not least, I would like to thank my family and in particular my wife Moria for her support, patience, and love all along the way.

Abstract

Online learning is the process of answering a sequence of questions given knowledge of the correct answers to previous questions and possibly additional available information. Answering questions in an intelligent fashion and being able to make rational decisions as a result is a basic feature of everyday life. Will it rain today (so should I take an umbrella)? Should I fight the wild animal that is after me, or should I run away? Should I open an attachment in an email message or is it a virus? The study of online learning algorithms is thus an important domain in machine learning, and one that has interesting theoretical properties and practical applications.

This dissertation describes a novel framework for the design and analysis of online learning algorithms. We show that various online learning algorithms can all be derived as special cases of our algorithmic framework. This unified view explains the properties of existing algorithms and also enables us to derive several new interesting algorithms.

Online learning is performed in a sequence of consecutive rounds, where at each round the learner is given a question and is required to provide an answer to this question. After predicting an answer, the correct answer is revealed and the learner suffers a loss if there is a discrepancy between his answer and the correct one.

The algorithmic framework for online learning we propose in this dissertation stems from a connection that we make between the notions of *regret* in online learning and *weak duality* in convex optimization. Regret bounds are the common thread in the analysis of online learning algorithms. A regret bound measures the performance of an online algorithm relative to the performance of a competing prediction mechanism, called a competing hypothesis. The competing hypothesis can be chosen in hindsight from a class of hypotheses, after observing the entire sequence of question-answer pairs. Over the years, competitive analysis techniques have been refined and extended to numerous prediction problems by employing complex and varied notions of progress toward a good competing hypothesis.

We propose a new perspective on regret bounds which is based on the notion of duality in convex optimization. Regret bounds are universal in the sense that they hold for any possible fixed hypothesis in a given hypothesis class. We therefore cast the universal bound as a lower bound for an optimization problem, in which we search for the optimal competing hypothesis. While the optimal competing hypothesis can only be found in hindsight, after observing the entire sequence of question-answer pairs, this viewpoint relates regret bounds to lower bounds of minimization problems.

The notion of duality, commonly used in convex optimization theory, plays an important role in obtaining lower bounds for the minimal value of a minimization problem. By generalizing the notion of Fenchel duality, we are able to derive a dual optimization problem, which can be optimized incrementally, as the online learning progresses. The main idea behind our derivation is the connection between regret bounds and Fenchel duality. This connection leads to a reduction from the process of online learning to the task of incrementally ascending the dual objective function.

In order to derive explicit quantitative regret bounds we make use of the weak duality property, which tells us that the dual objective lower bounds the primal objective. The analysis of our algorithmic framework uses the increase in the dual for assessing the progress of the algorithm. This contrasts most if not all previous works that have analyzed online algorithms by measuring the progress of the algorithm based on the correlation or distance between the online hypotheses and a competing hypothesis.

We illustrate the power of our framework by deriving various learning algorithms. Our framework yields the tightest known bounds for several known online learning algorithms. Despite the generality of our framework, the resulting analysis is more distilled than earlier analyses. The framework also serves as a vehicle for deriving various new algorithms. First, we obtain new algorithms for classic prediction problems by utilizing different techniques for ascending the dual objective. We further propose efficient optimization procedures for performing the resulting updates of the online hypotheses. Second, we derive novel algorithms for complex prediction problems, such as ranking and structured output prediction.

The generality of our approach enables us to use it in the batch learning model as well. In particular, we underscore a primal-dual perspective on boosting algorithms, which enables us to analyze boosting algorithms based on the framework. We also describe and analyze several generic online-to-batch conversion schemes.

The proposed framework can be applied in an immense number of possible real-world applications. We demonstrate a successful application of the framework in two different domains. First, we address the problem of online email categorization, which serves as an example of a natural online prediction task. Second, we tackle the problem of speech-to-text and music-to-score alignment. The alignment problem is an example of a complex prediction task in the batch learning model.

Contents

Chapter 1

Introduction

This introduction presents an overview of the online learning model and the contributions of this dissertation. The main concepts introduced here are covered in depth and more rigorously in later chapters.

1.1 Online Learning

Online learning takes place in a sequence of consecutive rounds. On each round, the learner is given a question and is required to provide an answer to this question. For example, a learner might receive an encoding of an email message and the question is whether the email is spam or not. To answer the question, the learner uses a prediction mechanism, termed a hypothesis, which is a mapping from the set of questions to the set of admissible answers. After predicting an answer, the learner gets the correct answer to the question. The quality of the learner's answer is assessed by a loss function that measures the discrepancy between the predicted answer and the correct one. The learner's ultimate goal is to minimize the cumulative loss suffered along its run. To achieve this goal, the learner may update the hypothesis after each round so as to be more accurate in later rounds.

As mentioned earlier, the performance of an online learning algorithm is measured by the cumulative loss suffered by the learning along his run on a sequence of question-answer pairs. We also use the term *example* to denote a question-answer pair. The learner tries to deduce information from previous examples so as to improve its predictions on present and future questions. Clearly, learning is hopeless if there is no correlation between past and present examples. Classic statistical theory of sequential prediction therefore enforces strong assumptions on the statistical properties of the input sequence (for example, it must form a stationary stochastic process).

In most of this dissertation we make no statistical assumptions regarding the origin of the sequence of examples. We allow the sequence to be deterministic, stochastic, or even adversarially adaptive to our own behavior (as in the case of spam email filtering). Naturally, an adversary can make the cumulative loss of our online learning algorithm arbitrarily large. For example, the adversary can ask the same question on each online round, wait for the learner's answer, and provide the opposite answer as the correct answer. To overcome this deficiency, we restate the learner's goal based on the notion of *regret*. To help understand this notion, note that the learner's prediction on each round is based on a hypothesis. The hypothesis is chosen from a predefined class of hypotheses. In this class, we define the optimal fixed hypothesis to be the hypothesis that minimizes the cumulative loss over the entire sequence of examples. The learner's regret is the difference between his cumulative loss and the cumulative loss of the optimal fixed hypothesis. This is termed 'regret' since it measures how 'sorry' the learner is, in retrospect, not to have followed the predictions of the optimal hypothesis. In the example above, where the adversary makes the learner's cumulative loss arbitrarily large, any competing fixed hypothesis would also suffer a large cumulative loss. Thus, the learner's regret in this case would not be large.

This dissertation presents an algorithmic framework for online learning that guarantees low regret. Specifically, we derive several bounds on the regret of the proposed online algorithms. The regret bounds we derive depend on certain properties of the loss functions, the hypothesis class, and the number of rounds we run the online algorithm.

1.2 Taxonomy of Online Learning Algorithms

Before delving into the description of our algorithmic framework for online learning, we would like to highlight connections to and put our work in context of some of the more recent work on online learning. For a more comprehensive overview of relevant publications see Section 1.6 below and the references in the papers cited there. Due to the centrality of the online learning setting, quite a few methods have been devised and analyzed in a diversity of research areas. Here, we focus on the machine learning and pattern recognition field. In this context, the Perceptron algorithm [1, 95, 91] is perhaps the first and simplest online learning algorithm and it will serve as our starting point.

The Perceptron is designed for answering yes/no questions. The algorithm assumes that questions are encoded as vectors in some vector space. We also use the term "instances" to denote the input vectors and the term "labels" to denote the answers. The class of hypotheses used by the Perceptron for predicting answers is the class of linear separators in the vector space. Therefore, each hypothesis can be described using a vector, often called a weight vector. For example, if the vector space is the two dimensional Euclidean space (the plane), then instances are points in the plane and hypotheses are lines. The weight vector is perpendicular to the line. The Perceptron answers

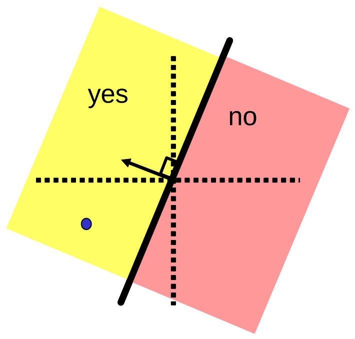

Figure 1.1: An illustration of linear separators in the plane ($\mathbb{R}^2$). The solid black line separates the plane into two regions. The circled point represents an input question. Since it falls into the "yes" region, the predicted answer will be "yes". The arrow designates a weight vector that represents the hypothesis.

"yes" if a point falls on one side of the line and otherwise it answers "no". See Figure 1.2 for an illustration.

The Perceptron updates its weight vector in an *additive* form, by adding (or subtracting) the input instance to the weight vector. In particular, if the predicted answer is negative whereas the true answer is positive then the Perceptron adds the instance vector to the weight vector. If the prediction is positive but the true answer is negative then the instance vector is subtracted from the weight vector. Finally, if the predicted answer is correct then the same weight vector is used in the subsequent round.

Over the years, numerous online learning algorithms have been suggested. The different approaches can be roughly divided into the following categories.

1.2.1 Update Type

Littlestone, Warmuth, Kivinen, and colleagues proposed online algorithms in which the weight vector is updated in a *multiplicative* way. Examples of algorithms that employ multiplicative updates are the Weighed Majority [85], Winnow [82], and Exponentiated Gradient (EG) algorithms [75]. Multiplicative updates are more efficient then additive updates when the instances contain many noisy elements. Later on, Gentile and Littlestone [59] proposed the family of p-norm algorithms that interpolates between additive and multiplicative updates.

This flurry of online learning algorithms sparked unified analyses of seemingly different online algorithms. Most notable is the work of Grove, Littlestone, and Schuurmans [62] on a quasi-additive family of algorithms, which includes both the Perceptron [95] and the Winnow [82] algorithms as special cases. A similar unified view for regression was derived by Kivinen and Warmuth [75, 76].

1.2.2 Problem Type

The Perceptron algorithm was originally designed for answering yes/no questions. In real-world applications we are often interested in more complex answers. For example, in multiclass categorization tasks, the learner needs to choose the correct answer out of k possible answers.

Simple adaptations of the Perceptron for multiclass categorization tasks date back to Kessler's construction [44]. Crammer and Singer [31] proposed more sophisticated variants of the Perceptron for multiclass categorization. The usage of online learning for more complex prediction problems has been further addressed by several authors. Some notable examples are multidimensional regression [76], discriminative training of Hidden Markov Models [23], and ranking problems [28, 29].

1.2.3 Aggressiveness Level

The update procedure used by the Perceptron is extremely simple and is rather conservative. First, no update is made if the predicted answer is correct. Second, all instances are added (subtracted) from the weight vector with a unit weight. Finally, only the most recent example is used for updating the weight vector. Older examples are ignored.

Krauth [78] proposed aggressive variants of the Perceptron in which updates are also performed if the Perceptron's answer is correct but the input instance lies too close to the decision boundary. The idea of trying to push instances away from the decision boundary is central to the Support Vector Machines literature [117, 33, 100]. In addition, various authors [68, 57, 80, 74, 103, 31, 28] suggested using more sophisticated learning rates, i.e., adding instances to the weight vector with different weights.

Finally, early works in game theory derive strategies for playing repeated games in which all past examples are used for updating the hypothesis. The most notable is follow-the-leader approaches [63].

1.3 Main Contributions

In this dissertation we introduce a general framework for the design and analysis of online learning algorithms. Our framework includes various online learning algorithms as special cases and yields the tightest known bounds for these algorithms. This unified view explains the properties of existing algorithms. Moreover, it also serves as a vehicle for deriving and analyzing new interesting online learning algorithms.

Our framework emerges from a new view on regret bounds, which are the common thread in the analysis of online learning algorithms. As mentioned in Section 1.1, a regret bound measures the performance of an online algorithm relative to the performance of a competing hypothesis. The competing hypothesis can be chosen in retrospect from a class of hypotheses, after observing the entire sequence of examples.

We propose an alternative view of regret bounds that is based on the notion of duality in convex optimization. Regret bounds are universal in the sense that they hold for any possible fixed hypothesis in a given hypothesis class. We therefore cast the universal bound as a lower bound for an optimization problem. Specifically, the cumulative loss of the online learner should be bounded above by the minimum value of an optimization problem in which we jointly minimize the cumulative loss and a "complexity" measure of a competing hypothesis. Note that the optimization problem can only be solved in hindsight after observing the entire sequence of examples. Nevertheless, this viewpoint implies that the cumulative loss of the online learner forms a lower bound for a minimization problem.

The notion of duality, commonly used in convex optimization theory, plays an important role in obtaining lower bounds for the minimal value of a minimization problem (see for example [89]). By generalizing the notion of Fenchel duality, we are able to derive a dual optimization problem, which can be optimized incrementally as the online learning progresses. In order to derive explicit quantitative regret bounds we make immediate use of the weak duality property, which tells us that the dual objective lower bounds the primal objective. We therefore reduce the process of online learning to the task of incrementally increasing the dual objective function. The amount by which the dual increases serves as a new and natural notion of progress. By doing so we are able to associate the cumulative loss of the competing hypothesis (as reflected by the primal objective value) and the cumulative loss of the online algorithm, using the increase in the dual.

Different online learning algorithms can be derived from our general framework by varying three components: the complexity function, the type of the loss function, and the dual ascending procedure. These three components correspond to the three categories described in the previous section; namely, to the update type, the problem type, and the aggressiveness level. We now briefly describe this correspondence.

First, recall that in the primal problem we jointly minimize the cumulative loss and the complexity of a competing hypothesis. It turns out that different complexity functions yield different types of updates. For example, using the squared Euclidean norm as a complexity function results in additive updates whereas by using the relative entropy we obtain multiplicative updates. Second, our framework can be used in conjunction with a large family of loss functions. Therefore, by constructing a loss function that fits a particular prediction problem, we immediately obtain online algorithms for solving this prediction problem. Last, the regret bounds we derive for the framework hold as long as we have a sufficient increment in the dual objective. By monitoring the increase in the dual we are able to control the aggressiveness level of the resulting online learning algorithm.

To make this dissertation coherent and due to the lack of space, some of my research work was omitted from this thesis. For example, I have also worked on boosting and online algorithms for regression problems with smooth loss functions [38, 40], online learning of pseudo-metrics [109], online learning of prediction suffix trees [39], online learning with various notions of margin [103], online learning with simultaneous projections [4], online learning with kernels on a budget [41], and stochastic optimization using online learning techniques [110].

1.4 Outline

The dissertation is divided into three main parts, titled *Theory, Algorithms,* and *Applications*. In each part, there are several chapters. The last section of each of the chapters includes a detailed review of previous work relevant to the specific contents described in the chapter.

1.4.1 Part I: Theory

In the theory part, we derive and analyze our algorithmic framework in its most general form. We start in Chapter 2 with a formal description of online learning and regret analysis. We then describe a more abstract framework called online convex programming and cast online learning as a special case of online convex programming. As its name indicates, online convex programming relies on convexity assumptions. We describe a common construction used when the natural loss function for an online learning task is not convex.

Next, in Chapter 3 we derive our algorithmic framework based on the relation between regret bounds and duality. Our presentation assumes some previous knowledge about convex analysis, which can be found in Chapter A in the appendix. We provide a general analysis for all algorithms that can be derived from the framework. To simplify the representation, we start the chapter with a description of a basic algorithmic framework that depends on a parameter. This parameter reflects the trade-off between the cumulative loss of the competing hypothesis and its complexity. We later suggest methods for automatically tuning this parameter. We conclude the chapter by discussing the tightness of our bounds.

The regret bounds we derive in Chapter 3 for our general algorithmic framework grow as the sqrt root of the number of online rounds. While this dependence is tight in the general case, it can be improved by imposing additional assumptions. In Chapter 4 we derive online learning algorithms with logarithmic regret, assuming that the loss functions are strongly convex.

So far, we have focused on the online learning model. Another popular learning model is the PAC learning framework. For completeness, in Chapter B given in the appendix, we discuss the applicability of our algorithmic framework to the PAC learning model. We start this chapter with a short introduction to the PAC learning model. Next, we discuss the relative difficulty of online learning and PAC learning. Finally, we propose general conversion schemes from online learning to the PAC setting.

1.4.2 Part II: Algorithms

The second part is devoted to more specific algorithms and implementation details. We start in Chapter 5 by deriving specific algorithms from our general algorithmic framework. In particular, we demonstrate that by varying the three components of the general framework we can design algorithms with different update types, different aggressiveness levels, and for different problem types.

Next, in Chapter 6 we show the applicability of our analysis for deriving boosting algorithms. While boosting algorithms do not fall under the online learning model, our general analysis fits naturally to general primal-dual incremental methods. As we discuss, the process of boosting can be viewed as a primal-dual game between a weak learner and a booster.

Finally, in Chapter 7 we discuss the computational aspects of the different update schemes. Depending on the loss function and update scheme at hand, we derive procedures for performing the update with increasing computational complexity.

1.4.3 Part III: Applications

In the last part of the dissertation we demonstrate the applicability of our algorithms to real world problems. We start with the problem of online email categorization, which is a natural online learning task. Next, we discuss the problem of alignment. The alignment problem is an example of a complex prediction task. We study the alignment problem in the PAC learning model and use our algorithmic framework for online learning along with the conversion schemes described in the appendix (Chapter B) to construct an efficient algorithm for alignment.

1.5 Notation

We denote scalars with lower case letters (e.g. x and λ), and vectors with bold face letters (e.g. $\mathbf{x}$ and $\boldsymbol{\lambda}$). The ith element of a vector $\mathbf{x}$ is denoted by x_i. Since online learning is performed in a sequence of rounds, we denote by $\mathbf{x}_t$ the tth vector in a sequence of vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T$. The ith element of $\mathbf{x}_t$ is denoted by $x_{t,i}$.

The inner product between vectors $\mathbf{x}$ and $\mathbf{w}$ is denoted by $\langle \mathbf{x}, \mathbf{w} \rangle$. Sets are designated by upper case letters (e.g. S). The set of real numbers is denoted by $\mathbb{R}$ and the set of non-negative real numbers is denoted by $\mathbb{R}_+$. For any $k \geq 1$, the set of integers $\{1, \ldots, k\}$ is denoted by $[k]$. Given a

Table 1.1: Summary of notations.

$\mathbb{R}$	The set of real numbers
$\mathbb{R}_+$	The set of non-negative real numbers
$[k]$	The set $\{1, 2, \ldots, k\}$
S	A set of vectors
x, λ	Scalars
$\mathbf{x}, \boldsymbol{\lambda}$	Vectors
$\mathbf{x}_1, \ldots, \mathbf{x}_T$	A sequence of vectors
$x_{t,i}$	ith element of the vector $\mathbf{x}_t$
$\langle \mathbf{x}, \mathbf{w} \rangle$	inner product
$[\![\pi]\!]$	1 if predicate π holds an 0 otherwise
$[a]_+$	Hinge function: $\max\{0, a\}$
$\|\mathbf{x}\|$,$\|\mathbf{x}\|_\star$	A norm and its dual norm
f, g, h	Functions
$\nabla f(\mathbf{x})$	Gradient of f at $\mathbf{x}$
$\nabla^2 f(\mathbf{x})$	Hessian of f at $\mathbf{x}$
$\partial f(\mathbf{x})$	The differential set of f at $\mathbf{x}$
$f^\star$	The Fenchel conjugate of function f
Z	A random variable
$\mathbb{P}[A]$	Probability that an event A occurs
$\mathbb{E}[Z]$	Expectation of Z

predicate π, we use the notation $[\![\pi]\!]$ to denote the function that outputs 1 if π holds and 0 otherwise. The hinge function is denoted by $[a]_+ = \max\{0, a\}$.

A norm of a vector $\mathbf{x}$ is denoted by $\|\mathbf{x}\|$. The dual norm is defined as $\|\boldsymbol{\lambda}\|_\star = \sup\{\langle \mathbf{x}, \boldsymbol{\lambda} \rangle : \|\mathbf{x}\| \leq 1\}$. For example, the Euclidean norm, $\|\mathbf{x}\|_2 = (\langle \mathbf{x}, \mathbf{x} \rangle)^{1/2}$ is dual to itself and the ℓ_1 norm, $\|\mathbf{x}\|_1 = \sum_i |x_i|$, is dual to the ℓ_∞ norm, $\|\mathbf{x}\|_\infty = \max_i |x_i|$.

Throughout the dissertation, we make extensive use of several notions from convex analysis. In the appendix we overview basic definitions and derive some useful tools. Here we summarize some of our notations. The gradient of a differentiable function f is denoted by ∇f and the Hessian is denoted by $\nabla^2 f$. If f is non-differentiable, we denote its sub-differential set by ∂f. We denote the Fenchel conjugate of a function $f(\mathbf{w})$ by $f^\star(\boldsymbol{\theta}) = \sup_{\mathbf{w}} \langle \mathbf{w}, \boldsymbol{\theta} \rangle - f(\mathbf{w})$ (see Section A.3 in the appendix for more details).

Random variables are designated using upper case letters (e.g. Z). We use the notation $\mathbb{P}[A]$ to denote the probability that an event A occurs. The expected value of a random variable is denoted by $\mathbb{E}[Z]$. In some situations, we have a deterministic function h that receives a random variable as input. We denote by $\mathbb{E}[h(Z)]$ the expected value of the random variable $h(Z)$. Occasionally, we omit the dependence of h on Z. In this case, we clarify the meaning of the expectation by using the notation $\mathbb{E}_Z[h]$.

Table 1.1 provides a summary of our notations.

1.6 Bibliographic Notes

How to predict rationally is a key issue in various research areas such as game theory, machine learning, and information theory. In this section we give a high level overview of related work in different research fields. The last section of each of the chapters below includes a detailed review of previous work relevant to the specific contents of each chapter.

In game theory, the problem of sequential prediction has been addressed in the context of playing repeated games with mixed strategies. A player who can achieve low regret (i.e. whose regret grows sublinearly with the number of rounds) is called a Hannan consistent player [63]. Hannan consistent strategies have been obtained by Hannan [63], Blackwell [9] (in his proof of the approachability theorem), Foster and Vohra [49, 50], Freund and Schapire [55], and Hart and Mas-collel [64]. Von Neumann's classical minimax theorem has been recovered as a simple application of regret bounds [55]. The importance of low regret strategies was further amplified by showing that if all players follow certain low regret strategies then the game converges to a correlated equilibrium (see for example [65, 10]). Playing repeated games with mixed strategies is closely related to the expert setting widely studied in the machine learning literature [42, 82, 85, 119].

Prediction problems have also intrigued information theorists since the early days of the information theory field. For example, Shannon estimated the entropy of the English language by letting humans predict the next symbol in English texts [111]. Motivated by applications of data compression, Ziv and Lempel [124] proposed an online universal coding system for arbitrary individual sequences. In the compression setting, the learner is not committed to a single prediction but rather assigns a probability over the set of possible outcomes. The success of the coding system is measured by the total likelihood of the entire sequence of symbols. Feder, Merhav, and Gutman [47] applied universal coding systems to prediction problems, where the goal is to minimize the number of prediction errors. Their basic idea is to use an estimation of the conditional probabilities of the outcomes given previous symbols, as calculated by the Lempel-Ziv coding system, and then to randomly guess the next symbol based on this conditional probability.

Another related research area is the "statistics without probability" approach developed by Dawid and Vovk [34, 35], which is based on the theory of prediction with low regret.

In an attempt to unify different sequential prediction problems and algorithms, Cesa-Bianchi and Lugosi developed a unified framework called potential-based algorithms [19]. See also their inspiring book [20] about learning, prediction, and games. The potential-based decision strategy formulated by Cesa-Bianchi and Lugosi differs from our construction, which is based on online convex programming [123]. The analysis presented in [19] relies on a generalized Blackwell's condition, which was proposed in [65]. This type of analysis is also similar to the analysis presented by [62] for the quasi-additive family of online learning algorithms. Our analysis is different and is based on the weak duality theorem, the generalized Fenchel duality, and strongly convex functions with respect to arbitrary norms.

Part I

Theory

Chapter 2

Online Convex Programming

2.1 Casting Online Learning as Online Convex Programming

In this chapter we formally define the setting of online learning. We then describe several assumptions under which the online learning setting can be cast as an online convex programming procedure.

Online learning is performed in a sequence of T consecutive rounds. On round t, the learner is first given a question, cast as a vector $\mathbf{x}_t$, and is required to provide an answer to this question. For example, $\mathbf{x}_t$ can be an encoding of an email message and the question is whether the email is spam or not. The learner's prediction is performed based on a hypothesis, $h_t : \mathcal{X} \to \mathcal{Y}$, where $\mathcal{X}$ is the set of questions and Y is the set of possible answers. In the aforementioned example, $\mathcal{Y}$ would be $\{+1, -1\}$ where $+1$ stands for a spam email and -1 stands for a benign one. After predicting an answer, the learner receives the correct answer to the question, denoted y_t, and suffers a loss according to a loss function $\ell(h_t, (\mathbf{x}_t, y_t))$. The function ℓ assesses the quality of the hypothesis h_t on the example $(\mathbf{x}_t, y_t)$. Formally, let $\mathcal{H}$ be the set of all possible hypotheses, then ℓ is a function from $\mathcal{H} \times (\mathcal{X} \times \mathcal{Y})$ into the reals. The ultimate goal of the learner is to minimize the cumulative loss he suffers along his run. To achieve this goal, the learner may choose a new hypothesis after each round so as to be more accurate in later rounds.

In most cases, the hypotheses used for prediction come from a parameterized set of hypotheses, $\mathcal{H} = \{h_{\mathbf{w}} : \mathbf{w} \in S\}$, where S is a subset of a vector space. For example, the set of linear classifiers which is used for answering yes/no questions, is defined as $\mathcal{H} = \{h_{\mathbf{w}}(\mathbf{x}) = \text{sign}(\langle \mathbf{w}, \mathbf{x} \rangle) : \mathbf{w} \in \mathbb{R}^n\}$. Thus, rather than saying that on round t the learner chooses a hypothesis, we can say that the learner chooses a parameter vector $\mathbf{w}_t$ and his hypothesis is $h_{\mathbf{w}_t}$. Next, we note that once the environment chooses a question-answer pair $(\mathbf{x}_t, y_t)$, the loss function becomes a function over the hypothesis space or equivalently over the set of parameter vectors S. We can therefore redefine the online learning process as follows. On round t, the learner chooses a vector $\mathbf{w}_t \in S$, which defines a hypothesis $h_{\mathbf{w}_t}$ to be used for prediction. Then, the environment chooses a question-answer pair $(\mathbf{x}_t, y_t)$, which induces the following loss function over the set of parameter vectors, $g_t(\mathbf{w}) = \ell(h_{\mathbf{w}}, (\mathbf{x}_t, y_t))$. Finally, the learner suffers the loss $g_t(\mathbf{w}_t) = \ell(h_{\mathbf{w}_t}, (\mathbf{x}_t, y_t))$.

Let us further assume that the set of admissible parameter vectors, S, is convex and that the loss functions g_t are convex functions (for an overview of convex analysis see the appendix). Under these assumptions, the online learning process can be described as an online convex programming procedure, defined as follows:

```
For $t = 1, 2, \ldots, T$:
    Learner chooses a vector $\mathbf{w}_t \in S$, where $S$ is a convex set
    Environment responds with a convex function $g_t : S \to \mathbb{R}$
    Outcome of the round is $g_t(\mathbf{w}_t)$
```

Figure 2.1: Online Convex Programming

In offline convex programming, the goal is to find a vector $\mathbf{w}$ within a convex set S that minimizes a convex objective function, $g : S \to \mathbb{R}$. In online convex programming, the set S is known in advance, but the objective function may change along the online process. The goal of the online optimizer, which we call the learner, is to minimize the cumulative objective value

$$\sum_{t=1}^{T} g_t(\mathbf{w}_t) \; .$$

In summary, we have shown that the online learning setting can be cast as the task of online convex programming, assuming that:

1. The hypotheses used for prediction come from a parameterized set of hypotheses, $\mathcal{H} = \{h_{\mathbf{w}} : \mathbf{w} \in S\}$, where S is a convex set.

2. The loss functions, $g_t(\mathbf{w}) = \ell(h_{\mathbf{w}}, (\mathbf{x}_t, y_t))$, are convex functions with respect to $\mathbf{w}$.

In Section 2.3 we discuss the case in which the second assumption above does not hold; namely, the loss functions are non-convex.

We conclude this section with a specific example in which the above assumptions clearly hold. The setting we describe is called online regression with squared loss. In online regression, the set of instances is the n-dimensional Euclidean space, $\mathcal{X} = \mathbb{R}^n$, and the set of targets (possible answers) is the reals, $\mathcal{Y} = \mathbb{R}$. The hypothesis set is

$$\mathcal{H} = \{h_{\mathbf{w}}(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle \mid \mathbf{w} \in \mathbb{R}^n\} \ ,$$

and thus $S = \mathbb{R}^n$. The loss function is defined to be

$$\ell(h_{\mathbf{w}}, (\mathbf{x}, y)) = (\langle \mathbf{w}, \mathbf{x} \rangle - y)^2 \ .$$

It is straightforward to verify that S is a convex set and that for all t, $g_t(\mathbf{w}) = \ell(h_{\mathbf{w}}, (\mathbf{x}_t, y_t))$ is convex with respect to $\mathbf{w}$.

2.2 Regret

As mentioned earlier, the performance of an online learning algorithm is measured by the cumulative loss it suffers along its run on a sequence of examples $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)$. Ideally, we would like to think of the correct answers as having been generated by an unknown yet *fixed* hypothesis h such that $y_t = h(\mathbf{x}_t)$ for all $t \in [T]$. Moreover, in a utopian case, the cumulative loss of h on the entire sequence is zero. In this case, we would like the cumulative loss of our online algorithm to be independent of T. In the more realistic case there is no h that correctly predicts the correct answers of all observed instances. In this case, we would like the cumulative loss of our online algorithm not to exceed by much the cumulative loss of *any* fixed hypothesis h. Formally, we assess the performance of the learner using the notion of *regret*. Given any fixed hypothesis $h \in \mathcal{H}$, we define the regret of an online learning algorithm as the excess loss for not consistently predicting with the hypothesis h,

$$R(h, T) \ = \ \sum_{t=1}^{T} \ell(h_t, (\mathbf{x}_t, y_t)) - \sum_{t=1}^{T} \ell(h, (\mathbf{x}_t, y_t)) \ .$$

Similarly, given any fixed vector $\mathbf{u} \in S$, we define the regret of an online convex programming procedure as the excess loss for not consistently choosing the vector $\mathbf{u} \in S$,

$$R(\mathbf{u}, T) \ = \ \sum_{t=1}^{T} g_t(\mathbf{w}_t) - \sum_{t=1}^{T} g_t(\mathbf{u}) \ .$$

In the next chapter, we present an algorithmic framework for online convex programming that guarantees a regret of $O(\sqrt{T})$ with respect to any vector $\mathbf{u} \in S$. Since we have shown that online learning can be cast as online convex programming, we also obtain a low regret algorithmic framework for online learning.

2.3 Non-convex loss functions and relative mistake bounds

In Section 2.1 we presented a reduction from online learning to online convex programming. This reduction is based on the assumption that for each round t, the loss function $g_t(\mathbf{w}) = \ell(\mathbf{w}, (\mathbf{x}_t, y_t))$ is convex with respect to $\mathbf{w}$. A well known example in which this assumption does not hold is online classification with the 0-1 loss function. In this section we describe the mistake bound model, which extends the utilization of online convex programming to online learning with non-convex loss functions.

For simplicity and for historical reasons, we focus on binary classification problems, in which the set of instances is $\mathcal{X} = \mathbb{R}^n$ and the set of possible answers is $\mathcal{Y} = \{+1, -1\}$. Extending the technique presented below to general non-convex loss functions is straightforward.

Define the hypothesis set of separating hyperplanes,

$$\mathcal{H} = \{h_{\mathbf{w}}(\mathbf{x}) = \text{sign}(\langle \mathbf{w}, \mathbf{x} \rangle) \,|\, \mathbf{w} \in \mathbb{R}^n\} ,$$

and the 0-1 loss function

$$\ell_{\text{0-1}}(h_{\mathbf{w}}, (\mathbf{x}, y)) = \begin{cases} 1 & \text{if } \ h_{\mathbf{w}}(\mathbf{x}) \neq y \\ 0 & \text{if } \ h_{\mathbf{w}}(\mathbf{x}) = y \end{cases}$$

Therefore, the cumulative 0-1 loss of the online learning algorithm is the number of prediction mistakes the online algorithm makes along its run.

We now show that no algorithm can obtain a sub-linear regret bound for the 0-1 loss function. To do so, let $\mathcal{X} = \{1\}$ so our problem boils down to finding the bias of a coin in an online manner. An adversary can make the number of mistakes of any online algorithm to be equal to T, by simply waiting for the learner's prediction and then providing the opposite answer as the true answer. In contrast, the number of mistakes of the constant prediction $u = \text{sign}(\sum_t y_t)$ is at most $T/2$. Therefore, the regret of any online algorithm with respect to the 0-1 loss function will be at least $T/2$.

To overcome the above hardness result, the common solution in the online learning literature is to find a convex loss function that upper bounds the original non-convex loss function. In binary classification, a popular convex loss function is the hinge-loss, defined as

$$\ell_{\text{hinge}}(h_{\mathbf{w}}, (\mathbf{x}, y)) = [1 - y\langle \mathbf{w}, \mathbf{x} \rangle]_+ \quad ,$$

where $[a]_+ = \max\{0, a\}$. It is straightforward to verify that $\ell_{\text{hinge}}(h_{\mathbf{w}}, (\mathbf{x}, y))$ is a convex function and $\ell_{\text{hinge}}(h_{\mathbf{w}}, (\mathbf{x}, y)) \geq \ell_{\text{0-1}}(h_{\mathbf{w}}, (\mathbf{x}, y))$. Therefore, for any $\mathbf{u} \in S$ we have

$$\begin{aligned} R(\mathbf{u}, T) &= \sum_{t=1}^{T} \ell_{\text{hinge}}(h_{\mathbf{w}_t}, (\mathbf{x}_t, y_t)) - \sum_{t=1}^{T} \ell_{\text{hinge}}(h_{\mathbf{u}}, (\mathbf{x}_t, y_t)) \\ &\geq \sum_{t=1}^{T} \ell_{\text{0-1}}(h_{\mathbf{w}_t}, (\mathbf{x}_t, y_t)) - \sum_{t=1}^{T} \ell_{\text{hinge}}(h_{\mathbf{u}}, (\mathbf{x}_t, y_t)) \ . \end{aligned}$$

As a direct corollary from the above inequality we get that a low regret algorithm for the (convex) hinge-loss function can be utilized for deriving an online learning algorithm for the 0-1 loss with the bound

$$\sum_{t=1}^{T} \ell_{\text{0-1}}(h_{\mathbf{w}_t}, (\mathbf{x}_t, y_t)) \ \leq \ \sum_{t=1}^{T} \ell_{\text{hinge}}(h_{\mathbf{u}}, (\mathbf{x}_t, y_t)) + R(\mathbf{u}, T) \ .$$

Denote by $\mathcal{M}$ the set of rounds in which $h_{\mathbf{w}_t}(\mathbf{x}_t) \neq y_t$ and note that the left-hand side of the above is equal to $|\mathcal{M}|$. Furthermore, we can remove the examples not in $\mathcal{M}$ from the sequence of examples, and obtain a bound of the form

$$|\mathcal{M}| \ \leq \ \sum_{t \in \mathcal{M}} \ell_{\text{hinge}}(h_{\mathbf{u}}, (\mathbf{x}_t, y_t)) + R(\mathbf{u}, |\mathcal{M}|) \ . \tag{2.1}$$

Such bounds are called relative mistake bounds.

In the next chapter, we present a low regret algorithmic framework for online learning. In particular, this framework yields an algorithm for online learning with the hinge-loss that attains the regret bound $R(\mathbf{u}, T) = X \, \|\mathbf{u}\| \, \sqrt{T}$, where $X = \max_t \|\mathbf{x}_t\|$. Running this algorithm on the examples in $\mathcal{M}$ results in the famous Perceptron algorithm [95]. Combining the regret bound with Eq. (2.1) we obtain the inequality

$$|\mathcal{M}| \ \leq \ \sum_{t \in \mathcal{M}} \ell_{\text{hinge}}(h_{\mathbf{u}}, (\mathbf{x}_t, y_t)) + X \, \|\mathbf{u}\| \, \sqrt{|\mathcal{M}|} \ ,$$

which implies that (see Lemma 19 in Section A.5)

$$|\mathcal{M}| \ \leq \ \sum_{t \in \mathcal{M}} \ell_{\text{hinge}}(h_{\mathbf{u}}, (\mathbf{x}_t, y_t)) + X \, \|\mathbf{u}\| \sqrt{\sum_{t \in \mathcal{M}} \ell_{\text{hinge}}(h_{\mathbf{u}}, (\mathbf{x}_t, y_t)) + X^2 \, \|\mathbf{u}\|^2} \ .$$

The bound we have obtained is the best relative mistake bound known for the Perceptron algorithm [58, 103, 106].

2.4 Bibliographic notes

The term "online convex programming" was introduced by Zinkevich [123] but this setting was introduced some years earlier by Gordon in [60]. This model is also closely related to the model of relative loss bounds presented by Kivinen and Warmuth [76, 75, 77]. Our presentation of relative mistake bounds follows the works of Littlestone [84], and Kivinen and Warmuth [76].

The impossibility of obtaining a regret bound for the 0-1 loss function dates back to Cover [26], who showed that any online predictor that makes *deterministic* predictions cannot have a vanishing regret universally for all sequences. One way to circumvent this difficulty is to allow the online predictor to make randomized predictions and to analyze its expected regret. In this dissertation we adopt another way and follow the mistake bound model, namely, we slightly modify the regret-based model in which we analyze our algorithm.

Chapter 3

Low Regret and Duality

3.1 Online Convex Programming by Incremental Dual Ascend

In this chapter we present a general low regret algorithmic framework for online convex programming. Recall that the regret of an online convex programming procedure for not consistently choosing the vector $\mathbf{u} \in S$ is defined to be,

$$R(\mathbf{u}, T) \;=\; \sum_{t=1}^{T} g_t(\mathbf{w}_t) - \sum_{t=1}^{T} g_t(\mathbf{u}) \;\; . \tag{3.1}$$

We derive regret bounds that take the following form

$$\forall \mathbf{u} \in S, \;\; R(\mathbf{u}, T) \;\leq\; (f(\mathbf{u}) + L)\,\sqrt{T} \;\; , \tag{3.2}$$

where $f : S \to \mathbb{R}_+$ and $L \in \mathbb{R}_+$. Informally, the function f measures the "complexity" of vectors in S and the scalar L is related to some generalized Lipschitz property of the functions $g_1, \ldots, g_T$. We defer the exact requirements we impose on f and L to later sections.

Our algorithmic framework emerges from a representation of the regret bound given in Eq. (3.2) using an optimization problem. Specifically, we rewrite Eq. (3.2) as follows

$$\sum_{t=1}^{T} g_t(\mathbf{w}_t) \;\leq\; \inf_{\mathbf{u} \in S} \; \sum_{t=1}^{T} g_t(\mathbf{u}) + (f(\mathbf{u}) + L)\,\sqrt{T} \;\; . \tag{3.3}$$

That is, the learner's cumulative loss should be bounded above by the minimum value of an optimization problem in which we jointly minimize the cumulative loss of $\mathbf{u}$ and the "complexity" of $\mathbf{u}$ as measured by the function f. Note that the optimization problem on the right-hand side of

Eq. (3.3) can only be solved in retrospect after observing the entire sequence of loss functions. Nevertheless, writing the regret bound as in Eq. (3.3) implies that the learner's cumulative loss forms a lower bound for a minimization problem.

The notion of duality, commonly used in convex optimization theory, plays an important role in obtaining lower bounds for the minimal value of a minimization problem (see for example [89]). By generalizing the notion of Fenchel duality, we are able to derive a dual optimization problem. While the primal minimization problem of finding the best vector in S can only be solved in hindsight, the dual problem can be optimized incrementally, as the online learning progresses. In order to derive explicit quantitative regret bounds we make immediate use of the fact that dual objective lower bounds the primal objective. We therefore reduce the process of online convex optimization to the task of incrementally increasing the dual objective function. The amount by which the dual increases serves as a new and natural notion of progress. By doing so we are able to link the primal objective value, the learner's cumulative loss, and the increase in the dual.

3.2 Generalized Fenchel Duality

In this section we derive our main analysis tool. First, it should be recalled that the Fenchel conjugate of a function f is the function (see Section A.3)

$$f^\star(\boldsymbol{\theta}) \;=\; \sup_{\mathbf{w} \in S} \langle \mathbf{w}, \boldsymbol{\theta} \rangle - f(\mathbf{w}) \ .$$

In the following, we derive the dual problem of the following optimization problem,

$$\inf_{\mathbf{w} \in S} \left(f(\mathbf{w}) + \sum_{t=1}^{T} g_t(\mathbf{w}) \right) \ .$$

An equivalent problem is

$$\inf_{\mathbf{w}_0, \mathbf{w}_1, \ldots, \mathbf{w}_T} \left(f(\mathbf{w}_0) + \sum_{t=1}^{T} g_t(\mathbf{w}_t) \right) \ \text{ s.t. } \ \mathbf{w}_0 \in S \ \text{ and } \ \forall t \in [T], \ \mathbf{w}_t = \mathbf{w}_0 \ .$$

Introducing T vectors $\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_T$, each $\boldsymbol{\lambda}_t \in \mathbb{R}^n$ is a vector of Lagrange multipliers for the equality constraint $\mathbf{w}_t = \mathbf{w}_0$, we obtain the following Lagrangian

$$\mathcal{L}(\mathbf{w}_0, \mathbf{w}_1, \ldots, \mathbf{w}_T, \boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_T) \;=\; f(\mathbf{w}_0) + \sum_{t=1}^{T} g_t(\mathbf{w}_t) + \sum_{t=1}^{T} \langle \boldsymbol{\lambda}_t, \mathbf{w}_0 - \mathbf{w}_t \rangle \ .$$

The dual problem is the task of maximizing the following dual objective value,

$$\begin{aligned}\mathcal{D}(\boldsymbol{\lambda}_1,\ldots,\boldsymbol{\lambda}_T) &= \inf_{\mathbf{w}_0\in S,\mathbf{w}_1,\ldots,\mathbf{w}_T} \mathcal{L}(\mathbf{w}_0,\mathbf{w}_1,\ldots,\mathbf{w}_T,\boldsymbol{\lambda}_1,\ldots,\boldsymbol{\lambda}_T) \\ &= -\sup_{\mathbf{w}_0\in S}\left(\langle \mathbf{w}_0, -\sum_{t=1}^T \boldsymbol{\lambda}_t\rangle - f(\mathbf{w}_0)\right) - \sum_{t=1}^T \sup_{\mathbf{w}_t}\left(\langle \mathbf{w}_t,\boldsymbol{\lambda}_t\rangle - g_t(\mathbf{w}_t)\right) \\ &= -f^\star\left(-\sum_{t=1}^T \boldsymbol{\lambda}_t\right) - \sum_{t=1}^T g_t^\star\left(\boldsymbol{\lambda}_t\right) \quad ,\end{aligned}$$

where $f^\star, g_1^\star,\ldots,g_T^\star$ are the Fenchel conjugate functions of $f, g_1,\ldots,g_T$. Therefore, the generalized Fenchel dual problem is

$$\sup_{\boldsymbol{\lambda}_1,\ldots,\boldsymbol{\lambda}_T} \; - f^\star\left(-\sum_{t=1}^T \boldsymbol{\lambda}_t\right) - \sum_{t=1}^T g_t^\star(\boldsymbol{\lambda}_t) \quad . \tag{3.4}$$

Note that when $T = 1$ the above duality is the so-called Fenchel duality [11].

3.3 A Low Regret Algorithmic Framework for Online Convex Programming

In this section we describe a template learning algorithm for online convex programming. Recall that we would like our learning algorithm to achieve a regret bound of the form given in Eq. (3.3). We start by rewriting Eq. (3.3) as follows

$$\sum_{t=1}^T g_t(\mathbf{w}_t) - c\,L \;\le\; \inf_{\mathbf{u}\in S}\left(c\,f(\mathbf{u}) + \sum_{t=1}^m g_t(\mathbf{u})\right) \quad , \tag{3.5}$$

where $c = \sqrt{T}$. Thus, up to the sublinear term $c\,L$, the learner's cumulative loss lower bounds the optimum of the minimization problem on the right-hand side of Eq. (3.5). Based on the previous section, the generalized Fenchel dual function of the right-hand side of Eq. (3.5) is

$$\mathcal{D}(\boldsymbol{\lambda}_1,\ldots,\boldsymbol{\lambda}_T) \;=\; -\,c\,f^\star\left(-\tfrac{1}{c}\sum_{t=1}^T \boldsymbol{\lambda}_t\right) - \sum_{t=1}^T g_t^\star(\boldsymbol{\lambda}_t) \quad , \tag{3.6}$$

where we used the fact that the Fenchel conjugate of $c\,f(\mathbf{w})$ is $-cf^\star(\boldsymbol{\theta}/c)$ (see Section A.3.1).

Our construction is based on the weak duality theorem stating that any value of the dual problem is smaller than the optimal value of the primal problem. The algorithmic framework we propose is therefore derived by incrementally ascending the dual objective function. Intuitively, by ascending the dual objective we move closer to the optimal primal value and therefore our performance becomes similar to the performance of the best fixed weight vector which minimizes the right-hand side of Eq. (3.5).

Our algorithmic framework utilizes the following important property of the dual function given in Eq. (3.6): Assume that for all t we have $\inf_{\mathbf{w}} g_t(\mathbf{w}) = 0$ and thus the definition of $g_t^\star$ implies that $g_t^\star(\mathbf{0}) = 0$. Then, for any sequence of t vectors $(\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_t)$ we have

$$\mathcal{D}(\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_t, \mathbf{0}, \ldots, \mathbf{0}) \;=\; -\,c\,f^\star\left(-\tfrac{1}{c}\sum_{i \le t} \boldsymbol{\lambda}_i\right) - \sum_{i \le t} g^\star(\boldsymbol{\lambda}_i) \;\; .$$

That is, if the tail of dual solutions is grounded to zero then the dual objective function does not depend on the *yet to be seen* tail of functions $g_{t+1}, \ldots, g_T$. This contrasts with the primal objective function where we must know all the functions $g_1, \ldots, g_T$ for calculating the primal objective value at some vector $\mathbf{w}$.

Based on the above property, we initially use the elementary dual solution $\boldsymbol{\lambda}_t^1 = \mathbf{0}$ for all t. Since f serves as a measure of "complexity" of vectors in S we enforce the requirement that the minimum of f over the vectors in S is zero. From this requirement we get that $\mathcal{D}(\boldsymbol{\lambda}_1^1, \ldots, \boldsymbol{\lambda}_T^1) = 0$. We assume in addition that f is strongly convex (see Definition 4 in Section A.4). Therefore, based on Lemma 15 in Section A.4, the function $f^\star$ is differentiable. At round t, the learner predicts the vector

$$\mathbf{w}_t \;=\; \nabla f^\star\left(-\frac{1}{c}\sum_{i=1}^{T} \boldsymbol{\lambda}_i^t\right) \;\; . \tag{3.7}$$

After predicting $\mathbf{w}_t$, the environment responds with the function g_t and the learner suffers the loss $g_t(\mathbf{w}_t)$. Finally, the learner finds a new set of dual variables as follows. Denote by ∂_t the differential set of g_t at $\mathbf{w}_t$, that is,

$$\partial_t \;=\; \{\boldsymbol{\lambda} \,:\, \forall \mathbf{w} \in S,\; g_t(\mathbf{w}) - g_t(\mathbf{w}_t) \;\ge\; \langle \boldsymbol{\lambda}, \mathbf{w} - \mathbf{w}_t \rangle\} \;\; . \tag{3.8}$$

The new dual variables $(\boldsymbol{\lambda}_1^{t+1}, \ldots, \boldsymbol{\lambda}_T^{t+1})$ are set to be *any* set of vectors that satisfy the condition:

$$\exists \boldsymbol{\lambda}' \in \partial_t \;\text{ s.t. }\; \mathcal{D}(\boldsymbol{\lambda}_1^{t+1}, \ldots, \boldsymbol{\lambda}_T^{t+1}) \;\ge\; \mathcal{D}(\boldsymbol{\lambda}_1^t, \ldots, \boldsymbol{\lambda}_{t-1}^t, \boldsymbol{\lambda}_t^t + \boldsymbol{\lambda}', \boldsymbol{\lambda}_{t+1}^t, \ldots, \boldsymbol{\lambda}_T^t) \;\; . \tag{3.9}$$

In practice, we cannot calculate the dual objective value at the end of round t unless the last $T - t$ dual vectors are still grounded to zero. In this case, we can rewrite Eq. (3.9) as follows

$$\exists \boldsymbol{\lambda}' \in \partial_t \;\text{ s.t. }\; \mathcal{D}(\boldsymbol{\lambda}_1^{t+1}, \ldots, \boldsymbol{\lambda}_t^{t+1}, \mathbf{0}, \ldots, \mathbf{0}) \;\ge\; \mathcal{D}(\boldsymbol{\lambda}_1^t, \ldots, \boldsymbol{\lambda}_{t-1}^t, \boldsymbol{\lambda}', \mathbf{0}, \ldots, \mathbf{0}) \;\; . \tag{3.10}$$

```
PARAMETERS: A strongly convex function $f : S \to \mathbb{R}$
            A positive scalar $c$
INITIALIZE: $\forall t,\ \boldsymbol{\lambda}_t^1 = \mathbf{0}$
FOR $t = 1, 2, \ldots, T$,
    Set: $\mathbf{w}_t \;=\; \nabla f^\star\left(-\frac{1}{c}\sum_{i=1}^T \boldsymbol{\lambda}_i^t\right)$
    Receive $g_t : S \to \mathbb{R}$
    Choose $(\boldsymbol{\lambda}_1^{t+1}, \ldots, \boldsymbol{\lambda}_T^{t+1})$ that satisfies Eq. (3.9)
```

Figure 3.1: An Algorithmic Framework for Online Convex Programming

Put another way, the increase in the dual objective due to the update step must be at least as large as the increase that would have been obtained had we only set the t variable to be $\boldsymbol{\lambda}'$ instead of its previous value of zero.

A summary of the algorithmic framework is given in Figure 3.1. In the next section we show that the above condition ensures that the increase of the dual at round t is proportional to the loss $g_t(\mathbf{w}_t)$.

We conclude this section with two update rules that trivially satisfy the above condition. The first update scheme simply finds $\boldsymbol{\lambda}' \in \partial_t$ and set

$$\boldsymbol{\lambda}_i^{t+1} \;=\; \begin{cases} \boldsymbol{\lambda}' & \text{if } i = t \\ \boldsymbol{\lambda}_i^t & \text{if } i \neq t \end{cases} \quad . \tag{3.11}$$

The second update defines

$$(\boldsymbol{\lambda}_1^{t+1}, \ldots, \boldsymbol{\lambda}_T^{t+1}) \;=\; \underset{\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_T}{\operatorname{argmax}} \; \mathcal{D}(\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_T) \qquad \text{s.t.} \quad \forall i \neq t,\ \boldsymbol{\lambda}_i = \boldsymbol{\lambda}_i^t \ . \tag{3.12}$$

3.4 Analysis

In this section we analyze the performance of the template algorithm given in the previous section. Our proof technique is based on monitoring the value of the dual objective function. The main result is the following lemma which upper and lower bounds the final value of the dual objective function.

Lemma 1 *Let f be a strongly convex function with respect to a norm $\|\cdot\|$ over a set S and assume that $\min_{\mathbf{w} \in S} f(\mathbf{w}) \geq 0$. Let $\|\cdot\|_\star$ be the dual norm of $\|\cdot\|$. Let $g_1, \ldots, g_T$ be a sequence of convex and closed functions such that $\inf_{\mathbf{w}} g_t(\mathbf{w}) \geq 0$ for all $t \in [T]$. Assume that an algorithm of the form* *given in Figure 3.1 is run on this sequence with the function f. Let $\mathbf{w}_1, \ldots, \mathbf{w}_T$ be the sequence of primal vectors that the algorithm generates and $\boldsymbol{\lambda}_1^{T+1}, \ldots, \boldsymbol{\lambda}_T^{T+1}$ be its final sequence of dual variables. Then, there exists a sequence of sub-gradients $\boldsymbol{\lambda}'_1, \ldots, \boldsymbol{\lambda}'_T$, where $\boldsymbol{\lambda}'_t \in \partial_t$ for all t, such that*

$$\sum_{t=1}^{T} g_t(\mathbf{w}_t) - \frac{1}{2\,c} \sum_{t=1}^{T} \|\boldsymbol{\lambda}'_t\|_\star^2 \;\le\; \mathcal{D}(\boldsymbol{\lambda}_1^{T+1}, \ldots, \boldsymbol{\lambda}_T^{T+1}) \;\le\; \inf_{\mathbf{w}\in S} c\, f(\mathbf{w}) + \sum_{t=1}^{T} g_t(\mathbf{w}) \ .$$

Proof The second inequality follows directly from the weak duality theorem. Turning to the left most inequality, we first note that the assumption $\min_{\mathbf{w}\in S} f(\mathbf{w}) \ge 0$ implies that

$$f^\star(\mathbf{0}) \;=\; \max_{\mathbf{w}\in S} \langle \mathbf{w}, \mathbf{0}\rangle - f(\mathbf{w}) \;=\; -\min_{\mathbf{w}\in S} f(\mathbf{w}) \le 0 \ .$$

Similarly, for all t we have $g_t^\star(\mathbf{0}) \le 0$, and thus $\mathcal{D}(\mathbf{0}, \ldots, \mathbf{0}) \ge 0$. Denote

$$\Delta_t \;=\; \mathcal{D}(\boldsymbol{\lambda}_1^{t+1}, \ldots, \boldsymbol{\lambda}_T^{t+1}) - \mathcal{D}(\boldsymbol{\lambda}_1^{t}, \ldots, \boldsymbol{\lambda}_T^{t})$$

and note that $\mathcal{D}(\boldsymbol{\lambda}_1^{T+1}, \ldots, \boldsymbol{\lambda}_T^{T+1})$ can be rewritten as

$$\mathcal{D}(\boldsymbol{\lambda}_1^{T+1}, \ldots, \boldsymbol{\lambda}_T^{T+1}) \;=\; \sum_{t=1}^{T} \Delta_t \; + \mathcal{D}(\boldsymbol{\lambda}_1^{1}, \ldots, \boldsymbol{\lambda}_T^{1}) \;\ge\; \sum_{t=1}^{T} \Delta_t \ . \tag{3.13}$$

The condition on the update of the dual variables given in Eq. (3.9) implies that

$$\Delta_t \;\ge\; \mathcal{D}(\boldsymbol{\lambda}_1^{t}, \ldots, \boldsymbol{\lambda}_{t-1}^{t}, \boldsymbol{\lambda}_t^{t} + \boldsymbol{\lambda}'_t, \boldsymbol{\lambda}_{t+1}^{t}, \ldots, \boldsymbol{\lambda}_T^{t}) - \mathcal{D}(\boldsymbol{\lambda}_1^{t}, \ldots, \boldsymbol{\lambda}_{t-1}^{t}, \boldsymbol{\lambda}_t^{t}, \boldsymbol{\lambda}_{t+1}^{t}, \ldots, \boldsymbol{\lambda}_T^{t}) \tag{3.14}$$

for some subgradient $\boldsymbol{\lambda}'_t \in \partial_t$. Denoting $\boldsymbol{\theta}_t = -\frac{1}{c}\sum_{j=1}^{T} \boldsymbol{\lambda}_j^t$ and using the definition of $\mathcal{D}$, we can rewrite Eq. (3.14) as,

$$\Delta_t \;\ge -c\, \left(f^\star(\boldsymbol{\theta}_t - \boldsymbol{\lambda}'_t/c) - f^\star(\boldsymbol{\theta}_t) \right) - g_t^\star(\boldsymbol{\lambda}'_t) \ .$$

Lemma 15 in Section A.4 tells us that if f is a strongly convex function w.r.t. a norm $\|\cdot\|$ then for any two vectors $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$ we have

$$f^\star(\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2) - f^\star(\boldsymbol{\mu}_1) \le \langle \nabla f^\star(\boldsymbol{\mu}_1), \boldsymbol{\mu}_2\rangle + \tfrac{1}{2}\|\boldsymbol{\mu}_2\|_\star^2 \ . \tag{3.15}$$

Using this lemma and the definition of $\mathbf{w}_t$ we get that

$$\Delta_t \;\ge\; \langle \nabla f^\star(\boldsymbol{\theta}_t), \boldsymbol{\lambda}'_t\rangle - g_t^\star(\boldsymbol{\lambda}'_t) - \frac{1}{2\,c}\|\boldsymbol{\lambda}'_t\|_\star^2 = \; \langle \mathbf{w}_t, \boldsymbol{\lambda}'_t\rangle - g_t^\star(\boldsymbol{\lambda}'_t) - \frac{1}{2\,c}\|\boldsymbol{\lambda}'_t\|_\star^2 \ . \tag{3.16}$$

Since $\boldsymbol{\lambda}'_t \in \partial_t$ and since we assume that g_t is closed and convex, we can apply Lemma 11 from Section A.3 to get that $\langle \mathbf{w}_t, \boldsymbol{\lambda}'_t \rangle - g_t^\star(\boldsymbol{\lambda}'_t) = g_t(\mathbf{w}_t)$. Plugging this equality into Eq. (3.16) and summing over t we obtain that

$$\sum_{t=1}^{T} \Delta_t \;\geq\; \sum_{t=1}^{T} g_t(\mathbf{w}_t) - \frac{1}{2\,c} \sum_{t=1}^{T} \|\boldsymbol{\lambda}'_t\|_\star^2 \;.$$

Combining the above inequality with Eq. (3.13) concludes our proof. ■

Rearranging the inequality in Lemma 1 we obtain that

$$\forall \mathbf{u} \in S, \quad R(\mathbf{u}, T) \;\leq\; c\, f(\mathbf{u}) + \frac{1}{2\,c} \sum_{t=1}^{T} \|\boldsymbol{\lambda}'_t\|_\star^2 \;. \tag{3.17}$$

To derive a regret bound from the above inequality we need to bound the cumulative sum of $\|\boldsymbol{\lambda}'_t\|_\star^2$. Let us first assume that $\|\cdot\|_\star$ is the Euclidean norm. In this case, we can bound $\|\boldsymbol{\lambda}'_t\|$ by assuming that g_t is a Lipschitz continuous function. More generally, let us define

Definition 1 *A function $g : S \to \mathbb{R}$ is L-Lipschitz with respect to a norm $\|\cdot\|$ if*

$$\forall \mathbf{w} \in S,\ \forall \boldsymbol{\lambda} \in \partial g(\mathbf{w}), \ \ \|\boldsymbol{\lambda}\|_\star \leq L \;.$$

The following regret bound follows as a direct corollary of Lemma 1.

Corollary 1 *Under the same conditions of Lemma 1. Assume that there exists a constant L such that for all t, the function g_t is $\sqrt{2}\,L$-Lipschitz with respect to the norm $\|\cdot\|_\star$. Then, for all $\mathbf{u} \in S$ we have,*

$$R(\mathbf{u}, T) \;=\; \sum_{t=1}^{T} g_t(\mathbf{w}_t) - \sum_{t=1}^{T} g_t(\mathbf{u}) \;\leq\; c\, f(\mathbf{u}) + \frac{L\,T}{c} \;.$$

In particular, for any constant $U > 0$, setting $c = \sqrt{T\,L/U}$ yields the regret bound $R(\mathbf{u}, T) \leq \left(f(\mathbf{u})/\sqrt{U} + \sqrt{U}\right)\sqrt{L\,T}$. Furthermore, if $f(\mathbf{u}) \leq U$ then $R(\mathbf{u}, T) \;\leq\; 2\sqrt{L\,U\,T}$.

The second regret bound we derive is based on a different bound on the norm of sub-gradients. We first need the following definition:

Definition 2 *A function $g : S \to \mathbb{R}$ is L-self-bounded with respect to a norm $\|\cdot\|$ if*

$$\forall \mathbf{w} \in S,\ \exists \boldsymbol{\lambda} \in \partial g(\mathbf{w}), \ \ \frac{1}{2}\|\boldsymbol{\lambda}\|_\star^2 \leq L\, g(\mathbf{w}) \;.$$

Based on the above definition, we are ready to state our second regret bound.

Theorem 1 *Under the same conditions of Lemma 1. Assume that there exists a constant L such that for all t, the function g_t is L-self-bounded with respect to the norm $\|\cdot\|_\star$. Let U_1 and U_2 be two positive scalars and set $c = L + \sqrt{L^2 + L\,U_2/U_1}$. Then, for any $\mathbf{u} \in S$ that satisfies $f(\mathbf{u}) \le U_1$ and $\sum_{t=1}^T g_t(\mathbf{u}) \le U_2$ we have,*

$$R(\mathbf{u},T) \;=\; \sum_{t=1}^T g_t(\mathbf{w}_t) - \sum_{t=1}^T g_t(\mathbf{u}) \;\le\; 2\sqrt{L\,U_1\,U_2} + 4\,L\,U_1 \;\;.$$

Proof Combining Eq. (3.17) with the assumption that $\{g_t\}$ are L-self-bounded yields

$$\sum_{t=1}^T g_t(\mathbf{w}_t) - \sum_{t=1}^T g_t(\mathbf{u}) \;\le\; c f(\mathbf{u}) + \frac{L}{c}\sum_{t=1}^T g_t(\mathbf{w}_t)\;.$$

Rearranging the above we get that

$$\sum_{t=1}^T g_t(\mathbf{w}_t) - \sum_{t=1}^T g_t(\mathbf{u}) \;\le\; \frac{c}{1-\frac{L}{c}}\,f(\mathbf{u}) + \left(\frac{1}{1-\frac{L}{c}} - 1\right)\sum_{t=1}^T g_t(\mathbf{u})\;.$$

Using the assumptions $f(\mathbf{u}) \le U_1$ and $\sum_{t=1}^T g_t(\mathbf{u}) \le U_2$ we obtain

$$R(\mathbf{u},T) \;\le\; \frac{c\,U_1}{1-\frac{L}{c}} + \left(\frac{1}{1-\frac{L}{c}} - 1\right) U_2 \;=\; \frac{U_2}{\frac{c}{L}-1}\left(1 + \frac{c^2}{L^2}\,\frac{U_1\,L}{U_2}\right) \;\;.$$

Plugging the definition of c into the above gives

$$\begin{aligned}
R(\mathbf{u},T) \;&\le\; \frac{U_2}{\sqrt{1+\frac{U_2}{U_1 L}}}\left(1 + \left(1 + \sqrt{1+\frac{U_2}{U_1 L}}\right)^2 \frac{U_1\,L}{U_2}\right)\\
&=\; \frac{U_2}{\sqrt{1+\frac{U_2}{U_1 L}}}\left(1 + \left(2 + 2\sqrt{1+\frac{U_2}{U_1 L}} + \frac{U_2}{U_1 L}\right) \frac{U_1\,L}{U_2}\right)\\
&=\; \frac{2U_2}{\sqrt{1+\frac{U_2}{U_1 L}}} + \frac{2U_1\,L}{\sqrt{1+\frac{U_2}{U_1 L}}} + 2U_1\,L\\
&\le\; \frac{2U_2}{\sqrt{1+\frac{U_2}{U_1 L}}} + 4U_1\,L \;=\; \frac{2U_2\sqrt{\frac{U_1\,L}{U_2}}}{\sqrt{\frac{U_1 L}{U_2}+1}} + 4U_1\,L\\
&\le\; 2\sqrt{U_1\,U_2\,L} + 4U_1\,L \;\;.
\end{aligned}$$

■

The regret bound in Theorem 1 will be small if there exists $\mathbf{u}$ for which both $f(\mathbf{u})$ and $\sum_t g_t(\mathbf{u})$ are small. In particular, setting U_1 to be a constant and $U_2 = \sqrt{T}$ we obtain that $R(\mathbf{u}, T) \leq O(\sqrt{T})$. Note that the bound can be much better if there exists $\mathbf{u}$ with $\sum_t g_t(\mathbf{u}) \ll \sqrt{T}$. Unfortunately, to benefit from this case we must know the upper bound U_2 in advance (since c depends on U_2). Similarly, in Corollary 1 the definition of c depends on the horizon T. In Section 3.5 we propose a method that automatically tunes the parameter c as the online learning progresses.

3.5 Automatically tuning the complexity tradeoff parameter

In the previous sections we described and analyzed an algorithmic framework for online convex programming that achieves low regret provided that the parameter c is appropriately tuned. In this section, we present methods for automatically tuning the parameter c as the online learning progresses.

We start by writing an instantaneous objective function

$$\mathcal{P}_t(\mathbf{w}) = c_t\, f(\mathbf{w}) + \sum_t g_t(\mathbf{w}) \; ,$$

where $0 < c_1 \leq c_2 \leq \ldots$. Using the generalized Fenchel duality derived in Section 3.2 we obtain that the dual objective of $\mathcal{P}_t$ is

$$\mathcal{D}_t(\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_T) = -c_t\, f^\star\left(-\frac{1}{c_t}\sum_{t=1}^{T} \boldsymbol{\lambda}_t\right) - \sum_{t=1}^{T} g_t^\star\left(\boldsymbol{\lambda}_t\right) \; .$$

We now describe an algorithmic framework for online convex programming in which the parameter c is changed from round to round. Initially, we define $\boldsymbol{\lambda}_1^1 = \ldots = \boldsymbol{\lambda}_T^1 = \mathbf{0}$ and $c_0 = 0$. At each round of the online algorithm we choose $c_t \geq c_{t-1}$ and set $\mathbf{w}_t$ to be

$$\mathbf{w}_t \;=\; \nabla f^\star\left(-\frac{1}{c_t}\sum_i \lambda_i^t\right) \; . \tag{3.18}$$

At the end of each online round, we set the new dual variables $(\boldsymbol{\lambda}_1^{t+1}, \ldots, \boldsymbol{\lambda}_T^{t+1})$ to be any set of vectors that satisfy the condition:

$$\exists \boldsymbol{\lambda}' \in \partial_t \text{ s.t. } \mathcal{D}_t(\boldsymbol{\lambda}_1^{t+1}, \ldots, \boldsymbol{\lambda}_T^{t+1}) \;\geq\; \mathcal{D}_t(\boldsymbol{\lambda}_1^t, \ldots, \boldsymbol{\lambda}_{t-1}^t, \boldsymbol{\lambda}_t^t + \boldsymbol{\lambda}', \boldsymbol{\lambda}_{t+1}^t, \ldots, \boldsymbol{\lambda}_T^t) \; . \tag{3.19}$$

```
PARAMETERS: A strongly convex function $f : S \to \mathbb{R}$
INITIALIZE: $c_0 = 0, \quad \forall t, \ \boldsymbol{\lambda}_t^1 = \mathbf{0}$
FOR $t = 1, 2, \ldots, T$,
    Choose $c_t \geq c_{t-1}$
    Set: $\mathbf{w}_t \ = \ \nabla f^\star \left( -\frac{1}{c_t} \sum_{i=1}^T \boldsymbol{\lambda}_i^t \right)$
    Receive $g_t : S \to \mathbb{R}$
    Choose $(\boldsymbol{\lambda}_1^{t+1}, \ldots, \boldsymbol{\lambda}_T^{t+1})$ that satisfies Eq. (3.19)
```

Figure 3.2: An Algorithmic Framework for Online Convex Programming with a variable learning rate.

The algorithmic framework is summarized in Figure 3.2.

The following lemma generalizes Lemma 1 to the case where c varies as learning progresses.

Lemma 2 *Let f be a strongly convex function with respect to a norm $\|\cdot\|$ over a set S and let $\|\cdot\|_\star$ be the dual norm of $\|\cdot\|$. Assume that $\min_{\mathbf{w}\in S} f(\mathbf{w}) \geq 0$. Let $g_1, \ldots, g_T$ be a sequence of convex and closed functions, such that $\inf_{\mathbf{w}} g_t(\mathbf{w}) \geq 0$ for all $t \in [T]$. Assume that an algorithm of the form given in Figure 3.2 is run on this sequence with the function f. Let $\mathbf{w}_1, \ldots, \mathbf{w}_T$ be the sequence of primal vectors that the algorithm generates and $\boldsymbol{\lambda}_1^{T+1}, \ldots, \boldsymbol{\lambda}_T^{T+1}$ be its final sequence of dual variables. Then, there exists a sequence of sub-gradients $\boldsymbol{\lambda}'_1, \ldots, \boldsymbol{\lambda}'_T$, where $\boldsymbol{\lambda}'_t \in \partial_t$ for all t, such that*

$$\forall \mathbf{u} \in S, \quad \sum_{t=1}^T g_t(\mathbf{w}_t) - \sum_{t=1}^T g_t(\mathbf{u}) \ \leq \ c_T\, f(\mathbf{u}) + \frac{1}{2} \sum_{t=1}^T \frac{\|\boldsymbol{\lambda}'_t\|_\star^2}{c_t} \ .$$

Proof We prove the lemma by bounding $\mathcal{D}_T(\boldsymbol{\lambda}_1^{T+1}, \ldots, \boldsymbol{\lambda}_T^{T+1})$ from above and below. The upper bound

$$\mathcal{D}_T(\boldsymbol{\lambda}_1^{T+1}, \ldots, \boldsymbol{\lambda}_T^{T+1}) \ \leq \ c_T f(\mathbf{u}) + \sum_{t=1}^T g_t(\mathbf{u}) \ , \tag{3.20}$$

follows directly from the weak duality theorem. Turning to derive a lower bound, we denote

$$\Delta_t = \mathcal{D}_t(\boldsymbol{\lambda}_1^{t+1}, \ldots, \boldsymbol{\lambda}_T^{t+1}) - \mathcal{D}_{t-1}(\boldsymbol{\lambda}_1^{t}, \ldots, \boldsymbol{\lambda}_T^{t}) \ ,$$

where for simplicity we set $\mathcal{D}_0 = \mathcal{D}_1$. We can now rewrite,

$$\mathcal{D}_T(\boldsymbol{\lambda}_1^{T+1}, \ldots, \boldsymbol{\lambda}_T^{T+1}) \ = \ \sum_{t=1}^T \Delta_t \ + \mathcal{D}_0(\boldsymbol{\lambda}_1^{1}, \ldots, \boldsymbol{\lambda}_T^{1}) \ \geq \ \sum_{t=1}^T \Delta_t \ , \tag{3.21}$$

where the last inequality follows from the assumptions, $\min_{\mathbf{w}\in S} f(\mathbf{w}) \geq 0$ and $\min_{\mathbf{w}\in S} g_t(\mathbf{w}) \geq 0$. Lemma 12 in Section A.3 tells us that conjugate functions preserve order. If a function f_1 is always greater than a function f_2 then $f_1^\star \leq f_2^\star$. Combining this fact with the assumption $c_t \geq c_{t-1}$ implies that for any $\boldsymbol{\theta}$, $c_t\, f^\star(-\boldsymbol{\theta}/c_t) \leq c_{t-1}\, f^\star(-\boldsymbol{\theta}/c_{t-1})$ and thus the definition of $\mathcal{D}$ yields that

$$\mathcal{D}_t(\boldsymbol{\lambda}_1,\ldots,\boldsymbol{\lambda}_T) \geq \mathcal{D}_{t-1}(\boldsymbol{\lambda}_1,\ldots,\boldsymbol{\lambda}_T) \ .$$

Therefore,

$$\Delta_t \geq \mathcal{D}_t(\boldsymbol{\lambda}_1^{t+1},\ldots,\boldsymbol{\lambda}_T^{t+1}) - \mathcal{D}_t(\boldsymbol{\lambda}_1^{t},\ldots,\boldsymbol{\lambda}_T^{t}) \ .$$

As in the proof of Lemma 1, using the definition of $\mathbf{w}_t$, Lemma 15 and Lemma 11 we get that

$$\Delta_t \geq g_t(\mathbf{w}_t) - \frac{1}{2c_t}\|\boldsymbol{\lambda}'_t\|_\star^2 \ .$$

We conclude our proof by summing over t and combining with Eq. (3.21) and Eq. (3.20). ■

Choosing for example $c_t \propto \sqrt{t}$ and using the inequality $\sum_{t=1}^T 1/\sqrt{t} \leq 2\sqrt{T}$ we obtain the following:

Corollary 2 *Under the same conditions of Lemma 2. Assume that there exists a constant L such that for all t, the function g_t is $\sqrt{2\,L}$-Lipschitz with respect to the norm $\|\cdot\|_\star$. Let $U > 0$ be a scalar and for all t set $c_t = \sqrt{t\,L/U}$. Then, for all $\mathbf{u} \in S$ we have,*

$$R(\mathbf{u},T) \ \leq \ 2\left(f(\mathbf{u})/\sqrt{U} + \sqrt{U}\right)\,\sqrt{L\,T} \ .$$

Furthermore, if $f(\mathbf{u}) \leq U$ then $R(\mathbf{u},T) \ \leq \ 4\sqrt{L\,U\,T}$.

The above corollary assumes that the Lipschitz constant L is known in advance. In some situations, the Lipschitz constant of each function g_t is known only at the beginning of round t (although the function itself is only revealed at the end of round t). This will be the case for example in online learning with the hinge-loss, where the Lipschitz constant depends on the instance whereas the function g_t also depends on the unknown label. The following corollary gives an improved parameter selection for this case.

Corollary 3 *Under the same conditions of Lemma 2. Assume that for all t, the function g_t is $\sqrt{2\,L_t}$-Lipschitz with respect to the norm $\|\cdot\|_\star$, where L_t is known to the algorithm at the beginning of round t. Denote $L = \max_t L_t$. Let $U > 0$ be a scalar and for all t set $c_t = \sqrt{t\,\max_{i\leq t} L_t/U}$.*

Then, for all $\mathbf{u} \in S$ we have,

$$R(\mathbf{u}, T) \;\le\; 2\left(f(\mathbf{u})/\sqrt{U} + \sqrt{U}\right)\sqrt{LT}\ .$$

Furthermore, if $f(\mathbf{u}) \le U$ then $R(\mathbf{u}, T) \;\le\; 4\sqrt{LUT}$.

Next, we turn to deriving a regret bound analogous to the bound in Theorem 1, without assuming prior knowledge of the parameter U_2. We first need the following technical lemma.

Lemma 3 *Let $0 = a_0 \le a_1 \le \ldots \le a_T$ be a sequence of scalars and denote $A = \max_t(a_t - a_{t-1})$. Then, for any $\beta \ge 1$ we have*

$$\sum_{t=1}^{T} \frac{a_t - a_{t-1}}{\sqrt{\beta + a_{t-1}}} \;\le\; 2\sqrt{1 + A/\beta}\,(\sqrt{\beta + a_T} - \sqrt{\beta})\ .$$

Proof Using Holder inequality, we get that

$$\begin{aligned}\sum_{t=1}^{T} \frac{a_t - a_{t-1}}{\sqrt{\beta + a_{t-1}}} \;&=\; \sum_{t=1}^{T} \frac{a_t - a_{t-1}}{\sqrt{\beta + a_t}}\sqrt{\frac{\beta + a_t}{\beta + a_{t-1}}} \\ &\le\; \left(\max_t \sqrt{\frac{\beta + a_t}{\beta + a_{t-1}}}\right)\left(\sum_{t=1}^{T} \frac{a_t - a_{t-1}}{\sqrt{\beta + a_t}}\right)\ .\end{aligned}$$

Next, we note that

$$\sum_{t=1}^{T} \frac{a_t - a_{t-1}}{\sqrt{\beta + a_t}} \;\le\; \int_0^{a_T} \frac{1}{\sqrt{\beta + x}}\,dx \;=\; 2(\sqrt{\beta + a_T} - \sqrt{\beta + 0})\ .$$

Finally, for all t we have

$$\frac{\beta + a_t}{\beta + a_{t-1}} \;=\; 1 + \frac{a_t - a_{t-1}}{\beta + a_{t-1}} \;\le\; 1 + \frac{a_t - a_{t-1}}{\beta} \;\le\; 1 + A/\beta\,.$$

Combining the above three equations we conclude our proof. ■

Based on the above lemma, we are ready to provide a regret bound for self-bounded functions.

Theorem 2 *Under the same conditions of Lemma 2. Assume that there exists a constant L such that for all t, the function g_t is L-self-bounded with respect to the norm $\|\cdot\|_\star$. Let $\beta_1 \ge 1$ and*

$\beta_2 > 0$ *be two scalars, and for all* t *define*

$$c_t = \beta_2 \sqrt{\beta_1 + \sum_{i<t} g_i(\mathbf{w}_i)} \ .$$

Denote $G = 2\sqrt{1 + \max_t g_t(\mathbf{w}_t)/\beta_1}$*. Then, for all* $\mathbf{u} \in S$ *we have,*

$$R(\mathbf{u}, T) \ \le \ \left(\beta_2 f(\mathbf{u}) + \frac{LG}{\beta_2}\right)\left(\sum_{t=1}^{T} g_t(\mathbf{u}) + \beta_1\right)^{\frac{1}{2}} \ + \ \left(\beta_2 f(\mathbf{u}) + \frac{LG}{\beta_2}\right)^2 \ .$$

Proof Combining the assumption that $\{g_t\}$ are L-self-bounded with Lemma 2 gives that

$$R(\mathbf{u}, T) \ \le \ c_T f(\mathbf{u}) + L \sum_{t=1}^{T} \frac{g_t(\mathbf{w}_t)}{c_t} \ . \tag{3.22}$$

Denote $a_t = \sum_{i\le t} g_i(\mathbf{w}_i)$ and note that $a_t - a_{t-1} = g_t(\mathbf{w}_t)$. Using Lemma 3 and the definition of c_t we get that

$$\sum_{t=1}^{T} \frac{g_t(\mathbf{w}_t)}{c_t} \ \le \ \frac{2}{\beta_2}\sqrt{a_T(1 + \max_i g_i(\mathbf{w}_i)/\beta_1)} = \frac{G}{\beta_2}\sqrt{a_T} \ \le \ \frac{G}{\beta_2}\sqrt{\beta_1 + a_T} \ .$$

Combining the above with Eq. (3.22), using the fact that $c_T \le \beta_2\sqrt{\beta_1 + a_T}$, and rearranging terms we obtain that

$$(\beta_1 + a_T) - \left(\beta_2 f(\mathbf{u}) + \frac{LG}{\beta_2}\right)\sqrt{\beta_1 + a_T} - \left(\sum_{t=1}^{T} g_t(\mathbf{u}) + \beta_1\right) \ \le \ 0 \ .$$

The above inequality holds only if (see Lemma 19 in Section A.5)

$$\beta_1 + a_T \ \le \ \sum_{t=1}^{T} g_t(\mathbf{u}) + \beta_1 + \left(\beta_2 f(\mathbf{u}) + \frac{LG}{\beta_2}\right)^2 + \left(\beta_2 f(\mathbf{u}) + \frac{LG}{\beta_2}\right)\sqrt{\sum_{t=1}^{T} g_t(\mathbf{u}) + \beta_1} \ .$$

Rearranging the above concludes our proof. ■

The regret bound in the above theorem depends on the variable G, which depends on the maximal value of $g_t(\mathbf{w}_t)$. To ensure that $g_t(\mathbf{w}_t)$ is bounded, one can restrict the set S so that $\max_{\mathbf{w}\in S} g_t(\mathbf{w})$ will be bounded. In this case, we can set β_1 to be $\max_t \max_{\mathbf{w}\in S} g_t(\mathbf{w})$, which gives that $G \le 2\sqrt{2}$, and we obtain the following corollary.

Corollary 4 *Under the same conditions of Theorem 2. Assume that for all t $\max_{\mathbf{w}\in S} g_t(\mathbf{w}) \le \beta_1$ and that $\max_{\mathbf{w}\in S} f(\mathbf{w}) \le U$. Set $\beta_2 = 8^{\frac{1}{4}}\sqrt{L/U}$. Then,*

$$R(\mathbf{u},T) \;\le\; 3.4\sqrt{L\,U}\left(\sum_{t=1}^{T} g_t(\mathbf{u}) + \beta_1\right)^{\frac{1}{2}} \;+\; 11.4\,L\,U\;.$$

3.6 Tightness of regret bounds

In the previous sections we presented algorithmic frameworks for online convex programming with regret bounds that depend on $\sqrt{T}$ and on the complexity of the competing vector as measured by $f(\mathbf{u})$. In this section we show that without imposing additional conditions our bounds are tight, in a sense that is articulated below.

First, we study the dependence of the regret bounds on $\sqrt{T}$.

Theorem 3 *For any online convex programming algorithm, there exists a sequence of 1-Lipschitz convex functions of length T such that the regret of the algorithm on this sequence with respect to a vector $\mathbf{u}$ with $\|\mathbf{u}\|_2 \le 1$ is $\Omega(\sqrt{T})$.*

Proof The proof is based on the probabilistic method [2]. Let $S = [-1,1]$ and $f(w) = \frac{1}{2}w^2$. We clearly have that $f(w) \le 1/2$ for all $w \in S$. Consider a sequence of linear functions $g_t(w) = \sigma_t\, w$ where $\sigma \in \{+1,-1\}$. Note that g_t is 1-Lipschitz for all t. Suppose that the sequence $\sigma_1,\ldots,\sigma_T$ is chosen in advance, independently with equal probability. Since w_t only depends on $\sigma_1,\ldots,\sigma_{t-1}$ it is independent of σ_t. Thus, $\mathbb{E}_{\sigma_t}[g_t(w_t)\,|\,\sigma_1,\ldots,\sigma_{t-1}] = 0$. On the other hand, for any $\sigma_1,\ldots,\sigma_T$ we have that

$$\min_{u\in S}\sum_{t=1}^{T} g_t(u) \;\le\; -\left|\sum_{t=1}^{T}\sigma_t\right| \;.$$

Therefore,

$$\begin{aligned}
\mathbb{E}_{\sigma_1,\ldots,\sigma_T}[R(\mathbf{u},T)] &= \mathbb{E}_{\sigma_1,\ldots,\sigma_T}[\sum_t g_t(w_t)] - \mathbb{E}_{\sigma_1,\ldots,\sigma_T}[\min_u \sum_t g_t(u)] \\
&\ge 0 + \mathbb{E}_{\sigma_1,\ldots,\sigma_T}\left[\,|\sum_t \sigma_t|\,\right] \;.
\end{aligned}$$

The right-hand side above is the expected distance after T steps of a random walk and is thus equal approximately to $\sqrt{2T/\pi} = \Omega(\sqrt{T})$ (see for example [88]). Our proof is concluded by noting that if the expected value of the regret is greater than $\Omega(\sqrt{T})$, then there must exist at least one specific sequence for which the regret is greater than $\Omega(\sqrt{T})$. ■

Next, we study the dependence on the complexity function $f(\mathbf{w})$.

Theorem 4 *For any online convex programming algorithm, there exists a strongly convex function f with respect to a norm $\|\cdot\|$, a sequence of 1-Lipschitz convex functions with respect to $\|\cdot\|_\star$, and a vector $\mathbf{u}$, such that the regret of the algorithm on this sequence is $\Omega(f(\mathbf{u}))$.*

Proof Let $S = \mathbb{R}^T$ and $f(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|_2^2$. Consider the sequence of functions in which $g_t(\mathbf{w}) = [1 - y_t\langle \mathbf{w}, \mathbf{e}_t\rangle]_+$ where $y_t \in \{+1, -1\}$ and $\mathbf{e}_t$ is the tth vector of the standard base, namely, $e_{t,i} = 0$ for all $i \neq t$ and $e_{t,t} = 1$. Note that $g_t(\mathbf{w})$ is 1-Lipschitz with respect to the Euclidean norm. Consider any online learning algorithm. If on round t we have $w_{t,i} \geq 0$ then we set $y_t = -1$ and otherwise we set $y_t = 1$. Thus, $g_t(\mathbf{w}_t) \geq 1$. On the other hand, the vector $\mathbf{u} = (y_1, \ldots, y_T)$ satisfies $f(\mathbf{u}) \leq T/2$ and $g_t(\mathbf{u}) = 0$ for all t. Thus, $R(\mathbf{u}, T) \geq T = \Omega(f(\mathbf{u}))$. ■

3.7 Bibliographic notes

We presented a new framework for designing and analyzing algorithms for online convex programming. The usage of duality for the design of online algorithms was first proposed in [106, 107], in the context of binary classification with the hinge-loss. The generalization of the framework to online convex programming is based on [104]. The self-tuning version we presented in Section 3.5 is new. Specific self-tuning online algorithms were originally proposed by [5]. An alternative way is to use the so called "doubling trick" [21]. The idea of automatically tuning the parameters was also used in the context of prediction suffix trees [39] and in online learning with kernels on a budget [41].

There is a voluminous amount of work on unifying approaches for deriving online learning algorithms. We refer the reader to [62, 76, 20]. By generalizing our framework beyond online learning, we can automatically utilize well known online learning algorithms, such as the EG and p-norm algorithms [76, 62], to the setting of online convex programming.

Zinkevich [123] presented several specific algorithms for online convex programming. His algorithms can be derived as special cases of our algorithmic framework by using the complexity function $f(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2$. Gordon [61] described another algorithmic framework for online convex programming that is closely related to the potential based algorithms described by Cesa-Bianchi and Lugosi [19]. Our work generalizes some of the theorems in [61] while providing a somewhat simpler analysis.

The generality of our approach enables us to analyze boosting algorithms based on the framework (see Chapter 6). Many authors have derived unifying frameworks for boosting algorithms [86, 56, 24]. Nonetheless, our general framework and the connection between online convex programming and the Fenchel duality highlights an interesting perspective on both online learning and boosting. We believe that this viewpoint has the potential of yielding new algorithms in both domains.

The analysis of our algorithmic framework is derived by using the increase in the dual for assessing the progress of the algorithm. Concretely, the regret bounds above were derived by reasoning about the increase in the dual due to the loss suffered by the online algorithm. Most if not all previous works have analyzed online algorithms by measuring the progress of the algorithm based on the correlation or distance between $\mathbf{w}_t$ and a fixed (yet unknown to the online algorithm) competitor, denoted here by $\mathbf{u}$. For example, Novikoff's analysis of the Perceptron [91] is based on the inner product between the competitor and the current predicted vector, $\Delta_t = \langle \mathbf{w}_t, \mathbf{u} \rangle - \langle \mathbf{w}_{t+1}, \mathbf{u} \rangle$. Another progress measure, which has been extensively used in previous analyses of online algorithms, is based on the squared Euclidean distance, $\Delta_t = \|\mathbf{w}_t - \mathbf{u}\|^2 - \|\mathbf{w}_{t+1} - \mathbf{u}\|^2$ (see for example [6, 58, 75, 76, 123] and the references therein). In [107] we show that we can represent these previous definitions of progress as an instantaneous difference of dual objective values by modifying the primal problem.

The probabilistic arguments we use in the proof of Theorem 3, showing the lower bound $\Omega(\sqrt{T})$ for the regret of online convex programming, is based on [66]. The construction we used in the proof of Theorem 4 is widely known for proving the tightness of the Perceptron's mistake bound.

Chapter 4

Logarithmic Regret for Strongly Convex Functions

In Chapter 3 we presented a general algorithmic framework for online convex programming and analyzed its regret. Specifically, we showed that under some mild conditions, the regret is bounded by $O(\sqrt{T})$. Moreover, in Section 3.6 we showed that this dependence is tight. In this chapter we introduce an algorithmic framework that attains logarithmic regret, assuming that each function g_t is *strongly* convex. We start this section with a definition of a generalized notion of strong convexity. Next, we present our algorithmic framework and analyze its regret.

4.1 Generalized Strong Convexity

We have already used the notion of strong convexity with respect to a norm in Chapter 3 for the complexity function f. We now generalize the notion of strong convexity. First, we recall the definition of Bregman divergence. Let f be a differentiable convex function over a set S. Then, f induces the following Bregman divergence over S:

$$B_f(\mathbf{u}\|\mathbf{v}) \;=\; f(\mathbf{u}) - f(\mathbf{v}) - \langle \mathbf{u} - \mathbf{v}, \nabla f(\mathbf{v})\rangle \;. \tag{4.1}$$

For example, the function $f(\mathbf{v}) = \frac{1}{2}\|\mathbf{v}\|_2^2$ yields the divergence $B_f(\mathbf{u}\|\mathbf{v}) = \frac{1}{2}\|\mathbf{u}-\mathbf{v}\|_2^2$. Since f is convex, the Bregman divergence is non-negative.

We recall that a function f is strongly convex over S with respect to a norm $\|\cdot\|$ if

$$\forall \mathbf{u}, \mathbf{v} \in S,\ \forall \boldsymbol{\lambda} \in \partial f(\mathbf{v}),\ \ f(\mathbf{u}) - f(\mathbf{v}) - \langle \mathbf{u} - \mathbf{v}, \boldsymbol{\lambda}\rangle \;\geq\; \frac{\|\mathbf{u}-\mathbf{v}\|^2}{2} \;.$$

```
PARAMETERS: A function $f : S \to \mathbb{R}$ and a scalar $\sigma > 0$
INITIALIZE: $\mathbf{w}_1 \in S$
FOR $t = 1, 2, \ldots, T$
    Predict $\mathbf{w}_t$
    Receive a function $g_t$
    Choose $\boldsymbol{\lambda}_t \in \partial g_t(\mathbf{w}_t)$
    Set $\eta_t = 1/(\sigma\, t)$
    Set $\mathbf{w}_{t+1} = \nabla f^\star\left(\nabla f(\mathbf{w}_t) - \eta_t \boldsymbol{\lambda}_t\right)$
```

Figure 4.1: An algorithmic framework for online strongly convex programming.

(See Section A.4.) A direct generalization of the above definition is achieved by replacing the squared norm at the right-hand side of the inequality with a Bregman divergence.

Definition 3 *A convex function g is strongly convex over S with respect to a convex and differentiable function f if*

$$\forall \mathbf{u}, \mathbf{v} \in S,\ \forall \boldsymbol{\lambda} \in \partial g(\mathbf{v}),\ \ g(\mathbf{u}) - g(\mathbf{v}) - \langle \mathbf{u} - \mathbf{v}, \boldsymbol{\lambda} \rangle \ \geq\ B_f(\mathbf{u} \| \mathbf{v})\ .$$

The following lemma provides a sufficient condition for strong convexity of g w.r.t. f.

Lemma 4 *Assume that f is a differentiable and convex function and let $g = \sigma f + h$ where $\sigma > 0$ and h is also a convex function. Then, g is strongly convex w.r.t. the function $\sigma\, f$.*

Proof Let $\mathbf{v}, \mathbf{u} \in S$ and choose a vector $\boldsymbol{\lambda} \in \partial g(\mathbf{v})$. Since $\partial g(\mathbf{v}) = \partial h(\mathbf{v}) + \partial(\sigma f)(\mathbf{v})$ (see Section A.2), we have that there exists $\boldsymbol{\lambda}_1 \in \partial h(\mathbf{v})$ s.t. $\boldsymbol{\lambda} = \boldsymbol{\lambda}_1 + \sigma\, \nabla f(\mathbf{v})$. Thus,

$$g(\mathbf{u}) - g(\mathbf{v}) - \langle \mathbf{u} - \mathbf{v}, \boldsymbol{\lambda} \rangle = B_{(\sigma f)}(\mathbf{u} \| \mathbf{v}) + h(\mathbf{u}) - h(\mathbf{v}) - \langle \boldsymbol{\lambda}_1, \mathbf{u} - \mathbf{v} \rangle \geq B_{(\sigma f)}(\mathbf{u} \| \mathbf{v})\ ,$$

where the last inequality follows from the convexity of h. ■

4.2 Algorithmic Framework

In Figure 4.1 we describe our algorithmic framework for online strongly convex programming. Before we turn to the analysis of the algorithm, let us first describe a couple of specific examples.

Example 1 *Let S be a closed convex set and let $f(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|_2^2$. The conjugate of f is,*

$$f^\star(\boldsymbol{\theta}) \;=\; \max_{\mathbf{w}\in S} \;\langle \mathbf{w}, \boldsymbol{\theta}\rangle - \frac{1}{2}\|\mathbf{w}\|_2^2 \;=\; \frac{1}{2}\|\boldsymbol{\theta}\|_2^2 - \min_{\mathbf{w}\in S} \frac{1}{2}\|\mathbf{w} - \boldsymbol{\theta}\|_2^2 \,.$$

Based on Lemma 15 we also obtain that $\nabla f^\star(\boldsymbol{\theta})$ is the projection of $\boldsymbol{\theta}$ onto S, that is,

$$\nabla f^\star(\boldsymbol{\theta}) \;=\; \operatorname*{argmin}_{\mathbf{w}\in S} \|\mathbf{w} - \boldsymbol{\theta}\|_2^2 \,.$$

Therefore, the update of our algorithm can be written as

$$\mathbf{w}_{t+1} = \operatorname*{argmin}_{\mathbf{w}\in S} \|\mathbf{w} - (\mathbf{w}_t - \eta_t \boldsymbol{\lambda}_t)\|_2^2 \;\;.$$

When g_t is differentiable, this specific update procedure was originally proposed in [66]. Note that when S is the n'th dimensional ball of radius ρ, $S = \{\mathbf{w} \,|\, \|\mathbf{w}\| \le \rho\}$, the projection of $\boldsymbol{\theta}$ on S amounts to scaling $\boldsymbol{\theta}$ by $\min\{1, \frac{\rho}{\|\boldsymbol{\theta}\|}\}$.

Example 2 *Let S be the n'th dimensional probability simplex,*

$$S = \{\mathbf{w} | \sum_{j=1}^n w_j = 1 \,,\, \forall j : \; w_j \ge 0\} \;\;,$$

and let $f(\mathbf{w}) = \sum_{j=1}^n w_j \log(w_j) + \log(n)$. The conjugate of f is,

$$\begin{aligned} f^\star(\boldsymbol{\theta}) \;&=\; \max_{\mathbf{w}\in S} \;\langle \mathbf{w}, \boldsymbol{\theta}\rangle - \sum_{j=1}^n w_j \log(w_j) - \log(n) \\ &=\; -\log(n) - \min_{\mathbf{w}\in S} \sum_{j=1}^n w_j \log\left(\frac{w_j}{e^{\theta_j}}\right) \;\;. \end{aligned}$$

Using again Lemma 15 we obtain that $\nabla f^\star(\boldsymbol{\theta})$ is the (entropic) projection of $\boldsymbol{\theta}$ onto the simplex S, that is,

$$\nabla f^\star(\boldsymbol{\theta}) \;=\; \operatorname*{argmin}_{\mathbf{w}\in S} \sum_{j=1}^n w_j \log\left(\frac{w_j}{e^{\theta_j}}\right) \;\;.$$

Algebraic manipulations yield that the update $\mathbf{w}_{t+1} = \nabla f^\star\left(\nabla f(\mathbf{w}_t) - \eta_t \boldsymbol{\lambda}_t\right)$, when $\nabla f^\star(\boldsymbol{\theta})$ is of the above form, can be written as follows,

$$w_{t+1,j} = \frac{w_{t,j} e^{-\eta_t \lambda_{t,j}}}{Z_t} \quad \textit{where} \;\; Z_t = \sum_{r=1}^n w_{t,r} e^{-\eta_t \lambda_{t,r}} \;\;.$$

In this case we recovered a version of the multiplicative update [75, 85, 82].

4.3 Analysis

We start with the following lemma.

Lemma 5 *Let f be a strongly convex function w.r.t. a norm $\|\cdot\|$ over S and $\mathbf{u}$ be an arbitrary vector in S. Then,*

$$\langle \mathbf{w}_t - \mathbf{u}, \boldsymbol{\lambda}_t \rangle \le \frac{B_f(\mathbf{u}\|\mathbf{w}_t) - B_f(\mathbf{u}\|\mathbf{w}_{t+1})}{\eta_t} + \eta_t \frac{\|\boldsymbol{\lambda}_t\|_\star^2}{2} .$$

Proof Denote $\Delta_t = B_f(\mathbf{u}\|\mathbf{w}_t) - B_f(\mathbf{u}\|\mathbf{w}_{t+1})$. Expanding the definition of B_f we have that

$$\Delta_t = \langle \mathbf{u} - \mathbf{w}_{t+1}, \nabla f(\mathbf{w}_{t+1}) - \nabla f(\mathbf{w}_t) \rangle + B_f(\mathbf{w}_{t+1}\|\mathbf{w}_t) .$$

Since f is strongly convex w.r.t. $\|\cdot\|$ we have that

$$\Delta_t \ge \langle \mathbf{u} - \mathbf{w}_{t+1}, \nabla f(\mathbf{w}_{t+1}) - \nabla f(\mathbf{w}_t) \rangle + \frac{1}{2}\|\mathbf{w}_{t+1} - \mathbf{w}_t\|^2 . \tag{4.2}$$

Let us denote by $\boldsymbol{\theta}_t$ the term $\nabla f(\mathbf{w}_t) - \eta_t \boldsymbol{\lambda}_t$. Using the last part of Lemma 15 with $\boldsymbol{\theta}$ set to be $\boldsymbol{\theta}_t$ and rewriting $\nabla f^\star(\boldsymbol{\theta}_t)$ as $\mathbf{w}_{t+1}$ (see Figure 4.1) we get that,

$$\begin{aligned} 0 &\ge \langle \mathbf{u} - \nabla f^\star(\boldsymbol{\theta}_t), \boldsymbol{\theta}_t - \nabla f(\nabla f^\star(\boldsymbol{\theta}_t)) \rangle \\ &= \langle \mathbf{u} - \mathbf{w}_{t+1}, \boldsymbol{\theta}_t - \nabla f(\mathbf{w}_{t+1}) \rangle \\ &= \langle \mathbf{u} - \mathbf{w}_{t+1}, \nabla f(\mathbf{w}_t) - \eta_t \boldsymbol{\lambda}_t - \nabla f(\mathbf{w}_{t+1}) \rangle \ . \end{aligned}$$

Rearranging terms we we obtain that

$$\langle \mathbf{u} - \mathbf{w}_{t+1}, \nabla f(\mathbf{w}_{t+1}) - \nabla f(\mathbf{w}_t) \rangle \ \ge \ \eta_t \langle \mathbf{w}_{t+1} - \mathbf{u}, \boldsymbol{\lambda}_t \rangle .$$

Combining the above with Eq. (4.2) gives that

$$\begin{aligned} \Delta_t &\ge \eta_t \langle \mathbf{w}_{t+1} - \mathbf{u}, \boldsymbol{\lambda}_t \rangle + \frac{1}{2}\|\mathbf{w}_{t+1} - \mathbf{w}_t\|^2 \\ &= \eta_t \langle \mathbf{w}_t - \mathbf{u}, \boldsymbol{\lambda}_t \rangle - \langle \mathbf{w}_{t+1} - \mathbf{w}_t, \eta_t \boldsymbol{\lambda}_t \rangle + \frac{1}{2}\|\mathbf{w}_{t+1} - \mathbf{w}_t\|^2 . \end{aligned}$$

Applying the inequality

$$|\langle \mathbf{v}, \boldsymbol{\theta} \rangle| \leq \|\mathbf{v}\| \, \|\boldsymbol{\theta}\|_\star = \frac{1}{2}\|\mathbf{v}\|^2 + \frac{1}{2}\|\boldsymbol{\theta}\|_\star^2 - \frac{1}{2}\left(\|\mathbf{v}\| - \|\boldsymbol{\theta}\|_\star\right)^2 \leq \frac{1}{2}\|\mathbf{v}\|^2 + \frac{1}{2}\|\boldsymbol{\theta}\|_\star^2 \, ,$$

which holds for all $\mathbf{v}$ and $\boldsymbol{\theta}$, we obtain that

$$\begin{aligned} \Delta_t \;&\geq\; \eta_t \langle \mathbf{w}_t - \mathbf{u}, \boldsymbol{\lambda}_t \rangle - \frac{1}{2}\|\mathbf{w}_{t+1} - \mathbf{w}_t\|^2 - \frac{1}{2}\|\eta_t \boldsymbol{\lambda}_t\|_\star^2 + \frac{1}{2}\|\mathbf{w}_{t+1} - \mathbf{w}_t\|^2 \\ &= \eta_t \langle \mathbf{w}_t - \mathbf{u}, \boldsymbol{\lambda}_t \rangle - \frac{\eta_t^2}{2}\|\boldsymbol{\lambda}_t\|_\star^2 \, . \end{aligned}$$

■

Theorem 5 *Let f be a strongly convex function w.r.t. a norm $\|\cdot\|$ over S. Let $\sigma > 0$ and assume that for all t, $\frac{1}{\sigma} g_t$ is a strongly convex function w.r.t. f. Additionally, let L be a scalar such that $\frac{1}{2}\|\boldsymbol{\lambda}_t\|_\star^2 \leq L$ for all t. Then, the following bound holds for all $T \geq 1$,*

$$\sum_{t=1}^{T} g_t(\mathbf{w}_t) - \sum_{t=1}^{T} g_t(\mathbf{u}) \;\leq\; \frac{L}{\sigma}\left(1 + \log(T)\right) .$$

Proof From Lemma 5 we have that

$$\langle \mathbf{w}_t - \mathbf{u}, \boldsymbol{\lambda}_t \rangle \leq \frac{B_f(\mathbf{u}\|\mathbf{w}_t) - B_f(\mathbf{u}\|\mathbf{w}_{t+1})}{\eta_t} + \eta_t \frac{\|\boldsymbol{\lambda}_t\|_\star^2}{2} \, . \tag{4.3}$$

Since $\frac{1}{\sigma} g_t$ is strongly convex w.r.t. f we can bound the left-hand side of the above inequality as follows,

$$\langle \mathbf{w}_t - \mathbf{u}, \boldsymbol{\lambda}_t \rangle \geq g_t(\mathbf{w}_t) - g_t(\mathbf{u}) + \sigma B_f(\mathbf{u}\|\mathbf{w}_t) \, . \tag{4.4}$$

Combining Eq. (4.3) with Eq. (4.4), and using the assumption $\frac{1}{2}\|\boldsymbol{\lambda}_t\|_\star^2 \leq L$ we get that

$$g_t(\mathbf{w}_t) - g_t(\mathbf{u}) \;\leq\; \left(\frac{1}{\eta_t} - \sigma\right) B_f(\mathbf{u}\|\mathbf{w}_t) - \frac{1}{\eta_t} B_f(\mathbf{u}\|\mathbf{w}_{t+1}) + \eta_t L \, .$$

Summing over t we obtain

$$\begin{aligned} \sum_{t=1}^{T} \left(g_t(\mathbf{w}_t) - g_t(\mathbf{u})\right) \;&\leq\; \left(\frac{1}{\eta_1} - \sigma\right) B_f(\mathbf{u}\|\mathbf{w}_1) - \left(\frac{1}{\eta_T}\right) B_f(\mathbf{u}\|\mathbf{w}_{T+1}) + \\ &\qquad \sum_{t=2}^{T} B_f(\mathbf{u}\|\mathbf{w}_t)\left(\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}} - \sigma\right) + L \sum_{t=1}^{T} \eta_t \, . \end{aligned}$$

Plugging the value of η_t into the above inequality, we obtain that the first and third summands on the right-hand side of the inequality vanish and that the second summand is negative. We therefore get,

$$\sum_{t=1}^{T} (g_t(\mathbf{w}_t) - g_t(\mathbf{u})) \ \leq \ L \sum_{t=1}^{T} \eta_t \ = \ \frac{L}{\sigma} \sum_{t=1}^{T} \frac{1}{t} \ \leq \ \frac{L}{\sigma} (\log(T) + 1) \ .$$

■

4.4 Bibliographic Notes

Hazan et al. [66] introduced the novel idea of using strong convexity assumptions for achieving logarithmic regret. They also described three algorithms for strongly convex functions and our algorithmic framework generalizes their first algorithm.

Part II

Algorithms

Chapter 5

Derived Algorithms

In chapter 3 we introduced a general framework for online convex programming based on the notion of duality. In this chapter we derive various online learning algorithms from the general framework. Algorithms that derive from the framework may vary in one of three ways. First, we can use different complexity functions, $f(\mathbf{w})$, which induces different mappings between the primal and dual variables. In Section 5.1 we demonstrate the effect of $f(\mathbf{w})$ by deriving the family of quasi-additive algorithms [62] from the framework. We show that our analysis produces the best known mistake bounds for several known quasi-additive algorithms such as the Perceptron, Winnow, and p-norm algorithms. We also use our self-tuning technique from Section 3.5 to derive a self-tuned variant of the Winnow algorithm for binary classification problems. Second, different algorithms may use different update schemes of the dual variables. We illustrate this fact in Section 5.2 by deriving novel online learning algorithms based on a hierarchy of dual ascending methods. The third way in which different algorithms may vary is the form the functions $\{g_t\}$ take. In Section 5.3 we exemplify this by deriving online learning algorithms for complex prediction problems.

5.1 Quasi-additive Online Classification Algorithms

In this section we analyze the family of quasi-additive online algorithms described in [62, 75, 76] using the algorithmic framework given in chapter 3. The family of quasi-additive algorithms includes several known algorithms such as the Perceptron algorithm [95], Balanced-Winnow [62], and the family of p-norm algorithms [58]. Quasi-additive algorithms are based on a link function, denoted $\mathcal{L}\text{ink} : \mathbb{R}^n \to \mathbb{R}^n$, and are summarized in Figure 5.1.

We now derive the quasi-additive family of algorithms from our algorithmic framework given in Figure 3.1. To do so, we choose a complexity function[1] $f : S \to \mathbb{R}$ so that $\mathcal{L}\text{ink} \equiv \nabla f^\star$. Denote

[1] Any strongly convex function over S with respect to some norm can be a complexity function (see Chapter 3).

```
PARAMETERS: Link function $\mathcal{L}\text{ink} : \mathbb{R}^n \to \mathbb{R}^n$
    Learning rate $c > 0$
INITIALIZE: $\boldsymbol{\theta}_1 = \mathbf{0}$
FOR $t = 1, 2, \ldots, T$
    Set $\mathbf{w}_t = \mathcal{L}\text{ink}(\boldsymbol{\theta}_t)$
    Receive an instance vector $\mathbf{x}_t$
    Predict $\hat{y}_t = \text{sign}(\langle \mathbf{w}_t, \mathbf{x}_t \rangle)$
    Receive the correct label $y_t$
    IF $\hat{y}_t \neq y_t$
        $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \frac{1}{c}\, y_t \mathbf{x}_t$
    ELSE
        $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t$
```

	$\mathcal{L}\text{ink}_i(\boldsymbol{\theta})$
Perceptron	θ_i
Winnow	$\exp(\theta_i)$
p-norm	$\frac{\text{sign}(\theta_i)\, \lvert\theta_i\rvert^{p-1}}{\lVert\boldsymbol{\theta}\rVert_p^{p-2}}$

Figure 5.1: Left: The Quasi-Additive Family of Online Learning Algorithms for Binary Classification. Right: Several known members of the quasi-additive family.

$\mathcal{M} = \{t \in [T] : \hat{y}_t \neq y_t\}$ to be the set of rounds in which the algorithm makes a prediction mistake. Since the algorithm does not change its vector whenever $t \notin \mathcal{M}$ we can simply ignore the rounds not in $\mathcal{M}$ (see the discussion in Section 2.3). For a round $t \in \mathcal{M}$, we set

$$g_t(\mathbf{w}) \;=\; [\gamma - y_t \langle \mathbf{w}_t, \mathbf{x}_t \rangle]_+ \quad , \tag{5.1}$$

where $[a]_+ = \max\{0, a\}$ is the hinge function and $\gamma > 0$ is a parameter. Finally, we use the update scheme given in Eq. (3.11) to ascend the dual objective.

We now prove the equivalence between the above family of algorithms and the quasi-additive family given in Figure 5.1. To do so, it suffices to show that for each round t, the vector $\boldsymbol{\theta}_t$ defined in Figure 5.1 equals to $-\frac{1}{c}\sum_{i=1}^{T} \boldsymbol{\lambda}_i^t$. This claim can be proven using a simple inductive argument. To simplify notation, denote $\mathbf{v}_t = \sum_{i=1}^{T} \boldsymbol{\lambda}_i^t$. Initially, $\boldsymbol{\theta}_1 = -\frac{1}{c}\mathbf{v}_1 = 0$ and the claim clearly holds. Assume that $\boldsymbol{\theta}_t = -\frac{1}{c}\mathbf{v}_t$. If $t \notin \mathcal{M}$ then $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t$ and $\mathbf{v}_{t+1} = \mathbf{v}_t$ and the claim holds for the round $t+1$ as well. Otherwise, the function g_t is differentiable at $\mathbf{w}_t$ and its gradient at $\mathbf{w}_t$ is $-y_t \mathbf{x}_t$. Thus, $\mathbf{v}_{t+1} = \mathbf{v}_t - y_t \mathbf{x}_t$, which implies that $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \frac{1}{c} y_t \mathbf{x}_t = -\frac{1}{c}(\mathbf{v}_t - y_t \mathbf{x}_t) = -\frac{1}{c}\mathbf{v}_{t+1}$. This concludes our inductive argument.

To analyze the quasi-additive family of algorithms we note that for $t \in \mathcal{M}$ we have $g_t(\mathbf{w}_t) \geq \gamma$. In addition, the functions $g_t(\mathbf{w})$ are clearly X-Lipschitz where $X = \max_{t \in \mathcal{M}} \lVert \mathbf{x}_t \rVert_\star$. Thus, the regret bound in Corollary 1 implies that

$$\gamma\, |\mathcal{M}| \;\leq\; c\, f(\mathbf{u}) + \frac{X^2\, |\mathcal{M}|}{2\, c} + \sum_{t \in |\mathcal{M}|} g_t(\mathbf{u}) \quad . \tag{5.2}$$

The above inequality can yield a mistake bound if we appropriately choose the parameters c and γ.

We now provide concrete analyses for specific complexity functions f. For each choice of f we derive the specific form the update takes and briefly discuss the implications of the resulting mistake bounds.

5.1.1 Perceptron

The Perceptron algorithm [95] can be derived from Figure 5.1 by setting $\mathcal{L}\mathrm{ink}$ to be the identity function. This is equivalent to defining $f(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2$ over the domain $S = \mathbb{R}^n$. To see this, we note that the conjugate of f is $f^\star(\boldsymbol{\theta}) = \frac{1}{2}\|\boldsymbol{\theta}\|^2$ (see Section A.3.1) and thus $\nabla f^\star(\boldsymbol{\theta}) = \boldsymbol{\theta}$. The update $\mathbf{w}_{t+1} = \mathcal{L}\mathrm{ink}(\boldsymbol{\theta}_{t+1})$ thus amounts to, $\mathbf{w}_{t+1} = \mathbf{w}_t + \frac{1}{c}\, y_t\, \mathbf{x}_t$, which is a *scaled* version of the well known Perceptron update.

To analyze the Perceptron algorithm we note that the function $f(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2$ is strongly convex over $S = \mathbb{R}^n$ with respect to the Euclidean norm $\|\cdot\|_2$. Since the Euclidean norm is dual to itself we obtain that each function $g_t(\mathbf{w})$ defined in Eq. (5.1) is X-Lipschitz where $X = \max_{t\in\mathcal{M}} \|\mathbf{x}_t\|_2$. Setting $\gamma = 1$ in Eq. (5.2) we obtain that

$$|\mathcal{M}| \;\le\; c\,\frac{\|\mathbf{u}\|^2}{2} + \frac{X^2\,|\mathcal{M}|}{2\,c} + \sum_{t\in|\mathcal{M}|} [1 - y_t\langle \mathbf{u}, \mathbf{x}_t\rangle]_+ \quad . \tag{5.3}$$

We next show that since we are free to choose the constant c, which acts here as a simple scaling, the above yields the tightest mistake bound that is known for the Perceptron. Note that on round t, the hypothesis of the Perceptron can be rewritten as,

$$\mathbf{w}_t \;=\; \frac{1}{c} \sum_{i\in\mathcal{M}:i<t} y_i\,\mathbf{x}_i \quad .$$

The above form implies that the predictions of the Perceptron algorithm do not depend on the actual value of c so long as $c > 0$. Therefore, we can choose c to be the minimizer of the bound given in Eq. (5.3), namely,

$$c = \frac{X}{\|\mathbf{u}\|}\,\sqrt{|\mathcal{M}|} \quad .$$

Plugging this value back into Eq. (5.3) yields

$$|\mathcal{M}| \;\le\; X\,\|\mathbf{u}\|\,\sqrt{|\mathcal{M}|} + \sum_{t\in|\mathcal{M}|} [1 - y_t\langle \mathbf{u}, \mathbf{x}_t\rangle]_+ \quad .$$

Rearranging the above and using Lemma 19 in Section A.5 we obtain the following:

Corollary 5 *Let $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)$ be a sequence of examples and assume that this sequence is presented to the Perceptron algorithm. Let $\mathcal{M} = \{t \in [T] : \hat{y}_t \neq y_t\}$ be the set of rounds on which the Perceptron predicts an incorrect label and define $X = \max_{t \in \mathcal{M}} \|\mathbf{x}_t\|_2$. Then, for all $\mathbf{u} \in \mathbb{R}^n$ we have*

$$|\mathcal{M}| \;\leq\; \sum_{t \in \mathcal{M}} [1 - y_t \langle \mathbf{u}, \mathbf{x}_t \rangle]_+ + X\,\|\mathbf{u}\| \sqrt{\sum_{t \in \mathcal{M}} [1 - y_t \langle \mathbf{u}, \mathbf{x}_t \rangle]_+} + X^2\,\|\mathbf{u}\|^2 \;\; .$$

Note that this bound is identical to the best known mistake bound for the Perceptron algorithm (see for example [58]). However, our proof technique is vastly different. Furthermore, the new technique also enables us to derive mistake and regret bounds for new algorithms such as the ones discussed in Section 5.2.

5.1.2 Winnow

We now analyze a version of the Winnow algorithm called Balanced-Winnow [62]. This algorithm is closely related to the Exponentiated-Gradient algorithm [75]. For brevity we refer to the algorithm we analyze simply as Winnow. The Winnow algorithm can be derived from Figure 5.1 by setting the i'th element of the link function to be

$$\mathcal{L}\text{ink}_i(\boldsymbol{\theta}) \;=\; \exp(\theta_i) \;\; . \tag{5.4}$$

The i'th element of $\mathbf{w}_{t+1}$ can thus be rewritten using a multiplicative update rule,

$$w_{t+1,i} \;=\; \exp\left(\theta_{t,i} + \frac{1}{c}\, y_t x_{t,i}\right) \;=\; w_{t,i}\, \beta^{y_t x_{t,i}} \;\; , \tag{5.5}$$

where $\beta = \exp(1/c)$.

To analyze the Winnow algorithm we set $f(\mathbf{w})$ to be

$$f(\mathbf{w}) = \sum_{i=1}^{n} w_i \log\left(\frac{w_i}{1/n}\right) \;\; ,$$

and set the domain S to be the probabilistic simplex, $S = \left\{\mathbf{w} \in \mathbb{R}_+^n : \sum_{i=1}^n w_i = 1\right\}$. The function f is the relative entropy between the probability vector $\mathbf{w}$ and the uniform vector $(\frac{1}{n}, \ldots, \frac{1}{n})$. The relative entropy is non-negative and measures the entropic divergence between two distributions. It attains a value of zero whenever the two vectors are equal. Therefore, the minimum value of $f(\mathbf{w})$ is zero and is attained for $\mathbf{w} = (\frac{1}{n}, \ldots, \frac{1}{n})$. The conjugate of f is the logarithm of the sum of exponentials (see Section A.3.1)

$$f^\star(\boldsymbol{\theta}) \;=\; \log\left(\frac{1}{n}\sum_{i=1}^{n}\exp(\theta_i)\right) \quad . \tag{5.6}$$

The i'th element of the gradient of $f^\star$ is,

$$\frac{\exp(\theta_i)}{\sum_{j=1}^{n}\exp(\theta_j)} \quad .$$

We note that although $\nabla f^\star(\boldsymbol{\theta})$ does not equal $\mathcal{L}\mathrm{ink}(\boldsymbol{\theta})$ as given in Eq. (5.4), we still have that $\mathrm{sign}(\langle \nabla f^\star(\boldsymbol{\theta}), \mathbf{x}\rangle) = \mathrm{sign}(\langle \mathcal{L}\mathrm{ink}(\boldsymbol{\theta}), \mathbf{x}\rangle)$ for all $\mathbf{x}$ and $\boldsymbol{\theta}$. Therefore, the two algorithms produce the same predictions and are thus equivalent.

The relative entropy is strongly convex over the probabilistic simplex with respect to the norm $\|\cdot\|_1$ (see Lemma 16). The dual norm in this case is $\|\cdot\|_\infty$ and thus each function $g_t(\mathbf{w})$ defined in Eq. (5.1) is X-Lipschitz with $X = \max_{t\in\mathcal{M}} \|\mathbf{x}_t\|_\infty$. Next, we note that for all $\mathbf{u} \in S$ we have

$$f(\mathbf{u}) \;=\; \sum_{i=1}^{n} u_i \log(u_i) + \log(n)\sum_{i=1}^{n} u_i \;\le\; \log(n) \quad .$$

Plugging the above into Eq. (5.2) we obtain that

$$|\mathcal{M}| \;\le\; \frac{1}{\gamma}\left(c\,\log(n) + \frac{X^2\,|\mathcal{M}|}{2\,c} + \sum_{t\in|\mathcal{M}|}[\gamma - y_t\langle \mathbf{u}, \mathbf{x}_t\rangle]_+\right) \quad . \tag{5.7}$$

Had we known the value of $|\mathcal{M}|$ and X, we could have set $c = X\sqrt{|\mathcal{M}|/(2\,\log(n))}$ and obtained the inequality

$$|\mathcal{M}| \;\le\; \frac{1}{\gamma}\left(X\,\sqrt{2\,|\mathcal{M}|\log(n)} + \sum_{t\in|\mathcal{M}|}[\gamma - y_t\langle \mathbf{u}, \mathbf{x}_t\rangle]_+\right) \quad ,$$

which yields the bound (see again Lemma 19)

$$|\mathcal{M}| \le \frac{1}{\gamma}\sum_{t\in|\mathcal{M}|}[\gamma - y_t\langle \mathbf{u}, \mathbf{x}_t\rangle]_+ + \frac{X}{\gamma^{3/2}}\sqrt{2\,\log(n)\sum_{t\in|\mathcal{M}|}[\gamma - y_t\langle \mathbf{u}, \mathbf{x}_t\rangle]_+} + \frac{2X^2\log(n)}{\gamma^2} \quad . \tag{5.8}$$

Naturally, knowing the value of $|\mathcal{M}|$ prior to the run of the algorithm is not realistic. In the next section we describe an automatic method for tuning the parameter c. The automatically tuned algorithm achieves a bound similar to the bound in Eq. (5.8) without assuming any prior knowledge.

To conclude this section, we describe the applicability of the Winnow algorithm to learning Boolean r-of-k functions. Formally, a function $h : \{0,1\}^n \to \{+1,-1\}$ is an r-of-k Boolean function if there exists k indices, $i_1,\ldots,i_k$ such that $h(\mathbf{x}) = 1$ iff $x_{i_1} + \ldots + x_{i_k} \geq r$. That is, h returns 1 if r out of the k bits $i_1,\ldots,i_k$ are 1. Otherwise, h returns -1. Let us now show that $h(\mathbf{x})$ can be rewritten as $\text{sign}(\langle \mathbf{u},\mathbf{x}\rangle)$ for some vector $\mathbf{u}$. For simplicity, assume that the last element of $\mathbf{x}$ is always -1 (if this is not the case, we can add another element to $\mathbf{x}$). Let $\mathbf{u}$ be defined as follows

$$u_p = \begin{cases} \frac{1}{r+k-\frac{1}{2}} & \text{if } \exists j \in [k] : p = i_j \\ \frac{r-\frac{1}{2}}{r+k-\frac{1}{2}} & \text{if } p = n \\ 0 & \text{otherwise} \end{cases} .$$

Clearly, $\mathbf{u} \in S$. In addition, setting $\gamma = \frac{1/2}{r+k-1/2}$ we get that if $x_{i_1} + \ldots + x_{i_k} \geq r$ than $\langle \mathbf{u},\mathbf{x}\rangle \geq \gamma$ and otherwise $\langle \mathbf{u},\mathbf{x}\rangle \leq -\gamma$. The mistake bound in Eq. (5.8) therefore gives that

$$|\mathcal{M}| \leq \frac{2\log(n)}{\gamma^2} \;=\; 8\,(r+k-1/2)^2\log(n) \;.$$

The above mistake bound is better than the mistake bound for the Perceptron algorithm given in Corollary 5 since the latter grows linearly with n.

5.1.3 Self-tuned Winnow

We now describe an adaptation of the Winnow algorithm in which the parameter c is automatically tuned. Specifically, Let $\mathcal{M}_t = \{i < t : \hat{y}_i \neq y_i\}$ be the set of rounds on which the algorithm errs up until round t. We also define $X_t = \max_{i \leq t} \|\mathbf{x}_i\|_\infty$. We set the learning rate parameter to be

$$c_t = X_t\,\sqrt{(|\mathcal{M}_t|+1)/\log(n)} \;.$$

A summary of the self-tuned variant of the Winnow algorithm is given in Figure 5.2.

We now analyze the self-tuned Winnow algorithm given in Figure 5.2. Based on Lemma 2, and using the facts that $f(\mathbf{u}) \leq \log(n)$ and $g_t(\mathbf{w})$ is $\|\mathbf{x}_t\|_\infty$-Lipschitz with respect to the ℓ_∞ norm we

```
INITIALIZE: $\boldsymbol{\theta}_1 = \mathbf{0}$, $X_0 = 0$, $M_1 = 0$
FOR $t = 1, 2, \ldots, T$
    Receive an instance vector $\mathbf{x}_t$
    Set $X_t = \max\{\|\mathbf{x}_t\|_\infty, X_{t-1}\}$
    Set $w_{t,i} = \exp(\theta_{t,i})$
    Predict $\hat{y}_t = \text{sign}(\langle \mathbf{w}_t, \mathbf{x}_t \rangle)$
    Receive the correct label $y_t$
    IF $\hat{y}_t \neq y_t$
        $c_t = X_t \sqrt{(M_t + 1)/\log(n)}$
        $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \frac{1}{c_t} y_t \mathbf{x}_t$
        $M_{t+1} = M_t + 1$
    ELSE
        $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t$
        $M_{t+1} = M_t$
```

Figure 5.2: A self-tuned variant of the Winnow algorithm for binary classification.

obtain that

$$\begin{aligned}
\sum_{t \in \mathcal{M}} g_t(\mathbf{w}_t) - \sum_{t \in \mathcal{M}} g_t(\mathbf{u}) &\leq c_T f(\mathbf{u}) + \frac{1}{2} \sum_{t \in \mathcal{M}} \frac{\|\mathbf{x}_t\|_\infty^2}{c_t} \\
&\leq X_T \sqrt{|\mathcal{M}| \log(n)} + \frac{\sqrt{\log(n)}}{2} \sum_{i=1}^{|\mathcal{M}|} \frac{X_t}{\sqrt{i}} \\
&\leq X_T \sqrt{|\mathcal{M}| \log(n)} + \frac{X_T \sqrt{\log(n)}}{2} \sum_{i=1}^{|\mathcal{M}|} \frac{1}{\sqrt{i}} \\
&\leq 2\, X_T \sqrt{|\mathcal{M}| \log(n)} \ .
\end{aligned}$$

Combining the above with the fact that for $t \in \mathcal{M}$ we have $g_t(\mathbf{w}_t) \geq \gamma$ and and using Lemma 19 we obtain the following:

Corollary 6 *Let $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)$ be a sequence of examples and assume that this sequence is presented to the self-tuned Winnow algorithm given in Figure 5.2. Let $\mathcal{M} = \{t \in [T] : \hat{y}_t \neq y_t\}$ be the set of rounds on which the algorithm predicts an incorrect label and define $X = \max_{t \in \mathcal{M}} \|\mathbf{x}_t\|_\infty$. Let $S = \{\mathbf{u} \in \mathbb{R}_+^n : \sum \mathbf{u}_i = 1\}$. Then, for all $\mathbf{u} \in S$ and $\gamma > 0$ we have*

$$|\mathcal{M}| \leq \frac{1}{\gamma} \sum_{t \in |\mathcal{M}|} [\gamma - y_t \langle \mathbf{u}, \mathbf{x}_t \rangle]_+ + \frac{2\,X}{\gamma^{3/2}} \sqrt{\log(n) \sum_{t \in |\mathcal{M}|} [\gamma - y_t \langle \mathbf{u}, \mathbf{x}_t \rangle]_+} + \frac{4X^2 \log(n)}{\gamma^2} \ .$$

5.1.4 p-norm algorithms

We conclude this section with the analysis of the family of p-norm algorithms [58, 62]. This family can be viewed as a bridge between the Perceptron algorithm and the Winnow algorithm. As we show in the sequel, the Perceptron algorithm is a special case of a p-norm algorithm, obtained by setting $p = 2$, whereas the Winnow algorithm can be approximated by setting p to a large number. The family of p-norm algorithms can be derived from Figure 5.1 by setting the i'th element of the link function to be

$$\mathcal{L}\text{ink}_i(\boldsymbol{\theta}) = \frac{\text{sign}(\theta_i)\,|\theta_i|^{p-1}}{\|\boldsymbol{\theta}\|_p^{p-2}} \quad . \tag{5.9}$$

To analyze any p-norm algorithm we assume that $p \geq 2$ and let q be $(1 - 1/p)^{-1}$. Thus, $\frac{1}{p} + \frac{1}{q} = 1$ and the norm $\|\cdot\|_p$ is dual to the norm $\|\cdot\|_q$. Define,

$$f(\mathbf{w}) \;=\; \frac{1}{2(q-1)}\|\mathbf{w}\|_q^2 \ ,$$

and let $S = \mathbb{R}^n$. The assumption $p \geq 2$ implies that $q \in (1, 2]$ and thus the function $f(\mathbf{w})$ is strongly convex with respect to the norm $\|\cdot\|_q$ (see Example 5 in Section A.4). The conjugate function of f in this case is $f^\star(\boldsymbol{\theta}) = \frac{1}{2(p-1)}\|\boldsymbol{\theta}\|_p^2$ (see Section A.3.1) and its gradient $\nabla f^\star$ equals the link function given in Eq. (5.9) divided by $p - 1$. Setting $\gamma = 1$ in Eq. (5.2) we obtain that

$$|\mathcal{M}| \;\leq\; \frac{c}{2(q-1)}\|\mathbf{u}\|_q^2 + \frac{X^2\,|\mathcal{M}|}{2\,c} + \sum_{t\in|\mathcal{M}|} g_t(\mathbf{u}) \ , \tag{5.10}$$

where $X = \max_{t\in\mathcal{M}} \|\mathbf{x}_t\|_p$.

We next show that the predictions of the p-norm algorithm do not depend on the actual value of c as long as $c > 0$. Recall that $\boldsymbol{\theta}_t$ for the update of the family of quasi-additive algorithms can be written as

$$\boldsymbol{\theta}_t = \frac{1}{c}\sum_{i\in\mathcal{M}:i<t} y_i\,\mathbf{x}_i \ .$$

Denote by $\mathbf{v} = \sum_{i\in\mathcal{M}:i<t} y_i\,\mathbf{x}_i$. Clearly, this vector does not depend on c. Since hypothesis $\mathbf{w}_t$ is defined from $\boldsymbol{\theta}_t$ as given by Eq. (5.9) we can rewrite the j'th component of $\mathbf{w}_t$ as,

$$w_{t,j} \;=\; \frac{\text{sign}(v_j)\,|v_j/c|^{p-1}}{\|\mathbf{v}/c\|_p^{p-2}} \;=\; \frac{1}{c}\,\frac{\text{sign}(v_j)\,|v_j|^{p-1}}{\|\mathbf{v}\|_p^{p-2}} \ .$$

We have therefore shown that the predictions of the algorithm do not depend on the specific value of c as long as $c > 0$, so we are free to choose c so as to minimize the right-hand side of Eq. (5.10).

Specifically, plugging the value

$$c \;=\; X\,\sqrt{|\mathcal{M}|\,(q-1)}/\|\mathbf{u}\|_q$$

into Eq. (5.10) we obtain that

$$|\mathcal{M}| \;\leq\; \frac{1}{\sqrt{q-1}}\,X\,\|\mathbf{u}\|_q\,\sqrt{|\mathcal{M}|} + \sum_{t\in|\mathcal{M}|} g_t(\mathbf{u}) \;\;.$$

Rearranging the above, using the equality $\frac{1}{q-1} = p-1$, and using Lemma 19 yield the following:

Corollary 7 *Let* $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)$ *be a sequence of examples and assume that this sequence is presented to a* p*-norm algorithm with* $p \geq 2$*. Let* $\mathcal{M} = \{t \in [T] : \hat{y}_t \neq y_t\}$ *be the set of rounds on which the algorithm predicts an incorrect label and define* $X = \max_{t\in\mathcal{M}} \|\mathbf{x}_t\|_p$*. Then, for all* $\mathbf{u} \in \mathbb{R}^n$ *we have*

$$|\mathcal{M}| \;\leq\; \sum_{t\in\mathcal{M}} [1 - y_t\langle \mathbf{u}, \mathbf{x}_t\rangle]_+ + X\,\|\mathbf{u}\|_q \sqrt{(p-1)\sum_{t\in\mathcal{M}} [1 - y_t\langle \mathbf{u}, \mathbf{x}_t\rangle]_+ \;+\; (p-1)\,X^2\,\|\mathbf{u}\|_q^2} \;\;,$$

where $q = (1 - 1/p)^{-1}$.

5.2 Deriving new online updates based on aggressive dual ascending procedures

In the previous section we described the family of quasi-additive online learning algorithms. The algorithms are based on the simple update procedure defined in Eq. (3.11) that leads to a conservative increase of the dual objective since we modify a *single* dual vector by setting it to a sub-gradient of $g_t(\mathbf{w}_t)$. Furthermore, such an update takes place solely on rounds for which there was a prediction mistake ($t \in \mathcal{M}$). Algorithms that update their hypotheses only on erroneous rounds are also called conservative. In this section we describe broader and, in practice, more powerful update procedures. The motivation for the new algorithms is as follows. Intuitively, update schemes that yield larger increases of the dual objective value on each online round are likely to "consume" more of the upper bound on the total possible increase in the dual as set by the minimal primal value. Thus, they are in practice likely to suffer a smaller number of mistakes.

Building on the exposition provided in Section 3.3 we cast the online learning problem as the task of incrementally increasing the dual objective function

$$\mathcal{D}(\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_T) \;=\; -\,cf^\star\left(-\frac{1}{c}\sum_{t=1}^{T}\boldsymbol{\lambda}_t\right) - \sum_{t=1}^{T} g_t^\star(\boldsymbol{\lambda}_t) \;\; . \tag{5.11}$$

To simplify our derivation, we focus on the problem of binary classification with the hinge-loss. Formally, let $(\mathbf{x}_t, y_t) \in \mathbb{R}^n \times \{\pm 1\}$ be the tth classification example. We set

$$g_t(\mathbf{w}) = [\gamma - y_t\langle \mathbf{w}, \mathbf{x}_t\rangle]_+ \;\; ,$$

where $\gamma > 0$ is a margin parameter, and $[a]_+ = \max\{0, a\}$ is the hinge function. Based on Section A.3, the Fenchel conjugate of $g_t(\mathbf{w})$ is the function

$$g_t^\star(\boldsymbol{\lambda}) \;=\; \begin{cases} -\gamma\,\alpha & \text{if} \quad \boldsymbol{\lambda} \in \{-\alpha y_t \mathbf{x}_t : \alpha \in [0,1]\} \\ \infty & \text{otherwise} \end{cases}$$

Since our goal is to maximize the dual objective, we can restrict ourselves to the case $\boldsymbol{\lambda}_t = -\alpha_t y_t \mathbf{x}_t$, where $\alpha_t \in [0, 1]$, and rewrite the dual objective as a function of the vector $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_T)$

$$\mathcal{D}(\boldsymbol{\alpha}) \;=\; -\,cf^\star\left(\frac{1}{c}\sum_{t=1}^{T}\alpha_t y_t \mathbf{x}_t\right) + \gamma\sum_{t=1}^{T}\alpha_t \;\; . \tag{5.12}$$

The update scheme we described in Section 5.1 for increasing the dual can be rewritten as

$$\alpha_i^{t+1} = \begin{cases} 1 & t \in \mathcal{M} \,\wedge\, i = t \\ \alpha_i^t & \text{otherwise} \end{cases} \quad . \tag{5.13}$$

This update modifies $\boldsymbol{\alpha}$ only on rounds on which there was a prediction mistake ($t \in \mathcal{M}$). The update is performed by setting the t'th element of $\boldsymbol{\alpha}$ to 1 and keeping the rest of the variables intact. This simple update can be enhanced in several ways. First, note that while setting α_t^{t+1} to 1 guarantees a sufficient increase in the dual, there might be other values α_t^{t+1} which would lead to even larger increases of the dual. Furthermore, we can also update $\boldsymbol{\alpha}$ on rounds on which the prediction was correct so long as the loss is non-zero. Last, we need not restrict our update to the t'th element of $\boldsymbol{\alpha}$. We can instead update several dual variables as long as their indices are in $[t]$.

We now describe and briefly analyze a few new updates which increase the dual more aggressively. The goal here is to illustrate the power of the approach and the list of new updates we outline is by no means exhaustive.

5.2.1 Aggressive quasi-additive online algorithms

We start by describing an update which sets α_t^{t+1} adaptively, so as to maximize the dual objective with respect to the tth element of $\boldsymbol{\alpha}$. This improved update constructs $\boldsymbol{\alpha}^{t+1}$ as follows:

$$\boldsymbol{\alpha}^{t+1} = \operatorname*{argmax}_{\boldsymbol{\alpha}} \mathcal{D}(\boldsymbol{\alpha}) \quad \text{s.t. } \forall i \neq t,\ \alpha_i = \alpha_i^t \quad . \tag{5.14}$$

We defer the problem of how to solve the optimization problem in the definition of the update to Section 7.2. The update in Eq. (5.14) is equivalent to the update described in Eq. (3.12). Clearly, this update satisfies Eq. (3.9). Assume that $f(\mathbf{w})$ is strongly convex with respect to some norm $\|\cdot\|$ and let $X = \max_t \|\mathbf{x}_t\|_\star$. Applying Corollary 1 we get the regret bound

$$\sum_{t=1}^{T} g_t(\mathbf{w}_t) - \sum_{t=1}^{T} g_t(\mathbf{u}) \ \le\ c\, f(\mathbf{u}) + \frac{X^2\, T}{2\, c} \quad .$$

The above yields a regret of $O(\sqrt{T})$ provided that $c = \Theta(\sqrt{T})$. We can also use the self-tuned version of our algorithmic framework given in Figure 3.2 and define the update to be

$$\boldsymbol{\alpha}^{t+1} = \operatorname*{argmax}_{\boldsymbol{\alpha}} \mathcal{D}_t(\boldsymbol{\alpha}) \quad \text{s.t. } \forall i \neq t,\ \alpha_i = \alpha_i^t \quad . \tag{5.15}$$

The conditions of Corollary 3 hold for this update. Therefore, by setting

$$c_t = \sqrt{\frac{t}{U}}\ \max_{i \le t} \|\mathbf{x}_t\|_\star$$

we obtain that the following regret bound holds for any $\mathbf{u} \in \{\mathbf{w} : f(\mathbf{w}) \le U\}$

$$\sum_{t=1}^{T} g_t(\mathbf{w}_t) - \sum_{t=1}^{T} g_t(\mathbf{u}) \ \le\ 2X\, \sqrt{U\, T} \quad . \tag{5.16}$$

Mistake Bounds The update given in Eq. (5.15) yields the regret bound given in Eq. (5.16). We can derive a mistake bound from Eq. (5.16) by noting that the hinge-loss upper bounds $\gamma\, |\mathcal{M}|$. However, the resulting mistake bound depends on $\sqrt{T}$ whereas the mistake bounds we derived in Section 5.1 for the conservative quasi-additive algorithms may be better whenever there exists a vector $\mathbf{u}$ that suffers a small cumulative loss over the sequence of examples. If our primary goal is to bound the number of *mistakes* we can set the variables c_t in a different way so as to obtain an improved mistake bound for the update given in Eq. (5.15).

```
PARAMETERS: A strongly convex function $f$ w.r.t. a norm $\|\cdot\|$
            A constant $U > 0$
            Margin parameter $\gamma > 0$
INITIALIZE: $\forall t,\ \boldsymbol{\alpha}^1 = \mathbf{0}$, $X_0 = 0$, $M_0 = 0$
FOR $t = 1, 2, \ldots, T$
    Receive an instance vector $\mathbf{x}_t$
    Set $X_t = \max\{\|\mathbf{x}_t\|_\infty, X_{t-1}\}$
    Set $c_t = X_t\,\sqrt{M_{t-1}+1}\,/\,\sqrt{U}$
    Set $\mathbf{w}_t = \nabla f^\star(-\frac{1}{c_t}\sum_{i<t}\alpha_i^t\, y_i\,\mathbf{x}_i)$
    Predict $\hat{y}_t = \mathrm{sign}(\langle \mathbf{w}_t, \mathbf{x}_t\rangle)$
    Receive the correct label $y_t$
    Update $\boldsymbol{\alpha}^{t+1}$ as in Eq. (5.15)
    IF $\hat{y}_t \neq y_t$
      $M_{t+1} = M_t + 1$
    ELSE
      $M_{t+1} = M_t$
```

Figure 5.3: A self-tuned variant of the aggressive quasi-additive family of algorithms for binary classification.

Specifically, we set

$$c_t \;=\; X_t \sqrt{\frac{|\mathcal{M}_{t-1}|+1}{U}} \;, \tag{5.17}$$

where $\mathcal{M}_t = \mathcal{M} \cap [t]$, $X_t = \max_{i\le t}\|\mathbf{x}_t\|_\star$, and U is a positive scalar. The following theorem bounds the number of prediction mistakes the resulting algorithm makes.

Theorem 6 *Let f be a strongly convex function over a set S with respect to a norm $\|\cdot\|$ and let $\|\cdot\|_\star$ be the norm dual to $\|\cdot\|$. Assume that $\min_{\mathbf{w}\in S} f(\mathbf{w}) \ge 0$. Let $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)$ be a sequence of classification examples and assume that this sequence is presented to an algorithm of the form given in Figure 5.3. Let $\mathcal{M} = \{t \in [T] : \hat{y}_t \neq y_t\}$ be the set of rounds on which the algorithm predicts an incorrect label and define $X = \max_{t\in[T]}\|\mathbf{x}_t\|_\star$. Then, for all $\mathbf{u} \in S$ that satisfies $f(\mathbf{u}) \le U$ we have*

$$|\mathcal{M}| \;\le\; \frac{1}{\gamma}\sum_{t=1}^{T} g_t(\mathbf{u}) + \frac{2\,X\,\sqrt{U}}{\gamma^{3/2}}\sqrt{\sum_{t=1}^{T} g_t(\mathbf{u}) + \frac{4\,X^2\,U}{\gamma^2}} \;\;.$$

Proof Denote $\Delta_t = \mathcal{D}_t(\boldsymbol{\alpha}^{t+1}) - \mathcal{D}_{t-1}(\boldsymbol{\alpha}^t)$. The main property of the update given in Eq. (5.15) that we utilize is that $\Delta_t \geq 0$ for all t. Furthermore, if $t \in \mathcal{M}$ then

$$\Delta_t \;\geq\; g_t(\mathbf{w}_t) - \frac{\|\mathbf{x}_t\|_\star^2}{2\,c_t} \;\geq\; \gamma - \frac{\|\mathbf{x}_t\|_\star^2}{2\,c_t} \;\geq\; \gamma - \frac{\sqrt{U}\,X_t}{2\,\sqrt{|\mathcal{M}_t|}} \;\geq\; \gamma - \frac{\sqrt{U}\,X}{2\,\sqrt{|\mathcal{M}_t|}}\;.$$

Therefore,

$$\mathcal{D}(\boldsymbol{\alpha}^{T+1}) \;\geq\; \sum_{t=1}^{T} \Delta_t \;\geq\; \sum_{t \in \mathcal{M}} \Delta_t \;\geq\; \gamma|\mathcal{M}| - \frac{\sqrt{U}\,X}{2} \sum_{i=1}^{|\mathcal{M}|} \frac{1}{\sqrt{i}} \;\geq\; \gamma|\mathcal{M}| - \sqrt{U}\,X\sqrt{|\mathcal{M}|}\;.$$

Next, we upper bound $\mathcal{D}(\boldsymbol{\alpha}^{T+1})$. Without loss of generality, we assume that $T \in \mathcal{M}$ (otherwise, we can simply ignore the last rounds). Therefore,

$$\mathcal{D}(\boldsymbol{\alpha}^{T+1}) \;\leq\; c_T\, f(\mathbf{u}) + \sum_{t=1}^{T} g_t(\mathbf{u}) \;=\; X\sqrt{|\mathcal{M}|/U}\, f(\mathbf{u}) + \sum_{t=1}^{T} g_t(\mathbf{u}) \;\leq\; X\sqrt{U\,|\mathcal{M}|} + \sum_{t=1}^{T} g_t(\mathbf{u})\;.$$

Combining the upper and lower bounds we obtain that

$$\gamma\,|\mathcal{M}| - 2X\,\sqrt{U\,|\mathcal{M}|} - \sum_{t=1}^{T} g_t(\mathbf{u}) \;\leq\; 0\;.$$

The bound in the theorem follows from the above inequality using Lemma 19. ■

Our focus thus far was on an update which modifies a *single* dual variable, albeit aggressively. We now examine another implication of our analysis that suggests the modification of *multiple* dual variables on each round. The simple argument presented below implies that this broader family of updates also achieves the mistake and regret bounds above.

5.2.2 Updating multiple dual variables

The new update given in Eq. (5.15) is advantageous compared to the previous conservative update given in Eq. (5.13) since in addition to the same increase in the dual on rounds with a prediction mistake it is also guaranteed to increase the dual on the rest of the rounds. Yet both updates are confined to the modification of a single dual variable on each round. We nonetheless can increase the dual more dramatically by modifying multiple dual variables on each round. We now outline two forms of updates that modify multiple dual variables on each round.

In the first update scheme we optimize the dual over a set of dual variables $I_t \subseteq [t]$, which includes t. Given I_t, we set $\boldsymbol{\alpha}^{t+1}$ to be

$$\boldsymbol{\alpha}^{t+1} = \operatorname*{argmax}_{\boldsymbol{\alpha}\in[0,1]^T} \mathcal{D}_t(\boldsymbol{\alpha}) \;\; \text{s.t.} \;\; \forall i \notin I_t,\; \alpha_i = \alpha_i^t \;\; . \tag{5.18}$$

The increase in the dual due to this update is guaranteed to be at least as large as the increase due to the update given in Eq. (5.15). Therefore, this more general update also achieves the regret and mistake bounds discussed in the previous section.

Let us examine a few choices for I_t. Setting $I_t = [t]$ for all t gives a regularized version of the follow-the-leader algorithm, originally suggested by Hannan [63].[2] This algorithm makes use of all the examples that have been observed and thus is likely to make the largest increase in the dual objective on each round. It does require however a full-blown optimization procedure. In contrast, Eq. (5.18) can be solved analytically when we employ the smallest possible set, $I_t = \{t\}$, with $f(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2$ (see Section 7.2). This algorithm was described in [27] and belongs to a family of Passive Aggressive algorithms. The mistake bound that we obtain as a by-product is however superior to the one in [27]. Naturally, we can interpolate between the minimal and maximal choices for I_t by setting the size of I_t to a predefined value k and choosing, say, the last k observed examples as the elements of I_t. For $k = 1$ and $k = 2$ we can solve Eq. (5.18) analytically while gaining modest increases in the dual. The full power of the update is unleashed for large values of k. However, Eq. (5.18) cannot be solved analytically and requires the usage of numerical solvers based on, for instance, interior point methods.

The second update scheme modifies multiple dual variables on each round as well, but it does not require solving an optimization problem with multiple variables. Instead, we perform k_t mini-updates each of which focuses on a single variable from the set $[t]$. Formally, let $i_1, \ldots, i_{k_t}$ be a sequence of indices such that $i_1 = t$ and $i_j \in [t]$ for all $j \in [k_t]$. We define a sequence of dual solutions in a recursive manner as follows. We start by setting $\hat{\boldsymbol{\alpha}}^0 = \boldsymbol{\alpha}^t$ and then perform a sequence of single variable updates of the form,

$$\hat{\boldsymbol{\alpha}}^j = \operatorname*{argmax}_{\boldsymbol{\alpha}\in[0,1]^T} \mathcal{D}_t(\boldsymbol{\alpha}) \quad \text{s.t.} \quad \forall p \neq i_j,\;\; \hat{\alpha}_p^j = \hat{\alpha}_p^{j-1} \;\; .$$

Finally, we update $\boldsymbol{\alpha}^{t+1} = \hat{\boldsymbol{\alpha}}^{k_t}$. In words, we first decide on an ordering of the dual variables and incrementally increase the dual by fixing all the dual variables but the current one under consideration. For this variable we find the optimal solution of the constrained dual. The first dual variable

[2]In contrast to follow-the-leader algorithms, our regularized version of the algorithm also takes the complexity of $\mathbf{w}$ in the form of $f(\mathbf{w})$ into account when constructing its predictors. Note that in general, follow-the-leader algorithms may not attain a regret bound and some perturbation is needed. Our regularized version of follow-the-leader does yield a regret bound without perturbation.

we update is α_t thus ensuring that the first step in the row of updates is identical to the aggressive update given in Eq. (5.15). Since on each operation we can only increase the dual we immediately conclude that the regret and mistake bounds discussed in the previous section hold for this composite update scheme as well. The main advantage of this update is its simplicity since each operation involves optimization over a single variable, which can be solved analytically. The increase in the dual due to this update is closely related to the so-called row action methods in optimization (see for example [16]).

5.3 Complex Prediction Problems

Our focus in previous sections was mainly on the binary classification problem. The goal of this section is to demonstrate the applicability of our algorithmic framework to deriving online learning algorithms for various prediction problems. Recall that on each online round, the learner receives a question, $\mathbf{x}_t \in \mathcal{X}$, and is required to predict an answer for this question taken from a domain $\mathcal{Y}$. The prediction is based on a hypothesis $h_{\mathbf{w}_t}$ where $\mathbf{w}_t$ is a parameter vector. After predicting an answer, the learner receives the correct answer, $y_t \in \mathcal{Y}$, and suffers loss $g_t(\mathbf{w}_t) = \ell(h_{\mathbf{w}_t}, (\mathbf{x}_t, y_t))$. For each prediction problem, we define the domains $\mathcal{X}, \mathcal{Y}$, the form the hypotheses take, and an appropriate loss function.

5.3.1 Regression

In regression problems, the questions domain is $\mathcal{X} = \mathbb{R}^n$ and the answers domain is $\mathcal{Y} = \mathbb{R}$. A widely used hypothesis class is the set of linear regressors where $h_{\mathbf{w}}(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle$. A common loss function is the squared error defined as

$$\ell(h_{\mathbf{w}}, (\mathbf{x}, y)) = \frac{1}{2}(h_{\mathbf{w}}(\mathbf{x}) - y)^2 = \frac{1}{2}(\langle \mathbf{w}, \mathbf{x} \rangle - y)^2 \ .$$

This loss function is $(2\,\|\mathbf{x}\|^2)$-self-bounded with respect to any norm $\|\cdot\|$. To see this, we note that ℓ is differentiable with respect to $\mathbf{w}$ and its gradient is $\boldsymbol{\lambda} = (\langle \mathbf{w}, \mathbf{x} \rangle - y)\,\mathbf{x}$. Therefore,

$$\|\boldsymbol{\lambda}\|^2 = (\langle \mathbf{w}, \mathbf{x} \rangle - y)^2\,\|\mathbf{x}\|^2 = \ell(h_{\mathbf{w}}, (\mathbf{x}, y))(2\,\|\mathbf{x}\|^2) \ .$$

Thus, we can use Figure 3.1 and Figure 3.2 for deriving online learning algorithms for regression with the above loss function and analyze them based on Theorem 1 and Theorem 2.

Another popular loss function for regression is the ϵ-insensitive absolute loss defined as

$$\ell(h_{\mathbf{w}}, (\mathbf{x}, y)) = \max\{0, |\langle \mathbf{w}, \mathbf{x} \rangle - y| - \epsilon\} \ ,$$

where ϵ is a predefined tolerance parameter. That is, we do not suffer loss for predictions in $y \pm \epsilon$. Otherwise, we pay linearly for the miss-prediction. It is simple to show that the ϵ-insensitive absolute loss is $\|\mathbf{x}\|$-Lipschitz with respect to any norm $\|\cdot\|$. Therefore, we can again use Figure 3.1 and Figure 3.2 to derive online learning algorithms for regression with the ϵ-insensitive loss function and analyze them based on Corollary 1 and Corollary 3.

5.3.2 Multiclass Categorization

In multiclass categorization, the questions domain is an arbitrary set $\mathcal{X}$ and the answers domain is a finite set of labels. For concreteness we assume that there are k different possible labels and denote the set of all possible labels by $\mathcal{Y} = \{1, \ldots, k\}$.

To describe a hypothesis class for multiclass categorization we assume that we are provided with a class dependent feature function $\boldsymbol{\phi} : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}^d$. That is, $\boldsymbol{\phi}$ takes as input an instance $\mathbf{x}$ and a possible label $r \in \mathcal{Y}$ and returns as output a feature vector in $\mathbb{R}^d$. The hypotheses we use are parameterized by a weight vector $\mathbf{w} \in \mathbb{R}^d$ and are defined as follows:

$$h_{\mathbf{w}}(\mathbf{x}) \;=\; \underset{r \in \mathcal{Y}}{\operatorname{argmax}} \, \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}, r) \rangle \ .$$

After making its prediction, the algorithm receives the correct label $y_t \in \mathcal{Y}$. Similar to the setting of binary classification, the primary goal of the online algorithm is to minimize the cumulative number of prediction mistakes it makes along its run. Put another way, we would like to minimize the cumulative $0-1$ loss function, $\ell_{0-1}(h_{\mathbf{w}}, (\mathbf{x}, y)) = [\![h_{\mathbf{w}}(\mathbf{x}) \neq y]\!]$. Unfortunately, the $0-1$ loss function is not convex so we could not apply our algorithmic framework to this loss function. Instead, we follow the methodology described in Section 2.3 that we used for binary classification and define the following convex upper bound for the $0-1$ loss function:

$$\ell(h_{\mathbf{w}}, (\mathbf{x}, y) \;=\; \max_{r \neq y} \left[\gamma - \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}, y) - \boldsymbol{\phi}(\mathbf{x}, r) \rangle \right]_+ \ , \tag{5.19}$$

where $[a]_+ = \max\{0, a\}$ and $\gamma > 0$ is a margin parameter. The above definition generalizes the definition of the hinge-loss for binary classification. In words, the function $\boldsymbol{\phi}$ maps the instance $\mathbf{x}$ into k vectors in $\mathbb{R}^d$, each of which is associated with one of the labels in $\mathcal{Y}$. Then, the parameter vector $\mathbf{w} \in \mathbb{R}^d$ maps points in $\mathbb{R}^d$ into the reals. After applying the above two mappings, we end up with a single scalar for each label. The loss function given in Eq. (5.19) requires that the scalar associated with the correct label is larger than the scalars associated with the rest of the labels by at least the margin parameter γ.

To apply our algorithmic framework for the loss function given in Eq. (5.19) we note that a subgradient of the function $g_t(\mathbf{w}) = \ell(h_{\mathbf{w}}, (\mathbf{x}, y))$ can be found as follows. Define

$$i \;=\; \operatorname*{argmax}_{r \neq y} \gamma - \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}, y) - \boldsymbol{\phi}(\mathbf{x}, r) \rangle \;=\; \operatorname*{argmax}_{r \neq y} \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}, r) \rangle \;.$$

Then, a subgradient of $g_t(\mathbf{w})$ at $\mathbf{w}$ is

$$\boldsymbol{\lambda} \;=\; \begin{cases} \boldsymbol{\phi}(\mathbf{x}, r) - \boldsymbol{\phi}(\mathbf{x}, y) & \text{if } g_t(\mathbf{w}) > 0 \\ \mathbf{0} & \text{otherwise} \end{cases} \;.$$

In particular, the above implies that the function $g_t(\mathbf{w})$ is X-Lipschitz with respect to any norm $\|\cdot\|$, where $X = \max_{r \neq y} \|\boldsymbol{\phi}(\mathbf{x}, r) - \boldsymbol{\phi}(\mathbf{x}, y)\|$. Therefore, we can use Figure 3.1 and Figure 3.2 to derive online learning algorithms for multiclass categorization and analyze them based on Corollary 1 and Corollary 3.

5.3.3 Learning with Structured Output

A useful application of machine learning is learning with structured output. In this setting, the set of possible labels are endowed with a predefined structure. Typically, the set of labels is very large and the structure plays a key role in constructing efficient learning and inference procedures. Notable examples of structured label sets are graphs (in particular trees) and strings [22, 3, 113, 115]. We now overview how our algorithmic frameworks described in Figure 3.1 and Figure 3.2 can be adapted to structured output settings. For concreteness, we focus on an adaptation for string prediction. Our derivation, however, can be easily mapped to other settings of learning with structured output.

In string prediction problems we are provided with a predefined alphabet $\Sigma = \{1, \ldots, k\}$. Each input question is associated with an answer which is a string over Σ. For simplicity we assume that the output string is of a fixed length m. Thus, on round t, the learning algorithm receives an instance $\mathbf{x}_t$ from an arbitrary domain $\mathcal{X}$ and then predicts an output string $\mathbf{y}'_t \in \mathcal{Y}$ where $\mathcal{Y} = \Sigma^m$.

As in the multiclass categorization task, we assume that there exists a class dependent feature function $\boldsymbol{\phi} : \mathcal{X} \times \Sigma^m \to \mathbb{R}^d$, which takes as its input an instance $\mathbf{x}$ and a string $\mathbf{y}$ and outputs a feature vector in $\mathbb{R}^d$. The hypotheses we use are parameterized by a weight vector $\mathbf{w} \in \mathbb{R}^d$ and are defined as follows

$$h_{\mathbf{w}}(\mathbf{x}) \;=\; \operatorname*{argmax}_{\mathbf{y} \in \Sigma^m} \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}, \mathbf{y}) \rangle \;. \tag{5.20}$$

Calculating $h_{\mathbf{w}}(\mathbf{x})$ in the general case may require as many as k^m evaluations of $\langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}, \mathbf{y}) \rangle$. This problem becomes intractable as m becomes large. We must therefore impose some restrictions on the feature representation $\boldsymbol{\phi}$ that will enable us to find $h_{\mathbf{w}}(\mathbf{x})$ efficiently. A possible restriction on the feature representation is to assume that each feature ϕ_j takes the form,

$$\phi_j(\mathbf{x}, \mathbf{y}) \;=\; \psi_j^0(y_1, \mathbf{x}) + \sum_{i=2}^{m} \psi_j(y_i, y_{i-1}, \mathbf{x}) \;\;, \tag{5.21}$$

where ψ_j^0 and ψ_j are any computable functions. This construction is analogous to imposing a first order Markovian structure on the output string. This form paves the way for an efficient inference, i.e. solving Eq. (5.20), using a dynamic programming procedure. Similar yet richer structures such as dynamic Bayes nets can be imposed so long as the solution to Eq. (5.20) can be computed efficiently.

Upon predicting, the algorithm receives the correct output string $\mathbf{y}_t \in \Sigma^m$ that is associated with $\mathbf{x}_t$. The learning algorithm is also provided with a cost function $\rho : \Sigma^m \times \Sigma^m \to \mathbb{R}_+$. The value of $\rho(\mathbf{y}, \mathbf{y}')$ represents the cost associated with predicting the string $\mathbf{y}'$ instead of the string $\mathbf{y}$. For example, ρ can be an edit distance between $\mathbf{y}$ and $\mathbf{y}'$. The primary goal of the algorithm is to minimize its cumulative cost $\sum_t \rho(\mathbf{y}_t, h_{\mathbf{w}_t}(\mathbf{x}_t))$. In general, ρ is not necessarily a convex function of $\mathbf{w}$. To resolve this issue, we suggest two strategies.

Maximal loss function The first approach is to define the following generalization of the hinge-loss as our loss function:

$$\ell(h_{\mathbf{w}}, (\mathbf{x}, \mathbf{y}) \;=\; \max_{\mathbf{y}' \neq \mathbf{y}} \left[\rho(\mathbf{y}, \mathbf{y}') - \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}, \mathbf{y}) - \boldsymbol{\phi}(\mathbf{x}, \mathbf{y}')\rangle\right]_+ \;\;. \tag{5.22}$$

To apply our algorithmic framework for the loss function given in Eq. (5.22) we again note that a subgradient of the function $g_t(\mathbf{w}) = \ell(h_{\mathbf{w}}, (\mathbf{x}, y))$ is

$$\boldsymbol{\lambda} \;=\; \begin{cases} \boldsymbol{\phi}(\mathbf{x}, \hat{\mathbf{y}}) - \boldsymbol{\phi}(\mathbf{x}, y) & \text{if } g_t(\mathbf{w}) > 0 \\ \mathbf{0} & \text{otherwise} \end{cases} \;\;,$$

where

$$\hat{\mathbf{y}} \;=\; \operatorname*{argmax}_{\mathbf{y}' \neq \mathbf{y}} \rho(\mathbf{y}', \mathbf{y}) - \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}, \mathbf{y}) - \boldsymbol{\phi}(\mathbf{x}, \mathbf{y}')\rangle \;\;. \tag{5.23}$$

In particular, the above implies that the function $g_t(\mathbf{w})$ is X-Lipschitz with respect to any norm $\|\cdot\|$, where $X = \max_{\mathbf{y}' \neq \mathbf{y}} \|\boldsymbol{\phi}(\mathbf{x}, \mathbf{y}') - \boldsymbol{\phi}(\mathbf{x}, \mathbf{y})\|$. Therefore, we can use Figure 3.1 and Figure 3.2 to derive online learning algorithms for structured output prediction problems and analyze them based on Corollary 1 and Corollary 3.

The disadvantage of the above approach is that in order to apply it we must impose structural restrictions on the function ρ since otherwise we cannot efficiently solve the optimization problem

```
PARAMETERS: A constant $U > 0$ ; Cost function $\rho : \Sigma^m \times \Sigma^m \to \{0\} \cup [1, \infty)$
INITIALIZE: $\mathbf{w}_1 = 0$ ; $X_0 = 0$ ; $M_0 = 0$
FOR $t = 1, 2, \ldots, T$
    Receive an instance $\mathbf{x}_t$
    Predict $\mathbf{y}'_t = \arg\max_{\mathbf{y}} \langle \mathbf{w}_t, \boldsymbol{\phi}(\mathbf{x}_t, \mathbf{y}) \rangle$
    Receive correct target $\mathbf{y}_t$
    IF $\rho(\mathbf{y}_t, \mathbf{y}'_t) > 0$
        Set $M_t = M_{t-1} + 1$
        Set $\mathbf{a}_t = \boldsymbol{\phi}(\mathbf{x}_t, \mathbf{y}_t) - \boldsymbol{\phi}(\mathbf{x}_t, \mathbf{y}'_t)$
        Set $X_t = \max\{\|\mathbf{a}_t\| \, , \, X_{t-1}\}$
        Set $c_t = X_t \sqrt{M_t/(2U)}$
        Set $\alpha_t = \min\{1/c_t \, , \, \frac{\rho(\mathbf{y}_t, \mathbf{y}'_t) - \langle \mathbf{w}_t, \mathbf{a}_t \rangle}{\|a_t\|^2}\}$
        Set $\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha_t \, \mathbf{a}_t$
    ELSE
        Set $X_t = X_{t-1}$ ; $c_t = c_{t-1}$ ; $\mathbf{w}_{t+1} = \mathbf{w}_t$
```

Figure 5.4: A self-tuned algorithm for learning with structured output when the range of the cost function ρ is $\{0\} \cup [1, \infty)$.

given in Eq. (5.23). For example, we can impose on ρ similar restrictions to the restrictions we imposed on $\boldsymbol{\phi}$ in Eq. (5.21). In this case, Eq. (5.23) can be solved efficiently using a dynamic programming procedure. The second approach we discuss below lifts this restriction.

Prediction based loss function In our second approach we define the loss function on round t to be

$$\ell(h_{\mathbf{w}}, (\mathbf{x}_t, \mathbf{y}_t) \;=\; [\rho(\mathbf{y}_t, \hat{\mathbf{y}}_t) - \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}_t, \mathbf{y}_t) - \boldsymbol{\phi}(\mathbf{x}_t, \hat{\mathbf{y}}_t) \rangle]_+ \quad , \tag{5.24}$$

where $\hat{\mathbf{y}}_t = h_{\mathbf{w}_t}(\mathbf{x}_t)$. That is, the loss depends on the actual prediction the algorithm makes on round t. While such a definition may seem unusual, note that in online convex programming, we receive the loss function $g_t(\mathbf{w})$ only after we define $\mathbf{w}_t$ and therefore the definition of $g_t(\mathbf{w})$ based on $\mathbf{w}_t$ does not pose any problem.

Next, note that a subgradient of $g_t(\mathbf{w})$ at $\mathbf{w}_t$ is the zero vector if $g_t(\mathbf{w}_t) = 0$ and otherwise $\boldsymbol{\lambda} = \boldsymbol{\phi}(\mathbf{x}_t, \hat{\mathbf{y}}_t) - \boldsymbol{\phi}(\mathbf{x}_t, \mathbf{y}_t)$. Thus, as opposed to the Maximal loss given in Eq. (5.22), the subgradient of the prediction based loss can be calculated without imposing any restrictions on ρ.

The prediction based loss function given in Eq. (5.24) is X-Lipschitz with respect to any norm $\|\cdot\|$, where $X = \|\boldsymbol{\phi}(\mathbf{x}_t, \hat{\mathbf{y}}_t) - \boldsymbol{\phi}(\mathbf{x}_t, \mathbf{y}_t)\|$. Therefore, we can use Figure 3.1 to derive online learning algorithms for structured output prediction problems and analyze them based on Corollary 1. In particular, combining Corollary 1 with the inequalities

$$\rho(\mathbf{y}_t, \hat{\mathbf{y}}_t) \;\le\; g_t(\mathbf{w}_t) \quad \text{and} \quad g_t(\mathbf{u}) \;\le\; \max_{\mathbf{y}' \neq \mathbf{y}_t} \left[\rho(\mathbf{y}_t, \mathbf{y}') - \langle \mathbf{u}, \boldsymbol{\phi}(\mathbf{x}_t, \mathbf{y}_t) - \boldsymbol{\phi}(\mathbf{x}_t, \mathbf{y}'_t)\rangle\right]_+ ,$$

we obtain that

$$\sum_{t=1}^{T} \rho(\mathbf{y}_t, \hat{\mathbf{y}}_t) \;\le\; \sum_{t=1}^{T} \max_{\mathbf{y}' \neq \mathbf{y}_t} \left[\rho(\mathbf{y}_t, \mathbf{y}') - \langle \mathbf{u}, \boldsymbol{\phi}(\mathbf{x}_t, \mathbf{y}_t) - \boldsymbol{\phi}(\mathbf{x}_t, \mathbf{y}'_t)\rangle\right]_+ \;+\; c\, f(\mathbf{u}) + \frac{X^2\, T}{2\, c} \;.$$

Setting $c = \sqrt{T}$ we obtain a bound that grows as $O(\sqrt{T})$. A similar bound can be obtained by using the algorithmic framework given in Figure 3.2 and setting c_t to be proportional to $\sqrt{t}$.

In some situations, the range of the function ρ is $\{0\} \cup [1, \infty)$. For example, ρ can classify strings as being reasonably correct ($\rho(\mathbf{y}, \mathbf{y}') = 0$), slightly incorrect ($\rho(\mathbf{y}, \mathbf{y}') = 1$), wrong ($\rho(\mathbf{y}, \mathbf{y}') = 2$), and grossly wrong ($\rho(\mathbf{y}, \mathbf{y}') = 3$). In Figure 5.4 we describe an algorithm for this case. This algorithm is derived from Figure 3.2 by setting $f(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|_2^2$ and using the update rule given in Eq. (3.12). The assumption that ρ is either 0 or greater than 1 yields the following improved bound:

Theorem 7 *Let* $(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_T, \mathbf{y}_T)$ *be a sequence of classification examples and assume that this sequence is presented to the algorithm given in Figure 5.4. Denote* $X = \max_t \max_{\mathbf{y}' \neq \mathbf{y}_t} \|\boldsymbol{\phi}(\mathbf{x}_t, \mathbf{y}') - \boldsymbol{\phi}(\mathbf{x}_t, \mathbf{y}_t)\|$ *and*

$$\mathrm{Loss}(\mathbf{u}) \;=\; \sum_{t=1}^{T} \max_{\mathbf{y}' \neq \mathbf{y}_t} \left[\rho(\mathbf{y}_t, \mathbf{y}') - \langle \mathbf{u}, \boldsymbol{\phi}(\mathbf{x}_t, \mathbf{y}_t) - \boldsymbol{\phi}(\mathbf{x}_t, \mathbf{y}')\rangle\right]_+ \;.$$

Then, for all $\mathbf{u} \in S$ *that satisfies* $\|\mathbf{u}\| \le \sqrt{2\,U}$ *we have*

$$\sum_{t=1}^{T} \rho(\mathbf{y}_t, \mathbf{y}'_t) \;\le\; \mathrm{Loss}(\mathbf{u}) + X\,\sqrt{8\,U\,\mathrm{Loss}(\mathbf{u})} + 8\,X^2\,U \;.$$

Proof Let $\mathcal{M} = \{t \in [T] : \rho(\mathbf{y}_t, \mathbf{y}'_t) > 0\}$. Using Corollary 3 on the set of rounds in $\mathcal{M}$ with the loss functions $g_t(\mathbf{w}) = [\rho(\mathbf{y}_t, \mathbf{y}'_t) - \langle \mathbf{w}, \mathbf{a}_t\rangle]_+$ and the update rule given in Eq. (3.12) we obtain that

$$\sum_{t \in \mathcal{M}} g_t(\mathbf{w}_t) - \sum_{t \in \mathcal{M}} g_t(\mathbf{u}) \;\le\; X\,\sqrt{8\,U\,|\mathcal{M}|} \;.$$

Next, note that for all $t \in \mathcal{M}$ we have $g_t(\mathbf{w}_t) \geq \rho(\mathbf{y}_t, \mathbf{y}'_t) \geq 1$. Therefore, the above yields

$$\sum_{t \in \mathcal{M}} \rho(\mathbf{y}_t, \mathbf{y}'_t) - \sum_{t \in \mathcal{M}} g_t(\mathbf{u}) \;\leq\; X \sqrt{8\,U \sum_{t \in \mathcal{M}} \rho(\mathbf{y}_t, \mathbf{y}'_t)} \;.$$

Using the facts that $\sum_{t \in \mathcal{M}} \rho(\mathbf{y}_t, \mathbf{y}'_t) = \sum_{t=1}^{T} \rho(\mathbf{y}_t, \mathbf{y}'_t)$ and that $\mathrm{Loss}(\mathbf{u}) \geq \sum_{t \in \mathcal{M}} g_t(\mathbf{u})$ we get that

$$\sum_{t=1}^{T} \rho(\mathbf{y}_t, \mathbf{y}'_t) - X \sqrt{8\,U \sum_{t=1}^{T} \rho(\mathbf{y}_t, \mathbf{y}'_t)} - \mathrm{Loss}(\mathbf{u}) \;\leq\; 0 \;.$$

Combining the above with Lemma 19 concludes our proof. ■

5.3.4 Instance Ranking

We now demonstrate the applicability of our algorithmic framework to the problem of instance ranking. We analyze this setting since several prediction problems, including multilabel prediction, and label ranking, can be cast as special cases of the instance ranking problem.

In instance ranking, the question is encoded by a matrix X_t of dimension $k_t \times n$ and the answer is a vector $\mathbf{y}_t \in \mathbb{R}^{k_t}$. The semantic of $\mathbf{y}_t$ is as follows. For any pair (i, j), if $y_{t,i} > y_{t,j}$ then we say that $\mathbf{y}_t$ *ranks* the i'th row of X_t ahead of the j'th row of X_t. We also interpret $y_{t,i} - y_{t,j}$ as the confidence in which the i'th row should be ranked ahead of the j'th row. For example, each row of X_t encompasses a representation of a movie while $y_{t,i}$ is the movie's rating, expressed as the number of stars this movie has been awarded by a movie reviewer. The learner's predictions are determined based on a weight vector $\mathbf{w}_t \in \mathbb{R}^n$ and are defined to be $\hat{\mathbf{y}}_t = X_t\,\mathbf{w}_t$. Finally, let us define two loss functions for ranking; both generalize the hinge-loss used in binary classification problems. Denote by E_t the set $\{(i, j) : y_{t,i} > y_{t,j}\}$. For all $(i, j) \in E_t$ we define a pair-based hinge-loss $\ell_{i,j}(\mathbf{w}, (X_t, \mathbf{y}_t)) = [(y_{t,i} - y_{t,j}) - \langle \mathbf{w}, \mathbf{x}_{t,i} - \mathbf{x}_{t,j} \rangle]_+$, where $[a]_+ = \max\{a, 0\}$ and $\mathbf{x}_{t,i}, \mathbf{x}_{t,j}$ are respectively the i'th and j'th rows of X_t. Note that $\ell_{i,j}$ is zero if $\mathbf{w}$ ranks $\mathbf{x}_{t,i}$ higher than $\mathbf{x}_{t,j}$ with sufficient confidence. Ideally, we would like $\ell_{i,j}(\mathbf{w}_t, (X_t, \mathbf{y}_t))$ to be zero for all $(i, j) \in E_t$. If this is not the case, we are penalized according to some combination of the pair-based losses $\ell_{i,j}$. For example, we can set $\ell(\mathbf{w}, (X_t, \mathbf{y}_t))$ to be the average over the pair losses,

$$\ell^{\mathrm{avg}}(\mathbf{w}, (X_t, \mathbf{y}_t)) \;=\; \frac{1}{|E_t|} \sum_{(i,j) \in E_t} \ell_{i,j}(\mathbf{w}, (X_t, \mathbf{y}_t)) \;. \tag{5.25}$$

This loss was suggested by several authors (see for example [121]). Another popular approach (see for example [28]) penalizes according to the maximal loss over the individual pairs,

$$\ell^{\max}(\mathbf{w},(X_t,\mathbf{y}_t)) \;=\; \max_{(i,j)\in E_t} \ell_{i,j}(\mathbf{w},(X_t,\mathbf{y}_t)) \;\; . \tag{5.26}$$

We can apply our algorithmic frameworks given in Figure 3.1 and Figure 3.2 for ranking, using for $g_t(\mathbf{w})$ either $\ell^{\text{avg}}(\mathbf{w},(X_t,\mathbf{y}_t))$ or $\ell^{\max}(\mathbf{w},(X_t,\mathbf{y}_t))$. Denote by X_t the maximum over $(i,j)\in E_t$ of $\|\mathbf{x}_{t,i}-\mathbf{x}_{t,j}\|^2$. Then, both $g_t(\mathbf{w}) = \ell^{\text{avg}}(\mathbf{w},(X_t,\mathbf{y}_t))$ and $g_t(\mathbf{w}) = \ell^{\max}(\mathbf{w},(X_t,\mathbf{y}_t))$ are X_t-Lipschitz with respect to the norm $\|\cdot\|$. Therefore, we can again use Corollary 1 and Corollary 3 for analyzing the resulting algorithms.

5.4 Bibliographic Notes

The Perceptron dates back to Rosenblatt [95]. An analysis for the separable case (i.e. there exists a vector that achieves zero cumulative hinge-loss) appears in [1, 87]. Freund and Schapire [53] presented an analysis for the non-separable case with a squared-hinge-loss based on a reduction to the separable case. An analysis for the inseparable case with the hinge-loss was given by Gentile [58].

Winnow was invented by Littlestone [82]. It is also closely related to the Weighted Majority algorithm [85] and to the EG updates [75]. The p-norm algorithm was developed by Gentile and Littlestone [59]. The class of quasi-additive algorithms was described in [62]. The analysis in [62] is for the separable case. A similar derivation for regression problems was made by [6, 75, 76]. Kivinen and Warmuth analyzed their algorithms in the non-separable case using the regret and mistake bound models. They use the terminology “relative loss bounds”.

Self-tuned algorithms for regression problems with the squared and absolute loss were proposed in [5]. They also discuss the difference between self-tuning techniques and the “doubling trick”.

A special case of the aggressive quasi-additive update as given in Eq. (5.14), in the special case where $f(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|_2^2$, was called a Passive-Aggressive update in [28]. The mistake bounds presented here are tighter than the mistake bounds given in [28]. In addition, the self-tuned version given in Figure 5.3 is new. The idea of updating multiple dual variables is also novel.

A simple reduction from online multiclass categorization to the binary classification setting dates back to Kessler (see Duda and Hart’s book [44]). A more sophisticated algorithm, called MIRA, was suggested by Crammer and Singer [31]. A detailed description of online learning with complex prediction problems can be found for example in [28].

The basic extension of online multiclass categorization to the structured output setting was suggested by Collins [22]. The idea of imposing structural restrictions on the feature mapping ϕ was also proposed in [3, 113, 115]. The introduction of the cost function in the context of online learning was suggested in [102]. Prediction based loss functions vs. Maximal loss functions were discussed in [28].

Chapter 6

Boosting

6.1 A Primal-Dual Perspective of Boosting

In this chapter we describe the applicability of our algorithmic framework to the analysis of boosting algorithms. A boosting algorithm uses a weak learning algorithm that generates weak hypotheses, whose performances are just slightly better than random guessing, to build a strong hypothesis, which can attain an arbitrarily low error. The first boosting algorithm was derived by Schapire [98] and later on, Freund and Schapire [54] proposed a more efficient boosting algorithm called AdaBoost.

The AdaBoost algorithm receives as input a training set of examples $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$ where for all $i \in [m]$, $\mathbf{x}_i$ is taken from an instance domain $\mathcal{X}$, and y_i is a binary label, $y_i \in \{+1, -1\}$. The boosting process proceeds in a sequence of consecutive rounds. At round t, the booster first defines a distribution, denoted $\mathbf{w}_t$, over the set of examples. Then, the booster passes the training set along with the distribution $\mathbf{w}_t$ to the weak learner. The weak learner is assumed to return a hypothesis $h_t : \mathcal{X} \to \{+1, -1\}$ whose average error is slightly smaller than $\frac{1}{2}$. That is, there exists a constant $\gamma > 0$ such that,

$$\epsilon_t \overset{def}{=} \sum_{i=1}^{m} w_{t,i} \, \frac{1 - y_i h_t(\mathbf{x}_i)}{2} \;\leq\; \frac{1}{2} - \gamma \ . \tag{6.1}$$

The goal of the boosting algorithm is to invoke the weak learner several times with different distributions, and to combine the hypotheses returned by the weak learner into a final, so-called strong hypothesis whose error is small. The final hypothesis combines linearly the T hypotheses returned by the weak learner with coefficients $\alpha_1, \ldots, \alpha_T$, and is defined to be the sign of $h_f(\mathbf{x})$ where

$h_f(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t\, h_t(\mathbf{x})$. The coefficients $\alpha_1, \ldots, \alpha_T$ are determined by the booster. In AdaBoost, the initial distribution is set to be the uniform distribution, $\mathbf{w}_1 = (\frac{1}{m}, \ldots, \frac{1}{m})$. At iteration t, the value of α_t is set to be $\frac{1}{2}\log((1-\epsilon_t)/\epsilon_t)$. The distribution is updated by the rule $w_{t+1,i} = w_{t,i}\, \exp(-\alpha_t\, y_i\, h_t(\mathbf{x}_i))/Z_t$, where Z_t is a normalization factor. Freund and Schapire [54] have shown that under the assumption given in Eq. (6.1), the error of the final strong hypothesis is at most $\exp(-2\,\gamma^2\,T)$.

Several authors [97, 86, 56, 24] have suggested that boosting is best seen as a coordinate-wise greedy optimization process. To do this, note first that h_f errs on an example $(\mathbf{x}, y)$ iff $y\, h_f(\mathbf{x}) \le 0$. Therefore, the exp-loss function, defined as $\exp(-y\ h_f(\mathbf{x}))$, is a smooth upper bound of the zero-one error, which equals 1 if $y\, h_f(\mathbf{x}) \le 0$ and 0 otherwise. Thus, we can restate the goal of boosting as minimizing the average exp-loss of h_f over the training set with respect to the variables $\alpha_1, \ldots, \alpha_T$. To simplify our derivation in the sequel, we prefer to say that boosting maximizes the negation of the loss, that is,

$$\max_{\alpha_1,\ldots,\alpha_T} \quad -\frac{1}{m}\sum_{i=1}^{m} \exp\left(-y_i \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}_i)\right) \quad . \tag{6.2}$$

In this view, boosting is an optimization procedure which iteratively maximizes Eq. (6.2) with respect to the variables $\alpha_1, \ldots, \alpha_T$. This view of boosting enables the hypotheses returned by the weak learner to be general functions into the reals, $h_t : \mathcal{X} \to \mathbb{R}$ (see for instance [97]).

In this section we view boosting as a primal-dual online convex programming procedure. To motivate our construction, note that boosting algorithms define weights in two different domains: the vectors $\mathbf{w}_t \in \mathbb{R}^m$ which assign weights to *examples* and the weights $\{\alpha_t : t \in [T]\}$ over *weak-hypotheses*. In the terminology used throughout this manuscript, the weights $\mathbf{w}_t \in \mathbb{R}^m$ are *primal* vectors while (as we show in the sequel) each weight α_t of the hypothesis h_t is related to a *dual* vector $\boldsymbol{\lambda}_t$. In particular, we show that Eq. (6.2) is exactly the Fenchel dual of a primal problem for an online convex programming; thus the algorithmic framework described so far for online convex programming naturally fits the problem of iteratively solving Eq. (6.2).

To derive the primal problem whose Fenchel dual is the problem given in Eq. (6.2) let us first denote by $\mathbf{v}_t$ the vector in $\mathbb{R}^m$ whose ith element is $v_{t,i} = y_i h_t(\mathbf{x}_i)$. For all t, we set g_t to be the function $g_t(\mathbf{w}) = [\langle \mathbf{w}, \mathbf{v}_t\rangle]_+$. Intuitively, g_t penalizes vectors $\mathbf{w}$ that assign large weights to examples which are predicted accurately, that is $y_i h_t(\mathbf{x}_i) > 0$. In particular, if $h_t(\mathbf{x}_i) \in \{+1, -1\}$ and $\mathbf{w}_t$ is a distribution over the m examples (as is the case in AdaBoost), $g_t(\mathbf{w}_t)$ reduces to $1 - 2\epsilon_t$ (see Eq. (6.1)). In this case, minimizing g_t is equivalent to maximizing the error of the individual hypothesis h_t over the examples. Consider the problem of minimizing $c\, f(\mathbf{w}) + \sum_{t=1}^{T} g_t(\mathbf{w})$ where $f(\mathbf{w}) = \log(m) + \sum_i w_i \log(w_i)$ is the relative entropy and c is a scalar to be set later. To derive its

Fenchel dual, we note that $g_t^\star(\boldsymbol{\lambda}_t) = 0$ if there exists $\beta_t \in [0,1]$ such that $\boldsymbol{\lambda}_t = \beta_t \mathbf{v}_t$ and otherwise $g_t^\star(\boldsymbol{\lambda}_t) = \infty$ (see Appendix A.3.1). In addition, let us define $\alpha_t = \frac{1}{c}\beta_t$. Since our goal is to maximize the dual, we can restrict $\boldsymbol{\lambda}_t$ to take the form $\boldsymbol{\lambda}_t = \beta_t \mathbf{v}_t = c\,\alpha_t \mathbf{v}_t$, and get that

$$\mathcal{D}(\boldsymbol{\lambda}_1,\ldots,\boldsymbol{\lambda}_T) = -c\, f^\star\left(-\frac{1}{c}\sum_{t=1}^{T}\beta_t\,\mathbf{v}_t\right) = -c\,\log\left(\frac{1}{m}\sum_{i=1}^{m} e^{-\sum_{t=1}^{T}\alpha_t y_i h_t(\mathbf{x}_i)}\right) . \tag{6.3}$$

Minimizing the exp-loss of the strong hypothesis is therefore the dual problem of the following primal minimization problem: find a distribution over the examples whose relative entropy to the uniform distribution is as small as possible, while the correlation of the distribution with each $\mathbf{v}_t$ is as small as possible. Since the correlation of $\mathbf{w}$ with $\mathbf{v}_t$ is inversely proportional to the error of h_t with respect to $\mathbf{w}$, we obtain that in the primal problem we want to *maximize* the error of each *individual* hypothesis, while in the dual problem we *minimize* the global error of the *strong* hypothesis. The intuition of finding distributions which in retrospect result in large error rates of individual hypotheses was also alluded to in [97, 56].

We can now apply our algorithmic framework from Section 3.3 to boosting. We describe the boosting process with the parameters α_t, where $\alpha_t \in [0, 1/c]$, and stress that in our case, $\boldsymbol{\lambda}_t = c\,\alpha_t\,\mathbf{v}_t$. At the beginning of the boosting process the booster sets all dual variables to be zero, $\forall t\ \alpha_t = 0$. At round t, the booster first constructs a primal weight vector $\mathbf{w}_t \in \mathbb{R}^m$, which assigns importance weights to the examples in the training set. The primal vector $\mathbf{w}_t$ is constructed as in Eq. (3.7), that is, $\mathbf{w}_t = \nabla f^\star(\boldsymbol{\theta}_t)$, where $\boldsymbol{\theta}_t = -\sum_{i<t}\alpha_i \mathbf{v}_i$. We can also rewrite the kth element of $\mathbf{w}_t$ as follows:

$$w_{t,k} = \frac{e^{\theta_{t,k}}}{Z_t} = \frac{e^{-\sum_{i<t}\alpha_i v_{i,k}}}{Z_t} ,$$

where Z_t is a normalization factor that ensures that $\mathbf{w}_t$ sums to 1. Next, the weak learner responds by presenting the loss function $g_t(\mathbf{w}) = [\langle \mathbf{w}, \mathbf{v}_t\rangle]_+$. Finally, the booster updates the dual variables so as to increase the dual objective function.

We next show that the update rule given in Eq. (3.11) results in a boosting algorithm, which is closely related to the "boost-by-majority" algorithm presented in [52]. To see this, we note that the definition of g_t and the weak learnability assumption given in Eq. (6.1) imply that $\mathbf{v}_t$ is a subgradient of $g_t(\mathbf{w})$ at $\mathbf{w}_t$. Since in our case $\boldsymbol{\lambda}_t = c\,\alpha_t\,\mathbf{v}_t$ we obtain that Eq. (3.11) is equivalent to the update rule $\alpha_t = 1/c$. Thus, the final strong hypothesis becomes $h_f(\mathbf{x}) = \frac{1}{c}\sum_t h_t(\mathbf{x})$, which is equivalent to the majority vote of the weak hypotheses if the range of each weak hypothesis is $\{+1, -1\}$. The update of the distribution vector $\mathbf{w}_t$ can be rewritten as

$$w_{t+1,k} = \frac{w_{t,k}\, e^{-\frac{1}{c} y_k h_t(\mathbf{x}_k)}}{Z_t} ,$$

where Z_t is a normalization factor.

To analyze the boosting algorithm, we note that the conditions given in Lemma 1 hold and therefore the left-hand side inequality given in Lemma 1 tells us that

$$\sum_{t=1}^{T} g_t(\mathbf{w}_t) - \frac{1}{2\,c} \sum_{t=1}^{T} \|\boldsymbol{\lambda}'_t\|_\infty^2 \;\leq\; \mathcal{D}(\boldsymbol{\lambda}_1^{T+1}, \ldots, \boldsymbol{\lambda}_T^{T+1}) \;. \tag{6.4}$$

The definition of g_t and the weak learnability assumption given in Eq. (6.1) imply that $\langle \mathbf{w}_t, \mathbf{v}_t \rangle \geq 2\,\gamma$ for all t. Thus, $g_t(\mathbf{w}_t) = \langle \mathbf{w}_t, \mathbf{v}_t \rangle \geq 2\,\gamma$, which also implies that $\boldsymbol{\lambda}'_t = \mathbf{v}_t$. Recall that $v_{t,i} = y_i h_t(\mathbf{x}_i)$. Assuming that the range of h_t is $[+1, -1]$ we get that $\|\boldsymbol{\lambda}'_t\|_\infty \leq 1$. Combining all the above with Eq. (6.4) we get that

$$2\,T\,\gamma - \frac{T}{2\,c} \;\leq\; \mathcal{D}(\boldsymbol{\lambda}_1^{T+1}, \ldots, \boldsymbol{\lambda}_T^{T+1}) \;=\; -\,c\,\log\left(\frac{1}{m}\sum_{i=1}^{m} e^{-y_i \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}_i)}\right) \;,$$

where in the last equality we used the definition of $\mathcal{D}$ as given in Eq. (6.3). Setting $c = 1/(2\,\gamma)$ and rearranging terms we recover the original boosting bound

$$\frac{1}{m}\sum_{i=1}^{m} e^{-y_i \sum_{t=1}^{T} \alpha_t\, h_t(\mathbf{x}_i)} \;\leq\; e^{-2\,\gamma^2\,T} \;.$$

A disadvantage of the algorithm above is that we need to know the parameter γ. The AdaBoost algorithm lifts this limitation. In the next section we show that the AdaBoost algorithm can be derived from our algorithmic framework by using the dual update rule given in Eq. (3.12). As a by-product, we also derive the Gentle-Boost algorithm [56] from our framework.

6.2 AdaBoost

Let us now consider the boosting process described above with the update rule given in Eq. (3.12). We assume that the range of each h_t is $\{+1, -1\}$ and thus the elements of $\mathbf{v}_t$ are also in $\{+1, -1\}$. Denote

$$w_t^+ = \sum_{i:v_{t,i}=1} w_{t,i} \quad ; \quad w_t^- = \sum_{i:v_{t,i}=-1} w_{t,i} \;.$$

Based on Eq. (7.16) given in Section 7.2 it is easy to verify that Eq. (3.12) yields:

$$\alpha_t \;=\; \min\left\{\tfrac{1}{c}\,,\,\tfrac{1}{2}\,\log\left(\tfrac{w_t^+}{w_t^-}\right)\right\} \;. \tag{6.5}$$

Note that the definition of ϵ_t given in Eq. (6.1) implies that $2\epsilon_t = 1 - w_t^+ + w_t^-$. Combining the above with the fact that $w_t^+ + w_t^- = 1$ gives that $w_t^- = \epsilon_t$ and $w_t^+ = 1 - \epsilon_t$. Assume that c is sufficiently small[1] we obtain that Eq. (6.5) can be rewritten as

$$\alpha_t = \tfrac{1}{2} \log\left(\tfrac{1-\epsilon_t}{\epsilon_t}\right) \ , \tag{6.6}$$

and we have recovered the original AdaBoost algorithm.

To analyze the AdaBoost algorithm, denote $\boldsymbol{\theta}_t = -\sum_{i<t} \alpha_i \mathbf{v}_i$ and note that the increase in the dual due to the update step given in Eq. (6.6) is

$$\begin{aligned}
\Delta_t &= -c\left(f^\star(\boldsymbol{\theta}_t - \alpha_t \mathbf{v}_t) - f^\star(\boldsymbol{\theta}_t)\right) & (6.7)\\
&\geq -c\left(-\alpha_t\langle \mathbf{w}_t, \mathbf{v}_t\rangle + \tfrac{1}{2}\alpha_t^2\|\mathbf{v}_t\|_\infty^2\right) & (6.8)\\
&\geq -c\left(-\alpha_t\langle \mathbf{w}_t, \mathbf{v}_t\rangle + \tfrac{1}{2}\alpha_t^2\right) & (6.9)\\
&\geq c\,\frac{(\langle \mathbf{w}_t, \mathbf{v}_t\rangle)^2}{2} & (6.10)\\
&\geq c\,2\,\gamma^2 \ . & (6.11)
\end{aligned}$$

In the above, Eq. (6.7) follows from the definition of $\mathcal{D}$, Eq. (6.8) follows from Lemma 15 in Section A.4 and the definition of $\mathbf{w}_t$, Eq. (6.9) follows from the assumption $|h_t(\mathbf{x})| \leq 1$, Eq. (6.10) follows from the fact that α_t maximizes Δ_t, and Eq. (6.11) follows from the weak learnability assumption given in Eq. (6.1). We note in passing that the same bound would have been obtained had we set α_t to be $\langle \mathbf{w}_t, \mathbf{v}_t\rangle$. In fact, this boosting variant was proposed by Friedman et al [56] and is called Gentle-Boost.

Summing over t and using the fact that $\mathcal{D}(0,\ldots,0) = 0$ we obtain that

$$c\,2\,\gamma^2\,T \ \leq\ \mathcal{D}(\alpha_1,\ldots,\alpha_T) \ =\ -\,c\,\log\left(\frac{1}{m}\sum_{i=1}^{m} e^{-y_i \sum_{t=1}^{T}\alpha_t h_t(\mathbf{x}_i)}\right) \ . \tag{6.12}$$

Rearranging terms in the above we recover the original celebrated boosting bound

$$\frac{1}{m}\sum_{i=1}^{m} e^{-y_i \sum_{t=1}^{T}\alpha_t\, h_t(\mathbf{x}_i)} \ \leq\ e^{-2\,\gamma^2\,T} \ .$$

[1]Note that the bound we derive later in Eq. (6.12) does not depend on the value of c and thus we are allowed to set c to be arbitrarily small. Note in addition that by setting $c = 1/(2\gamma)$ we obtain a variant of AdaBoost in which the weights α_t are capped above by $2\,\gamma$. This variant enjoys the same bound as AdaBoost. However, a disadvantage of this variant is that we need to know the parameter γ.

6.3 Bibliographic Notes

Boosting algorithms are effective batch learning procedures. Freund and Schapire [55] were the first to relate boosting to online learning. For an analysis of the generalization properties of boosting algorithms see for example [99] and the references therein. Our algorithmic framework can be used to derive additional boosting algorithms such as the LogitBoost [56] and TotalBoost [120] algorithms.

Chapter 7

Efficient Implementation

In previous chapters, we were mainly interested in understanding the regret properties of different update procedures. In this chapter we address the computational aspects of updating the online hypothesis. First, as we showed in Example 1 in Chapter 4, calculating the gradient of $f^\star$ may involve a projection operation. We start in Section 7.1 by describing efficient techniques for projecting onto ℓ_1 balls and onto the probabilistic simplex. We discuss both Euclidean and Entropic projections. The remaining sections in this chapter focus on the computational aspects of the aggressive update rule given in Eq. (3.12). Depending on the loss function at hand, we derive procedures for solving the optimization problem given in Eq. (3.12) with increasing computational complexity. In Section 7.2 we provide analytical solutions for the hinge-loss. Next, in Section 7.3 we extend ideas from Section 7.1 and derive efficient procedures for solving Eq. (3.12) for a specific loss function encountered in ranking problems. Finally, in Section 7.4 we describe an iterative procedure for a more general loss function that is widely used in various complex prediction problems.

7.1 Projections onto ℓ_1 Balls and onto the Probabilistic Simplex

7.1.1 Euclidean Projections onto the Probabilistic Simplex

In this section we describe an efficient procedure for solving the following problem:

$$\min_{\boldsymbol{\alpha}} \frac{1}{2}\|\boldsymbol{\alpha} - \boldsymbol{\mu}\|_2^2 \quad \text{s.t.} \quad \sum_{i=1}^{p} \alpha_i \ = \ z \ , \ \ \alpha_i \geq 0 \ . \tag{7.1}$$

If $z = 1$ the above problem is a Euclidean projection onto the probabilistic simplex. To simplify our notation, we use the shorthand $\|\cdot\|$ for denoting the Euclidean norm $\|\cdot\|_2$.

The Lagrangian of the problem in Eq. (7.1) is,

$$\mathcal{L} \;=\; \frac{1}{2}\|\boldsymbol{\alpha}-\boldsymbol{\mu}\|^2 + \theta\left(\sum_{i=1}^{p}\alpha_i - z\right) - \boldsymbol{\zeta}\cdot\boldsymbol{\alpha} \;\;,$$

where $\theta \in \mathbb{R}$ is a Lagrange multiplier and $\boldsymbol{\zeta} \in \mathbb{R}^p_+$ is a vector of non-negative Lagrange multipliers. Differentiating with respect to α_i and comparing to zero gives the following KKT condition,

$$\frac{d\mathcal{L}}{d\alpha_i} \;=\; \alpha_i - \mu_i + \theta - \zeta_i \;=\; 0 \;\;.$$

The complementary slackness KKT condition implies that whenever $\alpha_i > 0$ we must have that $\zeta_i = 0$. Thus, if $\alpha_i > 0$ we get that,

$$\alpha_i \;=\; \mu_i - \theta + \zeta_i \;=\; \mu_i - \theta \;\;. \tag{7.2}$$

Since all the non-negative elements of the vector $\boldsymbol{\alpha}$ are tied via a single variable we would have ended up with a much simpler problem had we known the indices of these elements. At first sight, this task seems difficult, as the number of potential subsets of $\boldsymbol{\alpha}$ is clearly exponential in the dimension of $\boldsymbol{\alpha}$. Fortunately, the particular form of the problem renders an efficient algorithm for identifying the non-zero elements of $\boldsymbol{\alpha}$. The following lemma is a key tool in deriving our procedure for identifying the non-zero elements.

Lemma 6 *Let $\boldsymbol{\alpha}$ be the optimal solution to the minimization problem in Eq. (7.1). Let s and j be two indices such that $\mu_s > \mu_j$. If $\alpha_s = 0$ then α_j must be zero as well.*

Proof Assume by contradiction that $\alpha_s = 0$ yet $\alpha_j > 0$. Let $\tilde{\boldsymbol{\alpha}} \in \mathbb{R}^k$ be a vector whose elements are equal to the elements of $\boldsymbol{\alpha}$ except for $\tilde{\alpha}_s$ and $\tilde{\alpha}_j$ which are interchanged, that is, $\tilde{\alpha}_s = \alpha_j$, $\tilde{\alpha}_j = \alpha_s$, and for every other $r \notin \{s, j\}$ we have $\tilde{\alpha}_r = \alpha_r$. It is immediately clear that the constraints of Eq. (7.1) still hold. In addition we have that,

$$\|\boldsymbol{\alpha}-\boldsymbol{\mu}\|^2 - \|\tilde{\boldsymbol{\alpha}}-\boldsymbol{\mu}\|^2 \;=\; \mu_s^2 + (\alpha_j-\mu_j)^2 - (\alpha_j-\mu_s)^2 - \mu_j^2 \;=\; 2\alpha_j(\mu_s-\mu_j) \;>\; 0 \;\;.$$

Therefore, we obtain that $\|\boldsymbol{\alpha}-\boldsymbol{\mu}\|^2 > \|\tilde{\boldsymbol{\alpha}}-\boldsymbol{\mu}\|^2$, which contradicts the fact that $\boldsymbol{\alpha}$ is the optimal solution. ■

Let I denote the set $\{i \in [p] : \alpha_i > 0\}$. The above lemma gives a simple characterization of the set I. Let us reorder the $\boldsymbol{\mu}$ such that $\mu_1 \geq \mu_2 \geq \ldots \geq \mu_p$. Simply put, Lemma 6 implies that after the reordering, the set I is of the form $\{1, \ldots, \rho\}$ for some $1 \leq \rho \leq p$. Had we known ρ we could have

```
INPUT: A vector $\boldsymbol{\mu} \in \mathbb{R}^p$ and a scalar $z > 0$
DO:
    Sort $\boldsymbol{\mu}$ so that $\mu_1 \geq \mu_2 \geq \ldots \geq \mu_p$
    Define $\rho = \max\{j \in [p] \ : \ \mu_j - \frac{1}{j}(\sum_{r=1}^{j} \mu_r - z) > 0\}$
    Define $\boldsymbol{\alpha}$ to be the vector so that
        For $i \leq \rho$:   $\alpha_i = \mu_i - \frac{1}{\rho}\left(\sum_{i=1}^{\rho} \mu_i - z\right)$
        For $i > \rho$:   $\alpha_i = 0$
OUTPUT: The vector $\boldsymbol{\alpha}$ is the solution of Eq. (7.1)
```

Figure 7.1: A procedure for projecting onto the probabilistic simplex.

simply used Eq. (7.2) and obtained that

$$\sum_{i=1}^{p} \alpha_i = \sum_{i=1}^{\rho} \alpha_i = \sum_{i=1}^{\rho} (\mu_i - \theta) = z \quad \Rightarrow \quad \theta = \frac{1}{\rho}\left(\sum_{i=1}^{\rho} \mu_i - z\right) \ .$$

To recapitulate, given ρ we can summarize the optimal solution for $\boldsymbol{\alpha}$ as follows,

$$\alpha_i \ = \ \begin{cases} \mu_i - \dfrac{1}{\rho}\left(\displaystyle\sum_{i=1}^{\rho} \mu_i - z\right) & i \leq \rho \\ 0 & i > \rho \end{cases} \ . \tag{7.3}$$

We are left with the problem of finding the optimal value of ρ. We could simply enumerate all possible values of ρ in $[p]$, for each possible value compute $\boldsymbol{\alpha}$ as given by Eq. (7.3), and then choose the value for which the objective function ($\|\boldsymbol{\alpha} - \boldsymbol{\mu}\|^2$) is the smallest. While this procedure can be implemented quite efficiently, the following lemma provides an even simpler solution once we reorder the elements of $\boldsymbol{\mu}$ to be in a non-increasing order.

Lemma 7 *Let $\boldsymbol{\alpha}$ be the optimal solution to the minimization problem given in Eq. (7.1) and assume that $\mu_1 \geq \mu_2 \geq \ldots \geq \mu_p$. Then, the number of strictly positive elements in $\boldsymbol{\alpha}$ is,*

$$\rho(z, \boldsymbol{\mu}) \ = \ \max\left\{ j \in [p] \ : \ \mu_j - \frac{1}{j}\left(\sum_{r=1}^{j} \mu_r - z\right) > 0 \right\} \ .$$

The proof of this technical lemma is given in Section A.5. A procedure for solving Eq. (7.1) is summarized in Figure 7.1.

7.1.2 Projections onto ℓ_1 balls

In this section we describe an efficient solution to the following problem,

$$\underset{\boldsymbol{\alpha}\in\mathbb{R}^n}{\operatorname{argmin}} \ \|\boldsymbol{\alpha}-\boldsymbol{\mu}\|_2^2 \quad \text{s.t.} \quad \|\boldsymbol{\alpha}\|_1 \le z \ . \tag{7.4}$$

We do so by presenting a reduction to the problem of projecting onto the simplex given in Eq. (7.1). First, we note that if $\|\boldsymbol{\mu}\|_1 \le z$ then the solution of Eq. (7.4) is $\boldsymbol{\alpha} = \boldsymbol{\mu}$. Therefore, from now on we assume that $\|\boldsymbol{\mu}\|_1 > z$. In this case, the optimal solution must be on the boundary of the constraint set and thus we can replace the inequality constraint $\|\boldsymbol{\alpha}\|_1 \le z$ with an equality constraint $\|\boldsymbol{\alpha}\|_1 = z$. Having done this, the sole difference between the problem in Eq. (7.4) and the one in Eq. (7.1) is that in the latter we have an additional positiveness constraint: $\boldsymbol{\alpha} \ge 0$.

The following lemma shows that the sign of the elements in $\boldsymbol{\mu}$ is preserved in $\boldsymbol{\alpha}$. We use this property later on to complete our reduction.

Lemma 8 *Let $\boldsymbol{\alpha}$ be an optimal solution of Eq. (7.4). Then, for all i, $\alpha_i\,\mu_i \ge 0$.*

Proof Assume by contradiction that the claim does not hold. Thus, there exists i for which $\alpha_i\,\mu_i < 0$. Let $\hat{\boldsymbol{\alpha}}$ be a vector such that $\hat{\alpha}_i = 0$ and for all $j \ne i$ we have $\hat{\alpha}_j = \alpha_j$. Therefore, $\|\hat{\boldsymbol{\alpha}}\|_1 = \|\boldsymbol{\alpha}\|_1 - |\alpha_i| \le z$ and hence $\hat{\boldsymbol{\alpha}}$ is a feasible solution. In addition,

$$\|\boldsymbol{\alpha}-\boldsymbol{\mu}\|_2^2 - \|\hat{\boldsymbol{\alpha}}-\boldsymbol{\mu}\|_2^2 \ = \ (\alpha_i-\mu_i)^2 - (0-\mu_i)^2 = \alpha_i^2 - 2\alpha_i\mu_i > \alpha_i^2 > 0 \,.$$

We have shown that $\hat{\boldsymbol{\alpha}}$ is a feasible solution whose objective value is smaller than that of $\boldsymbol{\alpha}$ which leads us to the desired contradiction. ■

Let $\boldsymbol{\nu}$ be a vector such that for all i, $\nu_i = |\mu_i|$. Based on the above lemma, we can replace Eq. (7.4) with the problem

$$\underset{\boldsymbol{\beta}\in\mathbb{R}^n}{\operatorname{argmin}} \ \|\boldsymbol{\beta}-\boldsymbol{\nu}\|_2^2 \quad \text{s.t.} \quad \|\boldsymbol{\beta}\|_1 \le z \ \text{ and } \ \boldsymbol{\beta} \ge 0 \ , \tag{7.5}$$

and construct a solution $\boldsymbol{\alpha}$ of Eq. (7.4) from a solution of Eq. (7.5) by letting $\alpha_i = \text{sign}(\mu_i)\,\beta_i$. A summary of the algorithm for solving Eq. (7.4) is given in Figure 7.2.

```
INPUT: A vector $\boldsymbol{\mu} \in \mathbb{R}^n$ and a scalar $z > 0$
DO:
    If $\|\boldsymbol{\mu}\|_1 \leq z$ set $\boldsymbol{\alpha} = \boldsymbol{\mu}$ and return
    Define $\boldsymbol{\nu} \in \mathbb{R}^n$ to be a vector such that for all $i$, $\nu_i = |\mu_i|$
    Call Figure 7.1 with $\boldsymbol{\nu}$ and $z$ and let $\boldsymbol{\beta}$ be the output of Figure 7.1
    Define $\boldsymbol{\alpha}$ to be the vector such that for all $i$, $\alpha_i = \text{sign}(\mu_i)\,\nu_i$.
OUTPUT: The vector $\boldsymbol{\alpha}$ is the solution of Eq. (7.4)
```

Figure 7.2: A procedure for projecting onto ℓ_1 balls.

7.1.3 Entropic projections onto the Probabilistic Simplex

In this section we describe an efficient solution to the following problem,

$$\underset{\mathbf{w}\in S}{\text{argmin}} \sum_{j=1}^{n} w_j \log\left(\frac{w_j}{u_j}\right) \quad \text{where} \quad S = \left\{ \mathbf{w} \mid \sum_{j=1}^{n} w_j = 1 \,,\, \forall j : \, w_j \geq \epsilon \right\} \quad . \tag{7.6}$$

If $\epsilon = 0$ the above problem is an Entropic projection onto the Probabilistic simplex.

We start by writing the Lagrangian of the above constrained optimization problem,

$$\mathcal{L} = \sum_{j=1}^{n} w_j \log(w_j/u_j) + \theta \left(\sum_{j=1}^{n} w_j - 1 \right) - \sum_{j=1}^{n} \beta_j (w_j - \epsilon) \quad ,$$

Here, θ is an unconstrained Lagrange multiplier and $\{\beta_j\}$ is a set of *non-negative* Lagrange multipliers for the inequality constraints, $w_j \geq \epsilon$. By taking the derivative of $\mathcal{L}$ with respect to u_i, we get that at the optimum the following should hold,

$$\log(w_i) - \log(u_i) + 1 + \theta - \beta_i = 0 \quad .$$

After rearranging terms, taking the exponent of the above equation, we can rewrite the above equation as follows,

$$w_i = u_i e^{\beta_i}/Z \quad ,$$

where $Z = e^{\theta+1}$. Since, θ is a Lagrange multiplier for the constraint $\sum_j u_j = 1$ we can also write Z as,

$$Z = \sum_{j=1}^{n} u_j e^{\beta_j} \quad .$$

From KKT conditions we know that at the saddle point $(\mathbf{w}^\star, \theta^\star, \{\beta_i^\star\})$ of the Lagrangian $\mathcal{L}$ the following holds,

$$\beta_i^\star(w_i^\star - \epsilon) = 0 \ .$$

Therefore, if the i'th coordinate of the optimal solution is strictly greater than ϵ we must have that $\beta_i^\star = 0$. In the case where $w_i^\star = \epsilon$ the Lagrange multiplier $\beta_i^\star$ is simply constrained to be non-negative, $\beta_i^\star \geq 0$. Thus, the optimal solution can be rewritten in the following distilled form,

$$w_i^\star = \begin{cases} u_i/Z & w_i^\star > \epsilon \\ \epsilon & \text{otherwise} \end{cases} \ , \tag{7.7}$$

where Z is set such that $\sum_i w_i^\star = 1$. The lingering question is what components of $\mathbf{u}^\star$ we need to set to ϵ. The following Lemma paves the way to an efficient procedure for determining the components of $\mathbf{w}^\star$ which are equal ϵ.

Lemma 9 *Let $\mathbf{w}^\star$ denote optimal solution of Eq. (7.6) and let i and j be two indices such that, (i) $u_i \leq u_j$ and (ii) $w_j^\star = \epsilon$, then $w_i^\star = \epsilon$.*

Proof Assume by contradiction that $w_i^\star > \epsilon$. We now use the explicit form of the optimal solution and write,

$$w_i^\star = u_i e^{\beta_i^\star}/Z \ \text{ and } \ w_j^\star = u_j e^{\beta_j^\star}/Z \ .$$

Since $w_j^\star = \epsilon$ we get that $\beta_j^\star \geq 0$ while the fact that $w_i^\star > \epsilon$ implies that $\beta_i^\star = 0$. (See also Eq. (7.7).) Combining these two facts we get that,

$$w_i^\star = u_i/Z \leq u_j/Z \leq u_j e^{\beta_j^\star}/Z = w_j^\star = \epsilon \ .$$

Thus, $w_i^\star \leq \epsilon$ which stands in contradiction to the assumption $w_i^\star > \epsilon$. ■

The above lemma implies that if we sort the elements of $\mathbf{u}$ in an ascending order, then there exists an index $l^\star$ such that the optimal solution $\mathbf{w}^\star$ takes the following form,

$$\mathbf{w}^\star = (\epsilon, \ldots, \epsilon, \frac{u_{l^\star+1}}{Z}, \frac{u_{l^\star+2}}{Z}, \ldots, \frac{u_n}{Z}) \ .$$

We are therefore left with the task of finding $l^\star$. We do so by examining all possible values for $l \in \{1, \ldots, n\}$. Given a candidate value for $l^\star$, denoted l, we need to check the feasibility of the solution induced by the assumption $l^\star = l$. We next calculate Z for this choice of l. Since the role of Z is to enforce the constraint $\sum_j w_j^\star = 1$ we get that,

$$Z = \frac{\sum_{j=l+1}^n u_j}{1 - l\epsilon} \ .$$

```
INPUT: Sorted vector $\mathbf{u} \in \mathbb{R}^n_+ (u_j \leq u_{j+1})$ ; A scalar $\epsilon > 0$
INITIALIZE: $Z = \sum_{i=1}^n u_i$
FOR $l = 1, \ldots, n$:
    If $u_l/Z \geq \epsilon$ Then
        $l^\star = l - 1$
        Break
    EndIf
    $Z = Z + \frac{\epsilon Z - u_l}{1 - l\epsilon}$
OUTPUT: $\mathbf{w}^\star$
    $w_j^\star = u_j/Z$  for  $j = l^\star + 1, \ldots, n$
    $w_j^\star = \epsilon$  for  $j = 1, \ldots, l^\star$
```

Figure 7.3: Pseudo-code for solving the Entropic projection problem given in Eq. (7.6).

If l indeed induces a feasible solution then for all $j > l$ we must have that $u_j/Z > \epsilon$. Since we assume that the vector $\mathbf{u}$ is sorted in an ascending order it is enough to verify that $u_{l+1}/Z > \epsilon$. In case that we find more than one single feasible solution, it is easy to verify that the smallest candidate for $l^\star$ should be taken. Last, note that the condition $u_{l+1}/Z > \epsilon$ can be efficiently calculated if we simply keep track of partial sums. Each partial sum is of the form $\sum_{j=l+1}^n u_j$ and is computed from its predecessor $\sum_{j=l}^n u_j$. The pseudo-code describing the entire algorithm is given in Figure 7.3.

7.2 Aggressive Updates for the Hinge-loss

In this section we derive analytical solutions for the aggressive update rule given in Eq. (3.12) when $g_t(\mathbf{w})$ is the hinge-loss. The hinge-loss is usually the primary choice for online learning of binary classifiers. The optimization problem in Eq. (3.12) is equivalent to the following problem:

$$\underset{\boldsymbol{\lambda}}{\text{argmax}} \; -cf^\star\left(-\tfrac{\boldsymbol{\theta}_t + \boldsymbol{\lambda}}{c}\right) - g_t^\star(\boldsymbol{\lambda}) \;\;, \tag{7.8}$$

where $\boldsymbol{\theta}_t = \sum_{i \neq t} \boldsymbol{\lambda}_i^t$.

The focus of this section is on the case where $g_t(\mathbf{w}) = [\gamma - \langle \mathbf{w}, \mathbf{x} \rangle]_+$, where $\gamma > 0$ and $\mathbf{x} \in \mathbb{R}^n$. In this case, $g_t^\star(\boldsymbol{\lambda}) = -\gamma\,\alpha$ if $\boldsymbol{\lambda} = -\alpha\mathbf{x}$ for some $\alpha \in [0,1]$ and otherwise $g^\star(\boldsymbol{\lambda}) = \infty$ (see Section A.3.1). Therefore, the optimization problem is equivalent to the following:

$$\underset{\alpha \in [0,1]}{\text{argmax}} \; \gamma\,\alpha - cf^\star\left(-\tfrac{\boldsymbol{\theta}_t - \alpha\,\mathbf{x}}{c}\right) \;\;. \tag{7.9}$$

The above problem is an optimization over a single variable over a small domain. Thus, it can be easily solved using line search methods (see for example [8] page 723). In the following, we derive analytical solutions for two important specific cases.

We start with the case in which $f(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|_2^2$. Thus, $f^\star(\boldsymbol{\theta}) = \frac{1}{2}\|\boldsymbol{\theta}\|_2^2$ and Eq. (7.9) becomes

$$\operatorname*{argmax}_{\alpha\in[0,1]} \gamma\,\alpha - \tfrac{1}{2\,c}\| - \boldsymbol{\theta}_t + \alpha\,\mathbf{x}\|_2^2 \;=\; \operatorname*{argmax}_{\alpha\in[0,1]} \alpha\left(\gamma - \tfrac{1}{c}\langle\boldsymbol{\theta}_t,\mathbf{x}\rangle\right) - \alpha^2\frac{\|\mathbf{x}\|_2^2}{2\,c} \quad . \tag{7.10}$$

Denote the objective function of the above by $Q(\alpha)$ and note that Q is a concave parabola whose maximum falls at the point

$$\alpha' \;=\; \frac{c\left(\gamma - \frac{1}{c}\langle\boldsymbol{\theta}_t,\mathbf{x}\rangle\right)}{\|\mathbf{x}\|_2^2} \quad .$$

The maximizer of Eq. (7.10) will be α' if $\alpha' \in [0,1]$. Otherwise, if $\alpha' < 0$ then the maximizer will be 0 and if $\alpha' > 1$ then the maximizer will be 1. In summary, the solution of Eq. (7.10) is:

$$\alpha \;=\; \min\left\{1\,,\,\frac{c\left[\gamma - \frac{1}{c}\langle\boldsymbol{\theta}_t,\mathbf{x}\rangle\right]_+}{\|\mathbf{x}\|_2^2}\right\} \quad . \tag{7.11}$$

Next, we turn to the case in which $f(\mathbf{w}) = \log(n) + \sum_{j=1}^n w_j\log(w_j)$ where S is the probabilistic simplex. To devise an analytic solution, we further assume that $\mathbf{x}$ is a vector in $\{-1,0,1\}^n$. While this assumption seems restrictive, many text processing applications use term appearances as features which distill to binary vectors. In this case the conjugate of f is $f^\star(\boldsymbol{\theta}) = \log\left(\sum_{j=1}^n \exp(\theta_j)\right) - \log(n)$. Omitting terms which do not depend on α, the objective of the optimization problem given in Eq. (7.9) becomes

$$\operatorname*{argmax}_{\alpha\in[0,1]} \gamma\,\alpha - c\,\log\left(\sum_{j=1}^n \exp(-\tfrac{\theta_{t,j}}{c})\,\exp(-\alpha\,\tfrac{x_j}{c})\right) \quad . \tag{7.12}$$

To find this optimal value we introduce the following variables,

$$\theta^+ \;=\; \sum_{j:x_j=1}\exp(-\tfrac{\theta_{t,j}}{c}) \quad , \quad \theta^- \;=\; \sum_{j:x_j=-1}\exp(-\tfrac{\theta_{t,j}}{c}) \quad , \quad \theta^0 \;=\; \sum_{j:x_j=0}\exp(-\tfrac{\theta_{t,j}}{c}) \quad .$$

We can thus rewrite Eq. (7.12) as

$$\operatorname*{argmax}_{\alpha\in[0,1]} \gamma\,\alpha - c\,\log\left(\theta^+\,\exp(-\tfrac{\alpha}{c}) + \theta^-\,\exp(\tfrac{\alpha}{c}) + \theta^0\right) \quad . \tag{7.13}$$

Taking the derivative of the above equation with respect to α and comparing the result to zero yields the following,

$$\gamma - \frac{\theta^- \exp(\frac{\alpha}{c}) - \theta^+ \exp(-\frac{\alpha}{c})}{\theta^+ \exp(-\frac{\alpha}{c}) + \theta^- \exp(\frac{\alpha}{c}) + \theta^0} = 0 \ . \tag{7.14}$$

Denote $\beta = \exp(\frac{\alpha}{c})$, the equation above is equivalent to the following equation in β,

$$\gamma \left(\theta^+ \beta^{-1} + \theta^- \beta + \theta^0\right) \ = \ \theta^- \beta - \theta^+ \beta^{-1}$$

Multiplying both sides of the above equation by β and rearranging terms, we obtain the following quadratic equation

$$(1-\gamma)\theta^- \beta^2 - \gamma\theta^0 \beta - (1+\gamma)\theta^+ \ = \ 0 \ ,$$

whose largest root is

$$\beta = \frac{\gamma\theta^0 + \sqrt{\gamma^2(\theta^0)^2 + 4(1-\gamma^2)\theta^+\theta^-}}{2(1-\gamma)\theta^-} \ . \tag{7.15}$$

Since α must reside in $[0,1]$ we set α to be the minimum between $c\,\log(\beta)$ and 1, yielding,

$$\alpha \ = \ \min\left\{1, c\,\log\left(\frac{\gamma\theta^0 + \sqrt{\gamma^2(\theta^0)^2 + 4(1-\gamma^2)\theta^+\theta^-}}{2(1-\gamma)\theta^-}\right)\right\} \ . \tag{7.16}$$

7.3 Aggressive Updates for Label Ranking

In this section we describe an efficient procedure for performing the aggressive update rule given in Eq. (7.8) when $g_t(\mathbf{w})$ is a loss function that is widely used in label ranking problems. A detailed formal description of the label ranking task is given in Section 8.2. Here, we only give a short definition of the loss function $g_t(\mathbf{w})$.

Let $\mathbf{x}$ be a vector in $\mathbb{R}^n$, let k be an integer and let Y be a subset of $[k]$. In label ranking, the parameter vector is grouped into k vectors in $\mathbb{R}^n$ and we denote it by $\bar{\mathbf{w}} = (\mathbf{w}_1, \ldots, \mathbf{w}_k)$ where $\mathbf{w}_i \in \mathbb{R}^n$ for each $i \in [k]$. The loss function $g_t(\bar{\mathbf{w}})$ takes the following form:

$$g_t(\bar{\mathbf{w}}) \ = \ \max_{r \in Y, s \notin Y} [\gamma - (\langle \mathbf{w}_r, \mathbf{x}\rangle - \langle \mathbf{w}_s, \mathbf{x}\rangle)]_+ \ , \tag{7.17}$$

where $\gamma > 0$ is a scalar and $[a]_+ = \max\{a, 0\}$.

We now derive efficient procedures for performing the aggressive update rule given in Eq. (7.8) when g_t takes the form given in Eq. (7.17). For simplicity of our notation, from now on we omit the index t. Let $\boldsymbol{\sigma} \in \{\pm 1\}^k$ be a vector such that $\sigma_i = 1$ for $i \in Y$ and otherwise $\sigma_i = -1$. Define:

$$A = \{\boldsymbol{\alpha} \in \mathbb{R}^k_+ \ : \ \|\boldsymbol{\alpha}\|_1 \le 1 \ \text{ and } \ \sum_{i=1}^k \sigma_i \alpha_i = 0\} \ ,$$

and let $\bar{\boldsymbol{\lambda}} = (\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_k)$. In Lemma 20 we show that $g^\star(\bar{\boldsymbol{\lambda}})$ is ∞ unless there exists a vector $\boldsymbol{\alpha} \in A$ such that $\forall i,\ \boldsymbol{\lambda}_i = -\sigma_i \alpha_i \mathbf{x}$. Let us also assume that the complexity function is decomposable, i.e., $\bar{f}(\bar{\mathbf{w}}) = \sum_i f(\mathbf{w}_i)$. Therefore, Eq. (7.8) can be rewritten as

$$\underset{\boldsymbol{\alpha}\in A}{\operatorname{argmax}}\ \tfrac{\gamma}{2} \sum_{r\in Y} \alpha_r - c \sum_{i=1}^{k} f^\star\left(-\tfrac{\boldsymbol{\theta}_i - \sigma_i\, \alpha_i \mathbf{x}}{c}\right) \ . \tag{7.18}$$

In the following, we describe efficient procedures for solving Eq. (7.18). We start by deriving a procedure for the case in which $f(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|_2^2$. In this case, we can find the exact solution of Eq. (7.18) by extending ideas from Section 7.1. The complexity of the resulting procedure is $O(k \log(k))$. Next, we derive an efficient interior point method for the general case. The interior point procedure reaches the optimal solution within accuracy ϵ in time $O(k^{3/2} \log(\log(\frac{1}{\epsilon})))$.

7.3.1 Exact solution for the Euclidean case:

When $f(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|_2^2$, the optimization problem in Eq. (7.18) becomes

$$\underset{\boldsymbol{\alpha}\in A}{\operatorname{argmax}}\ \tfrac{\gamma}{2} \sum_{r\in Y} \alpha_r - \frac{1}{2\,c} \sum_{i=1}^{k} \|\boldsymbol{\theta}_i - \sigma_i\, \alpha_i \mathbf{x}\|^2 \ . \tag{7.19}$$

To make our notation easy to follow, we define $p = |Y|$ and $q = k - p$ and construct two vectors $\boldsymbol{\mu} \in \mathbb{R}^p$ and $\boldsymbol{\nu} \in \mathbb{R}^q$ such that for each $i \in Y$ there is an element $(\frac{c\gamma}{2} + \langle \boldsymbol{\theta}_i, \mathbf{x}\rangle)/\|\mathbf{x}\|^2$ in $\boldsymbol{\mu}$ and for each $i \notin Y$ there is an element $-\langle \boldsymbol{\theta}_i, \mathbf{x}\rangle/\|\mathbf{x}\|^2$ in $\boldsymbol{\nu}$. The problem in Eq. (7.19) can now be rewritten as,

$$\begin{aligned} &\underset{\boldsymbol{\alpha}\in\mathbb{R}^p_+,\boldsymbol{\beta}\in\mathbb{R}^q_+}{\operatorname{argmin}} && \frac{1}{2}\|\boldsymbol{\alpha} - \boldsymbol{\mu}\|^2 + \frac{1}{2}\|\boldsymbol{\beta} - \boldsymbol{\nu}\|^2 \\ &\text{s.t.} && \sum_{i=1}^{p} \alpha_i = \sum_{j=1}^{q} \beta_j \le 1 \ . \end{aligned} \tag{7.20}$$

Note that the variables $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are tied together through a single equality constraint $\|\boldsymbol{\alpha}\|_1 = \|\boldsymbol{\beta}\|_1$. We represent this coupling of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ by rewriting the optimization problem in Eq. (7.20) as,

$$\min_{z\in[0,1]}\ g(z;\boldsymbol{\mu}) + g(z;\boldsymbol{\nu}) \ ,$$

where

$$g(z;\boldsymbol{\mu}) = \min_{\boldsymbol{\alpha}} \frac{1}{2}\|\boldsymbol{\alpha} - \boldsymbol{\mu}\|^2 \quad \text{s.t.} \quad \sum_{i=1}^{p} \alpha_i = z \,,\ \ \alpha_i \ge 0 \ , \tag{7.21}$$

and similarly

$$g(z;\boldsymbol{\nu}) \;=\; \min_{\boldsymbol{\beta}} \frac{1}{2}\|\boldsymbol{\beta}-\boldsymbol{\nu}\|^2 \quad \text{s.t.} \quad \sum_{j=1}^{q}\beta_j \;=\; z \;,\;\; \beta_j \geq 0 \;. \tag{7.22}$$

The function $g(z;\cdot)$ takes the same functional form whether we use $\boldsymbol{\mu}$ or $\boldsymbol{\nu}$ as the second argument. We therefore describe our derivation in terms of $g(z;\boldsymbol{\mu})$. Clearly, the same derivation is also applicable to $g(z;\boldsymbol{\nu})$.

In Section 7.1 we have shown that the solution to the minimization problem on the right-hand side of Eq. (7.21) is

$$\alpha_i \;=\; \begin{cases} \mu_i - \dfrac{1}{\rho(z,\boldsymbol{\mu})}\left(\displaystyle\sum_{i=1}^{\rho}\mu_i - z\right) & i \leq \rho(z,\boldsymbol{\mu}) \\ 0 & i > \rho(z,\boldsymbol{\mu}) \end{cases} \;, \tag{7.23}$$

where

$$\rho(z,\boldsymbol{\mu}) \;=\; \max\left\{ j \in [p] \;:\; \mu_j - \frac{1}{j}\left(\sum_{r=1}^{j}\mu_r - z\right) > 0 \right\} \;.$$

Had we known the optimal value of z, i.e. the argument attaining the minimum of $g(z;\boldsymbol{\mu}) + g(z;\boldsymbol{\nu})$ we could have calculated the optimal variables $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ by first finding $\rho(z,\boldsymbol{\mu})$ and $\rho(z,\boldsymbol{\nu})$ and then finding $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ using Eq. (7.23). This is a classic chicken-and-egg problem: we can easily calculate the optimal solution given some side information; however, obtaining the side information seems as difficult as finding the optimal solution. One option is to perform a search over an ϵ-net of values for z in $[0,1]$. For each candidate value for z from the ϵ-net we can find $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ and then choose the value which attains the lowest objective value $(g(z;\boldsymbol{\mu}) + g(z;\boldsymbol{\nu}))$. While this approach may be viable in many cases, it is still quite time consuming. We are rescued by the fact that $g(z;\boldsymbol{\mu})$ and $g(z;\boldsymbol{\nu})$ entertain a very special structure. Rather than enumerating over all possible values of z we need to check at most $k+1$ possible values for z. To establish the last part of our efficient algorithm which performs this search for the optimal value of z we need the following lemma. The lemma is stated with $\boldsymbol{\mu}$ but, clearly, it also holds for $\boldsymbol{\nu}$.

Lemma 10 *Let $g(z;\boldsymbol{\mu})$ be as defined in Eq. (7.21). For each $i \in [p]$, define*

$$z_i = \sum_{r=1}^{i}\mu_r - i\mu_i \;\;.$$

Then, for each $z \in [z_i, z_{i+1}]$ the function $g(z; \boldsymbol{\mu})$ is equivalent to the following quadratic function,

$$g(z; \boldsymbol{\mu}) \;=\; \frac{1}{i}\left(\sum_{r=1}^{i} \mu_r - z\right)^2 + \sum_{r=i+1}^{p} \mu_r^2 \ .$$

Moreover, g is continuous, continuously differentiable, and convex in $[0, 1]$.

The proof of this lemma is given in Section A.5. The good news that the theorem carries is that $g(z; \boldsymbol{\mu})$ and $g(z; \boldsymbol{\nu})$ are convex and therefore their sum is also convex. Furthermore, the function $g(z; \cdot)$ is piecewise quadratic and the points where it changes from one quadratic function to another are simple to compute. We refer to these points as knots. We next exploit the properties of the function g to devise an efficient procedure for finding the optimal value of z and from there the road to the optimal dual variables is clear and simple.

Due to the strict convexity of $g(z; \boldsymbol{\mu}) + g(z; \boldsymbol{\nu})$ its minimum is unique and well defined. Therefore, it suffices to search for a seemingly *local* minimum over all the sub-intervals in which the objective function is equivalent to a quadratic function. If such a local minimum point is found it is guaranteed to be the global minimum. Once we have the value of z which constitutes the global minimum we can decouple the optimization problems for $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ and quickly find the optimal solution. However, there is one last small obstacle: the objective function is the sum of two piecewise quadratic functions. We therefore need to efficiently go over the *union* of the knots derived from $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$. We now summarize the full algorithm for finding the optimum of the dual variables and wrap up with its pseudo-code.

Given $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$ we find the sets of knots for each vector, take the union of the two sets, and sort the set in an ascending order. Based on the theorems above, it follows immediately that each interval between two consecutive knots in the union is also quadratic. Since $g(z; \boldsymbol{\mu}) + g(z; \boldsymbol{\nu})$ is convex, the objective function in each interval can be characterized as falling into one of two cases. Namely, the objective function is either monotone (increasing or decreasing) or it attains its unique global minimum inside the interval. In the latter case the objective function clearly decreases, reaches the optimum where its derivative is zero, and then increases. If the objective function is monotone in all of the intervals then the minimum is obtained at one of the boundary points $z = 0$ or $z = 1$. Otherwise, we simply need to identify the interval bracketing the global minimum and then find the optimal value of z by finding the minimizer of the quadratic function associated with the interval.

We utilize the following notation. For each $i \in [p]$, define the knots derived from $\boldsymbol{\mu}$

$$z_i = \sum_{r=1}^{i} \mu_r - i\mu_i \ \ ,$$

```
INPUT:    instance $\mathbf{x} \in \mathcal{X}$ ; set $Y \subset [k]$
          current vectors $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_k$ ; regularization parameter $c$
MARGINS:
          $\boldsymbol{\mu} = \text{sort}\left\{(\frac{c\gamma}{2} + \langle \boldsymbol{\theta}_r, \mathbf{x} \rangle)/\|\mathbf{x}\|^2 \mid r \in Y\right\}$
          $\boldsymbol{\nu} = \text{sort}\left\{-\langle \boldsymbol{\theta}_r, \mathbf{x} \rangle / \|\mathbf{x}\|^2 \mid r \notin Y\right\}$
KNOTS:
          $\forall i \in [p] \,:\, z_i = \sum_{r=1}^{i} \mu_r - i\mu_i \qquad \forall j \in [q] \,:\, \tilde{z}_j = \sum_{s=1}^{j} \nu_s - j\nu_j$
          $\mathcal{Q} = \{z_i : z_i < 1\} \cup \{\tilde{z}_j : \tilde{z}_j < 1\} \cup \{1\}$
INTERVALS:
          $\forall z \in \mathcal{Q} : \quad R(z) = |\{z_i : z_i \le z\}| \quad ; \quad S(z) = |\{\tilde{z}_j : \tilde{z}_j \le z\}|$
          $\forall z \in \mathcal{Q} : \quad N(z) = \min\{z' \in \mathcal{Q} : z' > z\} \cup \{1\}$
LOCAL MIN:
          $O(z) \; = \; \left( S(z) \sum_{r=1}^{R(z)} \mu_r + R(z) \sum_{r=1}^{S(z)} \nu_r \right) / \left(R(z) + S(z)\right)$
GLOBAL MIN:
          If $(\exists z \in \mathcal{Q} \;\; \text{s.t.} \;\; O(z) \in [z, N(z)])$ Then
             $z^\star = O(z) \;\; ; \;\; i^\star = R(z) \;\; ; \;\; j^\star = S(z)$
          Else If $(\mu_1 + \nu_1 \le 0)$
             $z^\star = 0 \;\; ; \;\; i^\star = 1 \;\; ; \;\; j^\star = 1$
          Else
             $z^\star = 1 \;\; ; \;\; i^\star = R(1) \;\; ; \;\; j^\star = S(1)$
DUAL'S AUXILIARIES:
          $\zeta_1 = \frac{1}{i^\star}\left(\sum_{r=1}^{i^\star} \mu_r - z^\star\right) \;\; ; \;\; \zeta_2 = \frac{1}{j^\star}\left(\sum_{r=1}^{j^\star} \nu_r - z^\star\right)$
OUTPUT:
          $\forall r \in Y \,:\, \alpha_r \; = \; \left[(\frac{c\gamma}{2} + \langle \boldsymbol{\theta}_r, \mathbf{x} \rangle)/\|\mathbf{x}\|^2 - \zeta_1\right]_+$
          $\forall r \notin Y \,:\, \alpha_r \; = \; \left[-\langle \boldsymbol{\theta}_r, \mathbf{x} \rangle / \|\mathbf{x}\|^2 - \zeta_2\right]_+$
```

Figure 7.4: Pseudo-code for solving Eq. (7.18) when $f(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2$.

and similarly, for each $j \in [q]$ define

$$\tilde{z}_j = \sum_{r=1}^{j} \nu_r - j\nu_j \ .$$

From Lemma 10 we know that $g(z;\boldsymbol{\mu})$ is quadratic in each segment $[z_i, z_{i+1})$ and $g(z;\boldsymbol{\nu})$ is quadratic in each segment $[\tilde{z}_j, \tilde{z}_{j+1})$. Therefore, as already argued above, the function $g(z;\boldsymbol{\mu}) + g(z;\boldsymbol{\nu})$ is also piecewise quadratic in $[0,1]$ and its knots are the points in the set,

$$\mathcal{Q} = \{z_i : z_i < 1\} \cup \{\tilde{z}_j : \tilde{z}_j < 1\} \cup \{1\} \ .$$

For each knot $z \in \mathcal{Q}$, we denote by $N(z)$ its consecutive knot in $\mathcal{Q}$, that is,

$$N(z) = \min \ \big(\{z' \in \mathcal{Q} : z' > z\} \cup \{1\}\big) \ .$$

We also need to know for each knot how many knots precede it. Given a knot z we define

$$R(z) = |\{z_i : z_i \le z\}| \ \text{ and } \ S(z) = |\{\tilde{z}_i : \tilde{z}_i \le z\}| \ .$$

Using the newly introduced notation we can find for a given value z its bracketing interval, $z \in [z', N(z')]$. From Lemma 10 we get that the value of the dual objective function at z is,

$$g(z;\boldsymbol{\mu}) + g(z;\boldsymbol{\nu}) \ = \ \frac{1}{R(z')}\left(\sum_{r=1}^{R(z')} \mu_r - z\right)^2 + \sum_{r=R(z')+1}^{p} \mu_r^2 + \frac{1}{S(z')}\left(\sum_{r=1}^{S(z')} \nu_r - z\right)^2 + \sum_{r=S(z')+1}^{p} \nu_r^2 \ .$$

The unique minimum of the quadratic function above is attained at the point

$$O(z') = \left(S(z')\sum_{i=1}^{R(z')} \mu_i + R(z')\sum_{i=1}^{S(z')} \nu_i\right) / \big(R(z') + S(z')\big) \ .$$

Therefore, if $O(z') \in [z', N(z')]$, then the global minimum of the dual objective function is attained at $O(z')$. Otherwise, if no such interval exists, the optimum is either at $z = 0$ or at $z = 1$. The minimum is achieved at $z = 0$ iff the derivative of the objective function at $z = 0$ is non-negative, namely, $-\mu_1 - \nu_1 \ge 0$. In this case, the optimal solution is $\boldsymbol{\alpha} = \mathbf{0}$ and $\boldsymbol{\beta} = \mathbf{0}$ which implies that $\boldsymbol{\lambda}_r = \mathbf{0}$ for all r. If on the other hand $-\mu_1 - \nu_1 < 0$ then the optimum is attained at $z = 1$. The skeleton of the pseudo-code for the fast projection algorithm is given in Figure 7.4. The most expensive operation performed by the algorithm is the sorting of $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$. Since the sum of the dimensions of these vectors is k the time complexity of the algorithm is $\Theta(k\log(k))$.

7.3.2 Interior point method for the general case:

We now present an efficient primal-dual interior point algorithm (PDIP) for solving the optimization problem given in Eq. (7.18), which is applicable to a larger family of complexity functions. We describe the PDIP algorithm for a slightly more general optimization problem which still exploits the structure of the problem and leads to a very efficient PDIP algorithm. Let $\{f_r | f_r : \mathbb{R} \to \mathbb{R}\}_{r=1}^d$ be a set of d twice differentiable functions and denote by $\{f_r'\}$ and $\{f_r''\}$ their first and second derivatives. Let $\mathbf{p}$ and $\mathbf{q}$ be two vectors in $\mathbb{R}^d$, A be a $2 \times d$ matrix, and $\mathbf{b}$ a two dimensional vector over $\mathbb{R}$. Instead of the original problem defined by Eq. (7.18), we work with the following minimization problem,

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^d} \sum_{r=1}^d f_r(\alpha_r) \text{ s.t. } A\boldsymbol{\alpha} = \mathbf{b},\ \forall r\ p_r\alpha_r \le q_r \quad . \tag{7.24}$$

It is easy to verify that the problem defined by Eq. (7.18) can be reformatted and described as an instance of the problem defined by Eq. (7.24).

To motivate the derivation of the PDIP algorithm, let us first note that the dual of Eq. (7.24) is the problem $\max_{\boldsymbol{\lambda}\in\mathbb{R}_+^d, \boldsymbol{\nu}\in\mathbb{R}^2} \mathcal{D}(\boldsymbol{\lambda},\boldsymbol{\nu})$ where $\mathcal{D}(\boldsymbol{\lambda},\boldsymbol{\nu})$ is

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^d} \sum_{r=1}^d f_r(\alpha_r) + \sum_{r=1}^d \lambda_r(p_r\alpha_r - q_r) + \langle \boldsymbol{\nu}, (A\boldsymbol{\alpha} - \mathbf{b})\rangle \ . \tag{7.25}$$

Denote by $P(\boldsymbol{\alpha})$ the objective function of the problem in Eq. (7.24). As the name implies, the PDIP algorithm maintains strictly feasible primal ($\boldsymbol{\alpha}$) and dual ($\boldsymbol{\lambda},\boldsymbol{\nu}$) solutions at all times. (we remind the reader that a strictly feasible solution of a given problem satisfies all the constraints of the problem, where each inequality constraint holds with strict inequality.) Assume that we have at hand a strictly feasible primal-dual solution $(\boldsymbol{\alpha},\boldsymbol{\lambda},\boldsymbol{\nu})$. We now define the following function, $\eta(\boldsymbol{\alpha},\boldsymbol{\lambda}) = \sum_{r=1}^d \lambda_r(q_r - p_r\alpha_r)$. We next show that $\eta(\boldsymbol{\alpha},\boldsymbol{\lambda})$ is a lower bound on the duality gap of our primal-dual solution. The definition of $\mathcal{D}(\boldsymbol{\lambda},\boldsymbol{\nu})$ implies that,

$$\begin{aligned} \mathcal{D}(\boldsymbol{\lambda},\boldsymbol{\nu}) &\le \sum_{r=1}^d (f_r(\alpha_r) + \lambda_r(p_r\alpha_r - q_r)) + \langle \boldsymbol{\nu}, A\boldsymbol{\alpha} - \mathbf{b}\rangle \\ &= \quad P(\boldsymbol{\alpha}) + \eta(\boldsymbol{\alpha},\boldsymbol{\lambda}) \ , \end{aligned} \tag{7.26}$$

where the second equality is due to the fact that $\boldsymbol{\alpha}$ is a feasible dual solution, thus $A\boldsymbol{\alpha} = \mathbf{b}$. Therefore, the duality gap is bounded below by

$$P(\boldsymbol{\alpha}) - \mathcal{D}(\boldsymbol{\lambda}, \boldsymbol{\nu}) \;\geq\; \eta(\boldsymbol{\alpha}, \boldsymbol{\lambda}) \ . \tag{7.27}$$

Moreover, if

$$\forall r \in [d],\ f'_r(\alpha_r) + \lambda_r p_r + \nu_1 A_{1,r} + \nu_2 A_{2,r} \;=\; 0 \ , \tag{7.28}$$

then $\boldsymbol{\alpha}$ attains the minimum of Eq. (7.25). Therefore, both Eq. (7.26) and Eq. (7.27) hold with equality. In this case, $\eta(\boldsymbol{\alpha}, \boldsymbol{\lambda})$ amounts to be the duality gap of the primal-dual solution $(\boldsymbol{\alpha}, \boldsymbol{\lambda}, \boldsymbol{\nu})$.

The PDIP algorithm is an iterative procedure where on each iteration it finds a new strictly feasible primal-dual solution. The primary goal of the update is to decrease the duality gap. To do so, we use the fact that Eq. (7.27) establishes $\eta(\boldsymbol{\alpha}, \boldsymbol{\lambda})$ as a lower bound on the duality gap. Thus, the main goal of the update is to decrease $\eta(\boldsymbol{\alpha}, \boldsymbol{\lambda})$ on each iteration while ensuring that the actual duality gap, $P(\boldsymbol{\alpha}) - \mathcal{D}(\boldsymbol{\lambda}, \boldsymbol{\nu})$, stays close to $\eta(\boldsymbol{\alpha}, \boldsymbol{\lambda})$ as much as possible. Additionally, we need to make sure that the new primal-dual solution is also strictly feasible. We are now ready to describe the core update of the PDIP algorithm. Let us denote by $(\boldsymbol{\alpha}, \boldsymbol{\lambda}, \boldsymbol{\nu})$ the current primal-dual solution of the algorithm. The new primal-dual solution is obtained from the current solution by finding a step-size parameter, $s \in (0, 1)$ for a triplet $(\Delta\boldsymbol{\alpha}, \Delta\boldsymbol{\lambda}, \Delta\boldsymbol{\nu})$ and the update itself takes the form $\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} + s\Delta\boldsymbol{\alpha}$, $\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda} + s\Delta\boldsymbol{\lambda}$, and $\boldsymbol{\nu} \leftarrow \boldsymbol{\nu} + s\Delta\boldsymbol{\nu}$. To compute the triplet $(\Delta\boldsymbol{\alpha}, \Delta\boldsymbol{\lambda}, \Delta\boldsymbol{\nu})$ we linearize each summand of $\eta(\boldsymbol{\alpha} + \Delta\boldsymbol{\alpha}, \boldsymbol{\lambda} + \Delta\boldsymbol{\lambda})$ using a first order Taylor approximation and get,

$$\begin{aligned}(\lambda_r + \Delta\lambda_r)\,(q_r - p_r(\alpha_r + \Delta\alpha_r)) \;\approx&\\ &(\lambda_r + \Delta\lambda_r)(q_r - p_r\alpha_r) - \lambda_r p_r\,\Delta\alpha_r \ .\end{aligned}$$

We require that the value of η for the new solution is approximately a fraction of the value at the current solution. This is achieved by solving the following set of linear equalities in $\Delta\alpha_r$ and $\Delta\lambda_r$,

$$\forall r \in [d],\ (\lambda_r + \Delta\lambda_r)(q_r - p_r\alpha_r) - \lambda_r p_r\,\Delta\alpha_r = 0.1\,\tfrac{\eta(\boldsymbol{\alpha},\boldsymbol{\lambda})}{d} \ . \tag{7.29}$$

The choice of the contraction constant 0.1 was set empirically. Assuming that the above set of equations hold, then $\eta(\boldsymbol{\alpha} + \Delta\boldsymbol{\alpha}, \boldsymbol{\lambda} + \Delta\boldsymbol{\lambda}) \approx 0.1\,\eta(\boldsymbol{\alpha}, \boldsymbol{\lambda})$. To recap, solving the system of linear equations given by Eq. (7.29) serves as a proxy for achieving a substantial decrease in η. Next, we need to make sure that η at the new parameters provides a rather tight lower bound. We do so by making sure that the linearization of the left hand side of Eq. (7.28) is approximately zero by casting the following set of linear equations, to Eq. (7.28),

$$\begin{aligned}\forall r \in [d], \;\; & f_r'(\alpha_r) + f_r''(\alpha_r)\Delta\alpha_r + (\lambda_r + \Delta\lambda_r)p_r + \\ & (\nu_1 + \Delta\nu_1)A_{1,r} + (\nu_2 + \Delta\nu_2)A_{2,r} \;=\; 0 \;.\end{aligned} \tag{7.30}$$

Solving Eq. (7.30) helps us tighten the lower bound on the duality gap given in Eq. (7.26). Last, we need to make sure that the new set of parameters is indeed a feasible primal solution by requiring the equality $A(\boldsymbol{\alpha} + \Delta\boldsymbol{\alpha}) = \mathbf{b}$ to hold. The triplet $(\Delta\boldsymbol{\alpha}, \Delta\boldsymbol{\lambda}, \Delta\boldsymbol{\nu})$ is thus found by finding the solution of all the sets of linear equations described above.

What remains is to describe the choice of the step size s. We find s by using a backtracking search while making sure that $\boldsymbol{\lambda}$ and $\boldsymbol{\alpha}$ are strictly feasible. Since the set of linear equations is overly constrained we cannot guarantee that Eq. (7.28) holds with equality. Instead, we choose the step size s so that the overall approximation errors for the equations defined by Eq. (7.28) decreases with respect to the previous primal-dual solution. For more details on the backtracking procedure see for instance pp. 612-613 in [14].

There is both theoretical and empirical evidence that a PDIP algorithm reaches the optimal solution (within accuracy ϵ) after $O(\sqrt{d}\,\log(\log(\frac{1}{\epsilon})))$ iterations [14, 48, 90]. On each iteration we need to solve a set of $2d + 2$ linear equations. A direct implementation would require $O(d^3)$ operations for each iteration of the PDIP algorithm. However, as we now show, we can utilize the structure of the problem to solve the set of linear equations in linear time. Thus, the complexity of update III is $O(d\sqrt{d}) = O(k\sqrt{k})$. To obtain an efficient solver, we first eliminate the variables $\Delta\boldsymbol{\lambda}$ by rewriting Eq. (7.29) as

$$\forall r, \; (\lambda_r + \Delta\lambda_r) \;=\; \frac{\lambda_r p_r}{q_r - p_r\alpha_r}\,\Delta\alpha_r + \frac{0.1\eta(\boldsymbol{\alpha},\boldsymbol{\lambda})}{d(q_r - p_r\alpha_r)} \;\;, \tag{7.31}$$

and substituting the above into Eq. (7.30). We now define $u_r = -f_r'(\alpha_r) - \frac{0.1\eta(\boldsymbol{\alpha},\boldsymbol{\lambda})}{d(q_r - p_r\alpha_r)} - \nu_1 A_{1,r} - \nu_2 A_{2,r}$, and $z_r = f_r''(\alpha_r) + \lambda_r p_r/(q_r - p_r\alpha_r)$, and rewrite Eq. (7.30)

$$\forall r \in [d], \; z_r\,\Delta\alpha_r = u_r + A_{1,r}\Delta\nu_1 + A_{2,r}\Delta\nu_2 \;\;. \tag{7.32}$$

Finally, we rewrite the set of two equalities $A(\boldsymbol{\alpha} + \Delta\boldsymbol{\alpha}) = \mathbf{b}$ as $A\Delta\boldsymbol{\alpha} = \mathbf{0}$. Substituing $\Delta\alpha_r$ with the right hand side of Eq. (7.32) in the linear set of equalities $A\Delta\boldsymbol{\alpha} = \mathbf{0}$, we obtain a system of 2 linear equations in 2 variables which can be solved in constant time. From the solution we obtain $\Delta\boldsymbol{\nu}$ and then compute $\Delta\boldsymbol{\alpha}$ as described in Eq. (7.32). From $\Delta\boldsymbol{\alpha}$ we now compute $\Delta\boldsymbol{\lambda}$ using Eq. (7.31). The overall complexity of the procedure of assigning new values to the primal-dual feasible solution is thus $O(d)$.

7.4 Aggressive Updates for the Max-of-Hinge loss function

In this section we describe an efficient procedure for performing the aggressive update rule given in Eq. (7.8) when $g_t(\mathbf{w})$ is the max-of-hinge loss function. The max-of-hinge loss function is widely used in complex prediction problems such as multiclass categorization and instance ranking (see Section 5.3). Formally, let $(\mathbf{a}_1, b_1), \ldots, (\mathbf{a}_k, b_k)$, be a sequence of pairs such that for all $i \in [k]$, $(\mathbf{a}_i, b_i) \in \mathbb{R}^n \times \mathbb{R}_+$. We assume that $g_t(\mathbf{w})$ takes the form:

$$g_t(\mathbf{w}) \;=\; \max_{i \in [k]} \, [b_i - \langle \mathbf{w}, \mathbf{a}_i \rangle]_+ \quad . \tag{7.33}$$

The Fenchel conjugate of g_t is (see Section A.3.1)

$$g_t^\star(\boldsymbol{\lambda}) \;=\; \begin{cases} -\sum_{i=1}^k \alpha_i b_i & \text{if} \;\; \boldsymbol{\lambda} \in \left\{ -\sum_{i=1}^k \alpha_i \mathbf{a}_i \;:\; \boldsymbol{\alpha} \in \mathbb{R}_+^k \text{ and } \|\boldsymbol{\alpha}\|_1 \le 1 \right\} \\ \infty & \text{otherwise} \end{cases}$$

Therefore, Eq. (7.8) becomes

$$\operatorname*{argmax}_{\boldsymbol{\alpha} \in \mathbb{R}_+^k : \|\boldsymbol{\alpha}\|_1 \le 1} \; \sum_{i=1}^k b_i \, \alpha_i - c f^\star \left(-\tfrac{\boldsymbol{\theta}_t - \sum_i \alpha_i \, \mathbf{a}_i}{c} \right) \quad . \tag{7.34}$$

If k is not very large, the above optimization problem can be solved using a generic numerical solver based on, for instance, interior point methods. Alternatively, one can solve the above using a gradient descent procedure with projections onto the set $\{\boldsymbol{\alpha} \in \mathbb{R}_+^k : \|\boldsymbol{\alpha}\|_1 \le 1\}$. The projections can be performed using the ideas presented in Section 7.1.

In the following, we describe a simple and efficient iterative procedure for solving Eq. (7.34) based on the logarithmic regret algorithmic framework described in Chapter 4. To do so, we first note that Eq. (7.8) is the dual problem of the following primal minimization problem:

$$\operatorname*{argmin}_{\mathbf{w} \in S} \; c \, (f(\mathbf{w}) - \langle \boldsymbol{\theta}_t, \mathbf{w} \rangle) + g_t(\mathbf{w}) \quad . \tag{7.35}$$

The above can be proven based on Fenchel duality (see Section 3.2). Since the intersection of the domains of f and g_t is non-empty, strong duality holds (see for example [11]). Moreover, if $\mathbf{w}^\star$ is the optimal solution of Eq. (7.35) then an optimal solution of Eq. (7.8) satisfies the relation $\boldsymbol{\theta}_t + \boldsymbol{\lambda}^* = c \nabla f(\mathbf{w}^\star)$. Therefore, we can focus on the solution of Eq. (7.35).

For simplicity, we focus on the case $f(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|_2^2$ and $S = \mathbb{R}^n$. A generalization to other complexity functions can be done based on the ideas presented in Chapter 4. We first write $\mathbf{z} =$

```
PARAMETERS: Scalars $\gamma_1,\ldots,\gamma_k$ ; Vectors $\mathbf{a}_1,\ldots,\mathbf{a}_k$ ; Number of iterations $T$ ; Scalar $c$
INITIALIZE: $\mathbf{z}_1 = \mathbf{0}$ ; $U = \sqrt{\frac{2}{c}\max_i [\gamma_i]_+}$
FOR $t = 1,2,\ldots,T$
    Set $i = \arg\max_j(\gamma_j - \langle \mathbf{z}_t, \mathbf{a}_j\rangle)$
    If $\gamma_i > \langle \mathbf{z}_t, \mathbf{a}_i\rangle$
      Set $\mathbf{v}_t = c\,\mathbf{z}_t - \mathbf{a}_i$
    Else
      Set $\mathbf{v}_t = c\,\mathbf{z}_t$
    Set $\mathbf{z}'_t = \mathbf{z}_t - \frac{1}{c\,t}\,\mathbf{v}_t$
    Set $\mathbf{z}_{t+1} = \min\{1\,,\,\frac{U}{\|\mathbf{z}'_t\|}\}\,z'_t$
OUTPUT: $\frac{1}{T}\sum_t \mathbf{z}_t$
```

Figure 7.5: An iterative procedure for solving Eq. (7.36).

$\mathbf{w} - \boldsymbol{\theta}_t$ and rewrite Eq. (7.35) as

$$\boldsymbol{\theta}_t \;+\; \underset{\mathbf{z}\in\mathbb{R}^n}{\operatorname{argmin}} \left(\frac{c}{2}\|\mathbf{z}\|_2^2 + \max_{i\in[k]}\,[\gamma_i - \langle \mathbf{a}_i, \mathbf{z}\rangle]_+\right)$$

where $\gamma_i = b_i - \langle \mathbf{a}_i, \boldsymbol{\theta}_t\rangle$ for all $i \in [k]$. Denote $\phi(\mathbf{z}) = \frac{c}{2}\|\mathbf{z}\|_2^2 + \max_i\,[\gamma_i - \langle \mathbf{a}_i, \mathbf{z}\rangle]_+$. An iterative procedure for minimizing $\phi(\mathbf{z})$ is given in Figure 7.5.

To analyze the convergence properties of the procedure we make use of Theorem 5 from Chapter 4. First note that $\frac{1}{c}\phi$ is strongly convex with respect to f, while f is strongly convex with respect to the Euclidean norm. Second, we have

$$\phi(\mathbf{z}^*) \;\le\; \phi(0) = \max_i\,[\gamma_i]_+ \quad\Rightarrow\quad \|\mathbf{z}^*\|_2 \;\le\; \sqrt{\frac{2}{c}\,\max_i\,[\gamma_i]_+}\;.$$

Therefore, the problem of minimizing $\phi(\mathbf{z})$ over $\mathbb{R}^n$ is equivalent to the problem

$$\underset{\mathbf{z}}{\operatorname{argmin}}\;\; \frac{c}{2}\|\mathbf{z}\|_2^2 + \max_i\,[\gamma_i - \langle \mathbf{a}_i, \mathbf{z}\rangle]_+ \quad \text{s.t.} \quad \|\mathbf{z}\|_2 \le \sqrt{\frac{2}{c}\,\max_i\,[\gamma_i]_+} \tag{7.36}$$

This fact also implies that the norm of $\|\mathbf{v}_t\|$ given in Figure 7.5 is at most $\sqrt{2\,c\,\max_i\,[\gamma_i]_+} + \max_i \|\mathbf{a}_i\|$. Applying Theorem 5 and using the convexity of ϕ we obtain that

$$\phi\left(\frac{1}{T}\sum_t \mathbf{z}_t\right) \;\le\; \frac{1}{T}\sum_t \phi(\mathbf{z}_t) \;\le\; \phi(\mathbf{z}^*) + \frac{\left(\sqrt{2\,c\,\max_i\,[\gamma_i]_+} + \max_i\|\mathbf{a}_i\|\right)^2\,(1+\log(T))}{2\,c\,T}\;.$$

We have thus shown that the procedure given in Figure 7.5 converges to the optimal solution of Eq. (7.36) and the rate of convergence is $O(\log(T)/T)$.

7.5 Bibliographic Notes

The basic technique for efficiently projecting onto the probabilistic simplex is described in [31, 105]. The algorithm given in Figure 7.4 was derived in [105] in the context of batch learning of label ranking. There, it is called SOPOPO for Soft Projections onto Polyhedra. The adaptation of the general primal-dual interior point method for label ranking is presented in [108]. Last, the algorithm given in Figure 7.5 is a special case of the Pegasos algorithm described in [110].

Part III

Applications

Chapter 8

Online Email Categorization

In this chapter we describe the applicability of our framework to online email categorization. We cast the online categorization problem as a label ranking problem. Label ranking is the task of ordering labels with respect to their relevance to an input instance. We tackle the label ranking task based on our algorithmic framework for online learning described in previous chapters.

We demonstrate the potential of the algorithms we derived in Chapter 5 in a few experiments with email categorization tasks. In particular, we use both the squared Euclidean norm and the relative entropy as complexity measures to derive efficient additive and multiplicative algorithms for the label ranking task. We also use three dual ascending procedures which lead to three aggressiveness levels. Interestingly, our formal analysis is on a par with the experimental results, which gives further validation to the formal results presented in previous chapters.

8.1 Label Ranking

Label ranking is concerned with the task of ordering labels which are associated with a given instance in accordance to their relevance to the input instance. As an illustrative example consider an email categorization task in which the instances are email messages. Most email applications allow users to organize email messages into user-defined folders. For example, Google's Gmail users can tag each of their email messages with one or more labels. The set of labels is also user defined yet it is finite and typically is made up of a few tens if not hundreds of different labels. The advantage of this approach is the flexibility it gives users in categorizing emails, since an email may be associated with multiple labels. This approach contrasts with traditional systems in which an email is associated with a *single* physical folder. Users may browse all emails associated with a particular label and can also use the labels to search through their emails. However, manually attaching the relevant labels to each incoming email message can be an all-out effort. An online label ranking algorithm automatically learns how to rank-order labels in accordance with their relevance to each of the incoming email messages.

8.2 Hypothesis class for label ranking

Let $\mathcal{X} \subset \mathbb{R}^n$ be an instance domain and let $\mathcal{Y} = \{1, \ldots, k\}$ be a predefined set of labels. On round t the online algorithm first receives an instance $\mathbf{x}_t \in \mathcal{X}$ and is required to rank the labels in $\mathcal{Y}$ according to their relevance to the instance $\mathbf{x}_t$. For simplicity, we assume that the predicted ranking is given in the form of a vector $\boldsymbol{\rho}_t \in \mathbb{R}^k$, where $\rho_{t,r} > \rho_{t,s}$ means that label r is ranked ahead of label s. After the online learning algorithm has predicted the ranking $\boldsymbol{\rho}_t$ it receives as feedback a subset of labels $Y_t \subseteq \mathcal{Y}$, which are mostly relevant to $\mathbf{x}_t$. We say that the ranking predicted by the algorithm is correct if all the labels in Y_t are at the top of the list. That is, if for all $r \in Y_t$ and $s \notin Y_t$ we have that $\rho_{t,r} > \rho_{t,s}$. Otherwise, if there exist $r \in Y_t$ and $s \notin Y_t$ for which $\rho_{t,r} \leq \rho_{t,s}$, we say that the algorithm made a prediction mistake on round t. The ultimate goal of the algorithm is to minimize the total number of prediction mistakes it makes along its run.

We assume that the prediction of the algorithm at each round is determined by a linear function which is parameterized by k weight vectors, $\{\mathbf{w}_1^t, \ldots, \mathbf{w}_k^t\}$. Namely, for all $r \in \mathcal{Y}$, the value of $\rho_{t,r}$ is the inner product between $\mathbf{w}_r^t$ and $\mathbf{x}_t$, that is, $\rho_{t,r} = \langle \mathbf{w}_r^t, \mathbf{x}_t \rangle$. We use the notation $\bar{\mathbf{w}}^t$ as an abbreviation for the set $\{\mathbf{w}_1^t, \ldots, \mathbf{w}_k^t\}$. To evaluate the performance of $\bar{\mathbf{w}}^t$ on the example $(\mathbf{x}_t, Y_t)$ we check whether $\bar{\mathbf{w}}^t$ made a prediction mistake, by determining whether for all $r \in Y_t$ and $s \notin Y_t$ we have $\langle \mathbf{w}_r^t, \mathbf{x}_t \rangle > \langle \mathbf{w}_s^t, \mathbf{x}_t \rangle$. Since the $0-1$ loss function is not convex, we follow the methodology described in Section 2.3 and use a generalization of the *hinge-loss* function as a convex upper bound for the $0-1$ loss. Formally, we define:

$$g_t(\bar{\mathbf{w}}) \;=\; \ell(\bar{\mathbf{w}}, (\mathbf{x}_t, Y_t)) = \max_{r \in Y_t, s \notin Y_t} [\gamma - (\langle \mathbf{w}_r, \mathbf{x}_t \rangle - \langle \mathbf{w}_s, \mathbf{x}_t \rangle)]_+ \quad . \tag{8.1}$$

The term $\langle \mathbf{w}_r, \mathbf{x}_t \rangle - \langle \mathbf{w}_s, \mathbf{x}_t \rangle$ in the definition of the hinge-loss is a generalization of the notion of *margin* from binary classification. The hinge-loss penalizes $\bar{\mathbf{w}}$ for any margin less than γ. Additionally, if $\bar{\mathbf{w}}$ errs on $(\mathbf{x}_t, Y_t)$ then there exist $r \in Y_t$ and $s \notin Y_t$ such that $\langle \mathbf{w}_r, \mathbf{x}_t \rangle - \langle \mathbf{w}_s, \mathbf{x}_t \rangle \leq 0$ and thus $\ell^\gamma(\bar{\mathbf{w}}, (\mathbf{x}_t, Y_t)) \geq \gamma$. Thus, the *cumulative hinge-loss* suffered over a sequence of examples upper bounds γM.

8.3 Algorithms

We derive 6 algorithms from the algorithmic framework given in Figure 3.1 by using 2 types of complexity functions and 3 dual ascending procedures.

8.3.1 Complexity functions

Recall that our hypotheses are parameterized by a sequence of k vectors $\bar{\mathbf{w}} = (\mathbf{w}_1, \ldots, \mathbf{w}_k)$, where for each $r \in [k]$, $\mathbf{w}_r \in \mathbb{R}^n$. We denote the complexity function over $\bar{\mathbf{w}}$ by $\bar{f}$.

The first complexity function we use is the squared Euclidean norm,

$$\bar{f}(\bar{\mathbf{w}}) \;=\; \frac{1}{2}\|\bar{\mathbf{w}}\|_2^2 \;=\; \sum_{r=1}^{k} \frac{1}{2}\|\mathbf{w}_r\|^2 \;\;. \tag{8.2}$$

This complexity function is strongly convex on $\mathbb{R}^n$ with respect to the Euclidean norm. The second complexity function is obtained by summing the relative entropy, $f(\mathbf{w}) = \log(n) + \sum_i w_i \log(w_i)$, over the k weight vectors:

$$\bar{f}(\bar{\mathbf{w}}) \;=\; \sum_{r=1}^{k} \left(\log(n) + \sum_{i=1}^{n} w_{r,i} \log(w_{r,i}) \right) \;\;. \tag{8.3}$$

The relative entropy is strongly convex over the probabilistic simplex with respect to the norm $\|\cdot\|_1$. Therefore, the function $\bar{f}$ satisfies the conditions stated in Lemma 18. Thus, the analysis given in Lemma 1 can be repeated for $\bar{f}$ where the only difference is that Eq. (3.15) is replaced with the inequality given in Lemma 18.

8.3.2 Dual Ascending Methods

Since $\bar{\mathbf{w}}$ is a sequence of k vectors, each dual vector is also a sequence of k weight vector and we denote it by $\bar{\boldsymbol{\lambda}} = (\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_k)$. We adopt three dual ascending procedures which results in three update schemes.

Update I: Conservative subgradient update The first update we use is an adaptation of the Perceptron update rule for label ranking and is defined as follows. If on round t the algorithm did not make a prediction mistake we do not change the dual variables at all. If there was a prediction error we use the update given in Eq. (3.11); namely, we set the tth dual variable to be a subgradient of g_t at $\bar{\mathbf{w}}^t$. In our case, a subgradient can be found as follows. Let,

$$(r', s') \;=\; \underset{r \in Y_t, s \notin Y_t}{\operatorname{argmin}} \; \langle \mathbf{w}_r^t - \mathbf{w}_s^t, \mathbf{x}_t \rangle \;\;. \tag{8.4}$$

That is, the pair (r', s') designates the labels which mostly violate the required preference constraints. Since there was a prediction mistake, we get that $\langle \mathbf{w}_{r'}^t - \mathbf{w}_{s'}^t, \mathbf{x}_t \rangle \leq 0$. Based on this definition, a subgradient of g_t at $\bar{\mathbf{w}}^t$ is a vector $\bar{\boldsymbol{\lambda}}' = (\boldsymbol{\lambda}'_1, \ldots, \boldsymbol{\lambda}'_k)$ where $\boldsymbol{\lambda}'_{r'} = -\mathbf{x}_t$, $\boldsymbol{\lambda}'_{s'} = \mathbf{x}_t$, and for all other r we have $\boldsymbol{\lambda}'_r = \mathbf{0}$.

Update II: Aggressive subgradient update The second update we consider sets the tth dual variable to be $\alpha\,\bar{\boldsymbol{\lambda}}'$, where α is a scalar and $\bar{\boldsymbol{\lambda}}'$ is a subgradient of g_t at $\bar{\mathbf{w}}^t$ as described in the previous update. The value of alpha is set so as to maximize the dual objective function. That is,

$$\alpha \;=\; \underset{\alpha'}{\operatorname{argmax}}\; \mathcal{D}(\bar{\boldsymbol{\lambda}}_1, \ldots, \bar{\boldsymbol{\lambda}}_{t-1}, \alpha'\,\bar{\boldsymbol{\lambda}}', \mathbf{0}, \ldots, \mathbf{0})\ .$$

A closed-form solution for α can be obtained based on the derivation we presented in Section 7.2.

Update III: Optimal dual ascent The third update we use is the update given in Eq. (3.12). That is, we find the value of $\bar{\boldsymbol{\lambda}}_t$ that maximizes the increase in the dual. In Section 7.3 we presented efficient methods for implementing this update.

8.4 Experiments

In this section we present experimental results that demonstrate different aspects of our proposed algorithms. Our experiments compare the 6 algorithms described in the previous section. Note that using update I with the first complexity function yields an algorithm which was previously proposed and studied in [30] whereas using update II with the same complexity function yields the PA algorithm described in [28]. We experimented with the Enron email dataset (available from http://www.cs.umass.edu/∼ronb/datasets/enron_flat.tar.gz). The task is to automatically classify email messages into user defined folders. Thus, the instances in this dataset are email messages whereas the set of classes is the email folders. Note that our online setting naturally captures the essence of this email classification task. We represented each email message as a binary vector $\mathbf{x} \in \{0,1\}^n$ with a coordinate for each word, so that $x_i = 1$ if the word corresponding to the index i appears in the email message and zero otherwise. We ran the various algorithms on sequences of email messages from 7 users. For update II we used the closed form solution derived in Section 7.2 and for update III we used the algorithms given in Section 7.3. We found in our experiments that the number of iterations required by the interior point algorithm from Section 7.3 never exceeded 15. The performance of the different algorithms on the datasets is summarized in Table 8.1. It is apparent that regardless of the complexity function used, update III consistently outperforms update II which in turn consistently outperforms update I. However, the improvement of update II over update I is more significant than the improvement of update III over update II. Comparing the two complexity functions we note that regardless of the update used, the complexity function based on the entropy consistently outperforms the complexity function based on the squared norm. Note

Table 8.1: The average number of online mistakes for different algorithms on seven users from the Enron datasets.

			$\bar{f}(\bar{\mathbf{w}})$ as in Eq. (8.2)			$\bar{f}(\bar{\mathbf{w}})$ as in Eq. (8.3)		
username	$\lvert\mathcal{Y}\rvert$	m	update I	update II	update III	update I	update II	update III
beck-s	101	1971	58.5	55.2	51.9	54.0	50.2	47.1
farmer-d	25	3672	29.5	23.3	22.7	27.6	22.6	22.0
kaminski-v	41	4477	50.2	44.5	41.9	46.7	42.9	40.0
kitchen-l	47	4015	48.2	41.9	40.4	41.9	38.3	36.0
lokay-m	11	2489	24.9	19.1	18.4	24.0	18.7	18.2
sanders-r	30	1188	31.7	28.3	27.2	28.3	24.2	23.4
williams-w3	18	2769	5.0	4.5	4.4	4.2	3.4	3.1

that when using update I with the squared norm as a complexity function we obtain an adaptation of the Perceptron algorithm for the label ranking task whereas when using update I with the entropy complexity function we obtain an adaptation of the EG algorithm [75]. The superiority of the entropy-based complexity function over the squared norm was underscored in [75] for regression and classification problems.

8.5 Bibliographic Notes

Quite a few learning algorithms have been devised for the category ranking problem such as a multiclass version of AdaBoost called AdaBoost.MH [97], a generalization of Vapnik's Support Vector Machines to the multilabel setting by Elisseeff and Weston [45], and generalizations of the Perceptron algorithm to category ranking [30, 28].

The category ranking hypotheses this work employs are closely related to the ones presented and used in [45, 30, 32]. However, we depart from the standard paradigm which is confined to a specific form of a complexity function and give a unified account for online learning for label ranking problems.

Chapter 9

Speech-to-text and Music-to-score Alignment

In this chapter we describe a new approach to learning to align an audio signal, with a given sequence of events associated with the signal. This approach is based on our algorithmic framework for online convex programming. We focus on two applications of the alignment task: speech-to-phoneme alignment and music-to-score alignment. In speech-to-phoneme alignment the events are phonemes and the goal is to predict the start time of each phoneme in the spoken utterance. In music-to-score alignment we are given a sequence of musical notes (extracted from a musical score) along with a recording of the musical piece and the goal is to predict the start time of each note in the recorded audio signal.

Our proposed method is based on the usage of online convex programming for structured output learning (see Section 5.3.3). The alignment functions we devise are based on feature functions that map the audio signal and the sequence of events along with the target event timing sequence into an abstract vector-space. Based on this mapping, the alignment function distills to a classifier in the vector-space, which is aimed at separating correct timing sequences from incorrect ones. Based on our algorithmic framework for online convex programming, we derive a simple iterative algorithm for learning the alignment function and discuss its formal properties.

9.1 The Alignment Problem

In the alignment problem, we are provided with a signal which is accompanied by a discrete sequence of symbols or events and the goal is to align each of the events in the tagging sequence with its corresponding position in the signal. In speech-to-phoneme alignment, the events designate the phoneme uttered in the signal. In music-to-score alignment, the events are the notes in the score accompanying the signal. The alignment problem is the task of finding the start time of each tagged event in the input signal.

We represent a signal as a sequence of acoustic feature vectors $\bar{\mathbf{x}} = (\mathbf{x}_1, \ldots, \mathbf{x}_T)$, where $\mathbf{x}_t$ is a d-dimensional vector. For brevity we denote the domain of the feature vectors by $\mathcal{X} \subset \mathbb{R}^d$. Naturally, the length of the acoustic signal varies from one signal to another and thus T is not fixed. We denote by $\mathcal{X}^*$ the set of all finite-length sequences over $\mathcal{X}$. The sequence of events is denoted by $\bar{e} = (e_1, \ldots, e_K)$, where $e_k \in E$ for all $1 \leq k \leq K$ and E is the domain of the events. We assume that E is a finite set and we denote by E^* the set of all finite-length sequences over E. In summary, each input is a pair $(\bar{\mathbf{x}}, \bar{e})$ where $\bar{\mathbf{x}}$ is a sequence representing the acoustic signal and $\bar{e}$ is a sequence of events that occur in the signal. The alignment of the signal $\bar{\mathbf{x}}$ with the events $\bar{e}$ is a sequence of start-times $\bar{\mathbf{y}} = (\mathbf{y}_1, \ldots, \mathbf{y}_K)$ where $\mathbf{y}_k \in \{1, \ldots, T\}$ is the start-time of the event e_k in the acoustic signal. Our goal is to learn an *alignment function*, denoted h, which takes as input the pair $(\bar{\mathbf{x}}, \bar{e})$ and returns an event timing sequence $\bar{\mathbf{y}}$. That is, h is a function from $\mathcal{X}^* \times E^*$ to the set of finite-length sequences over the integers, $\mathbb{N}^*$.

We focus on two applications of the above general setting: speech-to-phoneme alignment and music-to-score alignment. In both problems, the acoustic representation $\bar{\mathbf{x}}$ is produced by dividing the acoustic signal into frames of several milliseconds, and extracting a d dimensional feature vector from each frame. In the speech-to-phoneme alignment problem the feature vector extracted from each frame is the Mel-frequency cepstrum coefficients (MFCC) along with their first and second derivatives. The sequence of events is a sequence of phoneme symbols from E, where E is the set of 48 American English phoneme symbols as proposed by [79]. We assume that the acoustic signal is an utterance of the phoneme sequence $\bar{e} = (e_1, \ldots, e_K)$ and our goal is to find the start time of each phoneme in the utterance.

In the music-to-score alignment problem, each acoustic feature vector $\mathbf{x}_t$ in the sequence $\bar{\mathbf{x}}$ is produced by calculating the short time Fourier transform of the tth frame of the signal. E is a set of "note-on" events. Formally, each "note-on" event is a pair $e_k = (p_k, s_k)$. The first element of the pair, $p_k \in \mathbb{P} = \{0, 1, \ldots, 127\}$ is the note's pitch value (coded using the MIDI standard). The second element, s, is assumed to be a positive integer ($s_k \in \mathbb{N}$) as it measures the (theoretical) start time of the note according to the musical score. Clearly, there are different ways to perform the same musical score. Therefore, the actual (or observed) start times of the notes in the perceived audio signal are very likely to be different from the symbolic start times. Our goal in the music score alignment task is to find the actual start time of each note in the acoustic signal.

9.2 Discriminative Supervised Learning

In this section we describe a discriminative supervised learning approach to learning an alignment function h from a training set of examples. Each example in the training set is composed of an acoustic signal, $\bar{\mathbf{x}}$, a sequence of events, $\bar{e}$, and the true event timing sequence, $\bar{\mathbf{y}}$. Our goal is to find an alignment function, h, which performs well on the training set as well as on unseen examples. First, we define a quantitative assessment of alignment functions. Let $(\bar{\mathbf{x}}, \bar{e}, \bar{\mathbf{y}})$ be an input example and let h be an alignment function. We denote by $\rho(\bar{\mathbf{y}}, h(\bar{\mathbf{x}}, \bar{e}))$ the cost of predicting the timing sequence $h(\bar{\mathbf{x}}, \bar{e})$ where the true timing sequence is $\bar{\mathbf{y}}$. Formally, $\rho : \mathbb{N}^* \times \mathbb{N}^* \to \mathbb{R}$ is a function that gets two timing sequences (of the same length) and returns a scalar which is the cost of predicting the second timing sequence where the true timing sequence is the first. We assume that $\rho(\bar{\mathbf{y}}, \bar{\mathbf{y}}') \geq 0$ for any two timing sequences $\bar{\mathbf{y}}, \bar{\mathbf{y}}'$ and that $\rho(\bar{\mathbf{y}}, \bar{\mathbf{y}}) = 0$. An example for a cost function is

$$\rho(\bar{\mathbf{y}}, \bar{\mathbf{y}}') = \frac{|\{i : |\mathbf{y}_i - y'_i| > \epsilon\}|}{|\bar{\mathbf{y}}|} \quad . \tag{9.1}$$

In words, the above cost is the average number of times the absolute difference between the predicted timing sequence and the true timing sequence is greater than ϵ. Recall that our goal is to find an alignment function h that achieves a low cost on unseen examples. Formally, let Q be any (unknown) distribution over the domain of the examples, $\mathcal{X}^* \times \mathbb{E}^* \times \mathbb{N}^\star$. The goal of the learning process is to minimize the risk of using the alignment function, defined as the expected cost of h on the examples, where the expectation is taken with respect to the distribution Q,

$$\mathrm{risk}(h) = \mathbb{E}_{(\bar{\mathbf{x}}, \bar{e}, \bar{\mathbf{y}}) \sim Q} \left[\rho(\bar{\mathbf{y}}, h(\bar{\mathbf{x}}, \bar{e})) \right] \quad .$$

To do so, we assume that the examples of our training set are identically and independently distributed (i.i.d.) according to the distribution Q. Note that we only observe the training examples but we do not know the distribution Q. The training set of examples is used as a restricted window through which we estimate the quality of alignment functions according to the distribution of unseen examples in the real world, Q. In the next section we show how to use the training set in order to find an alignment function, h, which achieves a low cost on unseen examples with high probability.

9.3 Hypothesis Class and a Learning Algorithm for Alignment

Recall that a supervised learning algorithm for alignment receives as an input a training set $\{(\bar{\mathbf{x}}_1, \bar{e}_1, \bar{\mathbf{y}}_1), \ldots, (\bar{\mathbf{x}}_m, \bar{e}_m, \bar{\mathbf{y}}_m)\}$ and returns an alignment function $h \in \mathcal{H}$, where $\mathcal{H}$ is a family of allowed alignment functions often called a hypothesis class. In this section we describe a parametric family of alignment functions.

To facilitate an efficient algorithm we confine ourselves to a restricted class of alignment functions. Specifically, we assume the existence of a predefined alignment feature functions, $\boldsymbol{\phi} : \mathcal{X}^* \times \mathbb{E}^* \times \mathbb{N}^* \to \mathbb{R}^n$. The alignment feature function gets the acoustic representation, $\bar{\mathbf{x}}$, and the sequence of events, $\bar{e}$, together with a candidate timing sequence, $\bar{\mathbf{y}}$, and returns a vector in $\mathbb{R}^n$. Intuitively, each element of the output vector $\boldsymbol{\phi}(\bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{e})$ represents a confidence value of the suggested timing sequence $\bar{\mathbf{y}}$. The construction of $\boldsymbol{\phi}$ is task dependent. As an example, let us briefly describe a single element of an alignment feature function for the speech-to-phoneme alignment task. This alignment feature sums a cepstral distance between the frames $\mathbf{x}_{y_i+1}$ and $\mathbf{x}_{y_i-1}$ over $i = 1, 2, \ldots, |\bar{\mathbf{y}}|$. For each i, if y_i is indeed the correct start time of phoneme i, we expect the distance between $\mathbf{x}_{y_i+1}$ and $\mathbf{x}_{y_i-1}$ to be large. On the other hand, if y_i does not reflect a true alignment point then the distance is likely to be small. Naturally, it is naive to assume that the above alignment feature can be used alone to find the correct timing sequence. However, as our experiments show, an appropriate combination of a few alignment features enables us to accurately predict the correct timing sequence.

The alignment functions we use are of the form

$$h(\bar{\mathbf{x}}, \bar{e}) = \underset{\bar{\mathbf{y}}}{\operatorname{argmax}} \; \langle \mathbf{w}, \boldsymbol{\phi}(\bar{\mathbf{x}}, \bar{e}, \bar{\mathbf{y}}) \rangle \ , \tag{9.2}$$

where $\mathbf{w} \in \mathbb{R}^n$ is a vector of importance weights that we need to learn. In words, h returns a suggestion for a timing sequence by maximizing a weighted sum of the confidence scores returned by each alignment feature function ϕ_j. Since h is parameterized by $\mathbf{w}$ we use the notation $h_{\mathbf{w}}$ for an alignment function h, which is defined as in Eq. (9.2). Note that the number of possible timing sequences, $\bar{\mathbf{y}}$, is exponentially large. Nevertheless, as we show later, under mild conditions on the form of the alignment feature function, $\boldsymbol{\phi}$, the optimization problem in Eq. (9.2) can be efficiently calculated using a dynamic programming procedure.

We now provide a simple iterative algorithm for learning an alignment function of the form given in Eq. (9.2) based on a training set of examples $\{(\bar{\mathbf{x}}_1, \bar{e}_1, \bar{\mathbf{y}}_1), \ldots, (\bar{\mathbf{x}}_m, \bar{e}_m, \bar{\mathbf{y}}_m)\}$. Our iterative algorithm first runs an online learning algorithm for structured output (see Section 5.3.3) on the training set. The online learning algorithm produces a sequence of weight vectors $\mathbf{w}_1, \ldots, \mathbf{w}_{m+1}$. Since our goal is to output a single weight vector that defines the output alignment function, we use an online-to-batch conversion technique as given in Section B.3. In our experiments, we used the algorithm given in Figure 5.4 and the "Last" conversion scheme described in Section B.3.

9.4 Efficient evaluation of the alignment function

So far we have put aside the problem of evaluation time of the function h given in Eq. (9.2). Recall that calculating h requires solving the following optimization problem,

$$h(\bar{\mathbf{x}}, \bar{e}) = \operatorname*{argmax}_{\bar{\mathbf{y}}} \langle \mathbf{w}, \boldsymbol{\phi}(\bar{\mathbf{x}}, \bar{e}, \bar{\mathbf{y}}) \rangle \ .$$

A direct search for the maximizer is not feasible since the number of possible timing sequences, $\bar{\mathbf{y}}$, is exponential in the number of events. Fortunately, as we show below, by imposing a few mild conditions on the structure of the alignment feature functions the problem can be solved in polynomial time.

For simplicity, we assume that each element of our feature function, ϕ_j, can be decomposed as follows. Let ψ_j be any function from $\mathcal{X}^* \times \mathbb{E}^* \times \mathbb{N}^3$ into the reals, which can be computed in a constant time. That is, ψ_j receives as input the signal, $\bar{\mathbf{x}}$, the sequence of events, $\bar{e}$, and three time points. Additionally, we use the convention $\mathbf{y}_0 = 0$ and $\mathbf{y}_{|\bar{e}|+1} = T + 1$. Using the above notation, we assume that each ϕ_j can be decomposed to be

$$\phi_j(\bar{\mathbf{x}}, \bar{e}, \bar{\mathbf{y}}) \ = \ \sum_{i=1}^{|\bar{\mathbf{y}}|} \psi_j(\bar{\mathbf{x}}, \bar{e}, y_{i-1}, y_i, y_{i+1}) \ . \tag{9.3}$$

The alignment feature functions we derive in later sections for speech-to-phoneme and music-to-score alignment can be decomposed as in Eq. (9.3).

We now describe an efficient algorithm for calculating the best timing sequence assuming that ϕ_j can be decomposed as in Eq. (9.3). Similar algorithms can be constructed for any base feature functions that can be described as a dynamic Bayesian network ([36, 113]). Given $i \in \{1, \ldots, |\bar{e}|\}$ and two time indices $t, t' \in \{1, \ldots, T\}$, denote by $D(i, t, t')$ the score for the prefix of the events sequence $1, \ldots, i$, assuming that their actual start times are $y_1, \ldots, y_i$, where $y_i = t'$ and assuming that $y_{i+1} = t$. This variable can be computed efficiently in a similar fashion to the forward variables calculated by the Viterbi procedure in HMMs (see for instance [92]). The pseudo code for computing $D(i, t, t')$ recursively is shown in Figure 9.1. The best sequence of actual start times, $\bar{\mathbf{y}}'$, is obtained from the algorithm by saving the intermediate values that maximize each expression in the recursion step. The complexity of the algorithm is $\mathcal{O}(|\bar{e}|\ |\bar{\mathbf{x}}|^3)$. However, in practice, we can use the assumption that the maximal length of an event is bounded, $t - t' \leq L$. This assumption reduces the complexity of the algorithm to be $\mathcal{O}(|\bar{e}|\ |\bar{\mathbf{x}}|\ L^2)$.

To conclude this section we discuss the global complexity of our proposed method. In the training phase, our algorithm performs m iterations, one iteration per each training example. At each iteration the algorithm evaluates the alignment function once and updates the alignment function,

```
INPUT:  audio signal $\bar{\mathbf{x}}$, sequence of events $\bar{e}$ ;
        weight vector $\mathbf{w}$ ; maximal length of an event $L$
INITIALIZE: $\forall(1 \le t \le L),\ D(0,t,0) = 0$
RECURSION:
 For $i = 1, \ldots, |\bar{e}|$
   For $t = 1, \ldots, |\bar{\mathbf{x}}|$
     For $t' = t - L, \ldots, t-1$
     $D(i,t,t') = \max_{t'-L \le t'' < t'} D(i-1,t',t'') + \langle \mathbf{w}, \boldsymbol{\psi}(\bar{\mathbf{x}}, \bar{e}, t'', t', t) \rangle$
TERMINATION: $D^\star = \max_{t'} D(|\bar{e}|, T, t')$
```

Figure 9.1: An efficient procedure for evaluating the alignment function given in Eq. (9.2).

if needed. Each evaluation of the alignment function takes an order of $\mathcal{O}(|\bar{e}|\ |\bar{\mathbf{x}}|\ L^2)$ operations. Therefore the total complexity of our method becomes $\mathcal{O}(m\ |\bar{e}|\ |\bar{\mathbf{x}}|\ L^2)$.

9.5 Speech-to-phoneme alignment

In this section we present the implementation details of our learning approach for the task of speech-to-phoneme alignment. Recall that our construction is based on a set of base alignment functions, $\{\phi_j\}_{j=1}^n$, which maps an acoustic-phonetic representation of a speech utterance as well as a suggested phoneme start time sequence into an abstract vector-space. All of our base alignment functions are decomposable as in Eq. (9.3) and therefore it suffices to describe the functions $\{\psi_j\}$. We start the section by introducing a specific set of base functions, which is highly adequate for the speech-to-phoneme alignment problem. Next, we report experimental results comparing our algorithm to alternative state-of-the-art approaches.

9.5.1 Base alignment functions

We utilize seven different base alignment functions ($n = 7$). These base functions are used for defining our alignment function $f(\bar{\mathbf{x}}, \bar{e})$ as in Eq. (9.2).

Our first four base functions aim at capturing transitions between phonemes. These base functions are based on the distance between frames of the acoustical signal at two sides of phoneme boundaries as suggested by a phoneme start time sequence $\bar{\mathbf{y}}$. The distance measure we employ, denoted by d, is the Euclidean distance between feature vectors. Our underlying assumption is that if two frames, $\mathbf{x}_t$ and $\mathbf{x}_{t'}$, are derived from the same phoneme then the distance $d(\mathbf{x}_t, \mathbf{x}_{t'})$ should be smaller than if the two frames are derived from different phonemes. Formally, our first 4 base functions are defined as

$$\psi_j(\bar{\mathbf{x}}, \bar{e}, y_{i-1}, y_i, y_{i+1}) = d(\mathbf{x}_{\mathbf{y}_i - j}, \mathbf{x}_{\mathbf{y}_i + j}), \ \ j \in \{1, 2, 3, 4\} \ . \tag{9.4}$$

If $\bar{\mathbf{y}}$ is the correct timing sequence then distances between frames across the phoneme change points are likely to be large. In contrast, an incorrect phoneme start time sequence is likely to compare frames from the same phoneme, often resulting small distances. Note that the first four base functions described above only use the start time of the ith phoneme and does not use the values of y_{i-1} and y_{i+1}.

The fifth base function we use is based on the framewise phoneme classifier described in [37]. Formally, for each phoneme event $e \in \mathbb{P}$ and frame $\mathbf{x} \in \mathcal{X}$, there is a confidence, denoted $g_e(\mathbf{x})$, that the phoneme e is pronounced in the frame $\mathbf{x}$. The resulting base function measures the cumulative confidence of the complete speech signal given the phoneme sequence and their start-times,

$$\psi_5(\bar{\mathbf{x}}, \bar{e}, \mathbf{y}_{i-1}, \mathbf{y}_i, \mathbf{y}_{i+1}) = \sum_{t=\mathbf{y}_i}^{\mathbf{y}_{i+1}-1} g_{e_i}(\mathbf{x}_t) \ . \tag{9.5}$$

The fifth base function use both the start time of the ith phoneme and the $(i+1)$th phoneme but ignores y_{i-1}.

Our next base function scores timing sequences based on phoneme durations. Unlike the previous base functions, the sixth base function is oblivious to the speech signal itself. It merely examines the length of each phoneme, as suggested by $\bar{\mathbf{y}}$, compared to the typical length required to pronounce this phoneme. Formally,

$$\psi_6(\bar{\mathbf{x}}, \bar{e}, \mathbf{y}_{i-1}, \mathbf{y}_i, \mathbf{y}_{i+1}) = \log \mathcal{N}(\mathbf{y}_{i+1} - \mathbf{y}_i; \hat{\mu}_{e_i} \ , \hat{\sigma}_{e_i}) \ \ , \tag{9.6}$$

where $\mathcal{N}$ is a Normal probability density function with mean $\hat{\mu}_e$ and standard deviation $\hat{\sigma}_e$. In our experiments, we estimated $\hat{\mu}_e$ and $\hat{\sigma}_e$ from the entire TIMIT training set, excluding SA1 and SA2 utterances.

Our last base function exploits assumptions on the speaking rate of a speaker. Intuitively, people usually speak in an almost steady rate and therefore a timing sequence in which speech rate is changed abruptly is probably incorrect. Formally, let $\hat{\mu}_e$ be the average length required to pronounce the eth phoneme. We denote by r_i the relative speech rate, $r_i = (\mathbf{y}_{i+1} - \mathbf{y}_i)/\hat{\mu}_e$. That is, r_i is the ratio between the actual length of phoneme e_i as suggested by $\bar{\mathbf{y}}$ to its average length. The relative

Table 9.1: Percentage of correctly positioned phoneme boundaries, given a predefined tolerance on the TIMIT corpus.

	$t \leq 10$ms	$t \leq 20$ms	$t \leq 30$ms	$t \leq 40$ms
TIMIT core test-set				
Discrim. Alignment	**79.7**	92.1	**96.2**	**98.1**
Brugnara *et al.* [15]	75.3	88.9	94.4	97.1
Hosom [69]		**92.57**		
TIMIT entire test-set				
Discrim. Alignment	**80.0**	**92.3**	**96.4**	**98.2**
Brugnara *et al.* [15]	74.6	88.8	94.1	96.8

speech rate presumably changes slowly over time. In practice the speaking rate ratios often differ from speaker to speaker and within a given utterance. We measure the local change in the speaking rate as $(r_i - r_{i-1})^2$ and we define the base function ψ_7 as the local change in the speaking rate,

$$\psi_7(\bar{\mathbf{x}}, \bar{e}, \mathbf{y}_{i-1}, \mathbf{y}_i, \mathbf{y}_{i+1}) = (r_i - r_{i-1})^2 \quad . \tag{9.7}$$

Note that ψ_7 relies on all three start-times it receives as an input, $\mathbf{y}_{i-1}, \mathbf{y}_i, \mathbf{y}_{i+1}$.

9.5.2 Experiments

To validate the effectiveness of the proposed approach we performed experiments with the TIMIT corpus. We first divided the training portion of TIMIT (excluding the SA1 and SA2 utterances) into two disjoint parts containing 500 and 3193 utterances, respectively. The first part of the training set was used for learning the functions g_{e_i} (Eq. (9.5)), which define the base function ψ_5. These functions were learned by the algorithm described in [37] using the MFCC+Δ+$\Delta\Delta$ acoustic features [46] and a Gaussian kernel ($\sigma = 6.24$ and $C = 5.0$). We ran our iterative alignment algorithm on the remaining utterances in the training set. We used the cost function

$$\rho(\bar{\mathbf{y}}, \bar{\mathbf{y}}') = \left\lfloor \frac{10 \; |\{i : |\mathbf{y}_i - y'_i| > \epsilon\}|}{|\bar{\mathbf{y}}|} \right\rfloor \quad .$$

where the value of ϵ was set to be 1 (i.e., 10 ms). That is, the output of ρ is in $\{0, 1, \ldots, 10\}$.

We evaluated the learned alignment functions on both the core test set and the entire test set of TIMIT. We compared our results to the results reported by Brugnara *et al.* [15] and the results obtained by Hosom [69]. The results are summarized in Table 9.1. For each tolerance value

$\tau \in \{10\,\mathrm{ms}, 20\,\mathrm{ms}, 30\,\mathrm{ms}, 40\,\mathrm{ms}\}$, we counted the number of predictions whose distance to the true boundary, $t = |y_i - y'_i|$, is less than τ. As can be seen in the table our discriminative large margin algorithm is comparable to the best results reported on TIMIT. Furthermore, we found in our experiments that the same level of accuracy is obtained when using solely the first 50 utterances (rather than the entire 3193 utterances that are available for training).

9.6 Music-to-score alignment

In this section we present the implementation details of our learning approach for the task of music-to-score alignment. We start the section by introducing a specific set of base alignment functions which is highly adequate for the music-to-score alignment problem. Next, we report experimental results comparing our algorithm to an alternative generative method for score alignment.

9.6.1 Base alignment functions

We utilized ten different base alignment functions ($n = 10$). Recall that each note-on event in the music-to-score alignment problem is a pair $e = (p, s)$, where p is the pitch of the note and s is the (theoretical) start time of the note. Our first nine base alignment functions ignore the value of s and thus, for these features, ψ_j only depends on $\bar{\mathbf{x}}, \bar{p}$, and $\bar{\mathbf{y}}$. Intuitively, the first nine base alignment functions measure the confidence that a pitch value p_i starts at time index y_i of the signal.

We now describe the specific form of each of the above base functions, starting with ψ_1. The function $\psi_1(\bar{\mathbf{x}}, \bar{e}, \mathbf{y}_{i-1}, \mathbf{y}_i, \mathbf{y}_{i+1})$ measures the energy of the acoustic signal at the frame $\mathbf{x}_{\mathbf{y}_i}$ and the frequency corresponding to the pitch p_i. Formally, let F_{p_i} denotes a band-pass filter with a center frequency at the first harmonic of the pitch p_i and cut-off frequencies of $1/4$ tone below and above p_i. Concretely, the lower cut-off frequency of F_{p_i} is $440 \cdot 2^{\frac{p_i - 57 - 0.5}{12}} Hz$ and the upper cut-off frequency is $440 \cdot 2^{\frac{p_i - 57 + 0.5}{12}} Hz$, where $p_i \in \mathbb{P} = \{0, 1, \ldots, 127\}$ is the pitch value (coded using the MIDI standard) and $440 \cdot 2^{\frac{p_i - 57}{12}}$ is the frequency value in Hz associated with the codeword p_i. Similarly, ψ_2 and ψ_3 are the output energies of band-pass filters centered at the second and third pitch harmonics, respectively. All the filters were implemented using the fast Fourier transform.

The above three local templates $\{\psi_j\}_{j=1}^3$ measure energy values for each time $\mathbf{y}_i$. Since we are interested in identifying note onset times, it is reasonable to compare energy values at time $\mathbf{y}_i$ with energy values at time $\mathbf{y}_i - 1$. However, the (discrete) first order derivative of the energy is highly sensitive to noise. Instead, we calculate the derivatives of a fitted second-order polynomial of each of the above local features. (This method is also a common technique in speech processing systems [92].) Therefore, the next six local templates, $\{\psi_j\}_{j=4}^9$, measure the first and second derivatives of the energy of the output signal of the three filters around the first three harmonics of

p_i.

While the first nine base alignment functions measure confidence of timing sequences based on spectral properties of the signal, the last alignment feature captures the similarity between $\bar{s}$ and $\bar{\mathbf{y}}$. Formally, let

$$r_i = \frac{y_{i+1} - y_i}{s_{i+1} - s_i} \tag{9.8}$$

be the ratio between the ith interval, according to $\bar{\mathbf{y}}$, to the interval according to $\bar{s}$. We also refer to r_i as the relative tempo. The sequence of relative tempo values is presumably constant in time, since $\bar{s}$ and $\bar{\mathbf{y}}$ represent two performances of the *same* musical piece. However, in practice the tempo ratios often differ from performance to performance and within a given performance. The local template ψ_{10} measures the local change in the tempo,

$$\psi_{10}(\bar{\mathbf{x}}, \bar{e}, \mathbf{y}_{i-1}, \mathbf{y}_i, \mathbf{y}_{i+1}) = (r_i - r_{i-1})^2 \quad .$$

The relative tempo of Eq. (9.8) is ill-defined whenever $s_{i+1} - s_i$ is zero (or relatively small). Since we are dealing with polyphonic musical pieces, very short intervals between notes are quite relevant. Therefore, we define the tempo r_i as in Eq. (9.8) but confine ourselves to indices i for which $s_{i+1} - s_i$ is greater than a predefined value τ (in our experiments we set $\tau = 60$ ms). Thus, if $s_{i+1} - s_i \le \tau$ or $s_i - s_{i-1} \le \tau$, then we set ψ_{10} to be zero.

9.6.2 Experiments

We now describe experiments with our alignment algorithm for the task of score alignment of polyphonic piano music pieces. Specifically, we compare our alignment algorithm to a generative method which is based on the Generalized Hidden Markov Model (GHMM). The details of the GHMM approach can be found in [102]. This GHMM method is closely related to graphical model approaches for score alignment, first proposed by Raphael [93, 94]. Note in passing that more advanced algorithms for real-time score alignment have been suggested (see for example [25] and the references therein). In our experiments we focus on the basic comparison between the discriminative and generative approaches for score alignment. Recall that our alignment algorithm uses a training set of examples to deduce an alignment function. We downloaded 12 musical pieces from `http://www.piano-midi.de/mp3.php` where sound and MIDI were both recorded. Here the sound serves as the acoustical signal $\bar{\mathbf{x}}$ and the MIDI is the actual start times $\bar{\mathbf{y}}$. We also downloaded other MIDI files of the same musical pieces from a variety of other web-sites and used these MIDI files to create the sequence of events $\bar{e}$. The complete dataset we used is available from `http://www.cs.huji.ac.il/~shais/alignment`.

Table 9.2: Summary of the LOO loss (in ms) for different algorithms for music-to-score alignment.

	GHMM-1	GHMM-3	GHMM-5	GHMM-7	Discrim.
1	10.0	188.9	49.2	69.7	**8.9**
2	15.3	159.7	31.2	20.7	**9.1**
3	22.5	48.1	29.4	37.4	**17.1**
4	12.7	29.9	15.2	17.0	**10.0**
5	54.5	82.2	55.9	53.3	**41.8**
6	12.8	46.9	26.7	23.5	**14.0**
7	336.4	75.8	30.4	43.3	**9.9**
8	11.9	24.2	15.8	17.1	**11.4**
9	11473	11206	51.6	12927	**20.6**
10	16.3	60.4	16.5	20.4	**8.1**
11	22.6	39.8	27.5	19.2	**12.4**
12	13.4	14.5	13.8	28.1	**9.6**
mean	1000.1	998.1	30.3	1106.4	**14.4**
std	3159	3078.3	14.1	3564.1	**9.0**
median	15.8	54.2	28.5	25.8	**10.7**

In the score alignment problem we report the average alignment error,

$$\frac{1}{|\bar{\mathbf{y}}|}\sum_{i=1}^{|\bar{\mathbf{y}}|} |\mathbf{y}_i - \mathbf{y}'_i| \ .$$

Since this dataset is rather small, we ran our iterative algorithm on the training set several times and chose the alignment function which minimizes the error on the training set. We used the leave-one-out (LOO) cross-validation procedure for evaluating the test results. In the LOO setup the algorithms are trained on all the training examples except one, which is used as a test set. The error between the predicted and true start times is computed for each of the algorithms. The GHMM approach uses a Gaussian Mixture Model (GMM) for modeling some of the probabilities. The number of Gaussians used by the GMM needs to be determined. We used the values of 1, 3, 5 and 7 as the number of Gaussians and we denote by GHMM-n the resulting generative model with n Gaussians. In addition, we used the EM algorithm to train the GMMs. The EM algorithm converges to a local maximum, rather than to the global maximum. A common solution to this problem is to use a random partition of the data to initialize the EM. In all our experiments with the GMM we used 15 random partitions of the data to initialize the EM and chose the one that led to the highest likelihood.

The LOO results for each of the 12 musical pieces are summarized in Table 9.2. As seen from the table, our discriminative learning algorithm outperforms all the variants of the GHMM method in all of the experiments. Moreover, in all but two of the experiments the error of the discriminative algorithm is less than 20 ms, which is the length of an acoustic frame in our experiments; thus it is the best accuracy one can hope for at this time resolution. It can be seen that the variance of the LOO loss obtained by the generative algorithms is rather high. This can be attributed to the fact that the EM algorithm converges to a local maximum which depends on initialization of the parameters.

9.7 Bibliographic Notes

Most of the previous work on speech-to-phoneme and music-to-score alignment focused on a generative model of the audio signal using Hidden Markov Models (HMM). See for example [15, 69, 114, 101, 112] and the references therein. Despite their popularity, HMM-based approaches have several drawbacks such as convergence of the EM procedure to local maxima and overfitting effects due to the large number of parameters. In this chapter we proposed an alternative approach to learning alignment functions that builds upon recent work on discriminative supervised learning. The advantage of discriminative learning algorithms stems from the fact that the objective function used during the learning phase is tightly coupled with the decision task one needs to perform. In addition, there is both theoretical and empirical evidence that discriminative learning algorithms are likely to outperform generative models for the same task (cf. [117, 33]). One of the best known discriminative learning algorithms is the support vector machine (SVM), which has been successfully applied in speech processing applications [96, 71, 81]. The classical SVM algorithm is designed for simple decision tasks such as binary classification and regression. Hence, its exploitation in signal processing systems so far has also been restricted to simple decision tasks such as phoneme classification and music genre classification. The alignment problem is more involved, since we need to predict a sequence of event timings rather than a single number. The main challenge is therefore to extend the notion of discriminative learning to the complex task of alignment.

Our proposed method is based on recent advances in kernel machines and large margin classifiers for sequences [23, 113, 115], which in turn build on the pioneering work of Vapnik and colleagues [117, 33]. The learning algorithm we present shares similarities with the SVM method for structured output prediction [113, 115]. However, the weight vector generated by our method is not identical to the one obtained by directly solving the SVM optimization problem. It is worth noting that by applying our regret bounds from Section 5.3 along with the risk bounds given in Section B.3 we obtain generalization bounds which are comparable to the generalization bounds derived for the SVM method (see for example [113]). The major advantage of our method over directly solving the SVM problem is its simplicity and efficiency.

Chapter 10

Discussion

In this dissertation we presented a new framework for the design and analysis of online convex programming algorithms. We illustrated the applicability of the framework to online learning and boosting. Our framework yields the tightest known bounds for existing algorithms and also paves the way for the design of new algorithms.

The main idea behind our derivation is the connection between regret bounds and Fenchel duality. This connection leads to a reduction from online convex programming to the task of incrementally ascending the dual objective function. The analysis of our algorithms makes use of the weak duality theorem. Despite the generality of this framework, the resulting analysis is more distilled than earlier analyses. The strongly convex property we employ both simplifies the analysis and enables more intuitive conditions in our theorems.

The connection between online learning and optimization was first derived by us in [106, 107]. There, we focused on the problem of binary classification with the hinge loss, and used the Lagrange duality. The generalization of the framework beyond the specific setting of online learning with the hinge-loss to the general setting of online convex programming appears in [104]. This generalization enables us to automatically utilize well known online learning algorithms, such as the EG and p-norm algorithms [76, 62], in the setting of online convex programming. The self-tuned algorithmic framework we presented in Section 3.5 has never been published. The online-to-batch conversion schemes described in Chapter B are partially based on the analysis of stochastic gradient descents described in [110]. The analysis of boosting algorithms based on our framework is an improvement over the analysis in [104]. Some of the algorithms that appear in Chapter 7 as well as the application to online email categorization were published in [105, 108]. The application of music-to-score and text-to-speech alignment was published in [102, 73, 72].

There are various possible extensions of the work that we did not pursue here due to lack of space. For instance, our framework can naturally be used for the analysis of other settings such as repeated games (see [55, 123]). The reduction from online learning to dual incrementing algorithms may enable us to apply algorithms proposed in convex analysis to online learning (see for example [16]). We also conjecture that our primal-dual view of boosting will lead to new methods for regularizing boosting algorithms, thus improving their generalization capabilities. Another interesting direction is the applicability of our framework to settings where the target hypothesis is not fixed but rather drifts with the sequence of examples. Finally, we believe that the proposed framework will be useful in deriving additional online learning algorithms for various prediction problems.

Bibliography

[1] S. Agmon. The relaxation method for linear inequalities. *Canadian Journal of Mathematics*, 6(3):382–392, 1954.

[2] N. Alon and J. H. Spencer. *The Probabilistic Method*. Wiley-Interscience, second edition, 2000.

[3] Y. Altun, I. Tsochantaridis, and T. Hofmann. Hidden markov support vector machines. In *Proceedings of the Twentieth International Conference on Machine Learning*, 2003.

[4] Y. Amit, S. Shalev-Shwartz, and Y. Singer. Online classification for complex problems using simultaneous projections. In *Advances in Neural Information Processing Systems 20*, 2006.

[5] P. Auer, N. Cesa-Bianchi, and C. Gentile. Adaptive and self-confident on-line learning algorithms. *Journal of Computer and System Sciences*, 64(1):48–75, 2002.

[6] K. Azoury and M. Warmuth. Relative loss bounds for on-line density estimation with the exponential family of distributions. *Machine Learning*, 43(3):211–246, 2001.

[7] K. Azuma. Weighted sums of certain dependent random variables. *Tohoku Mathematical Journal*, 68:357–367, 1967.

[8] D.P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1999.

[9] D. Blackwell. An analog of the minimax theorem for vector payoffs. *Pacific Journal of Mathematics*, 6(1):1–8, Spring 1956.

[10] A. Blum and Y. Mansour. From external to internal regret. *Journal of Machine Learning Research*, 8:1307–1324, Jun 2007.

[11] J. Borwein and A. Lewis. *Convex Analysis and Nonlinear Optimization*. Springer, 2006.

[12] S. Boucheron, O. Bousquet, and G. Lugosi. Concentration inequalities. In O. Bousquet, U.v. Luxburg, and G. Ratsch, editors, *Advanced Lectures in Machine Learning*, pages 208–240. Springer, 2004.

[13] S. Boucheron, O. Bousquet, and G. Lugosi. Theory of classification: a survey of recent advances. *ESAIM: Probability and Statistics*, 9:323–375, 2005.

[14] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[15] F. Brugnara, D. Falavigna, and M. Omologo. Automatic segmentation and labeling of speech based on hidden markov models. *Speech Communication*, 12:357–370, 1993.

[16] Y. Censor and S.A. Zenios. *Parallel Optimization: Theory, Algorithms, and Applications*. Oxford University Press, New York, NY, USA, 1997.

[17] N. Cesa-Bianchi, A. Conconi, and C. Gentile. On the generalization ability of on-line learning algorithms. *IEEE Transactions on Information Theory*, 50(9):2050–2057, September 2004.

[18] N. Cesa-Bianchi and C. Gentile. Improved risk tail bounds for on-line algorithms. In *Advances in Neural Information Processing Systems 19*, 2006.

[19] N. Cesa-Bianchi and G. Lugosi. Potential-based algorithms in on-line prediction and game theory. *Machine Learning Journal*, 3(51):239–261, 2003.

[20] N. Cesa-Bianchi and G. Lugosi. *Prediction, learning, and games*. Cambridge University Press, 2006.

[21] Nicolò Cesa-Bianchi, Yoav Freund, David Haussler, David P. Helmbold, Robert E. Schapire, and Manfred K. Warmuth. How to use expert advice. *Journal of the Association for Computing Machinery*, 44(3):427–485, May 1997.

[22] M. Collins. Discriminative reranking for natural language parsing. In *Machine Learning: Proceedings of the Seventeenth International Conference*, 2000.

[23] M. Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Conference on Empirical Methods in Natural Language Processing*, 2002.

[24] M. Collins, R.E. Schapire, and Y. Singer. Logistic regression, AdaBoost and Bregman distances. *Machine Learning*, 47(2/3):253–285, 2002.

[25] A. Cont. Realtime audio to score alignment for polyphonic music instruments using sparse non-negative constraints and hierarchical HMMs. In *IEEE International Conference in Acoustics and Speech Signal Processing*, 2006.

[26] Thomas M. Cover. Behavior of sequential predictors of binary sequences. *Trans. 4th Prague Conf. Information Theory Statistical Decision Functions, Random Processes*, 1965.

[27] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive aggressive algorithms. Technical report, The Hebrew University, 2005.

[28] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585, Mar 2006.

[29] K. Crammer and Y. Singer. A new family of online algorithms for category ranking. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2002.

[30] K. Crammer and Y. Singer. A new family of online algorithms for category ranking. *Journal of Machine Learning Research*, 3:1025–1058, 2003.

[31] K. Crammer and Y. Singer. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3:951–991, 2003.

[32] K. Crammer and Y. Singer. Loss bounds for online category ranking. In *Proceedings of the Eighteenth Annual Conference on Computational Learning Theory*, 2005.

[33] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, 2000.

[34] A. P. Dawid. Statistical theory: The prequential approach. *Journal of the Royal Statistical Society, Series A*, 147:278–292, 1984.

[35] A.P. Dawid and V. Vovk. Prequential probability: principles and properties. *Bernoulli*, 3:1–38, 1997.

[36] T. Dean and K. Kanazawa. A model for reasoning about persistence and causation. *Computational Intelligence*, 5(3):142–150, 1989.

[37] O. Dekel, J. Keshet, and Y. Singer. Online algorithm for hierarchical phoneme classification. In *Workshop on Multimodal Interaction and Related Machine Learning Algorithms; Lecture Notes in Computer Science*, pages 146–159. Springer-Verlag, 2004.

[38] O. Dekel, S. Shalev-Shwartz, and Y. Singer. Smooth epsilon-insensitive regression by loss symmetrization. In *Proceedings of the Sixteenth Annual Conference on Computational Learning Theory*, 2003.

[39] O. Dekel, S. Shalev-Shwartz, and Y. Singer. The power of selective memory: Self-bounded learning of prediction suffix trees. In *Advances in Neural Information Processing Systems 17*, 2005.

[40] O. Dekel, S. Shalev-Shwartz, and Y. Singer. Smooth epsilon-insensitive regression by loss symmetrization. *Journal of Machine Learning Research*, 6:711–741, May 2005.

[41] O. Dekel, S. Shalev-Shwartz, and Y. Singer. The Forgetron: A kernel-based perceptron on a fixed budget. In *Advances in Neural Information Processing Systems 18*, 2006.

[42] A. DeSantis, G. Markowsky, and M.N. Wegman. Learning probabilistic prediction functions. In *COLT*, pages 312–328, 1988.

[43] L. Devroye, L. Györfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer, 1996.

[44] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. Wiley, 1973.

[45] A. Elisseeff and J. Weston. A kernel method for multi-labeled classification. In *Advances in Neural Information Processing Systems 14*, 2001.

[46] ETSI Standard, ETSI ES 201 108, 2000.

[47] M. Feder, N. Merhav, and M. Gutman. Universal prediction of individual sequences. *IEEE Transactions on Information Theory*, 38:1258–1270, 1992.

[48] A. Forsgren, P. E. Gill, and M. H. Wright. Interior methods for nonlinear optimization. *SIAM Review*, 44(4), 2002.

[49] D.P. Foster and R.V. Vohra. A randomization rule for selecting forecasts. *Operations Research*, 41(4):704–709, July–August 1993.

[50] D.P. Foster and R.V. Vohra. Asymptotic calibration. *Biometrika*, 85(2):379–390, 1998.

[51] D. A. Freedman. On tail probabilities for martingales. *The Annals of Probability*, 3(1):100–118, February 1975.

[52] Y. Freund. Boosting a weak learning algorithm by majority. In *Proceedings of the Third Annual Workshop on Computational Learning Theory*, pages 202–216, August 1990. To appear, *Information and Computation*.

[53] Y. Freund and R. E. Schapire. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296, 1999.

[54] Y. Freund and R.E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Computational Learning Theory: Second European Conference, EuroCOLT '95*, pages 23–37. Springer-Verlag, 1995.

[55] Y. Freund and R.E. Schapire. Game theory, on-line prediction and boosting. In *Proceedings of the Ninth Annual Conference on Computational Learning Theory*, pages 325–332, 1996.

[56] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28(2):337–374, April 2000.

[57] C. Gentile. A new approximate maximal margin classification algorithm. *Journal of Machine Learning Research*, 2:213–242, 2001.

[58] C. Gentile. The robustness of the p-norm algorithms. *Machine Learning*, 53(3), 2002.

[59] C. Gentile and N. Littlestone. The robustness of the p-norm algorithms. In *Proceedings of the Twelfth Annual Conference on Computational Learning Theory*, 1999.

[60] G. Gordon. Regret bounds for prediction problems. In *Proceedings of the Twelfth Annual Conference on Computational Learning Theory*, 1999.

[61] G. Gordon. No-regret algorithms for online convex programs. In *Advances in Neural Information Processing Systems 20*, 2006.

[62] A. J. Grove, N. Littlestone, and D. Schuurmans. General convergence results for linear discriminant updates. *Machine Learning*, 43(3):173–210, 2001.

[63] J. Hannan. Approximation to Bayes risk in repeated play. In M. Dresher, A. W. Tucker, and P. Wolfe, editors, *Contributions to the Theory of Games*, volume III, pages 97–139. Princeton University Press, 1957.

[64] S. Hart and A. Mas-Colell. A general class of adaptive strategies. *Journal of Economic Theory*, 98(1):26–54, 2000.

[65] S. Hart and A. Mas-Colell. A simple adaptive procedure leading to correlated equilibrium. *Econometrica*, 68:1127–1150, 2000.

[66] E. Hazan, A. Kalai, S. Kale, and A. Agarwal. Logarithmic regret algorithms for online convex optimization. In *Proceedings of the Nineteenth Annual Conference on Computational Learning Theory*, 2006.

[67] D. P. Helmbold and M. Warmuth. On weak learning. *Journal of Computer and System Sciences*, 50:551–573, 1995.

[68] M. Herbster. Learning additive models online with fast evaluating kernels. In *Proceedings of the Fourteenth Annual Conference on Computational Learning Theory*, pages 444–460, 2001.

[69] J.P. Hosom. Automatic phoneme alignment based on acoustic-phonetic modeling. In *Proceedings of the Seventh International Conference on Spoken Language Processing*, pages 357–360, 2002.

[70] M.J. Kearns and U.V. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, 1994.

[71] J. Keshet, D. Chazan, and B.-Z. Bobrovsky. Plosive spotting with margin classifiers. In *Proceedings of the Seventh European Conference on Speech Communication and Technology*, pages 1637–1640, 2001.

[72] J. Keshet, S. Shalev-Shwartz, S. Bengio, Y. Singer, and D. Chazan. Discriminative kernel-based phoneme sequence recognition. In *Interspeech*, 2006.

[73] J. Keshet, S. Shalev-Shwartz, Y. Singer, and D. Chazan. Phoneme alignment based on discriminative learning. In *Interspeech*, 2005.

[74] J. Kivinen, A. J. Smola, and R. C. Williamson. Online learning with kernels. *IEEE Transactions on Signal Processing*, 52(8):2165–2176, 2002.

[75] J. Kivinen and M. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1–64, January 1997.

[76] J. Kivinen and M. Warmuth. Relative loss bounds for multidimensional regression problems. *Journal of Machine Learning*, 45(3):301–329, July 2001.

[77] J. Kivinen and M.K. Warmuth. Additive versus exponentiated gradient updates for linear prediction. In *stoc95*, pages 209–218, 1995. See also technical report UCSC-CRL-94-16, University of California, Santa Cruz, Computer Research Laboratory.

[78] W. Krauth and M. Mézard. Learning algorithms with optimal stability in neural networks. *Journal of Physics A.*, 20:745, 1987.

[79] K.-F. Lee and H.-W. Hon. Speaker independent phone recognition using hidden markov models. *IEEE Trans. Acoustic, Speech and Signal Proc.*, 37(2):1641–1648, 1989.

[80] Y. Li and P. M. Long. The relaxed online maximum margin algorithm. *Machine Learning*, 46(1–3):361–387, 2002.

[81] Z. Litichever and D. Chazan. Classification of transition sounds with application to automatic speech recognition. In *EUROSPEECH*, 2001.

[82] N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2:285–318, 1988.

[83] N. Littlestone. From on-line to batch learning. In *Proceedings of the Second Annual Workshop on Computational Learning Theory*, pages 269–284, July 1989.

[84] N. Littlestone. *Mistake bounds and logarithmic linear-threshold learning algorithms*. PhD thesis, U. C. Santa Cruz, March 1989.

[85] N. Littlestone and M. Warmuth. Relating data compression and learnability. Unpublished manuscript, November 1986.

[86] L. Mason, J. Baxter, P. Bartlett, and M. Frean. Functional gradient techniques for combining hypotheses. In *Advances in Large Margin Classifiers*. MIT Press, 1999.

[87] M. Minsky and S. Papert. *Perceptrons: An Introduction to Computational Geometry*. The MIT Press, 1969.

[88] F. Mosteller, R. E. K. Rourke, and G. B. Thomas. *Probability and Statistics*. Addison-Wesley, 1961.

[89] Y. Nesterov. Primal-dual subgradient methods for convex problems. Technical report, Center for Operations Research and Econometrics (CORE), Catholic University of Louvain (UCL), 2005.

[90] Y. Nesterov and A. Nemirovski. *Interior-Point Polynomial Algorithms in Convex Programming*. SIAM Studies in Applied Mathematics, 1994.

[91] A. B. J. Novikoff. On convergence proofs on perceptrons. In *Proceedings of the Symposium on the Mathematical Theory of Automata*, volume XII, pages 615–622, 1962.

[92] L. Rabiner and B.H. Juang. *Fundamentals of Speech Recognition*. Prentice Hall, 1993.

[93] C. Raphael. Automatic segmentation of acoustic musical signals using hidden markov models. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 21(4), April 1999.

[94] C. Raphael. A hybrid graphical model for aligning polyphonic audio with musical scores. In *Proceedings of the 5th International Conference on Music Information Retrieval*, 2004.

[95] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–407, 1958. (Reprinted in *Neurocomputing* (MIT Press, 1988).).

[96] J. Salomon, S. King, and M. Osborne. Framewise phone classification using support vector machines. In *Proceedings of the Seventh International Conference on Spoken Language Processing*, pages 2645–2648, 2002.

[97] R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):1–40, 1999.

[98] R.E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990.

[99] R.E. Schapire, Y. Freund, P. Bartlett, and W.S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5):1651–1686, October 1998.

[100] B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond*. MIT Press, 2002.

[101] S. Shalev-Shwartz, S. Dubnov, N. Friedman, and Y. Singer. Robust temporal and spectral modeling for query by melody. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2002.

[102] S. Shalev-Shwartz, J. Keshet, and Y. Singer. Learning to align polyphonic music. In *Proceedings of the 5th International Conference on Music Information Retrieval*, 2004.

[103] S. Shalev-Shwartz and Y. Singer. A new perspective on an old perceptron algorithm. In *Proceedings of the Eighteenth Annual Conference on Computational Learning Theory*, 2005.

[104] S. Shalev-Shwartz and Y. Singer. Convex repeated games and fenchel duality. In *Advances in Neural Information Processing Systems 20*, 2006.

[105] S. Shalev-Shwartz and Y. Singer. Efficient learning of label ranking by soft projections onto polyhedra. *Journal of Machine Learning Research*, 7 (July):1567–1599, 2006.

[106] S. Shalev-Shwartz and Y. Singer. Online learning meets optimization in the dual. In *Proceedings of the Nineteenth Annual Conference on Computational Learning Theory*, 2006.

[107] S. Shalev-Shwartz and Y. Singer. A primal-dual perspective of online learning algorithms. *Machine Learning Journal*, 2007.

[108] S. Shalev-Shwartz and Y. Singer. A unified algorithmic approach for efficient online label ranking. In *aistat07*, 2007.

[109] S. Shalev-Shwartz, Y. Singer, and A. Ng. Online and batch learning of pseudo-metrics. In *Proceedings of the Twenty-First International Conference on Machine Learning*, 2004.

[110] S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub-gradient solver for svm. In *Proceedings of the 24th International Conference on Machine Learning*, 2007.

[111] C.E. Shannon. Prediction and entropy of printed english. *Bell Sys. Tech. Jour.*, 30:51–64, 1951.

[112] F. Soulez, X. Rodet, and D. Schwarz. Improving polyphonic and poly-instrumental music to score alignment. In *Proceedings of the International Symposium on Music Information Retrieval*, 2003.

[113] B. Taskar, C. Guestrin, and D. Koller. Max-margin markov networks. In *Advances in Neural Information Processing Systems 17*, 2003.

[114] D.T. Toledano, L.A.H. Gomez, and L.V. Grande. Automatic phoneme segmentation. *IEEE Trans. Speech and Audio Proc.*, 11(6):617–625, 2003.

[115] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the Twenty-First International Conference on Machine Learning*, 2004.

[116] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, November 1984.

[117] V. N. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

[118] V. N. Vapnik and A. Ya. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its applications*, XVI(2):264–280, 1971.

[119] V. Vovk. Competitive on-line statistics. *International Statistical Review*, 69:213–248, 2001.

[120] M. Warmuth, J. Liao, and G. Ratsch. Totally corrective boosting algorithms that maximize the margin. In *Proceedings of the 23rd international conference on Machine learning*, 2006.

[121] J. Weston and C. Watkins. Support vector machines for multi-class pattern recognition. In *Proceedings of the Seventh European Symposium on Artificial Neural Networks*, April 1999.

[122] T. Zhang. Data dependent concentration bounds for sequential prediction algorithms. In *Proceedings of the Eighteenth Annual Conference on Computational Learning Theory*, pages 173–187, 2005.

[123] M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the Twentieth International Conference on Machine Learning*, 2003.

[124] J. Ziv and A. Lempel. Compression of individual sequences via variable rate coding. *IEEE Transactions on Information Theory*, 24:530–536, 1978.

Appendix A

Convex Analysis

In this chapter we recall a few definitions from convex analysis and derive several tools used throughout this dissertation. For a more thorough introduction see for example [11, 14].

A.1 Convex sets and functions

A set S is convex if for any two vectors $\mathbf{w}_1, \mathbf{w}_2$ in S, all the line between $\mathbf{w}_1$ and $\mathbf{w}_2$ is also within S. That is, for any $\alpha \in [0,1]$ we have that $\alpha \mathbf{w}_1 + (1-\alpha)\mathbf{w}_2 \in S$. A set S is open if every point in S has a neighborhood lying in S. A set S is closed if its complement is an open set. A function $f : S \to \mathbb{R}$ is closed and convex if for any scalar $\alpha \in \mathbb{R}$, the level set $\{\mathbf{w} : f(\mathbf{w}) \leq \alpha\}$ is closed and convex. Throughout this work, we deal solely with closed functions.

A.2 Gradients, Subgradients, and Differential Sets

A vector $\boldsymbol{\lambda}$ is a sub-gradient of a function f at $\mathbf{v}$ if

$$\forall \mathbf{u} \in S, \;\; f(\mathbf{u}) - f(\mathbf{v}) \geq \langle \mathbf{u} - \mathbf{v}, \boldsymbol{\lambda} \rangle \; .$$

The differential set of f at $\mathbf{v}$, denoted $\partial f(\mathbf{v})$, is the set of all sub-gradients of f at $\mathbf{v}$. A function f is convex iff $\partial f(\mathbf{v})$ is non-empty for all $\mathbf{v} \in S$. If f is convex and differentiable at $\mathbf{v}$ then $\partial f(\mathbf{v})$ consists of a single vector which amounts to the gradient of f at $\mathbf{v}$ and is denoted by $\nabla f(\mathbf{v})$. As a consequence we obtain that a differential function f is convex iff for all $\mathbf{v}, \mathbf{u} \in S$ we have that

$$f(\mathbf{u}) - f(\mathbf{v}) - \langle \mathbf{u} - \mathbf{v}, \nabla f(\mathbf{v}) \rangle \geq 0 \; .$$

If f is differentiable at $\mathbf{w}$ then $\partial f(\mathbf{w})$ consists of a single vector which amounts to the gradient of f at $\mathbf{w}$ and is denoted by $\nabla f(\mathbf{w})$.

We conclude this section with some basic rules that help us to claculate subgradients.

- **Scaling:** For $\alpha > 0$ we have $\partial(\alpha f) = \alpha\, \partial f$.
- **Addition:** $\partial(f_1 + f_2) = \partial f_1 + \partial f_2$.
- **Affine transformation:** If $g(\mathbf{x}) = f(Ax + b)$ for a matrix A and a vector b then $\partial g(\mathbf{x}) = A^t \partial f(Ax + b)$, where A^t is the transpose of A.
- **Pointwise Maximum:** If $f = \max_{i\in[m]} f_i$ then ∂f at a vector $\mathbf{x}$ is the convex hull of the union of ∂f_i at $\mathbf{x}$ over $i \in \{j : f_j(\mathbf{x}) = f(\mathbf{x})\}$.

A.3 Fenchel Conjugate

The Fenchel conjugate of a function $f : S \to \mathbb{R}$ is defined as

$$f^\star(\boldsymbol{\theta}) \;=\; \sup_{\mathbf{w}\in S} \langle \mathbf{w}, \boldsymbol{\theta}\rangle - f(\mathbf{w}) \ . \tag{A.1}$$

Since $f^\star$ is defined as a supremum of linear functions it is convex. If f is closed and convex then the Fenchel conjugate of $f^\star$ is f itself. The Fenchel-Young inequality states that for any $\mathbf{w}$ and $\boldsymbol{\theta}$ we have that $f(\mathbf{w}) + f^\star(\boldsymbol{\theta}) \geq \langle \mathbf{w}, \boldsymbol{\theta}\rangle$.

Sub-gradients play an important role in the definition of Fenchel conjugate. In particular, the following lemma states that if $\boldsymbol{\lambda} \in \partial f(\mathbf{w})$ then Fenchel-Young inequality holds with equality.

Lemma 11 *Let f be a closed and convex function and let $\partial f(\mathbf{w}')$ be its differential set at $\mathbf{w}'$. Then, for all $\boldsymbol{\lambda}' \in \partial f(\mathbf{w}')$ we have, $f(\mathbf{w}') + f^\star(\boldsymbol{\lambda}') \;=\; \langle \boldsymbol{\lambda}', \mathbf{w}'\rangle$.*

Proof Since $\boldsymbol{\lambda}' \in \partial f(\mathbf{w}')$, we know that $f(\mathbf{w}) - f(\mathbf{w}') \;\geq\; \langle \boldsymbol{\lambda}', \mathbf{w} - \mathbf{w}'\rangle$ for all $\mathbf{w} \in S$. Equivalently

$$\langle \boldsymbol{\lambda}', \mathbf{w}'\rangle - f(\mathbf{w}') \;\geq\; \sup_{\mathbf{w}\in S} \big(\langle \boldsymbol{\lambda}', \mathbf{w}\rangle - f(\mathbf{w})\big) \ .$$

The right-hand side of the above equals to $f^\star(\boldsymbol{\lambda}')$ and thus,

$$\langle \boldsymbol{\lambda}', \mathbf{w}'\rangle - f(\mathbf{w}') \;\geq\; f^\star(\boldsymbol{\lambda}') \quad \Rightarrow \quad \langle \boldsymbol{\lambda}', \mathbf{w}'\rangle - f^\star(\boldsymbol{\lambda}') \;\geq\; f(\mathbf{w}') \ . \tag{A.2}$$

The assumption that f is closed and convex implies that f is the Fenchel conjugate of $f^\star$. Thus,

$$f(\mathbf{w}') = \sup_{\boldsymbol{\lambda}} \big(\langle \boldsymbol{\lambda}, \mathbf{w}'\rangle - f^\star(\boldsymbol{\lambda})\big) \;\geq\; \langle \boldsymbol{\lambda}', \mathbf{w}'\rangle - f^\star(\boldsymbol{\lambda}') \ .$$

Combining the above with Eq. (A.2) gives,

$$\langle \boldsymbol{\lambda}', \mathbf{w}' \rangle - f^\star(\boldsymbol{\lambda}') \;\geq\; f(\mathbf{w}') \;\text{ and }\; f(\mathbf{w}') \;\geq\; \langle \boldsymbol{\lambda}', \mathbf{w}' \rangle - f^\star(\boldsymbol{\lambda}') \;.$$

Therefore, each of the two inequalities above must hold with equality which concludes the proof. ■

The next lemma shows that conjugate functions preserve order.

Lemma 12 *Let f_1, f_2 be two functions and assume that for all $\mathbf{w} \in S$, $f_1(\mathbf{w}) \leq f_2(\mathbf{w})$. Then, for all $\boldsymbol{\theta}$ we have $f_1^\star(\boldsymbol{\theta}) \geq f_2^\star(\boldsymbol{\theta})$.*

Proof

$$f_2^\star(\boldsymbol{\theta}) \;=\; \sup_{\mathbf{w}} \langle \mathbf{w}, \boldsymbol{\theta} \rangle - f_2(\mathbf{w}) \;\leq\; \sup_{\mathbf{w}} \langle \mathbf{w}, \boldsymbol{\theta} \rangle - f_1(\mathbf{w}) \;=\; f_1^\star(\boldsymbol{\theta}) \;.$$

■

A.3.1 Some Fenchel Conjugate Pairs

We now describe a few useful Fenchel-conjugate pairs.

Half Norm Squared: Let $\|\cdot\|$ be any norm on $\mathbb{R}^n$ and let $f(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2$ with $S = \mathbb{R}^n$. Then $f^\star(\boldsymbol{\theta}) = \frac{1}{2}\|\boldsymbol{\theta}\|_\star^2$ where $\|\cdot\|_\star$ is the dual norm of $\|\cdot\|$. The domain of $f^\star$ is also $\mathbb{R}^n$. For example, if $f(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|_2^2$ then $f^\star(\boldsymbol{\theta}) = \frac{1}{2}\|\boldsymbol{\theta}\|_2^2$ since the ℓ_2 norm is dual to itself. For a proof see pp. 93-94 in [14].

Hinge-loss: Let $f(\mathbf{w}) = [\gamma - \langle \mathbf{w}, \mathbf{x} \rangle]_+$ where $\gamma \in \mathbb{R}_+$ and $\mathbf{x} \in \mathbb{R}^n$ with $S = \mathbb{R}^n$. Then, the conjugate of f is,

$$f^\star(\boldsymbol{\theta}) \;=\; \begin{cases} -\gamma\,\alpha & \text{if } \boldsymbol{\theta} \in \{-\alpha\,\mathbf{x} : \alpha \in [0,1]\} \\ \infty & \text{otherwise} \end{cases} \;.$$

The above is a special case of the max-of-hinge function below.

Max-of-hinge: Let $f(\mathbf{w}) = \max_{i \in [k]} [b_i - \langle \mathbf{w}, \mathbf{x}_i \rangle]_+$ where for all $i \in [k]$, $b_i \in \mathbb{R}_+$ and $\mathbf{x}_i \in \mathbb{R}^n$. The conjugate of f is,

$$f^\star(\boldsymbol{\theta}) \;=\; \begin{cases} -\sum_{i=1}^k \alpha_i b_i & \text{if } \;\boldsymbol{\theta} \in \left\{-\sum_{i=1}^k \alpha_i \mathbf{x}_i \;:\; \boldsymbol{\alpha} \in \mathbb{R}_+^k \text{ and } \|\boldsymbol{\alpha}\|_1 \leq 1\right\} \\ \infty & \text{otherwise} \end{cases} \tag{A.3}$$

To show the above, we rewrite $f(\mathbf{w})$ as follows:

$$\begin{aligned} f(\mathbf{w}) &= \min_{\xi \geq 0} \left(\xi + \sum_{i=1}^{k} \max_{\alpha_i \geq 0} \alpha_i (b_i - \langle \mathbf{w}, \mathbf{x}_i \rangle - \xi) \right) \\ &= \min_{\xi \geq 0} \max_{\boldsymbol{\alpha} \in \mathbb{R}_+^k} \xi (1 - \sum_i \alpha_i) + \sum_i \alpha_i (b_i - \langle \mathbf{w}, \mathbf{x}_i \rangle) \\ &= \max_{\boldsymbol{\alpha} \in \mathbb{R}_+^k} \min_{\xi \geq 0} \xi (1 - \sum_i \alpha_i) + \sum_i \alpha_i (b_i - \langle \mathbf{w}, \mathbf{x}_i \rangle) \\ &= \max_{\boldsymbol{\alpha} \in \mathbb{R}_+^k : \|\boldsymbol{\alpha}\|_1 \leq 1} \sum_i \alpha_i (b_i - \langle \mathbf{w}, \mathbf{x}_i \rangle) \ . \end{aligned}$$

The third equality above follows from the strong max-min property that clearly holds for bilinear functions (see [14] page 115). Therefore, using the definition of $f^\star(\boldsymbol{\theta})$ we get that

$$\begin{aligned} f^\star(\boldsymbol{\theta}) &= \max_{\mathbf{w}} \ (\langle \boldsymbol{\theta}, \mathbf{w} \rangle - f(\mathbf{w})) \\ &= \max_{\mathbf{w}} \left(\langle \boldsymbol{\theta}, \mathbf{w} \rangle - \max_{\boldsymbol{\alpha} \in \mathbb{R}_+^k : \|\boldsymbol{\alpha}\|_1 \leq 1} \sum_i \alpha_i (b_i - \langle \mathbf{w}, \mathbf{x}_i \rangle) \right) \\ &= \max_{\mathbf{w}} \min_{\boldsymbol{\alpha} \in \mathbb{R}_+^k : \|\boldsymbol{\alpha}\|_1 \leq 1} \left(- \sum_i \alpha_i b_i + \langle \mathbf{w}, \boldsymbol{\theta} + \sum_i \alpha_i \mathbf{x}_i \rangle \right) \\ &= \min_{\boldsymbol{\alpha} \in \mathbb{R}_+^k : \|\boldsymbol{\alpha}\|_1 \leq 1} \max_{\mathbf{w}} \left(- \sum_i \alpha_i b_i + \langle \mathbf{w}, \boldsymbol{\theta} + \sum_i \alpha_i \mathbf{x}_i \rangle \right) \\ &= \min_{\boldsymbol{\alpha} \in \mathbb{R}_+^k : \|\boldsymbol{\alpha}\|_1 \leq 1} \left(- \sum_i \alpha_i b_i + \max_{\mathbf{w}} \langle \mathbf{w}, \boldsymbol{\theta} + \sum_i \alpha_i \mathbf{x}_i \rangle \right) \ . \end{aligned}$$

The fourth equality above again follows from the strong max-min property. Finally, we note that for any $\boldsymbol{\alpha}$, the value of $\max_{\mathbf{w}} \langle \mathbf{w}, \boldsymbol{\theta} + \sum_i \alpha_i \mathbf{x}_i \rangle$ is zero if $\boldsymbol{\theta} + \sum_i \alpha_i \mathbf{x}_i = \mathbf{0}$ and otherwise the value is ∞. This fact clearly implies that Eq. (A.3) holds.

Relative Entropy and Logarithm of Sum of Exponentials: Let $S = \{\mathbf{w} \in \mathbb{R}_+^n : \sum_{i=1}^n w_i = 1\}$ and let

$$f(\mathbf{w}) = \sum_{i=1}^{n} w_i \log \left(\frac{w_i}{1/n} \right) \ .$$

Then, the Fenchel conjugate of f is,

$$f^\star(\boldsymbol{\theta}) = \log \left(\frac{1}{n} \sum_{i=1}^{n} \exp(\boldsymbol{\theta}_i) \right) \ . \tag{A.4}$$

For a proof see p. 93 in [14].

Binary-entropy and Logistic-loss: Let $S = \{\mathbf{w} \in \mathbb{R}^n : \forall i \in [n], w_i \in [0,1]\}$ and let

$$f(\mathbf{w}) \;=\; -\sum_{i=1}^{n} \left(w_i \log(w_i) + (1-w_i)\log(1-w_i)\right) \;\; .$$

Then, the Fenchel conjugate of f is,

$$f^\star(\boldsymbol{\theta}) \;=\; \sum_{i=1}^{n} \log(1+e^{\theta_i}) \;\; . \tag{A.5}$$

The above Fenchel-pair correspondence can be shown by using the fact that for differentiable functions we have that $\nabla f(\mathbf{w})$ is the inverse of the function $\nabla f^\star(\boldsymbol{\theta})$.

Effect of Scaling and Shifting: We conclude this section with the following useful property of conjugate pairs. Let f be a function and let $f^\star$ be its Fenchel conjugate. For $a > 0$ and $b \in \mathbb{R}$, the Fenchel conjugate of $g(\mathbf{w}) = af(\mathbf{w}) + b$ is $g^\star(\boldsymbol{\theta}) = af^\star(\boldsymbol{\theta}/a) - b$. For a proof see page 95 in [14].

A.4 Strongly Convex Functions

Definition 4 *A continuous function f is strongly convex over a convex set S with respect to a norm $\|\cdot\|$ if S is contained in the domain of f and for all $\mathbf{v},\mathbf{u} \in S$ and $\alpha \in [0,1]$ we have*

$$f(\alpha\,\mathbf{v} + (1-\alpha)\,\mathbf{u}) \;\le\; \alpha\, f(\mathbf{v}) + (1-\alpha)\, f(\mathbf{u}) - \frac{1}{2}\,\alpha\,(1-\alpha)\,\|\mathbf{v}-\mathbf{u}\|^2 \;\; . \tag{A.6}$$

The following lemma gives an alternative definition for strong convexity.

Lemma 13 *f is strongly convex iff for all $\mathbf{u},\mathbf{v} \in S$ we have*

$$\forall \boldsymbol{\theta} \in \partial f(\mathbf{u}),\;\; \langle \boldsymbol{\theta}, \mathbf{v}-\mathbf{u}\rangle \le f(\mathbf{v}) - f(\mathbf{u}) - \frac{1}{2}\|\mathbf{v}-\mathbf{u}\|^2 \;\; . \tag{A.7}$$

Proof Assume f satisfies Eq. (A.6). Rearranging, we get,

$$\frac{f(\mathbf{u}+\alpha(\mathbf{v}-\mathbf{u})) - f(\mathbf{u})}{\alpha} \le f(\mathbf{v}) - f(\mathbf{u}) - \frac{1}{2}(1-\alpha)\|\mathbf{v}-\mathbf{u}\|^2 \, .$$

Taking $\alpha \to 0$ we obtain,

$$f'(\mathbf{u};\mathbf{v}-\mathbf{u}) \le f(\mathbf{v}) - f(\mathbf{u}) - \frac{1}{2}\|\mathbf{v}-\mathbf{u}\|^2 \, ,$$

where $f'(\mathbf{u};\cdot)$ is the directional derivative of f at $\mathbf{u}$ (see [11], page 15). Next, using Proposition 3.1.6 in [11] we obtain that

$$\forall \boldsymbol{\theta} \in \partial f(\mathbf{u}),\ \ \langle \boldsymbol{\theta}, \mathbf{v}-\mathbf{u}\rangle \le f'(\mathbf{u};\mathbf{v}-\mathbf{u})\ ,$$

and therefore Eq. (A.7) holds. Now assume that Eq. (A.7) holds. Take two points $\mathbf{u}_1$ and $\mathbf{u}_2$ and set $\mathbf{v} = \alpha\mathbf{u}_1 + (1-\alpha)\mathbf{u}_2$. Take some $\boldsymbol{\theta} \in \partial f(\mathbf{v})$. We have,

$$\begin{aligned}\langle \boldsymbol{\theta}, \mathbf{u}_1 - \mathbf{v}\rangle &\le f(\mathbf{u}_1) - f(\mathbf{v}) - \frac{1}{2}\|\mathbf{v}-\mathbf{u}_1\|^2 \\ \langle \boldsymbol{\theta}, \mathbf{u}_2 - \mathbf{v}\rangle &\le f(\mathbf{u}_2) - f(\mathbf{v}) - \frac{1}{2}\|\mathbf{v}-\mathbf{u}_2\|^2\ .\end{aligned}$$

Summing the above two equations with coefficients α and $(1-\alpha)$ we obtain

$$0 \le \alpha f(\mathbf{u}_1) + (1-\alpha) f(\mathbf{u}_2) - f(\mathbf{v}) - \frac{1}{2}\left(\alpha\|\mathbf{v}-\mathbf{u}_1\|^2 + (1-\alpha)\|\mathbf{v}-\mathbf{u}_2\|^2\right)$$

Using the definition of $\mathbf{v}$ and rearranging terms we obtain

$$f(\alpha\mathbf{u}_1 + (1-\alpha)\mathbf{u}_2) \le \alpha f(\mathbf{u}_1) + (1-\alpha) f(\mathbf{u}_2) - \frac{1}{2}\alpha(1-\alpha)\|\mathbf{u}_1-\mathbf{u}_2\|^2\ ,$$

and therefore Eq. (A.6) holds. ■

The following lemma gives useful tools for proving strong convexity.

Lemma 14 *Assume that f is differentiable. Then f is strongly convex iff*

$$\langle \nabla f(\mathbf{u}) - \nabla f(\mathbf{v}), \mathbf{u}-\mathbf{v}\rangle \ge \|\mathbf{u}-\mathbf{v}\|^2\ . \tag{A.8}$$

Additionally, if f is twice differentiable then a sufficient condition for strong convexity of f is

$$\forall \mathbf{w},\mathbf{x},\ \ \langle \nabla^2 f(\mathbf{w})\,\mathbf{x}\,,\,\mathbf{x}\rangle \ge \|\mathbf{x}\|^2\ . \tag{A.9}$$

Proof Assume f is strongly convex. Then, from Lemma 13 we have

$$\begin{aligned}\langle \nabla f(\mathbf{u}), \mathbf{v}-\mathbf{u}\rangle &\le f(\mathbf{u}) - f(\mathbf{v}) - \frac{1}{2}\|\mathbf{u}-\mathbf{v}\|^2 \\ \langle \nabla f(\mathbf{v}), \mathbf{u}-\mathbf{v}\rangle &\le f(\mathbf{v}) - f(\mathbf{u}) - \frac{1}{2}\|\mathbf{v}-\mathbf{u}\|^2\end{aligned}$$

Adding these two inequalities we obtain Eq. (A.8). Assume now that Eq. (A.8) holds. Define $h(\alpha) = f(\mathbf{v} + \alpha(\mathbf{u} - \mathbf{v}))$, and denote $\mathbf{w} = \mathbf{v} + \alpha(\mathbf{u} - \mathbf{v})$. Then,

$$h'(\alpha) - h'(0) = \langle \nabla f(\mathbf{w}) - \nabla f(\mathbf{v}), \mathbf{u} - \mathbf{v} \rangle = \frac{1}{\alpha} \langle \nabla f(\mathbf{w}) - \nabla f(\mathbf{v}), \mathbf{w} - \mathbf{v} \rangle \, .$$

Using Eq. (A.8) we obtain that

$$h'(\alpha) - h'(0) \geq \frac{1}{\alpha} \|\mathbf{w} - \mathbf{v}\|^2 = \alpha \|\mathbf{u} - \mathbf{v}\|^2 \, .$$

Therefore,

$$f(\mathbf{u}) - f(\mathbf{v}) - \langle \nabla f(\mathbf{v}), \mathbf{u} - \mathbf{v} \rangle = h(1) - h(0) - h'(0) = \int_0^1 (h'(\alpha) - h'(0)) d\alpha \geq \frac{1}{2} \|\mathbf{u} - \mathbf{v}\|^2 \, ,$$

and thus f is strongly convex.

Finally assume that f is twice differentiable and Eq. (A.9) holds. Then,

$$h''(\alpha) \;=\; \langle \nabla^2 f(\mathbf{v} + \alpha(\mathbf{u} - \mathbf{v}))\,(\mathbf{u} - \mathbf{v})\,,\,(\mathbf{u} - \mathbf{v}) \rangle \;\geq\; \|\mathbf{u} - \mathbf{v}\|^2 \;\; .$$

Since there exists $\phi \in [0, 1]$ such that $h(1) = h(0) + h'(0) + h''(\phi)/2$ we obtain that

$$f(\mathbf{u}) = h(1) \;=\; h(0) + h'(0) + \frac{1}{2} h''(\phi) \;\geq\; f(\mathbf{v}) + \langle \nabla f(\mathbf{v}), \mathbf{u} - \mathbf{v} \rangle + \frac{1}{2} \|\mathbf{u} - \mathbf{v}\|^2 \;\; ,$$

which implies that f is strongly convex (based on Lemma 13). ■

Strongly convex functions play an important role in our analysis primarily due to the following lemma.

Lemma 15 *Let f be a closed and strongly convex function over S with respect to a norm $\|\cdot\|$. Let $f^\star$ be the Fenchel conjugate of f. Then,*

1. *$f^\star$ is differentiable*
2. $\nabla f^\star(\boldsymbol{\theta}) = \arg\max_{\mathbf{w} \in S} \; \langle \mathbf{w}, \boldsymbol{\theta} \rangle - f(\mathbf{w})$
3. $\forall \boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \quad \|\nabla f^\star(\boldsymbol{\theta}_1) - \nabla f^\star(\boldsymbol{\theta}_2)\| \leq \|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|_\star$ *, where $\|\cdot\|_\star$ is the norm dual to $\|\cdot\|$*
4. $\forall\, \boldsymbol{\theta}, \boldsymbol{\lambda} \in \mathbb{R}^n, \quad f^\star(\boldsymbol{\theta} + \boldsymbol{\lambda}) - f^\star(\boldsymbol{\theta}) \;\leq\; \langle \nabla f^\star(\boldsymbol{\theta}), \boldsymbol{\lambda} \rangle + \frac{1}{2} \|\boldsymbol{\lambda}\|_\star^2.$
5. *If f is differentiable then $\forall \boldsymbol{\theta} \in \mathbb{R}^n$, $\forall \mathbf{u} \in S$, $\langle \mathbf{u} - \nabla f^\star(\boldsymbol{\theta}), \boldsymbol{\theta} - \nabla f(\nabla f^\star(\boldsymbol{\theta})) \rangle \leq 0$*

Proof

1-2. Since f is strongly convex the maximizer of $\max_{\mathbf{w}\in S}\langle \mathbf{w},\boldsymbol{\theta}\rangle - f(\mathbf{w})$ exists and is unique (see [11] page 19). Denote it by $\pi(\boldsymbol{\theta})$. Since $f^\star$ is a convex function, to prove the lemma it suffices to show that $\partial f^\star(\boldsymbol{\theta}) = \{\pi(\boldsymbol{\theta})\}$. From the definition of $\pi(\boldsymbol{\theta})$ we clearly have that

$$\forall \mathbf{u}\in S,\;\; \langle \pi(\boldsymbol{\theta}),\boldsymbol{\theta}\rangle - f(\pi(\boldsymbol{\theta})) \ge \langle \mathbf{u},\boldsymbol{\theta}\rangle - f(\mathbf{u})\ ,$$

and thus $\boldsymbol{\theta}\in\partial f(\pi(\boldsymbol{\theta}))$. Therefore, using Lemma 11 we obtain that

$$\langle \pi(\boldsymbol{\theta}),\boldsymbol{\theta}\rangle = f(\pi(\boldsymbol{\theta})) + f^\star(\boldsymbol{\theta})\ . \tag{A.10}$$

Let $\boldsymbol{\lambda}$ be an arbitrary vector and to simplify our notation denote $\mathbf{w} = \pi(\boldsymbol{\theta})$ and $\mathbf{u} = \pi(\boldsymbol{\lambda})$. Based on Eq. (A.10) we have that,

$$\begin{aligned} f^\star(\boldsymbol{\lambda}) - f^\star(\boldsymbol{\theta}) &= \langle \mathbf{u},\boldsymbol{\lambda}\rangle - f(\mathbf{u}) - \langle \mathbf{w},\boldsymbol{\theta}\rangle + f(\mathbf{w}) \\ &= f(\mathbf{w}) - f(\mathbf{u}) - \langle \mathbf{w}-\mathbf{u},\boldsymbol{\lambda}\rangle + \langle \mathbf{w},\boldsymbol{\lambda}-\boldsymbol{\theta}\rangle \\ &\ge \langle \mathbf{w},\boldsymbol{\lambda}-\boldsymbol{\theta}\rangle\ , \end{aligned}$$

which implies that $\mathbf{w}\in\partial f^\star(\boldsymbol{\theta})$. Finally, we show that $\mathbf{w}$ is the only element in $\partial f^\star(\boldsymbol{\theta})$ and thus $\mathbf{w} = \nabla f^\star(\boldsymbol{\theta})$. Let $\mathbf{w}_0\in\partial f^\star(\boldsymbol{\theta})$. Thus $f^\star(\boldsymbol{\theta}) = \langle \mathbf{w}_0,\boldsymbol{\theta}\rangle - f(\mathbf{w}_0) = \langle \mathbf{w},\boldsymbol{\theta}\rangle - f(\mathbf{w})$. But the uniqueness of the solution of $\max_{\mathbf{w}\in S}\langle \mathbf{w},\boldsymbol{\theta}\rangle - f(\mathbf{w})$ implies that $\mathbf{w}_0 = \mathbf{w}$.

3. Let $\boldsymbol{\theta}_1,\boldsymbol{\theta}_2$ be two vectors and let $\alpha\in(0,1)$. Denote $\mathbf{w}_1 = \pi(\boldsymbol{\theta}_1)$, $\mathbf{w}_2 = \pi(\boldsymbol{\theta}_2)$, and $\mathbf{w}_\alpha = \alpha\mathbf{w}_1 + (1-\alpha)\mathbf{w}_2$. As we have shown before, $\boldsymbol{\theta}_1\in\partial f(\mathbf{w}_1)$ and $\boldsymbol{\theta}_2\in\partial f(\mathbf{w}_2)$. Therefore, using Lemma 13 we have

$$\begin{aligned} f(\mathbf{w}_\alpha) - f(\mathbf{w}_1) - \langle \boldsymbol{\theta}_1,\mathbf{w}_\alpha-\mathbf{w}_1\rangle &\ge \frac{1}{2}\|\mathbf{w}_\alpha-\mathbf{w}_1\|^2 \\ f(\mathbf{w}_\alpha) - f(\mathbf{w}_2) - \langle \boldsymbol{\theta}_2,\mathbf{w}_\alpha-\mathbf{w}_2\rangle &\ge \frac{1}{2}\|\mathbf{w}_\alpha-\mathbf{w}_2\|^2\ . \end{aligned}$$

Summing these two inequalities with coefficients α and $(1-\alpha)$, using the definition of $\mathbf{w}_\alpha$, and rearranging terms we obtain

$$\begin{aligned} f(\mathbf{w}_\alpha) &- (\alpha f(\mathbf{w}_1) + (1-\alpha)f(\mathbf{w}_2)) + \alpha(1-\alpha)\langle \boldsymbol{\theta}_2-\boldsymbol{\theta}_1,\mathbf{w}_2-\mathbf{w}_1\rangle \\ &\ge\ \alpha(1-\alpha)\,\frac{1}{2}\|\mathbf{w}_1-\mathbf{w}_2\|^2\ . \end{aligned} \tag{A.11}$$

Since f is strongly convex we have that $f(\mathbf{w}_\alpha) - (\alpha f(\mathbf{w}_1) + (1-\alpha)f(\mathbf{w}_2)) \ge \alpha(1-\alpha)\frac{1}{2}\|\mathbf{w}_1 - \mathbf{w}_2\|^2$. In addition, using Holder inequality we get that $\langle \boldsymbol{\theta}_2-\boldsymbol{\theta}_1,\mathbf{w}_2-\mathbf{w}_1\rangle \le \|\boldsymbol{\theta}_2-\boldsymbol{\theta}_1\|_\star\,\|\mathbf{w}_2-\mathbf{w}_1\|_\star$.

Combining the above two inequalities with Eq. (A.11) and rearranging terms yield that

$$\|\mathbf{w}_1 - \mathbf{w}_2\| \le \|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|_\star \ .$$

4. Let T be an arbitrary positive integer and for all $t \in \{0, 1, \ldots, T\}$ define $\alpha_t = t/T$. We have:

$$\begin{aligned} f^\star(\boldsymbol{\theta} + \boldsymbol{\lambda}) - f^\star(\boldsymbol{\theta}) &= f^\star(\boldsymbol{\theta} + \alpha_T\, \boldsymbol{\lambda}) - f^\star(\boldsymbol{\theta} + \alpha_0\, \boldsymbol{\lambda}) \\ &= \sum_{t=0}^{T-1} \left(f^\star(\boldsymbol{\theta} + \alpha_{t+1}\, \boldsymbol{\lambda}) - f^\star(\boldsymbol{\theta} + \alpha_t\, \boldsymbol{\lambda}) \right) \end{aligned} \tag{A.12}$$

Since $f^\star$ is convex we have that for all t,

$$\begin{aligned} f^\star(\boldsymbol{\theta} + \alpha_{t+1}\, \boldsymbol{\lambda}) - f^\star(\boldsymbol{\theta} + \alpha_t\, \boldsymbol{\lambda}) &\le \langle \nabla f^\star(\boldsymbol{\theta} + \alpha_{t+1}\, \boldsymbol{\lambda}), (\alpha_{t+1} - \alpha_t)\, \boldsymbol{\lambda} \rangle \\ &= \frac{1}{T} \langle f^\star(\boldsymbol{\theta} + \alpha_{t+1}\, \boldsymbol{\lambda}), \boldsymbol{\lambda} \rangle \ . \end{aligned} \tag{A.13}$$

In addition, using Holder inequality and claim 3 of the lemma we get that

$$\begin{aligned} \langle \nabla f^\star(\boldsymbol{\theta} + \alpha_{t+1}\, \boldsymbol{\lambda}), \boldsymbol{\lambda} \rangle &= \langle \nabla f^\star(\boldsymbol{\theta}), \boldsymbol{\lambda} \rangle + \langle \nabla f^\star(\boldsymbol{\theta} + \alpha_{t+1} \boldsymbol{\lambda}) - \nabla f^\star(\boldsymbol{\theta}), \boldsymbol{\lambda} \rangle \\ &\le \langle \nabla f^\star(\boldsymbol{\theta}), \boldsymbol{\lambda} \rangle + \|\nabla f^\star(\boldsymbol{\theta} + \alpha_{t+1} \boldsymbol{\lambda}) - \nabla f^\star(\boldsymbol{\theta})\| \, \|\boldsymbol{\lambda}\|_\star \\ &\le \langle \nabla f^\star(\boldsymbol{\theta}), \boldsymbol{\lambda} \rangle + \|\alpha_{t+1}\, \boldsymbol{\lambda}\|_\star \, \|\boldsymbol{\lambda}\|_\star \\ &= \langle \nabla f^\star(\boldsymbol{\theta}), \boldsymbol{\lambda} \rangle + \alpha_{t+1} \|\boldsymbol{\lambda}\|_\star^2 \ . \end{aligned}$$

Combining the above with Eq. (A.13) and Eq. (A.12) we obtain

$$\begin{aligned} f^\star(\boldsymbol{\theta} + \boldsymbol{\lambda}) - f^\star(\boldsymbol{\theta}) &\le \langle \nabla f^\star(\boldsymbol{\theta}), \boldsymbol{\lambda} \rangle + \|\boldsymbol{\lambda}\|_\star^2 \, \frac{1}{T} \sum_{t=0}^{T-1} \alpha_{t+1} \\ &= \langle \nabla f^\star(\boldsymbol{\theta}), \boldsymbol{\lambda} \rangle + \frac{T+1}{2\,T} \|\boldsymbol{\lambda}\|_\star^2 \ . \end{aligned}$$

Taking $T \to \infty$ concludes our proof.

5. Denote $\mathbf{v} = \nabla f^\star(\boldsymbol{\theta})$ and note that

$$\mathbf{v} = \operatorname*{argmax}_{\mathbf{w} \in S} \langle \mathbf{w}, \boldsymbol{\theta} \rangle - f(\mathbf{w}) \ .$$

Denote the objective of the maximization problem above by $P(\mathbf{w})$. The assumption that f is differentiable implies that P is also differentiable with $\nabla P(\mathbf{w}) = \boldsymbol{\theta} - \nabla f(\mathbf{w})$. Finally, the optimality of $\mathbf{v}$ implies that for all $\mathbf{u} \in S$ $\langle \mathbf{u} - \mathbf{v}, \nabla P(\mathbf{v}) \rangle \leq 0$, which concludes our proof. ■

Several notable examples of strongly convex functions which we use are as follows.

Example 3 *The function $f(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|_2^2$ is strongly convex over $S = \mathbb{R}^n$ with respect to the Euclidean norm. Its conjugate function is $f^\star(\boldsymbol{\theta}) = \frac{1}{2}\|\boldsymbol{\theta}\|_2^2$.*

Example 4 *The function $f(\mathbf{w}) = \sum_{i=1}^n w_i \log(w_i/\frac{1}{n})$ is strongly convex over the probabilistic simplex, $S = \{\mathbf{w} \in \mathbb{R}_+^n : \|\mathbf{w}\|_1 = 1\}$, with respect to the ℓ_1 norm (see Lemma 16 below). Its conjugate function is $f^\star(\boldsymbol{\theta}) = \log(\frac{1}{n}\sum_{i=1}^n \exp(\theta_i))$.*

Example 5 *For $q \in (1,2)$, the function $f(\mathbf{w}) = \frac{1}{2(q-1)}\|\mathbf{w}\|_q^2$ is strongly convex over $S = \mathbb{R}^n$ with respect to the ℓ_q norm (see Lemma 17 below). Its conjugate function is $f^\star(\boldsymbol{\theta}) = \frac{1}{2(p-1)}\|\boldsymbol{\theta}\|_p^2$, where $p = (1-1/q)^{-1}$.*

Lemma 16 *The function $f(\mathbf{w}) = \log(n) + \sum_{i=1}^n w_i \log(w_i)$ is strongly convex over the probabilistic simplex, $S = \{\mathbf{w} \in \mathbb{R}_+^n : \|\mathbf{w}\|_1 = 1\}$, with respect to the ℓ_1 norm.*

Proof Based on Lemma 14, it suffices to show that for all $\mathbf{u}, \mathbf{v} \in S$ we have

$$\langle \nabla f(\mathbf{u}) - \nabla f(\mathbf{v}), \mathbf{u} - \mathbf{v} \rangle \geq \|\mathbf{u} - \mathbf{v}\|^2 . \tag{A.14}$$

The i'th element of $\nabla f(\mathbf{u})$ is $\log(u_i) + 1$ and thus we need to show that

$$\sum_i (\log(u_i) - \log(v_i))(u_i - v_i) \geq \|\mathbf{u} - \mathbf{v}\|_1^2 .$$

Let $x_i = (\log(u_i) - \log(v_i))(u_i - v_i)$ and note that $x_i \geq 0$. Denote $I = \{i \in [n] : x_i > 0\}$. Based on the Cauchy-Schwartz inequality, we can bound the right-hand side of the above as follows

$$\begin{aligned}
\|\mathbf{u} - \mathbf{v}\|_1^2 = \left(\sum_i |u_i - v_i|\right)^2 &\leq \left(\sum_{i \in I} \sqrt{x_i}\frac{|u_i - v_i|}{\sqrt{x_i}}\right)^2 \\
&\leq \sum_{i \in I} x_i \sum_{i \in I} \frac{|u_i - v_i|^2}{x_i} \\
&= \sum_{i \in I} (\log(u_i) - \log(v_i))\,(u_i - v_i) \sum_{i \in I} \frac{u_i - v_i}{\log(u_i) - \log(v_i)} .
\end{aligned}$$

Therefore, to prove that Eq. (A.14) holds it is left to show that

$$\sum_{i \in I} \frac{u_i - v_i}{\log(u_i) - \log(v_i)} \leq 1. \tag{A.15}$$

To do so, we show in the sequel that for all i

$$\frac{u_i - v_i}{\log(u_i) - \log(v_i)} \le \frac{u_i + v_i}{2} \,, \tag{A.16}$$

which immediately implies that Eq. (A.15) holds. The left-hand side of the above is positive and thus we can assume w.l.g. that $u_i \ge v_i$. Fix v_i and consider the function

$$\phi(u) = 2(u - v_i) - (u + v_i)(\log(u) - \log(v_i)) \,.$$

Clearly, $\phi(v_i) = 0$. In addition, the derivative of ϕ is negative,

$$\phi'(u) = 2 - \log(u) - \frac{u + v_i}{u} + \log(v_i) = 1 + \log(\frac{v_i}{u}) - \frac{v_i}{u} \le 0 \,,$$

where in the last inequality we used $\log(a) \le a - 1$. Thus, ϕ monotonically decreases and thus $\phi(u) \le 0$ in $(v_i, 1]$. Therefore, Eq. (A.16) holds and our proof is concluded. ■

Lemma 17 *For $q \in (1, 2)$, the function $f(\mathbf{w}) = \frac{1}{2(q-1)}\|\mathbf{w}\|_q^2$ is strongly convex over $\mathbb{R}^n$ with respect to the ℓ_q norm.*

Proof f is twice differentiable and therefore, based on Lemma 14, it suffices to show that for any $\mathbf{w}$ and $\mathbf{x}$ we have

$$\langle \nabla^2 f(\mathbf{w})\,\mathbf{x}\,,\ \mathbf{x}\rangle \ \ge\ \|\mathbf{x}\|_q^2 \ .$$

Let us rewrite $f(\mathbf{w}) = \Psi(\sum_i \phi(w_i))$ where $\Psi(a) = \frac{a^{2/q}}{2(q-1)}$ and $\phi(a) = |a|^q$. Note that the first and second derivatives of Ψ are

$$\Psi'(a) = \frac{1}{q\,(q-1)} a^{\frac{2}{q}-1} \quad , \quad \Psi''(a) = \frac{1}{q\,(q-1)} \left(\frac{2}{q} - 1\right) a^{\frac{2}{q}-2} \ .$$

Similarly, the first and second derivatives of ϕ are

$$\phi'(a) = q\,\mathrm{sign}(a)\,|a|^{q-1} \quad , \quad \phi''(a) = q\,(q-1)\,|a|^{q-2} \ .$$

Denote $H = \nabla^2 f(\mathbf{w})$. The value of the element (i, j) of H for $i \ne j$ is,

$$H_{i,j}(\mathbf{w}) \ = \ \Psi''\left(\sum_{r=1}^{n} \phi(w_r)\right) \phi'(w_i)\phi'(w_j) \ ,$$

and the i'th diagonal element of H is,

$$H_{i,i}(\mathbf{w}) \;=\; \Psi''\left(\sum_{r=1}^{n}\phi(w_r)\right)\left(\phi'(w_i)\right)^2 + \Psi'\left(\sum_{r=1}^{n}\phi(w_r)\right)\phi''(w_i) \;\; .$$

Therefore,

$$\langle H\,\mathbf{x},\mathbf{x}\rangle \;=\; \Psi''\left(\sum_{r=1}^{n}\phi(w_r)\right)\left(\sum_i \phi'(w_i)x_i\right)^2 + \Psi'\left(\sum_{r=1}^{n}\phi(w_r)\right)\sum_i \phi''(w_i)x_i^2 \;\; .$$

The first term above is non-negative since we assume that $q \in (1,2)$. Unraveling the second term we obtain that,

$$\langle H\,\mathbf{x},\mathbf{x}\rangle \;\geq\; \frac{\|\mathbf{w}\|_q^{q(\frac{2}{q}-1)}}{q(q-1)}\sum_i q(q-1)|w_i|^{q-2}x_i^2 \;=\; \|\mathbf{w}\|_q^{2-q}\sum_i |w_i|^{q-2}\,x_i^2 \;\; . \tag{A.17}$$

We next show that the right-hand side of the above upper bounds $\|\mathbf{x}\|_q^2$. Denote $y_i = |w_i|^{(2-q)\frac{q}{2}}$. Using Holder inequality with the dual norms $\frac{2}{q}$ and $\frac{2}{2-q}$ we obtain that

$$\begin{aligned}\|\mathbf{x}\|_q^2 \;&=\; \left(\sum_i x_i^q\right)^{\frac{2}{q}} \;=\; \left(\sum_i y_i\frac{x_i^q}{y_i}\right)^{\frac{2}{q}} \;\leq\; \left(\left(\sum_i y_i^{\frac{2}{2-q}}\right)^{\frac{2-q}{2}}\left(\sum_i \frac{x_i^2}{y_i^{2/q}}\right)^{\frac{q}{2}}\right)^{\frac{2}{q}} \\ &=\; \left(\sum_i |w_i|^q\right)^{\frac{2-q}{q}}\left(\sum_i |w_i|^{q-2}\,x_i^2\right) \;=\; \|\mathbf{w}\|_q^{2-q}\sum_i |w_i|^{q-2}\,x_i^2 \,.\end{aligned}$$

Comparing the above with Eq. (A.17) we get that $\langle H\,\mathbf{x},\mathbf{x}\rangle \geq \|\mathbf{x}\|_q^2$, which concludes our proof. ■

We conclude this section with an extension of Lemma 18 to cases in which the same strongly convex function is concatenated.

Lemma 18 *Let f be a closed and strongly convex function over $S \subseteq \mathbb{R}^n$ with respect to a norm $\|\cdot\|$. Let $f^\star$ be the Fenchel conjugate of f. Let k be an integer and denote $\bar{S} = S^k$. Define a function $\bar{f} : \bar{S} \to \mathbb{R}$ such that for $\bar{\mathbf{w}} = (\mathbf{w}_1, \ldots, \mathbf{w}_k) \in \bar{S}^k$ we have*

$$\bar{f}(\bar{\mathbf{w}}) = \sum_{i=1}^{k} f(\mathbf{w}_i) \;\; .$$

Let $\bar{f}^\star$ be the Fenchel conjugate of $\bar{f}$. Then,

1. *$\bar{f}^\star$ is differentiable*
2. *If $\bar{\boldsymbol{\lambda}} = \nabla \bar{f}^\star(\bar{\boldsymbol{\theta}})$ then $\boldsymbol{\lambda}_i = \arg\max_{\mathbf{w}_i \in S} \ \langle \mathbf{w}_i, \boldsymbol{\theta}_i\rangle - f(\mathbf{w}_i)$*
3. *$\forall\, \bar{\boldsymbol{\theta}}, \bar{\boldsymbol{\lambda}} \in \mathbb{R}^n$, $\ \bar{f}^\star(\bar{\boldsymbol{\theta}}+\bar{\boldsymbol{\lambda}}) - \bar{f}^\star(\bar{\boldsymbol{\theta}}) \ \le \ \langle \nabla \bar{f}^\star(\bar{\boldsymbol{\theta}}), \bar{\boldsymbol{\lambda}}\rangle + \frac{1}{2}\sum_{i=1}^k \|\boldsymbol{\lambda}_i\|_\star^2$, where $\|\cdot\|_\star$ is the norm dual to $\|\cdot\|$*

Proof The definition of $\bar{f}$ implies that

$$\bar{f}^\star(\bar{\boldsymbol{\theta}}) = \max_{\bar{\mathbf{w}}} \langle \bar{\mathbf{w}}, \bar{\boldsymbol{\theta}}\rangle - \bar{f}(\bar{\mathbf{w}}) \ = \ \sum_{i=1}^k \ \max_{\mathbf{w}_i} \langle \mathbf{w}_i, \boldsymbol{\theta}_i\rangle - f(\mathbf{w}_i) \ = \ \sum_{i=1}^k f^\star(\boldsymbol{\theta}_i) \ .$$

The claims now follow directly from the fact that f is strongly convex. ■

A.5 Technical lemmas

Lemma 19 *Let $x, b, c \in \mathbb{R}_+$ and assume that $x - b\sqrt{x} - c \le 0$. Then, $x \le c + b^2 + b\sqrt{c}$.*

Proof Denote $Q(y) = y^2 - b\,y - c$ and note that Q is a convex parabola. Thus, $Q(\sqrt{x}) \le 0$ whenever $\sqrt{x}$ is between the two roots of $Q(y)$,

$$r_{1,2} = \frac{1}{2}b \pm \sqrt{(\frac{1}{2}b)^2 + c} \ .$$

In particular, $\sqrt{x}$ is smaller than the larger root of $Q(y)$, and thus

$$\sqrt{x} \le \frac{1}{2}b + \sqrt{(\frac{1}{2}b)^2 + c} \ .$$

Since both sides of the above are non-negative we obtain that

$$x \le \left(\frac{1}{2}b + \sqrt{(\frac{1}{2}b)^2 + c}\right)^2 = \frac{1}{2}b^2 + c + b\sqrt{(\frac{1}{2}b)^2 + c} \le c + b^2 + b\sqrt{c}\,.$$

■

Lemma 20 *Let k be an integer, $Y \subset [k]$, and $\mathbf{x} \in \mathbb{R}^n$. Let $\bar{\mathbf{w}} = (\mathbf{w}_1, \ldots, \mathbf{w}_k)$ where for all $i \in [k]$, $\mathbf{w}_i \in \mathbb{R}^n$. Let $\gamma > 0$ and define*

$$g(\bar{\mathbf{w}}) = \max_{r \in Y, s \notin Y} [\gamma - \langle \mathbf{w}_r - \mathbf{w}_s, \mathbf{x} \rangle]_+ \quad .$$

Define $A = \{\boldsymbol{\alpha} \in \mathbb{R}_+^k \ : \ \|\boldsymbol{\alpha}\|_1 \le 1 \ \text{and} \ \sum_{r \in Y} \alpha_r = \sum_{s \notin Y} \alpha_s\}$. Then, for any $\bar{\boldsymbol{\lambda}} = (\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_k)$ we have

$$g^\star(\boldsymbol{\theta}) \ = \ \begin{cases} -\frac{\gamma}{2} \sum_{r \in Y} \alpha_r & \text{if } \exists \boldsymbol{\alpha} \in A, \ \text{s.t. } \forall r \in Y, \ \boldsymbol{\lambda}_r = -\alpha_r \mathbf{x}, \ \forall s \notin Y, \ \boldsymbol{\lambda}_s = \alpha_s \mathbf{x} \\ \infty & \text{otherwise} \end{cases} \tag{A.18}$$

Proof Denote $\bar{Y} = [k] \setminus Y$ and let $E = Y \times \bar{Y}$. We start be rewriting $g(\bar{\mathbf{w}})$ as the solution of a minimization problem:

$$g(\bar{\mathbf{w}}) \ = \ \min_{\xi \ge 0} \xi \quad \text{s.t. } \forall (r,s) \in E, \ \ \gamma - \langle \mathbf{w}_r - \mathbf{w}_s, \mathbf{x} \rangle \le \xi \ . \tag{A.19}$$

The core idea for deriving the conjugate function is to show that the optimization problem on the right-hand side of Eq. (A.19) is equivalent to the more compact optimization problem defined as follows,

$$\min_{b, \xi \ge 0} \xi \quad \text{s.t. } \forall r \in Y, \ \langle \mathbf{w}_r, \mathbf{x} \rangle \ge b + \gamma/2 - \xi/2 \ ; \ \forall s \in \bar{Y}, \ \langle \mathbf{w}_s, \mathbf{x} \rangle \le b - \gamma/2 + \xi/2 \ . \tag{A.20}$$

Since the objective function of Eq. (A.20) and Eq. (A.19) are identical and b has no effect on the objective function, but rather on the constraints, it suffices to show that for any feasible solution ξ of Eq. (A.20) there exists a feasible solution (b, ξ) of Eq. (A.19) and vice versa.

Let (b, ξ) be a feasible solution of Eq. (A.20). Then, for any pair of labels $r \in Y$ and $s \in \bar{Y}$ we get that,

$$\langle \mathbf{w}_r - \mathbf{w}_s, \mathbf{x} \rangle \ge \gamma/2 + b - \xi/2 - (b - \gamma/2 + \xi/2) = \gamma - \xi \ .$$

Therefore, ξ is a feasible solution of Eq. (A.19). Proving that if ξ is a feasible solution of Eq. (A.19) then there exists b such that (b, ξ) is a feasible solution of Eq. (A.20) is a bit more complex to show. We do so by first defining the following two variables

$$\begin{aligned} \bar{b} &= \min_{r \in Y} \langle \mathbf{w}_r, \mathbf{x} \rangle - \gamma/2 + \xi/2 \ ; \\ \underline{b} &= \max_{s \notin Y} \langle \mathbf{w}_s, \mathbf{x} \rangle + \gamma/2 - \xi/2 \ . \end{aligned} \tag{A.21}$$

Let j and l denote the indices of the labels which attain, respectively, the minimum and maximum of the problems defined by Eq. (A.21). Then, by construction we get that,

$$\bar{b} - \underline{b} = \langle \mathbf{w}_j - \mathbf{w}_l, \mathbf{x} \rangle - \gamma + \xi \geq 0 \ ,$$

where the last inequality is due to feasibility of the solution ξ with respect to the problem defined by Eq. (A.19). We now define $b = (\bar{b} + \underline{b})/2$ which immediately implies that $\underline{b} \leq b \leq \bar{b}$. We therefore get that for any label $r \in Y$ the following inequality hold,

$$\begin{aligned} \langle \mathbf{w}_r, \mathbf{x} \rangle \ \geq \ \langle \mathbf{w}_j, \mathbf{x} \rangle \ &\geq \ \bar{b} + \gamma/2 - \xi/2 \\ &\geq \ b + \gamma/2 - \xi/2 \ , \end{aligned}$$

and similarly for any label $s \in \bar{Y}$ we get,

$$\begin{aligned} \langle \mathbf{w}_s, \mathbf{x} \rangle \ \leq \ \langle \mathbf{w}_l, \mathbf{x} \rangle \ &\leq \ \underline{b} - \gamma/2 + \xi/2 \\ &\leq \ b - \gamma/2 + \xi/2 \ . \end{aligned}$$

We have thus established a feasible solution (b, ξ) as required.

Based on the equivalence between Eq. (A.19) and Eq. (A.20) we can rewrite $g(\mathbf{w})$ as follows:

$$\begin{aligned} g(\bar{\mathbf{w}}) &= \min_{b, \xi \geq 0} \max_{\boldsymbol{\alpha} \geq 0} \ \xi + \sum_{r \in Y} \alpha_r \left(b + \tfrac{\gamma}{2} - \tfrac{\xi}{2} - \langle \mathbf{w}_r, \mathbf{x} \rangle \right) + \sum_{s \in \bar{Y}} \alpha_s \left(\langle \mathbf{w}_s, \mathbf{x} \rangle - b + \tfrac{\gamma}{2} - \tfrac{\xi}{2} \right) \\ &= \max_{\boldsymbol{\alpha} \geq 0} \min_{b, \xi \geq 0} \ \xi + \sum_{r \in Y} \alpha_r \left(b + \tfrac{\gamma}{2} - \tfrac{\xi}{2} - \langle \mathbf{w}_r, \mathbf{x} \rangle \right) + \sum_{s \in \bar{Y}} \alpha_s \left(\langle \mathbf{w}_s, \mathbf{x} \rangle - b + \tfrac{\gamma}{2} - \tfrac{\xi}{2} \right) \\ &= \max_{\boldsymbol{\alpha} \geq 0} \Bigg(\sum_{r \in Y} \alpha_r \left(\tfrac{\gamma}{2} - \langle \mathbf{w}_r, \mathbf{x} \rangle \right) + \sum_{s \in \bar{Y}} \alpha_s \left(\tfrac{\gamma}{2} + \langle \mathbf{w}_s, \mathbf{x} \rangle \right) \\ &\qquad + \min_{\xi \geq 0} \xi \big(1 - \sum_i \alpha_i \big) + \min_b b \big(\sum_{r \in Y} \alpha_r - \sum_{s \in \bar{Y}} \alpha_s \big) \Bigg) \\ &= \max_{\boldsymbol{\alpha} \in A} \ \tfrac{\gamma}{2} \sum_{r \in Y} \alpha_r - \sum_{r \in Y} \alpha_r \langle \mathbf{w}_r, \mathbf{x} \rangle + \sum_{s \in \bar{Y}} \alpha_s \langle \mathbf{w}_s, \mathbf{x} \rangle \ , \end{aligned}$$

where $A = \{ \boldsymbol{\alpha} \in \mathbb{R}_+^k \ : \ \|\boldsymbol{\alpha}\|_1 \leq 1 \ \text{ and } \ \sum_{r \in Y} \alpha_r = \sum_{s \in \bar{Y}} \alpha_s \}$. The second equality above follows from the strong max-min property that clearly holds for bilinear functions (see [14] page

115). Therefore, using the definition of $g^\star(\bar{\boldsymbol{\lambda}})$ we get that

$$\begin{aligned}
g^\star(\bar{\boldsymbol{\lambda}}) &= \max_{\bar{\mathbf{w}}} \left(\langle \bar{\boldsymbol{\lambda}}, \bar{\mathbf{w}}\rangle - g(\bar{\mathbf{w}})\right) \\
&= \max_{\bar{\mathbf{w}}} \sum_i \langle \boldsymbol{\lambda}_i, \mathbf{w}_i\rangle - \max_{\boldsymbol{\alpha}\in A} \left(\tfrac{\gamma}{2}\sum_{r\in Y}\alpha_r - \sum_{r\in Y}\alpha_r\langle \mathbf{w}_r, \mathbf{x}\rangle + \sum_{s\in\bar{Y}}\alpha_s\langle \mathbf{w}_s, \mathbf{x}\rangle\right) \\
&= \max_{\bar{\mathbf{w}}} \min_{\boldsymbol{\alpha}\in A} \left(-\tfrac{\gamma}{2}\sum_i \alpha_i + \sum_{r\in Y}\langle \mathbf{w}_r, \boldsymbol{\lambda}_r + \alpha_r\mathbf{x}\rangle + \sum_{s\in\bar{Y}}\langle \mathbf{w}_s, \boldsymbol{\lambda}_s - \alpha_s\mathbf{x}\rangle\right) \\
&= \min_{\boldsymbol{\alpha}\in A} \max_{\bar{\mathbf{w}}} \left(-\tfrac{\gamma}{2}\sum_i \alpha_i + \sum_{r\in Y}\langle \mathbf{w}_r, \boldsymbol{\lambda}_r + \alpha_r\mathbf{x}\rangle + \sum_{s\in\bar{Y}}\langle \mathbf{w}_s, \boldsymbol{\lambda}_s - \alpha_s\mathbf{x}\rangle\right) \\
&= \min_{\boldsymbol{\alpha}\in A} -\tfrac{\gamma}{2}\sum_i \alpha_i + \max_{\bar{\mathbf{w}}} \sum_{r\in Y}\langle \mathbf{w}_r, \boldsymbol{\lambda}_r + \alpha_r\mathbf{x}\rangle + \sum_{s\in\bar{Y}}\langle \mathbf{w}_s, \boldsymbol{\lambda}_s - \alpha_s\mathbf{x}\rangle \ .
\end{aligned}$$

The fourth equality above again follows from the strong max-min property. Finally, we note that for any $\boldsymbol{\alpha}$, the maximum over $\bar{\mathbf{w}}$ equals to 0 if $\boldsymbol{\lambda}_r = -\alpha_r\mathbf{x}$ for $r \in Y$ and $\boldsymbol{\lambda}_s = \alpha_s\mathbf{x}$ for $s \in \bar{Y}$. Otherwise, the maximum is ∞. This concludes our proof. ■

Proof of Lemma 7

Throughout the proof we assume that the elements of the vector $\boldsymbol{\mu}$ are sorted in a non-ascending order, namely, $\mu_1 \geq \mu_2 \geq \ldots \geq \mu_p$. Recall that the definition of $\rho(z, \boldsymbol{\mu})$ is,

$$\rho(z,\boldsymbol{\mu}) \ = \ \max\left\{ j \in [p] \ : \ \mu_j - \frac{1}{j}\left(\sum_{r=1}^{j}\mu_r - z\right) > 0 \right\} \ .$$

For brevity, we refer to $\rho(z, \boldsymbol{\mu})$ simply as ρ. Denote by $\boldsymbol{\alpha}^\star$ the optimal solution of the constrained optimization problem of Eq. (7.1) and let

$$\rho^\star = \max\{j \ : \ \alpha^\star_j > 0\} \ .$$

From Eq. (7.3) we know that $\alpha^\star_r = \mu_r - \theta^\star > 0$ for $r \leq \rho^\star$ where

$$\theta^\star = \frac{1}{\rho^\star}\left(\sum_{j=1}^{\rho^\star}\mu_j - z\right) \ ,$$

and therefore $\rho \geq \rho^\star$. We thus need to prove that $\rho = \rho^\star$. Assume by contradiction that $\rho > \rho^\star$. Let us denote by $\boldsymbol{\alpha}$ the vector induced by the choice of ρ, that is, $\alpha_r = 0$ for $r > \rho$ and $\alpha_r = \mu_r - \theta$ for

$r \leq \rho$, where,

$$\theta = \frac{1}{\rho}\left(\sum_{j=1}^{\rho} \mu_j - z\right) .$$

From the definition of ρ, we must have that $\alpha_\rho = \mu_\rho - \theta > 0$. Therefore, since the elements of $\boldsymbol{\mu}$ are sorted in a non-ascending order, we get that $\alpha_r = \mu_r - \theta > 0$ for all $r \leq \rho$. In addition, the choice of θ implies that $\|\boldsymbol{\alpha}\|_1 = z$. We thus get that $\boldsymbol{\alpha}$ is a feasible solution as it satisfies the constraints of Eq. (7.1). Examining the objective function attained at $\boldsymbol{\alpha}$ we get that,

$$\begin{aligned}
\|\boldsymbol{\alpha} - \boldsymbol{\mu}\|^2 &= \sum_{r=1}^{\rho^\star} \theta^2 + \sum_{r=\rho^\star+1}^{\rho} \theta^2 + \sum_{r=\rho+1}^{p} \mu_r^2 \\
&< \sum_{r=1}^{\rho^\star} \theta^2 + \sum_{r=\rho^\star+1}^{\rho} \mu_r^2 + \sum_{r=\rho+1}^{p} \mu_r^2 \\
&= \sum_{r=1}^{\rho^\star} \theta^2 + \sum_{r=\rho^\star+1}^{p} \mu_r^2 ,
\end{aligned}$$

where to derive the inequality above we used the fact that $\mu_r - \theta > 0$ for all $r \leq \rho$. We now need to analyze two cases depending on whether $\theta^\star$ is greater than θ or not. If $\theta^\star \geq \theta$ than we can further bound $\|\boldsymbol{\alpha} - \boldsymbol{\mu}\|^2$ from above as follows,

$$\|\boldsymbol{\alpha} - \boldsymbol{\mu}\|^2 < \sum_{r=1}^{\rho^\star} \theta^2 + \sum_{r=\rho^\star+1}^{p} \mu_r^2 \leq \sum_{r=1}^{\rho^\star} (\theta^\star)^2 + \sum_{r=\rho^\star+1}^{p} \mu_r^2 = \|\boldsymbol{\alpha}^\star - \boldsymbol{\mu}\|^2 ,$$

which contradicts the optimality of $\boldsymbol{\alpha}^\star$. We are thus left to show that the case $\theta > \theta^\star$ also leads to a contradiction. We do so by constructing a vector $\tilde{\boldsymbol{\alpha}}$ from $\boldsymbol{\alpha}^\star$. We show that this vector satisfies the constraints of Eq. (7.1) hence it is a feasible solution. Finally, we show that the objective function attained by $\tilde{\boldsymbol{\alpha}}$ is strictly smaller than that of $\boldsymbol{\alpha}^\star$. We define the vector $\tilde{\boldsymbol{\alpha}} \in \mathbb{R}^k$ as follows,

$$\tilde{\alpha}_r = \begin{cases} \alpha^\star_{\rho^\star} - \epsilon & r = \rho^\star \\ \epsilon & r = \rho^\star + 1 \\ \alpha^\star_r & \text{otherwise} \end{cases} ,$$

where $\epsilon = \frac{1}{2}(\mu_{\rho^\star+1} - \theta^\star)$. Since we assume that $\theta > \theta^\star$ and $\rho > \rho^\star$ we know that $\alpha_{\rho^\star+1} = \mu_{\rho^\star+1} - \theta > 0$ which implies that

$$\tilde{\alpha}_{\rho^\star+1} = \frac{1}{2}(\mu_{\rho^\star+1} - \theta^\star) > \frac{1}{2}(\mu_{\rho^\star+1} - \theta) = \frac{1}{2}\alpha_{\rho^\star+1} > 0 .$$

Furthermore, we also get that,

$$\tilde{\alpha}_{\rho^\star} \;=\; \mu_{\rho^\star} - \frac{1}{2}\mu_{\rho^\star+1} - \frac{1}{2}\theta^\star \;>\; \frac{1}{2}\left(\mu_{\rho^\star+1} - \theta\right) \;=\; \frac{1}{2}\alpha_{\rho^\star+1} \;>\; 0 \;\; .$$

In addition, by construction we get that the rest of components of $\tilde{\boldsymbol{\alpha}}$ are also non-negative. Our construction also preserves the norm, that is $\|\tilde{\boldsymbol{\alpha}}\|_1 = \|\boldsymbol{\alpha}^\star\|_1 = z$. Thus, the vector $\tilde{\boldsymbol{\alpha}}$ is also a feasible solution for the set of constraints defined by Eq. (7.1). Alas, examining the difference in the objective functions attained by $\tilde{\boldsymbol{\alpha}}$ and $\boldsymbol{\alpha}^\star$ we get,

$$\begin{aligned}
\|\boldsymbol{\alpha}^\star - \boldsymbol{\mu}\|^2 - \|\tilde{\boldsymbol{\alpha}} - \boldsymbol{\mu}\|^2 \;&=\; (\theta^\star)^2 + \mu_{\rho^\star+1}^2 - \Big((\theta^\star + \epsilon)^2 + (\mu_{\rho^\star+1} - \epsilon)^2\Big) \\
&=\; 2\epsilon\underbrace{(\mu_{\rho^\star+1} - \theta^\star)}_{=2\epsilon} - 2\epsilon^2 \;=\; 2\,\epsilon^2 \;>\; 0 \;\; .
\end{aligned}$$

We thus obtained the long desired contradiction which concludes the proof. ■

Proof of Lemma 10

Plugging the value of the optimal solution $\boldsymbol{\alpha}$ from Eq. (7.23) into the objective $\|\boldsymbol{\alpha} - \boldsymbol{\mu}\|^2$ and using Lemma 7 give that,

$$g(z;\boldsymbol{\mu}) \;=\; \frac{1}{\rho(z;\boldsymbol{\mu})}\left(\sum_{r=1}^{\rho(z;\boldsymbol{\mu})} \mu_r - z\right)^2 + \sum_{r=\rho(z;\boldsymbol{\mu})+1} \mu_r^2 \;\; ,$$

where, to remind the reader, the number of strictly positive $\boldsymbol{\alpha}$'s is,

$$\rho(z;\boldsymbol{\mu}) = \max\left\{\rho : \mu_\rho - \frac{1}{\rho}\left(\sum_{r=1}^{\rho}\mu_r - z\right) \geq 0\right\} \;\; .$$

Throughout the proof $\boldsymbol{\mu}$ is fixed and known. We therefore abuse our notation and use the shorthand $\rho(z)$ for $\rho(z;\boldsymbol{\mu})$. Recall that $\boldsymbol{\mu}$ is given in a non-ascending order, $\mu_{i+1} \leq \mu_i$ for $i \in [p-1]$. Therefore, we get that

$$\begin{aligned}
z_{i+1} \;&=\; \sum_{r=1}^{i+1}\mu_r - (i+1)\mu_{i+1} \;=\; \sum_{r=1}^{i}\mu_r + \mu_{i+1} - \mu_{i+1} - i\,\mu_{i+1} \\
&=\; \sum_{r=1}^{i}\mu_r - i\,\mu_{i+1} \;\geq\; \sum_{r=1}^{i}\mu_r - i\,\mu_i \;=\; z_i \;\; .
\end{aligned}$$

Thus, the sequence $z_1, z_2, \ldots, z_p$ is monotonically non-decreasing and the intervals $[z_i, z_{i+1})$ are well defined. The definition of $\rho(z)$ implies that for all $z \in [z_i, z_{i+1})$ we have $\rho(z) = \rho(z_i) = i$. Hence, the value of $g(z; \boldsymbol{\mu})$ for each $z \in [z_i, z_{i+1})$ is,

$$g(z;\boldsymbol{\mu}) \;=\; \frac{1}{i}\left(\sum_{r=1}^{i}\mu_r - z\right)^2 + \sum_{r=i+1}^{p}\mu_r^2 \ .$$

We have thus established the fact that $g(z; \boldsymbol{\mu})$ is a quadratic function in each interval (z_i, z_{i+1}) and in particular it is continuous in each such sub-interval. To show that g is continuous in $[0, C]$ we need to examine all of its knots z_i. Computing the left limit and the right limit of g at each knot we get that,

$$\begin{aligned}
\lim_{z\downarrow z_i} g(z;\boldsymbol{\mu}) &= \lim_{z\downarrow z_i}\frac{1}{i}\left(\sum_{r=1}^{i}\mu_r - z\right)^2 + \sum_{r=i+1}^{p}\mu_r^2 \\
&= \frac{1}{i}\left(\sum_{r=1}^{i}\mu_r - \sum_{r=1}^{i}\mu_r + i\mu_i\right)^2 + \sum_{r=i+1}^{p}\mu_r^2 \\
&= i\mu_i^2 + \sum_{r=i+1}^{p}\mu_r^2 \ ,
\end{aligned}$$

and

$$\begin{aligned}
\lim_{z\uparrow z_i} g(z;\boldsymbol{\mu}) &= \lim_{z\uparrow z_i}\frac{1}{i-1}\left(\sum_{r=1}^{i-1}\mu_r - z\right)^2 + \sum_{r=i}^{p}\mu_r^2 \\
&= \frac{1}{i-1}\left(\sum_{r=1}^{i-1}\mu_r - \sum_{r=1}^{i}\mu_r + i\mu_i\right)^2 + \sum_{r=i}^{p}\mu_r^2 \\
&= (i-1)\mu_i^2 + \sum_{r=i}^{p}\mu_r^2 \;=\; i\mu_i^2 + \sum_{r=i+1}^{p}\mu_r^2 \ .
\end{aligned}$$

Therefore, $\lim_{z\downarrow z_i} g(z;\boldsymbol{\mu}) = \lim_{z\uparrow z_i} g(z;\boldsymbol{\mu})$ and g is indeed continuous. The continuity of the derivative of g is shown by using the same technique of examining the right and left limits at each knot z_i for the function,

$$g'(z;\boldsymbol{\mu}) = \frac{2}{i}\left(z - \sum_{r=1}^{i}\mu_r\right) \ .$$

Finally, we use the fact that a continuously differentiable function is convex iff its derivative is monotonically non-decreasing. Since g is quadratic in each segment $[z_i, z_{i+1}]$, g' is indeed monotonically non-decreasing in each segment. Furthermore, from the continuity of g' we get that g' is monotonically non-decreasing on the entire interval $[0, 1]$. Thus, g is convex on $[0, 1]$. ■

Appendix B

Using Online Convex Programming for PAC learning

In this chapter we outline the applicability of our algorithmic framework from the previous chapter to *Probably Approximately Correct* (PAC) learning [116]. The PAC setting is also referred to as *batch* learning. We start this chapter with a short introduction to PAC learning. Next, in Section B.2 we relate two important notions of learnability: the notion of mistake bound in online learning and the notion of VC dimension used in PAC learning. Based on this comparison, we demonstrate that online learning is more difficult than PAC learning. Despite this disparity, as we have shown in previous chapters, many classes of hypotheses can be efficiently learned in the online model. In Section B.3 we derive several simple conversions from the online setting to the batch setting. The end result is that the existence of a good online learner implies the existence of good batch learning algorithms. These online-to-batch conversions are general and do not necessarily rely on our specific algorithms for online learning.

B.1 Brief Overview of PAC Learning

In this section we briefly define the PAC model of learning from examples. Similar to the online model, we have a set of questions (or instances) $\mathcal{X}$ and a set of answers (or targets) $\mathcal{Y}$. We also have a class of hypotheses $\mathcal{H} = \{h : \mathcal{X} \to \mathcal{Y}\}$; namely, each hypothesis $h \in \mathcal{H}$ is a question answering mechanism. In contrast to the online model, in the PAC learning model we also have a joint distribution Q over $\mathcal{X} \times \mathcal{Y}$. The input of the learning algorithm is a batch of T training examples $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)\}$. Based on these examples, the learner needs to find an output hypothesis $h_o \in \mathcal{H}$. We assess the quality of the output hypothesis using its expected loss, where expectation is taken with respect to Q. Formally, let $\ell : \mathcal{H} \times (\mathcal{X} \times \mathcal{Y}) \to \mathbb{R}_+$ be a loss function and define the *risk* of using a hypothesis h to be

$$\mathrm{risk}(h) \;=\; \mathbb{E}_{(\mathbf{x},y)\sim Q}[\ell(h,(\mathbf{x},y))] \;. \tag{B.1}$$

In words, the risk of h is its loss on a randomly sampled example from $\mathcal{X} \times \mathcal{Y}$, where sampling is based on the distribution Q.

To achieve this goal, we assume that the input examples the learner receives are identically and independently distributed (i.i.d.) according to the distribution Q. Note that we only observe the training examples but we do not know the distribution Q. The training set of examples is used as a restricted window through which we estimate the quality of a hypothesis according to the distribution of unseen examples in the real world, Q.

Clearly, if the learning algorithm is "unlucky" the training set of examples might not correctly reflect the underlying distribution Q. We therefore must give the learning algorithm some slack. In the PAC model, we require the algorithm to be probably approximately correct. By "probably" correct we mean that there is a small probability that the learning algorithm will fail and by "approximately" correct we mean that the risk of the output hypothesis can be slightly larger than that of the optimal hypothesis in $\mathcal{H}$. Formally, assume that the minimum risk is achievable and denote by $h^\star$ a hypothesis s.t. $\mathrm{risk}(h^\star) = \min_{h\in\mathcal{H}} \mathrm{risk}(h)$. We are interested in risk bounds of the form

$$\mathbb{P}\left[\mathrm{risk}(h_o) - \mathrm{risk}(h^\star) \geq \epsilon\right] \;\leq\; \delta \;, \tag{B.2}$$

where $\epsilon > 0$ is the "approximately" parameter and $\delta \in (0,1)$ is the "probably" parameter. The probability above is over the choice of the input training examples.[1]

A natural question is what values of ϵ and δ are plausible. Intuitively, since the training set is our "window" through which we observe the world, increasing the size of the training set should yield more accurate output hypotheses. The theory of sample complexity relates the values of ϵ and δ to the size of the training set and to a measure of complexity of $\mathcal{H}$. One classical complexity measure is the so-called VC dimension.

The VC dimension was originally designed for binary hypotheses (i.e. $\mathcal{Y} = \{+1,-1\}$) and is defined as follows:

$$VC(\mathcal{H}) \;=\; \sup\{\, d \in \mathbb{N} : \exists \mathbf{x}_1,\ldots,\mathbf{x}_d \text{ s.t. } \forall y_1,\ldots,y_d,\; \exists h \in H,\; \text{ s.t. } \forall i \in [d],\; y_i = h(\mathbf{x}_i) \,\} \;.$$

In words, the VC dimension of $\mathcal{H}$ is the maximal size of a set of instances that can be shattered by $\mathcal{H}$, that is, any labeling of the instances can be explained by $\mathcal{H}$. The basic VC theorem tells us

[1] Note that $\mathrm{risk}(h_o)$ depends on the input training examples and thus it is a random variable. In contrast, while $\mathrm{risk}(h^\star)$ is unknown to us, it does not constitute a random variable.

that a class $\mathcal{H}$ can be learned based on a finite training set if and only if its VC dimension is finite. Intuitively, if the VC dimension of $\mathcal{H}$ is larger than d, then it can perfectly explain any training set of size d and thus we gain no information from the specific training set at hand.

B.2 Mistake bounds and VC dimension

In the previous section we described the PAC learning model and outlined the basic VC theorem. Littlestone [82] proposed the mistake bound model which is another learnability criterion for a class of hypotheses $\mathcal{H}$. Formally, a class $\mathcal{H}$ is M-learnable in the mistake bound model if there exists an online learning algorithm that makes at most M prediction mistakes on any sequence of examples of the form $(\mathbf{x}_1, h^\star(y_1)), \ldots, (\mathbf{x}_T, h^\star(y_T))$ for some $h^\star \in \mathcal{H}$.

The following theorem, whose proof is given in [82], relates the two notions of learnability.

Theorem 8 *Let $\mathcal{H}$ be a set of hypotheses and assume that $\mathcal{H}$ is M-learnable in the mistake bound model. Then,* $\mathrm{VC}(H) \leq M$.

The converse is not true as the following example demonstrates.

Example 6 *Let $\mathcal{X} = [n]$, $\mathcal{Y} = \{+1, -1\}$, and let $\mathcal{H} = \{h_i(x) = [\![x \leq i]\!] : i \in [n]\}$, where $[\![\pi]\!] = 1$ if the predicate π holds and -1 otherwise. Let us assume for simplicity that n is a power of 2. It is easy to verify that* $\mathrm{VC}(\mathcal{H}) = 1$. *In contrast, we prove in the following that for any online learning algorithm we can construct a sequence of $T = \log_2(n)$ examples such that the online algorithm errs on all of them. In other words, for any $M < \log_2(n)$ the set $\mathcal{H}$ is* not *M-learnable in the mistake bound model. To construct a worst case sequence we initialized $l = 1$ and $r = n$. We now present the instance $x_t = (r - l + 1)/2 + 1$ to the online algorithm. If the prediction of the algorithm is $+1$ we provide the feedback -1 and re-assign $r = x_t$. Otherwise, we provide the feedback $+1$ and re-assign $l = x_t + 1$. We can repeat this process $\log_2(n)$ times, which concludes the construction of our worst case sequence.*

B.3 Online-to-Batch Conversions

In this section we show that an online algorithm that attains low regret can be converted into a batch learning algorithm that attains low risk. Such online-to-batch conversions are interesting both from the practical as from the theoretical perspective. First, from the practical perspective, the simplicity and efficiency of online learning algorithms along with online-to-batch conversions automatically yield efficient batch learning algorithms. Second, from the theoretical perspective, the regret bounds we derive in previous chapters yield the best known risk bounds for batch learning. The analysis, however, is much simpler.

Recall that in this chapter we assume that the sequence of examples are independently and identically distributed according to an unknown distribution Q over $\mathcal{X} \times \mathcal{Y}$. To emphasize the above fact and to simplify our notation, we denote by Z_t the tth example in the sequence and use the shorthand

$$Z_1^j = (Z_1, \ldots, Z_j) = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_j, y_j)) .$$

We denote the tth hypothesis that the online learning algorithm generates by h_t. Note that h_t is a function of Z_1^{t-1} and thus it is a random variable. We denote the average loss of the online algorithm by

$$M_T(Z_1^T) = \frac{1}{T} \sum_{t=1}^{T} \ell(h_t, Z_t) . \tag{B.3}$$

We often omit the dependence of M_T on Z_1^T and use the shorthand M_T for denoting $M_T(Z_1^T)$.

The rest of this section is organized as follows. In Section B.3.1 we show that the expected value of M_T equals the expected value of $\frac{1}{T}\sum_{t=1}^{T} \mathrm{risk}(h_t)$. Thus, the online loss is an un-biased estimator for the average risk of the ensemble $(h_1, \ldots, h_T)$. Next, in Section B.3.2 we underscore that regret bounds (i.e., bounds on M_T) can yield bounds on the average risk of $(h_1, \ldots, h_T)$. Therefore, there must exist at least one hypothesis in the ensemble $(h_1, \ldots, h_T)$ whose risk is low. Since our goal in batch learning is to output a *single* hypothesis (with low risk), we must find a way to choose a single good hypothesis from the ensemble. In Section B.3.3, we discuss several simple procedures for choosing a single good hypothesis from the ensemble.

B.3.1 Online loss and ensemble's risk

Our first theorem shows that the expected value of M_T equals the expected value of the risk of the ensemble $(h_1, \ldots, h_T)$.

Theorem 9 *Let $Z_1, \ldots, Z_T$ be a sequence of independent random variables, each of which is distributed according to a distribution Q over $\mathcal{X} \times \mathcal{Y}$. Let $h_1, \ldots, h_T$ be the sequence of hypotheses generated by an online algorithm when running on the sequence $Z_1, \ldots, Z_T$, and let* $\mathrm{risk}(h)$ *be as defined in Eq. (B.1). Then,*

$$\mathbb{E}_{Z_1,\ldots,Z_T}\left[\frac{1}{T}\sum_{t=1}^{T} \mathrm{risk}(h_t)\right] = \mathbb{E}_{Z_1,\ldots,Z_T}\left[\frac{1}{T}\sum_{t=1}^{T} \ell(h_t, Z_t)\right] .$$

Proof Using the linearity of expectation and the fact that h_t only depends on Z_1^{t-1} we have,

$$\mathbb{E}_{Z_1^T}[\frac{1}{T}\sum_{t=1}^{T} \ell(h_t, Z_t)] = \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{Z_1^T}[\ell(h_t, Z_t)] = \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{Z_1^t}[\ell(h_t, Z_t)] . \tag{B.4}$$

Recall that the law of total expectation implies that for any two random variables R_1, R_2, and a function f, $\mathbb{E}_{R_1}[f(R_1)] = \mathbb{E}_{R_2}\mathbb{E}_{R_1}[f(R_1)|R_2]$. Setting $R_1 = Z_1^t$ and $R_2 = Z_1^{t-1}$ we get that

$$\mathbb{E}_{Z_1^t}[\ell(h_t, Z_t)] \;=\; \mathbb{E}_{Z_1^{t-1}}[\mathbb{E}_{Z_1^t}[\ell(h_t, Z_t)|Z_1^{t-1}]] \;=\; \mathbb{E}_{Z_1^{t-1}}[\mathrm{risk}(h_t)] \;=\; \mathbb{E}_{Z_1^T}[\mathrm{risk}(h_t)] \;.$$

Combining the above with Eq. (B.4) concludes our proof. ■

The above theorem tells us that in expectation, the online loss equals the average risk of the ensemble of hypotheses generated by the online algorithm. The next step is to use our regret bounds from previous chapters to derive bounds on the online loss.

B.3.2 From Regret Bounds to Risk Bounds

In the previous section we analyzed the risks of the hypotheses generated by an online learning algorithm based on the average online loss, M_T. In the previous chapter we analyzed the regret of online algorithms by bounding the online loss, M_T, in terms of the loss of any competing hypothesis in $\mathcal{H}$. In particular, the bound holds for the hypothesis in $\mathcal{H}$ whose risk is minimal. Formally, assume that the minimum risk is achievable and denote by $h^\star$ a hypothesis s.t. $\mathrm{risk}(h^\star) = \min_{h\in\mathcal{H}} \mathrm{risk}(h)$. In this section, we derive bounds on M_T in terms of $\mathrm{risk}(h^\star)$.

We encountered two forms of regret bounds. In the simplest form of regret bounds, there exists a deterministic function $B : \mathbb{N} \to \mathbb{R}$ such that

$$\forall h \in \mathcal{H},\;\; \frac{1}{T}\sum_{t=1}^{T} \ell(h_t, Z_t) \;\le\; \frac{1}{T}\sum_{t=1}^{T} \ell(h, Z_t) + \frac{B(T)}{T} \;. \tag{B.5}$$

For example, the regret bounds given in Corollary 1 and Corollary 3 can be written in the form given in Eq. (B.5) with $B(T) = O(\sqrt{T})$.

In the second form of regret bounds, the function B depends on the actual sequence of examples the online algorithm receives. Formally, the function B is defined as follows

$$B(Z_1^T) \;=\; \min_{h\in\mathcal{H}} \sum_{t=1}^{T} \ell(h, (\mathbf{x}_t, y_t)) \;. \tag{B.6}$$

We derived regret bounds of the form

$$\forall h \in \mathcal{H},\;\; \frac{1}{T}\sum_{t=1}^{T} \ell(h_t, (\mathbf{x}_t, y_t)) \le \frac{B(Z_1^T) + \mu\sqrt{B(Z_1^T)} + \nu}{T} \;, \tag{B.7}$$

where μ, ν are positive constants. For example, the regret bound we derived in Corollary 4 can be written in the form given in Eq. (B.7), where using the notation given in Corollary 4, $\mu = 3.4\sqrt{L\,U}$ and $\nu = 3.4\sqrt{L\,U\,\beta_1} + 11.4LU$.

In the rest of this section we assume that the online algorithm satisfies either the regret bound given in Eq. (B.5) or the regret bound given in Eq. (B.7). Based on this assumption, we derive risk bounds for the hypotheses generated by the online algorithm.

We start by deriving a bound on the average risk of the ensemble of hypotheses that the online algorithm generates.

Theorem 10 *Assume that the condition stated in Theorem 9 holds and that the online algorithm satisfies Eq. (B.5). Then,*

$$\mathbb{E}_{Z_1^T}\left[\tfrac{1}{T}\sum_{t=1}^{T}\mathrm{risk}(h_t)\right] \;\le\; \mathrm{risk}(h^\star) + \frac{B(T)}{T} \;\;.$$

Proof Taking expectation of the inequality given in Eq. (B.5) we obtain

$$E_{Z_1^T}[\frac{1}{T}\sum_{t=1}^{T}\ell(h_t, Z_t)] \;\le\; \mathbb{E}_{Z_1^T}[\frac{1}{T}\sum_{t=1}^{T}\ell(h^\star, Z_t)] + \frac{B(T)}{T} \;\;. \tag{B.8}$$

Since $h^\star$ does not depend on the choice of Z_1^T, we have,

$$\mathbb{E}_{Z_1^T}[\frac{1}{T}\sum_{t=1}^{T}\ell(h^\star, Z_t)] \;=\; \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_{Z_1^T}[\ell(h^\star, Z_t)] \;=\; \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_{Z_t}[\ell(h^\star, Z_t)] \;=\; \mathrm{risk}(h^\star)\;. \tag{B.9}$$

Combining the above with Eq. (B.8) and Theorem 9 we conclude our proof. ■

Next, we turn to deriving risk bounds, assuming that the online algorithm satisfies the regret bound given in Eq. (B.7). In contrast to the previous form of regret bounds, it is now more complicated to analyze the expected value of the bound given in Eq. (B.7), due to the sqrt root of $B(Z_1^T)$. We thus take a different course and first derive a bound on $B(Z_1^T)$ based on $\mathrm{risk}(h^\star)$ that holds with high probability. Next, we bound M_T with high probability and finally we use these bounds to analyze the expected value of M_T.

Lemma 21 *Let $B(Z_1^T)$ be as defined in Eq. (B.6) and assume that the range of ℓ is $[0,1]$. Let*

$h^\star \in \arg\min_{h\in\mathcal{H}} \mathrm{risk}(h)$. *Then, for any* $\delta \in (0,1)$, *with a probability of at least* $1-\delta$ *we have*

$$\tfrac{1}{T} B(Z_1^T) \;\le\; \mathrm{risk}(h^\star) + \sqrt{\frac{2\ln\left(\frac{1}{\delta}\right)\,\mathrm{risk}(h^\star)}{T}} + \frac{2\ln\left(\frac{1}{\delta}\right)}{3\,T} \quad .$$

Proof We prove the lemma using Bernstein's inequality.[2] Since $\ell(h^\star, Z_t) \in [0,1]$ we obtain that $\mathrm{Var}[\ell(h^\star, Z_t)] \le \mathbb{E}[\ell(h^\star, Z_t)] = \mathrm{risk}(h^\star)$. Therefore, Bernstein's inequality implies that

$$\mathbb{P}[\tfrac{1}{T}\sum_{t=1}^{T} \ell(h^\star, Z_t) - \mathrm{risk}(h^\star) \ge \epsilon] \quad\le\quad \exp\left(-\frac{T\,\epsilon^2}{2(\mathrm{risk}(h^\star)+\epsilon/3)}\right) \quad .$$

Denote by δ the right-hand side of the above. Thus,

$$T\,\epsilon^2 - \tfrac{2}{3}\,\ln\left(\tfrac{1}{\delta}\right)\,\epsilon - 2\,\ln\left(\tfrac{1}{\delta}\right)\,\mathrm{risk}(h^\star) = 0 \quad .$$

Solving for ϵ (while using the fact that $\epsilon > 0$) we obtain that

$$\epsilon \;=\; \frac{\frac{1}{3}\,\ln\left(\frac{1}{\delta}\right) + \sqrt{(\frac{1}{3}\,\ln\left(\frac{1}{\delta}\right))^2 + 2\,T\,\ln\left(\frac{1}{\delta}\right)\,\mathrm{risk}(h^\star)}}{T} \;\le\; \sqrt{\frac{2\ln\left(\frac{1}{\delta}\right)\,\mathrm{risk}(h^\star)}{T}} + \frac{\frac{2}{3}\ln\left(\frac{1}{\delta}\right)}{T} \quad .$$

Therefore, with a probability of at least $1-\delta$ we have that

$$\tfrac{1}{T}\sum_{t=1}^{T} \ell(h^\star, Z_t) \;\le\; \mathrm{risk}(h^\star) + \sqrt{\frac{2\ln\left(\frac{1}{\delta}\right)\,\mathrm{risk}(h^\star)}{T}} + \frac{\frac{2}{3}\ln\left(\frac{1}{\delta}\right)}{T} \quad .$$

The bound in the lemma follows from the above by noticing that $\frac{1}{T}B(Z_1^T) \le \frac{1}{T}\sum_{t=1}^{T} \ell(h^\star, Z_t)$. ■

Lemma 22 *Assume that the conditions stated in Lemma 21 hold and that the online algorithm satisfies the regret bound given in Eq. (B.7). Then, with a probability of at least* $1-\delta$ *we have*

$$M_T \;\le\; \mathrm{risk}(h^\star) + \mu_1 \sqrt{\frac{\mathrm{risk}(h^\star)}{T}} + \frac{\mu_2}{T} \quad ,$$

where $\mu_1 = \sqrt{2}\,(\mu + \sqrt{\ln\left(\frac{1}{\delta}\right)}\,)$ *and* $\mu_2 = \nu + 3\ln\left(\frac{1}{\delta}\right) + \mu\,\sqrt{3\,\ln\left(\frac{1}{\delta}\right)}$.

[2] To remind the reader, we state the inequality (see for example Theorem 3 in [12]). Let $X_1, \ldots, X_T$ be a sequence of independent random variables with zero mean, and assume that $X_t \le 1$ with probability 1. Let $\sigma^2 = \frac{1}{T}\sum_t \mathrm{Var}[X_t]$. Then, for any $\epsilon > 0$, $\mathbb{P}[\sum_t X_t \ge \epsilon] \le \exp(-\frac{T\,\epsilon^2}{2(\sigma^2+\epsilon/3)})$.

Proof First, assume that $\text{risk}(h^\star) \le \frac{2\ln(1/\delta)}{T}$. In this case, Lemma 21 implies that

$$\frac{B(Z_1^T)}{T} \le \text{risk}(h^\star) + \frac{3\,\ln\left(\frac{1}{\delta}\right)}{T} \; .$$

Combining the above with Eq. (B.7) we obtain

$$\begin{aligned} M_T &\le \text{risk}(h^\star) + \mu\sqrt{\frac{\text{risk}(h^\star) + 3\,\ln\left(\frac{1}{\delta}\right)/T}{T}} + \frac{\nu + 3\,\ln\left(\frac{1}{\delta}\right)}{T} \\ &\le \text{risk}(h^\star) + \mu\sqrt{\frac{\text{risk}(h^\star)}{T}} + \frac{\nu + 3\ln\left(\frac{1}{\delta}\right) + \mu\sqrt{3\,\ln\left(\frac{1}{\delta}\right)}}{T} \; . \end{aligned} \tag{B.10}$$

Next, assume that $\text{risk}(h^\star) > \frac{2\ln(1/\delta)}{T}$. Thus, Lemma 21 implies that

$$\frac{B(Z_1^T)}{T} \le \text{risk}(h^\star) + \sqrt{\frac{2\ln\left(\frac{1}{\delta}\right)\,\text{risk}(h^\star)}{T}} + \frac{\frac{2}{3}\ln\left(\frac{1}{\delta}\right)}{T} < 2\,\text{risk}(h^\star) + \frac{\frac{2}{3}\ln\left(\frac{1}{\delta}\right)}{T} \; .$$

Combining the above with Eq. (B.7) we obtain

$$\begin{aligned} M_T &\le \frac{B(Z_1^T)}{T} + \frac{\mu}{\sqrt{T}}\sqrt{\frac{B(Z_1^T)}{T}} + \frac{\nu}{T} \\ &\le \text{risk}(h^\star) + \sqrt{\frac{2\ln\left(\frac{1}{\delta}\right)\,\text{risk}(h^\star)}{T}} + \frac{\frac{2}{3}\ln\left(\frac{1}{\delta}\right)}{T} + \frac{\mu}{\sqrt{T}}\left(\sqrt{2\text{risk}(h^\star)} + \sqrt{\frac{\frac{2}{3}\ln\left(\frac{1}{\delta}\right)}{T}}\right) + \frac{\nu}{T} \\ &\le \text{risk}(h^\star) + \sqrt{\frac{2\,\text{risk}(h^\star)}{T}}\left(\sqrt{\ln\left(\frac{1}{\delta}\right)} + \mu\right) + \frac{\nu + \frac{2}{3}\ln\left(\frac{1}{\delta}\right) + \mu\sqrt{\frac{2}{3}\,\ln\left(\frac{1}{\delta}\right)}}{T} \; . \end{aligned}$$

Combining the above with Eq. (B.10) we conclude our proof. ■

We are now ready to derive a risk bound from the regret bound given in Eq. (B.7).

Theorem 11 *Assume that the condition stated in Theorem 9 holds and that the online algorithm satisfies Eq. (B.7). Additionally, assume that the range of the loss function, ℓ, is $[0,1]$ and let $h^\star \in \arg\min_{h\in\mathcal{H}} \text{risk}(h)$. Then,*

$$\mathbb{E}_{Z_1^T}\left[\frac{1}{T}\sum_{t=1}^{T}\text{risk}(h_t)\right] \le \text{risk}(h^\star) + (\mu + e)\sqrt{\frac{2\,\text{risk}(h^\star)}{T}} + \frac{\nu + e(\sqrt{3}\,\mu + 3)}{T} \; .$$

Proof Using Theorem 9, the left-hand side of the bound equals $\mathbb{E}[M_T]$. Define the random variable

$$R = M_T - \text{risk}(h^\star) - \mu \sqrt{\frac{2\,\text{risk}(h^\star)}{T}} - \frac{\nu}{T} \ .$$

Additionally, let $\alpha_1 = \mu\sqrt{3}\,/T + \sqrt{2\,\text{risk}(h^\star)/T}$ and $\alpha_2 = 3/T$ and for all $s = 0, 1, 2, \ldots$ define $t_s = \alpha_1\sqrt{s} + \alpha_2\, s$. Based on Lemma 22 we have that $\mathbb{P}[R \geq t_s] \leq e^{-s}$. Therefore,

$$\begin{aligned} \mathbb{E}[R] \ &\leq\ \int_0^\infty x\mathbb{P}[R = x]dx \ \leq\ \sum_{s=1}^{\infty} t_s \mathbb{P}[R \geq t_{s-1}] \ \leq\ \alpha_1 \sum_{s=1}^{\infty} \sqrt{s}\, e^{-(s-1)} + \alpha_2 \sum_{s=1}^{\infty} s\, e^{-(s-1)} \\ &\leq\ (\alpha_1 + \alpha_2) \sum_{s=1}^{\infty} s\, e^{-(s-1)} \ \leq\ e\,(\alpha_1 + \alpha_2) \ , \end{aligned}$$

where the last inequality can be proven by bounding the tail $\sum_{s=4}^{\infty} s\, e^{-(s-1)}$ with the integral $\int_5^\infty x e^{-(x-1)} dx$. Our proof follows by plugging the definitions of R, α_1, and α_2 into the above. ■

B.3.3 Choosing a hypothesis from the ensemble

In the previous section we showed that regret bounds yield bounds on the average risk of $(h_1, \ldots, h_T)$. The goal of this section is two-fold. First, we need to output a single hypothesis and thus we must choose a single good hypothesis from $(h_1, \ldots, h_T)$. Second, the analysis in the previous section focuses on the expected value of the risk whereas in practice we have a single training set. Therefore, we must analyze the concentration properties of our online-to-batch conversion schemes.

Last (with random stopping time)

The simplest conversion scheme runs the online algorithm on a sequence of r examples and returns the last hypothesis h_r. To analyze the risk of h_r we assume that r is chosen uniformly at random from $[T]$, where T is a predefined integer. The following lemma shows that the bounds on $\mathbb{E}[\frac{1}{T}\sum_t \text{risk}(h_t)]$ we derived in the previous section can be transformed into a bound on $\text{risk}(h_r)$.

Lemma 23 *Assume that the conditions stated in Theorem 9 hold. Let $h^\star$ be a hypothesis in $\mathcal{H}$ whose risk is minimal and let $\delta \in (0, 1)$. Assume that there exists a scalar α such that*

$$\mathbb{E}_{Z_1^T}\left[\frac{1}{T}\sum_{t=1}^{T} \text{risk}(h_t)\right] \ \leq\ \text{risk}(h^\star) + \alpha \ .$$

Let $r \in [T]$ and assume that r is uniformly chosen at random from $[T]$. Then, with a probability of at least $1-\delta$ over the choices of Z_1^T and r we have

$$\mathrm{risk}(h_r) \;\le\; \mathrm{risk}(h^\star) + \frac{\alpha}{\delta} \;\; .$$

Proof Let R be the random variable $(\mathrm{risk}(h_r) - \mathrm{risk}(h^\star))$. From the definition of $h^\star$ as the minimizer of $\mathrm{risk}(h)$ we clearly have that R is a non-negative random variable. In addition, the assumption in the lemma implies that $\mathbb{E}[R] \le \alpha$. Thus, from the Markov inequality

$$\mathbb{P}[R \ge a] \le \frac{\mathbb{E}[R]}{a} \le \frac{\alpha}{a} \;\; .$$

Setting $\frac{\alpha}{a} = \delta$ we conclude our proof. ■

Combining the above lemma with Theorem 10 and Theorem 11 we obtain the following:

Corollary 8 *Assume that the conditions stated in Theorem 10 hold. Let $h^\star$ be a hypothesis in $\mathcal{H}$ whose risk is minimal and let $\delta \in (0,1)$. Then, with a probability of at least $1-\delta$ over the choices of $(Z_1,\ldots,Z_T)$ and the index r we have that*

$$\mathrm{risk}(h_r) \;\le\; \mathrm{risk}(h^\star) + \frac{B(T)}{\delta\, T} \;\; .$$

Similarly, if the conditions stated in Theorem 11 hold then with a probability of at least $1-\delta$ we have that

$$\mathrm{risk}(h_r) \;\le\; \mathrm{risk}(h^\star) + \frac{\mu + e}{\delta}\sqrt{\frac{2\,\mathrm{risk}(h^\star)}{T}} + \frac{\nu + e(\sqrt{3}\,\mu + 3)}{\delta\, T} \;\; .$$

Corollary 8 implies that by running the online algorithm on the first r examples and outputting h_r we obtain a batch learning algorithm with a guaranteed risk bound. However, the concentration bound given in Corollary 8 depends linearly on $1/\delta$, where δ is the confidence parameter. Our next conversion scheme is preferable when we are aiming for very high confidence.

Validation

In the validation conversion scheme, we first pick a subset of hypotheses from $h_1,\ldots,h_T$ and then use a fresh validation set to decide which hypothesis to output. We start by using a simple amplification technique (a.k.a. boosting the confidence), to construct a few candidate hypotheses such that with confidence $1-\delta$ the risk of at least one of the hypotheses is low.

Theorem 12 *Assume that the conditions stated in Corollary 8 hold. Let s be a positive integer. Assume that we reset the online algorithm after each block of T/s examples. Let $h'_1,\ldots,h'_s$ be a* *sequence of hypotheses where for each $i \in [s]$, the hypothesis h'_i is picked uniformly at random from $\{h_{i\,s+1}, \ldots, h_{i\,s+s}\}$. If the conditions stated in Theorem 10 hold, then with a probability of at least $1 - e^{-s}$, there exists $i \in [s]$ such that*

$$\mathrm{risk}(h'_i) \;\leq\; \mathrm{risk}(h^\star) + \frac{e\ s\,B(T/s)}{T} \enspace .$$

Similarly, if the conditions stated in Theorem 10 hold then

$$\mathrm{risk}(h'_i) \;\leq\; \mathrm{risk}(h^\star) + e\left((\mu+e)\sqrt{\frac{2\,\mathrm{risk}(h^\star)\,s}{T}} + \frac{(\nu + e(\sqrt{3}\,\mu+3))s}{T}\right) \enspace .$$

Proof Using Corollary 8 with a confidence value of $1/e$, we know that for all $i \in [s]$, with a probability of at least $1 - 1/e$ we have that

$$\mathrm{risk}(h'_i) - \mathrm{risk}(h^\star) \leq e\,\alpha(T/s) \tag{B.11}$$

where $\alpha(k) = B(k)/k$ if the conditions stated in Theorem 10 hold and $\alpha(k) = (\mu + e)\sqrt{\frac{2\,\mathrm{risk}(h^\star)}{k}} + \frac{\nu+e(\sqrt{3}\,\mu+3)}{k}$ if the conditions stated in Theorem 11 hold. Therefore, the probability that for *all* blocks the above inequality does not hold is at most e^{-s}. Hence, with a probability of at least $1 - e^{-s}$, at least one of the hypotheses h'_i satisfies Eq. (B.11). ■

The above theorem tells us that there exists at least one hypothesis $h_g \in \{h'_1, \ldots, h'_s\}$ such that $\mathrm{risk}(h_g)$ is small. To find such a good hypothesis we can use a validation set and choose the hypothesis whose loss on the validation set is minimal. Formally, let $Z'_1, \ldots, Z'_m$ be a sequence of random variables that represents a fresh validation set and let h_o be the hypothesis in $\{h'_1, \ldots, h'_s\}$ whose loss over $Z'_1, \ldots, Z'_m$ is minimal. Applying standard generalization bounds (e.g. Eq. (21) in [13]) on the finite hypothesis class $\{h'_1, \ldots, h'_s\}$ we obtain that there exists a constant C such that

$$\mathrm{risk}(h_o) - \mathrm{risk}(h_g) \;\leq\; C\left(\sqrt{\mathrm{risk}(h_g)\,\frac{\log(s)\,\log(m) + \ln\left(\frac{1}{\delta}\right)}{m}} + \frac{\log(s)\,\log(m) + \ln\left(\frac{1}{\delta}\right)}{m}\right) \enspace .$$

Averaging

If the set of hypotheses is convex and the loss function, ℓ, is convex with respect to h then the risk function is also convex with respect to h. Therefore, Jensen's inequality implies that

$$\operatorname{risk}\left(\tfrac{1}{T}\sum_{t=1}^{T} h_t\right) \leq \tfrac{1}{T}\sum_{t=1}^{T}\operatorname{risk}(h_t) \ .$$

We can use the above inequality in conjunction with Theorem 10 and Theorem 11 to derive bounds on the expected risk of the averaged hypothesis $\bar{h} = \frac{1}{T}\sum_{t=1}^{T} h_t$. In particular, the bounds we derived in Corollary 8 hold for $\bar{h}$ as well. If we want to have very high confidence, we can use the amplification technique described in the previous conversion scheme. Alternatively, we can use the following theorem, which follows from Theorem 3 in [122]. A similar bound was derived in [18].

Theorem 13 *Assume that the conditions stated in Theorem 9 hold. In addition, assume that $\mathcal{H}$ is a convex set and that ℓ is a convex function with respect to its first argument whose range is $[0,1]$. Then, for any $\delta \in (0,1)$ the following holds with a probability of at least $1-\delta$:*

$$\operatorname{risk}\left(\tfrac{1}{T}\sum_{t=1}^{T} h_t\right) \leq \frac{1}{T}\sum_{t=1}^{T}\operatorname{risk}(h_t) \leq M_T + \frac{3\,\ln\left(\frac{T+3}{\delta}\right) + 2\sqrt{T\,M_T\ln\left(\frac{T+3}{\delta}\right)}}{T} \ .$$

The above bound can be combined with our concentration bound on M_T given in Lemma 22, to yield a bound on $\operatorname{risk}(\bar{h})$ in terms of $\operatorname{risk}(h^\star)$.

B.4 Bibliographic Notes

The PAC learning model was introduced by Valiant [116]. The VC dimension was put forward by Vapnik and Chervonenkis [118]. For a more detailed overview of PAC learning and generalization bounds see for example [70, 117, 43, 13].

Littlestone [83] initiated the study of online to batch conversions. Freund and Schapire [53] demonstrated that using the Perceptron algorithm along with the voting online-to-batch technique [67] yields an efficient replacement for Support Vector Machines [117, 33, 100]. Concentration bounds on the risk of the ensemble that rely on the online loss were derived by Zhang [122] and Cesa-Bianchi et al [18, 17]. The bounds are based on concentration inequalities for martingales, such as the Azuma inequality [7] and a Bernstein-like inequality for martingales described by Freedman [51]. Our presentation relies on expectations and on the basic Markov inequality and is thus much simpler. Nevertheless, the bounds we obtain for the validation scheme share the same form as the bounds of the conversions scheme described in [122, 17, 18].