Foundations and Trends® in
Machine Learning
Vol. 4, No. 2 (2011) 107–194

DOI: 10.1561/2200000018

Online Learning and Online Convex Optimization

By Shai Shalev-Shwartz

Contents

1 Introduction **108**

1.1 Examples 111
1.2 A Gentle Start 112
1.3 Organization and Scope 116
1.4 Notation and Basic Definitions 117

2 Online Convex Optimization **119**

2.1 Convexification 120
2.2 Follow-the-leader 124
2.3 Follow-the-Regularized-Leader 127
2.4 Online Gradient Descent: Linearization of Convex Functions 130
2.5 Strongly Convex Regularizers 134
2.6 Online Mirror Descent 141
2.7 The Language of Duality 146
2.8 Bounds with Local Norms 152
2.9 Bibliographic Remarks 155

3 Online Classification **157**

3.1 Finite Hypothesis Class and Experts Advice 158
3.2 Learnability and the Standard Optimal Algorithm 160

3.3 Perceptron and Winnow 168
3.4 Bibliographic Remarks 175

4 Limited Feedback (Bandits) 177

4.1 Online Mirror Descent with Estimated Gradients 178
4.2 The Multi-armed Bandit Problem 179
4.3 Gradient Descent Without a Gradient 182
4.4 Bibliographic Remarks 185

5 Online-to-Batch Conversions 186

5.1 Bibliographic Remarks 190

Acknowledgments 191

References 192

Foundations and Trends® in
Machine Learning
Vol. 4, No. 2 (2011) 107–194

DOI: 10.1561/2200000018

Online Learning and Online Convex Optimization

Shai Shalev-Shwartz

Benin School of Computer Science and Engineering, The Hebrew University of Jerusalem, Israel, shais@cs.huji.ac.il

Abstract

Online learning is a well established learning paradigm which has both theoretical and practical appeals. The goal of online learning is to make a sequence of accurate predictions given knowledge of the correct answer to previous prediction tasks and possibly additional available information. Online learning has been studied in several research fields including game theory, information theory, and machine learning. It also became of great interest to practitioners due the recent emergence of large scale applications such as online advertisement placement and online web ranking. In this survey we provide a modern overview of online learning. Our goal is to give the reader a sense of some of the interesting ideas and in particular to underscore the centrality of convexity in deriving efficient online learning algorithms. We do not mean to be comprehensive but rather to give a high-level, rigorous yet easy to follow, survey.

1

Introduction

Online learning is the process of answering a sequence of questions given (maybe partial) knowledge of the correct answers to previous questions and possibly additional available information. The study of online learning algorithms is an important domain in machine learning, and one that has interesting theoretical properties and practical applications.

Online learning is performed in a sequence of consecutive rounds, where at round t the learner is given a question, $\mathbf{x}_t$, taken from an instance domain $\mathcal{X}$, and is required to provide an answer to this question, which we denote by p_t. After predicting an answer, the correct answer, y_t, taken from a target domain $\mathcal{Y}$, is revealed and the learner suffers a loss, $l(p_t, y_t)$, which measures the discrepancy between his answer and the correct one. While in many cases p_t is in $\mathcal{Y}$, it is sometimes convenient to allow the learner to pick a prediction from a larger set, which we denote by D.

```
Online Learning

for $t = 1, 2, \ldots$
  receive question $\mathbf{x}_t \in \mathcal{X}$
  predict $p_t \in D$
  receive true answer $y_t \in \mathcal{Y}$
  suffer loss $l(p_t, y_t)$
```

The specific case of yes/no answers and predictions, namely $D = \mathcal{Y} = \{0,1\}$, is called online classification. In this case it is natural to use the 0–1 loss function: $l(p_t, y_t) = |p_t - y_t|$. That is, $l(p_t, y_t)$ indicates if $p_t = y_t$ (the prediction is correct) or $p_t \neq y_t$ (the prediction is wrong).

For example, consider the problem of predicting whether it is going to rain tomorrow. On day t, the question $\mathbf{x}_t$ can be encoded as a vector of meteorological measurements. Based on these measurements, the learner should predict if it's going to rain tomorrow. In the following day, the learner knows the correct answer.

We can also allow the learner to output a prediction in $[0,1]$, which can be interpreted as the probability of raining tomorrow. This is an example of an application in which $D \neq \mathcal{Y}$. We can still use the loss function $l(p_t, y_t) = |p_t - y_t|$, which can now be interpreted as the probability to err if predicting that it's going to rain with probability p_t.

The learner's ultimate goal is to minimize the cumulative loss suffered along its run, which translates to making few prediction mistakes in the classification case. The learner tries to deduce information from previous rounds so as to improve its predictions on present and future questions. Clearly, learning is hopeless if there is no correlation between past and present rounds. Classic statistical theory of sequential prediction therefore enforces strong assumptions on the statistical properties of the input sequence (e.g., that it is sampled i.i.d. according to some unknown distribution).

In this review we survey methods which make no statistical assumptions regarding the origin of the sequence of examples. The sequence is allowed to be deterministic, stochastic, or even adversarially adaptive to the learner's own behavior (as in the case of spam email filtering). Naturally, an adversary can make the cumulative loss to 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 make nontrivial statements we must further restrict the problem. We consider two natural restrictions.

The first restriction is especially suited to the case of online classification. We assume that all the answers are generated by some target mapping, $h^\star : \mathcal{X} \to \mathcal{Y}$. Furthermore, $h^\star$ is taken from a fixed set, called a hypothesis class and denoted by $\mathcal{H}$, which is known to the learner. With this restriction on the sequence, which we call *the realizable case*, the learner should make as few mistakes as possible, assuming that both $h^\star$ and the sequence of questions can be chosen by an adversary. For an online learning algorithm, A, we denote by $M_A(\mathcal{H})$ the maximal number of mistakes A might make on a sequence of examples which is labeled by some $h^\star \in \mathcal{H}$. We emphasize again that both $h^\star$ and the sequence of questions can be chosen by an adversary. A bound on $M_A(\mathcal{H})$ is called a *mistake-bound* and we will study how to design algorithms for which $M_A(\mathcal{H})$ is minimal.

Alternatively, the second restriction of the online learning model we consider is a relaxation of the realizable assumption. We no longer assume that all answers are generated by some $h^\star \in \mathcal{H}$, but we require the learner to be competitive with the best fixed predictor from $\mathcal{H}$. This is captured by the *regret* of the algorithm, which measures how "sorry" the learner is, in retrospect, not to have followed the predictions of some hypothesis $h^\star \in \mathcal{H}$. Formally, the regret of the algorithm relative to $h^\star$ when running on a sequence of T examples is defined as

$$\mathrm{Regret}_T(h^\star) = \sum_{t=1}^{T} l(p_t, y_t) - \sum_{t=1}^{T} l(h^\star(x_t), y_t), \tag{1.1}$$

and the regret of the algorithm relative to a hypothesis class $\mathcal{H}$ is

$$\mathrm{Regret}_T(\mathcal{H}) = \max_{h^\star \in \mathcal{H}} \mathrm{Regret}_T(h^\star). \tag{1.2}$$

We restate the learner's goal as having the lowest possible regret relative to $\mathcal{H}$. We will sometime be satisfied with "low regret" algorithms, by which we mean that $\mathrm{Regret}_T(\mathcal{H})$ grows sub-linearly with the number of rounds, T, which implies that the difference between the *average* loss of the learner and the average loss of the best hypothesis in $\mathcal{H}$ tends to zero as T goes to infinity.

1.1 Examples

We already mentioned the problem of online classification. To make the discussion more concrete, we list several additional online prediction problems and possible hypothesis classes.

Online Regression In regression problems, $\mathcal{X} = \mathbb{R}^d$ which corresponds to a set of measurements (often called features), and $\mathcal{Y} = D = \mathbb{R}$. For example, consider the problem of estimating the fetal weight based on ultrasound measurements of abdominal circumference and femur length. Here, each $\mathbf{x} \in \mathcal{X} = \mathbb{R}^2$ is a two-dimensional vector corresponds to the measurements of the abdominal circumference and the femur length. Given these measurements the goal is to predict the fetal weight. Common loss functions for regression problems are the squared loss, $\ell(p, y) = (p - y)^2$, and the absolute loss, $\ell(p, y) = |p - y|$. Maybe the simplest hypothesis class for regression is the class of linear predictors, $\mathcal{H} = \{\mathbf{x} \mapsto \sum_{i=1}^{d} w[i]x[i] : \forall i, w[i] \in \mathbb{R}\}$, where $w[i]$ is the ith element of $\mathbf{w}$. The resulting problem is called *online linear regression.*

Prediction with Expert Advice On each online round the learner has to choose from the advice of d given experts. Therefore, $\mathbf{x}_t \in \mathcal{X} \subset \mathbb{R}^d$, where $x_t[i]$ is the advice of the ith expert, and $D = \{1, \ldots, d\}$. Then, the learner receives the true answer, which is a vector $\mathbf{y}_t \in \mathcal{Y} = [0, 1]^d$, where $y_t[i]$ is the cost of following the advice of the ith expert. The loss of the learner is the cost of the chosen expert, $\ell(p_t, \mathbf{y}_t) = y_t[p_t]$. A common hypothesis class for this problem is the set of constant predictors, $\mathcal{H} = \{h_1, \ldots, h_d\}$, where $h_i(\mathbf{x}) = i$ for all $\mathbf{x}$. This implies that the regret of the algorithm is measured relative to the performance of the strategies which always predict according to the same expert.

Online Ranking On round t, the learner receives a query $\mathbf{x}_t \in \mathcal{X}$ and is required to order k elements (e.g., documents) according to their relevance to the query. That is, D is the set of all permutations of $\{1,\ldots,k\}$. Then, the learner receives the true answer $y_t \in \mathcal{Y} = \{1,\ldots,k\}$, which corresponds to the document which best matches the query. In web applications, this is the document that the user clicked on. The loss, $\ell(p_t,y_t)$, is the position of y_t in the ranked list p_t.

1.2 A Gentle Start

We start with studying online classification problem, in which $\mathcal{Y} = \mathcal{D} = \{0,1\}$, and $\ell(p,y) = |p-y|$ is the 0–1 loss. That is, on each round, the learner receives $\mathbf{x}_t \in \mathcal{X}$ and is required to predict $p_t \in \{0,1\}$. Then, it receives $y_t \in \{0,1\}$ and pays the loss $|p_t - y_t|$. We make the following simplifying assumption:

- *Finite Hypothesis Class:* We assume that $|\mathcal{H}| < \infty$.

Recall that the goal of the learner is to have a low regret relative to the hypotheses set, $\mathcal{H}$, where each function in $\mathcal{H}$ is a mapping from $\mathcal{X}$ to $\{0,1\}$, and the regret is defined as

$$\text{Regret}_T(\mathcal{H}) = \max_{h\in\mathcal{H}} \left(\sum_{t=1}^{T} |p_t - y_t| - \sum_{t=1}^{T} |h(\mathbf{x}_t) - y_t| \right).$$

We first show that this is an impossible mission — no algorithm can obtain a sublinear regret bound even if $|\mathcal{H}| = 2$. Indeed, consider $\mathcal{H} = \{h_0,h_1\}$, where h_0 is the function that always returns 0 and h_1 is the function that always returns 1. 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, for any sequence of true answers, $y_1,\ldots,y_T$, let b be the majority of labels in $y_1,\ldots,y_T$, then the number of mistakes of h_b is at most $T/2$. Therefore, the regret of any online algorithm might be at least $T - T/2 = T/2$, which is not a sublinear with T. This impossibility result is attributed to Cover [13].

To sidestep Cover's impossibility result, we must further restrict the power of the adversarial environment. In the following we present two ways to do this.

1.2.1 Realizability Assumption

The first way to sidestep Cover's impossibility result is by making one additional assumption:

- *Realizability:* We assume that all target labels are generated by some $h^\star \in \mathcal{H}$, namely, $y_t = h^\star(x_t)$ for all t. Our goal is to design an algorithm with an optimal mistake bound. Namely, an algorithm for which $M_A(\mathcal{H})$ is minimal. See definition of $M_A(\mathcal{H})$ in the prequel.

Next, we describe and analyze online learning algorithms assuming both a finite hypothesis class and realizability of the input sequence. The most natural learning rule is to use (at any online round) any hypothesis which is consistent with all past examples.

```
Consistent

input: A finite hypothesis class $\mathcal{H}$
initialize: $V_1 = \mathcal{H}$
for $t = 1, 2, \ldots$
  receive $\mathbf{x}_t$
  choose any $h \in V_t$
  predict $p_t = h(\mathbf{x}_t)$
  receive true answer $y_t = h^\star(\mathbf{x}_t)$
  update $V_{t+1} = \{h \in V_t : h(\mathbf{x}_t) = y_t\}$
```

The `Consistent` algorithm maintains a set, V_t, of all the hypotheses which are consistent with $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_{t-1}, y_{t-1})$. This set is often called the version space. It then picks any hypothesis from V_t and predicts according to this hypothesis.

Obviously, whenever `Consistent` makes a prediction mistake, at least one hypothesis is removed from V_t. Therefore, after making M mistakes we have $|V_t| \leq |\mathcal{H}| - M$. Since V_t is always nonempty (by the realizability assumption it contains $h^\star$) we have $1 \leq |V_t| \leq |\mathcal{H}| - M$.

Rearranging, we obtain

Corollary 1.1. Let $\mathcal{H}$ be a finite hypothesis class. The `Consistent` algorithm enjoys the mistake bound $M_{\texttt{Consistent}}(\mathcal{H}) \leq |\mathcal{H}| - 1$.

It is rather easy to construct a hypothesis class and a sequence of examples on which `Consistent` will indeed make $|\mathcal{H}| - 1$ mistakes. Next, we present a better algorithm in which we choose $h \in V_t$ in a smarter way. We shall see that this algorithm is guaranteed to make exponentially fewer mistakes. The idea is to predict according to the majority of hypotheses in V_t rather than according to some arbitrary $h \in V_t$. That way, whenever we err, we are guaranteed to remove at least half of the hypotheses from the version space.

```
Halving

input: A finite hypothesis class $\mathcal{H}$
initialize: $V_1 = \mathcal{H}$
for $t = 1, 2, \ldots$
  receive $\mathbf{x}_t$
  predict $p_t = \text{argmax}_{r \in \{0,1\}} |\{h \in V_t : h(\mathbf{x}_t) = r\}|$
    (in case of a tie predict $p_t = 1$)
  receive true answer $y_t$
  update $V_{t+1} = \{h \in V_t : h(\mathbf{x}_t) = y_t\}$
```

Theorem 1.2. Let $\mathcal{H}$ be a finite hypothesis class. The `Halving` algorithm enjoys the mistake bound $M_{\texttt{Halving}}(\mathcal{H}) \leq \log_2(|\mathcal{H}|)$.

Proof. We simply note that whenever the algorithm errs we have $|V_{t+1}| \leq |V_t|/2$. (Hence the name Halving.) Therefore, if M is the total number of mistakes, we have

$$1 \leq |V_{T+1}| \leq |\mathcal{H}| \, 2^{-M}.$$

Rearranging the above inequality we conclude our proof. □

Of course, `Halving`'s mistake bound is much better than `Consistent`'s mistake bound. Is this the best we can do? What is an optimal algorithm for a given hypothesis class (not necessarily finite)? We will get back to this question in Section 3.

1.2.2 Randomization

In the previous subsection we sidestepped Cover's impossibility result by relying on the realizability assumption. This is a rather strong assumption on the environment. We now present a milder restriction on the environment and allow the learner to randomize his predictions. Of course, this by itself does not circumvent Cover's impossibility result as in deriving the impossibility result we assumed nothing on the learner's strategy. To make the randomization meaningful, we force the adversarial environment to decide on y_t without knowing the random coins flipped by the learner on round t. The adversary can still know the learner's forecasting strategy and even the random bits of previous rounds, but it doesn't know the actual value of the random bits used by the learner on round t. With this (mild) change of game, we analyze the *expected* 0–1 loss of the algorithm, where expectation is with respect to the learner's own randomization. That is, if the learner outputs $\hat{y}_t$ where $\mathbb{P}[\hat{y}_t = 1] = p_t$, then the expected loss he pays on round t is

$$\mathbb{P}[\hat{y}_t \neq y_t] = |p_t - y_t|.$$

Put another way, instead of having the predictions domain being $D = \{0,1\}$ we allow it to be $D = [0,1]$, and interpret $p_t \in D$ as the probability to predict the label 1 on round t. To summarize, we assume:

- *Randomized Predictions and Expected Regret*: We allow the predictions domain to be $D = [0,1]$ and the loss function is still $l(p_t, y_t) = |p_t - y_t|$.

With this assumption it is possible to derive a low regret algorithm as stated in the following theorem.

Theorem 1.3. Let $\mathcal{H}$ be a finite hypothesis class. There exists an algorithm for online classification, whose predictions come from $D = [0,1]$, that enjoys the regret bound

$$\sum_{t=1}^{T} |p_t - y_t| - \min_{h \in \mathcal{H}} \sum_{t=1}^{T} |h(\mathbf{x}_t) - y_t| \leq \sqrt{0.5 \ln(|\mathcal{H}|) T}.$$

We will provide a constructive proof of the above theorem in the next section.

To summarize, we have presented two different ways to sidestep Cover's impossibility result: realizability or randomization. At first glance, the two approaches seem to be rather different. However, there is a deep underlying concept that connects them. Indeed, we will show that both methods can be interpreted as *convexification* techniques. Convexity is a central theme in deriving online learning algorithms. We study it in the next section.

1.3 Organization and Scope

How to predict rationally is a key issue in various research areas such as game theory, machine learning, and information theory. The seminal book of Cesa-Bianchi and Lugosi [12] thoroughly investigates the connections between online learning, universal prediction, and repeated games. In particular, results from the different fields are unified using the prediction with expert advice framework.

We feel that convexity plays a central role in the derivation of online learning algorithms, and therefore start the survey with a study of the important sub-family of online learning problems, which is called *online convex optimization*. In this family, the prediction domain is a convex set and the loss function is a convex function with respect to its first argument. As we will show, many previously proposed algorithms for online classification and other problems can be jointly analyzed based on the online convex optimization framework. Furthermore, convexity is important because it leads to *efficient* algorithms.

In Section 3 we get back to the problem of online classification. We characterize a standard optimal algorithm for online classification. In addition, we show how online convex optimization can be used for deriving efficient online classification algorithms.

In Section 4 we study online learning in a limited feedback model, when the learner observes the loss value $l(p_t, y_t)$ but does not observe the actual correct answer y_t. We focus on the classic multi-armed bandit problem and derive an algorithm for this problem based on the online convex optimization algorithmic framework. We also present a low regret algorithm for the general problem of bandit online convex optimization.

Finally, in Section 5 we discuss several implications of online learning to batch learning problems, in which we assume that the examples are sampled i.i.d. from an unknown probability source.

Part of our presentation shares similarities with other surveys on online prediction problems. In particular, Rakhlin's lecture notes [34] and Hazan's book section [22] are good recent surveys on online convex optimization. While part of our presentation shares similarities with these surveys, we sometimes emphasize different techniques. Furthermore, we connect and relate the new results on online convex optimization to classic results on online classification, thus providing a fresh modern perspective on some classic algorithms. A more classic treatment can be found in Blum's survey [8].

1.4 Notation and Basic Definitions

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 $\langle \mathbf{x},\mathbf{w}\rangle$. Whenever we do not specify the vector space we assume that it is the d-dimensional Euclidean space and then $\langle \mathbf{x},\mathbf{w}\rangle = \sum_{i=1}^{d} x[i]w[i]$. 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 $\mathbb{R}_+$. The set of natural numbers is denoted by $\mathbb{N}$. For any $k \geq 1$, the set of integers $\{1,\ldots,k\}$ is denoted by $[k]$. Given a predicate π, we use the notation $\mathbf{1}_{[\pi]}$ to denote the indicator function that outputs 1 if π holds and 0 otherwise. The hinge function is denoted by $[a]_+ = \max\{0,a\}$.

The Euclidean (or ℓ_2) **norm** of a vector $\mathbf{w}$ is $\|\mathbf{w}\|_2 = \sqrt{\langle \mathbf{w},\mathbf{w}\rangle}$. We omit the subscript when it is clear from the context. We also use other ℓ_p

norms, $\|\mathbf{w}\|_p = (\sum_i |w[i]|^p)^{1/p}$, and in particular $\|\mathbf{w}\|_1 = \sum_i |w[i]|$ and $\|\mathbf{w}\|_\infty = \max_i |w[i]|$. A generic norm of a vector $\mathbf{w}$ is denoted by $\|\mathbf{w}\|$ and its **dual norm** is defined as

$$\|\mathbf{x}\|_\star = \max\{\langle \mathbf{w}, \mathbf{x}\rangle : \|\mathbf{w}\| \le 1\}.$$

The definition of the dual norm immediately implies the inequality

$$\langle \mathbf{w}, \mathbf{z}\rangle \le \|\mathbf{w}\| \, \|\mathbf{z}\|_\star. \tag{1.3}$$

For the ℓ_2 norm (which is dual to itself), this is the well known Cauchy–Schwartz inequality. For $p, q \ge 1$ such that $\frac{1}{p} + \frac{1}{q} = 1$ we have that the ℓ_p and ℓ_q norms are dual, and Equation (1.3) becomes Holder's inequality.

A function f is called L-**Lipschitz** over a set S with respect to a norm $\|\cdot\|$ if for all $\mathbf{u}, \mathbf{w} \in S$ we have $|f(\mathbf{u}) - f(\mathbf{w})| \le L\|\mathbf{u} - \mathbf{w}\|$.

The **gradient** of a differentiable function f is denoted by ∇f and the **Hessian** is denoted by $\nabla^2 f$.

Throughout the review, we make use of basic notions from convex analysis. A set S is **convex** if for all $\mathbf{w}, \mathbf{v} \in S$ and $\alpha \in [0,1]$ we have that $\alpha\mathbf{w} + (1-\alpha)\mathbf{v} \in S$ as well. Similarly, a function $f : S \to \mathbb{R}$ is convex if for all $\mathbf{w}, \mathbf{v}$ and $\alpha \in [0,1]$ we have $f(\alpha\mathbf{w} + (1-\alpha)\mathbf{v}) \le \alpha f(\mathbf{w}) + (1-\alpha) f(\mathbf{v})$.

It is convenient to allow convex functions to output the value ∞. The **domain** of a function f is the set of points on which f is finite. This is convenient, for example, for constraining the solution of an optimization problem to be within some set A. Indeed, instead of solving $\min_{\mathbf{x}\in A} f(\mathbf{x})$ we can solve $\min_{\mathbf{x}} f(\mathbf{x}) + I_A(\mathbf{x})$, where I_A is the function that outputs 0 if $\mathbf{x} \in A$ and ∞ if $\mathbf{x} \notin A$. In the next section we make use of some additional definitions and tools from convex analysis. For clarity, we define them as per need.

The expected value of a random variable, ψ, is denoted by $\mathbb{E}[\psi]$. In some situations, we have a deterministic function h that receives a random variable as input. We denote by $\mathbb{E}[h(\psi)]$ the expected value of the random variable $h(\psi)$. Occasionally, we omit the dependence of h on ψ. In this case, we may clarify the meaning of the expectation by using the notation $\mathbb{E}_\psi[h]$ or $\mathbb{E}_{\psi\sim P}[h]$ if ψ is distributed according to some distribution P.

2

Online Convex Optimization

In recent years, the design of many efficient online learning algorithms has been influenced by convex optimization tools. Furthermore, it was observed that most previously proposed efficient algorithms can be jointly analyzed based on the following elegant model:

```
Online Convex Optimization (OCO)

input: A convex set $S$
for $t = 1, 2, \ldots$
  predict a vector $\mathbf{w}_t \in S$
  receive a convex loss function $f_t : S \to \mathbb{R}$
  suffer loss $f_t(\mathbf{w}_t)$
```

In this section we describe algorithms for online convex optimization and analyze their regret. Recall that the regret of an online algorithm with respect to a competing hypothesis, which here will be some vector $\mathbf{u}$, is defined as

$$\mathrm{Regret}_T(\mathbf{u}) = \sum_{t=1}^{T} f_t(\mathbf{w}_t) - \sum_{t=1}^{T} f_t(\mathbf{u}). \tag{2.1}$$

As before, the regret of the algorithm relative to a set of competing vectors, U, is defined as

$$\mathrm{Regret}_T(U) = \max_{\mathbf{u} \in U} \mathrm{Regret}_T(\mathbf{u}).$$

Remark 2.1. (U vs. S) In the online convex optimization problem, the predictions of the learner should come from the set S, while we analyze the regret with respect to the set U. While in some situations it makes sense to set $U = S$, this is not always the case. Whenever we do not specify the value of U we use the default value $U = S$. Additionally, our default setting for S will be $S = \mathbb{R}^d$.

The rest of this section is organized as follows. We start with convexification techniques, showing how to utilize the online convex optimization framework in nonconvex problems. Next, we start describing and analyzing an algorithmic framework for online convex optimization. First, we describe the Follow-the-Leader approach, in which the learner simply picks the vector which performed best on past rounds. Next we describe a regularized form of Follow-the-Leader, which stabilizes the predictions of the algorithm, and show how stability leads to low regret. We proceed with deriving Online Gradient Descent and Online Mirror Descent from Follow-the-Regularized-Leader by a linearization trick. We derive several specific algorithms from the Online Mirror Descent framework. Finally, we describe additional proof techniques and also derive local-norm bounds, that will be used in the next sections.

2.1 Convexification

Some online prediction problems can be seamlessly cast in the online convex optimization framework.

Example 2.1 (Online linear regression). Recall the online linear regression problem described in Section 1.1. On each online round the learner first receives a vector of features, $\mathbf{x}_t \in A \subset \mathbb{R}^d$, and then needs to predict a scalar, p_t. Next, the learner receives the "true" target, $y_t \in \mathbb{R}$, and pays the loss $|p_t - y_t|$. The learner should be competitive with the set of linear predictors of the form $\mathbf{x} \mapsto \langle \mathbf{w}, \mathbf{x} \rangle$. If the predictions of the learner are also based on linear functions, then we can easily cast this online prediction problem in the online convex optimization framework as follows. The learner should decide on a vector $\mathbf{w}_t$, which yields the prediction $p_t = \langle \mathbf{w}_t, \mathbf{x}_t \rangle$. The loss function becomes $|p_t - y_t| = |\langle \mathbf{w}_t, \mathbf{x}_t \rangle - y_t|$. Therefore, letting $f_t(\mathbf{w}) = |\langle \mathbf{w}, \mathbf{x}_t \rangle - y_t|$, which is indeed a convex function, we obtain that $f_t(\mathbf{w}_t) = l(p_t, y_t)$.

Other online prediction problems do not seem to fit into the online convex optimization framework. For example, in online classification problems, the predictions domain or the loss functions are not convex. In this section we describe two convexification techniques that allow us to utilize the online convex optimization framework in additional scenarios.

2.1.1 Convexification by Randomization

To demonstrate the randomization technique, consider the problem of prediction with expert advice, where on each online round the learner has to choose from the advice of d given experts. Denote by $p_t \in \{1, \ldots, d\}$ the chosen expert. Then, the learner receives a vector $\mathbf{y}_t \in [0,1]^d$, where $y_t[i]$ is the cost of following the advice of the ith expert. The learner pays the loss $y_t[p_t]$. In this prediction problem, the decision space is discrete, hence nonconvex.

Furthermore, the problem of online classification with a finite hypothesis class we encountered in Section 1.2 can be easily cast as a special case of the prediction with expert advice problem. Therefore, Cover's impossibility result (see again Section 1.2) implies that there is no algorithm that can attain low regret for the prediction with expert advice problem.

However, as we show below, by allowing the learner to randomize his predictions we can cast the problem in the online convex optimization framework, and therefore can obtain low regret algorithm for this problem. Formally, let $S = \{\mathbf{w} \in \mathbb{R}^d : \mathbf{w} \geq 0 \wedge \|\mathbf{w}\|_1 = 1\}$ be the probability simplex, which forms a convex set. At round t, the learner chooses $\mathbf{w}_t \in S$ and based on $\mathbf{w}_t$ picks an expert at random according to $\mathbb{P}[p_t = i] = w_t[i]$. Then, the cost vector $\mathbf{y}_t$ is revealed and the learner pays for his expected cost

$$\mathbb{E}[y_t[p_t]] = \sum_{i=1}^{d} \mathbb{P}[p_t = i] y_t[i] = \langle \mathbf{w}_t, \mathbf{y}_t \rangle.$$

Note that by analyzing the expected cost of the learner we implicitly restrict the power of the adversarial environment — it cannot base the vector $\mathbf{y}_t$ on the random bits the learner employs on round t.

Now we can cast the problem as online convex optimization since S is a convex set and the loss function, $f_t(\mathbf{w}) = \langle \mathbf{w}, \mathbf{y}_t \rangle$, happens to be a linear function (hence, convex). Let the set of competing vectors, U, be the d singletons, namely the vectors of the form $(0,\ldots,0,1,0,\ldots,0)$. These vectors correspond to always following the advice of a single expert. Hence, a regret bound with respect to U implies a regret bound with respect to always predicting the advice of a single expert.

2.1.2 Convexification by Surrogate Loss Functions

To explain the second convexification technique we again start with the specific problem of online classification with a finite hypothesis class. Recall that one of the techniques we used to sidestep Cover's impossibility result relied on the realizability assumption. That is, we assumed that there exists $h^\star \in \mathcal{H}$ such that $y_t = h^\star(\mathbf{x}_t)$ for all t. With this assumption at hand, we described the `Halving` algorithm and showed that it makes at most $\log_2(|\mathcal{H}|)$ prediction mistakes.

We now derive a similar guarantee using the language of online convex optimization. Let us write $\mathcal{H} = \{h_1,\ldots,h_d\}$ and let $S = \{\mathbf{w} \in [0,1]^d : \sum_i w[i] = 1\}$ be the probability simplex. For each online round, define $\mathbf{v}_t = (h_1(\mathbf{x}_t),\ldots,h_d(\mathbf{x}_t)) \in \{0,1\}^d$. Our algorithm will maintain $\mathbf{w}_t \in S$ and will predict the label according to

$$p_t = \begin{cases} 1 & \text{if } \langle \mathbf{w}_t, \mathbf{v}_t \rangle \ge 1/2 \\ 0 & \text{if } \langle \mathbf{w}_t, \mathbf{v}_t \rangle < 1/2 \end{cases}$$

Let $\mathcal{M} = \{t : p_t \neq y_t\}$ be the rounds on which our algorithm makes a prediction mistake. We define

$$f_t(\mathbf{w}) = \begin{cases} 2|\langle \mathbf{w}, \mathbf{v}_t \rangle - y_t| & \text{if } t \in \mathcal{M} \quad p_t \neq y_t \\ 0 & \text{if } t \notin \mathcal{M}. \quad p_t = y_t \end{cases}$$

$p_t = 1 \quad y_t = 0 \quad |\langle w_t, v_t \rangle \geq \frac{1}{2}| \quad f_t \geq 1$

$p_t = 0 \quad y_t = 1 \quad |\langle w_t v_t \rangle < \frac{1}{2}| \quad f_t \geq 1$

Note that f_t depends on $\mathcal{M}$, and thus depends on $\mathbf{w}_t$. This does not pose any problem since in the online convex optimization model the environment picks the function f_t after observing $\mathbf{w}_t$. The two key properties of f_t are

- f_t is a convex function
- $f_t(\mathbf{w}_t) \geq |p_t - y_t|$, namely, the convex loss upper bounds the original nonconvex loss.

Hence the name *surrogate convex loss*. Since S is a convex set and f_t is a convex function for all t we have obtained an online convex optimization problem.

In the next sections we will derive algorithms for online convex optimization problems. In particular, one of these algorithms enjoys the regret bound

$$\forall \mathbf{u} \in S, \quad \sum_{t=1}^{T} f_t(\mathbf{w}_t) \leq \sum_{t=1}^{T} f_t(\mathbf{u}) + \frac{\log(d)}{\eta} + 2\eta \sum_{t=1}^{T} L_t,$$

where η is a parameter, which we will set here to be $\eta = 1/4$, and L_t is a Lipschitz parameter of the function f_t (with respect to the ℓ_1 norm). In our case, $L_t = 1$ if $t \in \mathcal{M}$ and $L_t = 0$ if $t \notin \mathcal{M}$. Hence,

$$\forall \mathbf{u} \in S, \quad \sum_{t=1}^{T} f_t(\mathbf{w}_t) \leq \sum_{t=1}^{T} f_t(\mathbf{u}) + 4\log(d) + \frac{1}{2}|\mathcal{M}|.$$

By the surrogate property of f_t, we can lower bound the left-hand side by $|\mathcal{M}|$. Rearranging, we obtain:

$$|\mathcal{M}| \leq 2\sum_{t=1}^{T} f_t(\mathbf{u}) + 8\log(d).$$

This type of bound, where the number of mistakes is upper bounded by the convex surrogate loss of a competing hypothesis, is often called a *relative loss bound*.

In the realizable case, we can further simplify the relative loss bound as follows. Since the bound holds for all $\mathbf{u} \in S$ it holds in particular for the vector $\mathbf{u} = (0,\ldots,0,1,0,\ldots,0)$, where the 1 is placed in the coordinate corresponding to the true hypothesis $h^\star$. By our construction, $f_t(\mathbf{u}) = 0$ for all t, which yields

$$|\mathcal{M}| \le 8\log(d).$$

We have obtained a mistake bound of the same order as the `Halving`'s mistake bound.

More generally, the first step of the technique involves a reparameterization of the problem such that the decision space becomes convex (instead of maintaing the set V_t in `Halving` we now maintain the vector $\mathbf{w}_t \in S$). In the second step we construct a function f_t of the predicted parameter that satisfies two requirements: It should be convex and it should upper bound the original loss function. Last, we would of course like to construct a convex surrogate for which there exists some $\mathbf{u} \in S$ that attains a low cumulative loss. Otherwise, the resulting bound will be meaningless. Typically, this is done by assuming more on the problem at hand. For example, in the above, the realizability assumption enabled us to construct a surrogate for which there was $\mathbf{u} \in S$ such that $f_t(\mathbf{u}) = 0$ for all t.

2.2 Follow-the-leader

By now, we hope that the reader is convinced that the online convex optimization framework is an important model, so we turn to deriving algorithms for online convex optimization.

The most natural learning rule is to use (at any online round) any vector which has minimal total loss on all past rounds. This is in the same spirit of the `Consistent` algorithm we encountered in Section 1.2 but in the context of online convex optimization it is usually referred to as `Follow-The-Leader`.

`Follow-The-Leader (FTL)`

$$\forall t, \quad \mathbf{w}_t = \operatorname*{argmin}_{\mathbf{w}\in S} \sum_{i=1}^{t-1} f_i(\mathbf{w}) \quad \text{(break ties arbitrarily)}$$

To analyze FTL, we first show that the regret of FTL is upper bounded by the cumulative difference between the loss of $\mathbf{w}_t$ and $\mathbf{w}_{t+1}$.

Lemma 2.1. Let $\mathbf{w}_1, \mathbf{w}_2, \ldots$ be the sequence of vectors produced by FTL. Then, for all $\mathbf{u} \in S$ we have

$$\text{Regret}_T(\mathbf{u}) = \sum_{t=1}^{T} (f_t(\mathbf{w}_t) - f_t(\mathbf{u})) \leq \sum_{t=1}^{T} (f_t(\mathbf{w}_t) - f_t(\mathbf{w}_{t+1})).$$

Proof. Subtracting $\sum_t f_t(\mathbf{w}_t)$ from both sides of the inequality and rearranging, the desired inequality can be rewritten as

$$\sum_{t=1}^{T} f_t(\mathbf{w}_{t+1}) \leq \sum_{t=1}^{T} f_t(\mathbf{u}).$$

We prove this inequality by induction. The base case of $T = 1$ follows directly from the definition of $\mathbf{w}_{t+1}$. Assume the inequality holds for $T - 1$, then for all $\mathbf{u} \in S$ we have

$w_{t+1} = \arg\min_u f_t(u)$

$$\sum_{t=1}^{T-1} f_t(\mathbf{w}_{t+1}) \leq \sum_{t=1}^{T-1} f_t(\mathbf{u}).$$

Adding $f_T(\mathbf{w}_{T+1})$ to both sides we get

$$\sum_{t=1}^{T} f_t(\mathbf{w}_{t+1}) \leq f_T(\mathbf{w}_{T+1}) + \sum_{t=1}^{T-1} f_t(\mathbf{u}).$$

$= \min$

The above holds for all $\mathbf{u}$ and in particular for $\mathbf{u} = \mathbf{w}_{T+1}$. Thus,

$$\sum_{t=1}^{T} f_t(\mathbf{w}_{t+1}) \leq \sum_{t=1}^{T} f_t(\mathbf{w}_{T+1}) = \min_{\mathbf{u} \in S} \sum_{t=1}^{T} f_t(\mathbf{u}),$$

where the last equation follows from the definition of $\mathbf{w}_{T+1}$. This concludes our inductive argument. □

We next use Lemma 2.1 to derive a regret bound for the following sub-family of online convex optimization.

Definition 2.1 (Online Quadratic Optimization). This is an online convex optimization problem where at each round $f_t(\mathbf{w}) = \frac{1}{2}\|\mathbf{w} - \mathbf{z}_t\|_2^2$ for some vector $\mathbf{z}_t$.

We further assume that $S = \mathbb{R}^d$. For this case, it is easy to verify that the FTL rule becomes

$$\mathbf{w}_t = \frac{1}{t-1}\sum_{i=1}^{t-1}\mathbf{z}_i,$$

namely, $\mathbf{w}_t$ is the average of $\mathbf{z}_1,\ldots,\mathbf{z}_{t-1}$. Note that we can rewrite

$$\mathbf{w}_{t+1} = \frac{1}{t}(\mathbf{z}_t + (t-1)\mathbf{w}_t) = \left(1 - \frac{1}{t}\right)\mathbf{w}_t + \frac{1}{t}\mathbf{z}_t$$

which yields

$$\mathbf{w}_{t+1} - \mathbf{z}_t = \left(1 - \frac{1}{t}\right)(\mathbf{w}_t - \mathbf{z}_t).$$

Therefore,

$$\begin{aligned} f_t(\mathbf{w}_t) - f_t(\mathbf{w}_{t+1}) &= \frac{1}{2}\|\mathbf{w}_t - \mathbf{z}_t\|^2 - \frac{1}{2}\|\mathbf{w}_{t+1} - \mathbf{z}_t\|^2 \\ &= \frac{1}{2}\left(1 - \left(1 - \frac{1}{t}\right)^2\right)\|\mathbf{w}_t - \mathbf{z}_t\|^2 \\ &\le \frac{1}{t}\|\mathbf{w}_t - \mathbf{z}_t\|^2. \end{aligned}$$

Let $L = \max_t \|\mathbf{z}_t\|$. Since $\mathbf{w}_t$ is the average of $\mathbf{z}_1,\ldots,\mathbf{z}_{t-1}$ we have that $\|\mathbf{w}_t\| \le L$ and therefore, by the triangle inequality, $\|\mathbf{w}_t - \mathbf{z}_t\| \le 2L$. We have therefore obtained:

$$\sum_{t=1}^{T}(f_t(\mathbf{w}_t) - f_t(\mathbf{w}_{t+1})) \le (2L)^2\sum_{t=1}^{T}\frac{1}{t}.$$

Combining the above with Lemma 2.1 and using the inequality $\sum_{t=1}^{T} 1/t \le \log(T) + 1$ we conclude that

---

Corollary 2.2. Consider running FTL on an Online Quadratic Optimization problem with $S = \mathbb{R}^d$ and let $L = \max_t \|\mathbf{z}_t\|$. Then, the regret of FTL with respect to all vectors $\mathbf{u} \in \mathbb{R}^d$ is at most $4L^2(\log(T) + 1)$.

While the above result seems promising, we next show that the FTL rule does not guarantee low regret for another important sub-family.

Definition 2.2 (Online Linear Optimization). This is an online convex optimization problem where at each round $f_t(\mathbf{w}) = \langle \mathbf{w}, \mathbf{z}_t \rangle$ for some vector $\mathbf{z}_t$.

Example 2.2 (Failure of FTL). Let $S = [-1,1] \subset \mathbb{R}$ and consider the sequence of linear functions such that $f_t(w) = z_t w$ where

$$z_t = \begin{cases} -0.5 & \text{if } t = 1 \\ 1 & \text{if } t \text{ is even} \\ -1 & \text{if } t > 1 \ \wedge \ t \text{ is odd} \end{cases}$$

Then, the predictions of FTL will be to set $w_t = 1$ for t odd and $w_t = -1$ for t even. The cumulative loss of the FTL algorithm will therefore be T while the cumulative loss of the fixed solution $u = 0 \in S$ is 0. Thus, the regret of FTL is T !

Intuitively, FTL fails in the above example because its predictions are *not stable* — $\mathbf{w}_t$ shifts drastically from round to round where we only added a single loss function to the objective of the optimization problem in the definition of $\mathbf{w}_t$. In contrast, FTL works fine for the quadratic game since $\mathbf{w}_{t+1}$ is "close" to $\mathbf{w}_t$. One way to stabilize FTL is by adding regularization, which is the topic of the next section.

2.3 Follow-the-Regularized-Leader

Follow-the-Regularized-Leader is a natural modification of the basic FTL algorithm in which we minimize the loss on all past rounds plus a regularization term. The goal of the regularization term is to stabilize the solution. Formally, for a regularization function, $R : S \to \mathbb{R}$ we define

```
Follow-the-Regularized-Leader (FoReL)
```

$$\forall t, \quad \mathbf{w}_t = \operatorname*{argmin}_{\mathbf{w} \in S} \sum_{i=1}^{t-1} f_i(\mathbf{w}) + R(\mathbf{w})$$

Naturally, different regularization functions will yield different algorithms with different regret bounds. We discuss properties of different regularization functions later. But, first, let us specify FoReL for the case of linear functions and squared-ℓ_2-norm regularization, which we often call the Euclidean regularization case.

Example 2.3. Consider the Online Linear Optimization problem where $f_t(\mathbf{w}) = \langle \mathbf{w}, \mathbf{z}_t \rangle$ and let $S = \mathbb{R}^d$. Suppose we run FoReL with the regularization function $R(\mathbf{w}) = \frac{1}{2\eta}\|\mathbf{w}\|_2^2$ for some positive scalar η. Then, it is easy to verify that

$$\mathbf{w}_{t+1} = -\eta \sum_{i=1}^{t} \mathbf{z}_i = \mathbf{w}_t - \eta \mathbf{z}_t. \tag{2.2}$$

Note that $\mathbf{z}_t$ is the gradient of f_t at $\mathbf{w}_t$ (in fact, at any point). Therefore, the recursive rule, $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \mathbf{z}_t$, can be rewritten as $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \nabla f_t(\mathbf{w}_t)$. Hence, this rule is often called Online Gradient Descent. We shall re-visit the Online Gradient Descent rule for general convex functions in the next section.

We next turn to the analysis of FoReL. As with the analysis of FTL, we first relate the regret of FoReL to the cumulative difference between the loss of $\mathbf{w}_t$ and $\mathbf{w}_{t+1}$.

Lemma 2.3. Let $\mathbf{w}_1, \mathbf{w}_2, \ldots$ be the sequence of vectors produced by FoReL. Then, for all $\mathbf{u} \in S$ we have

$$\sum_{t=1}^{T} (f_t(\mathbf{w}_t) - f_t(\mathbf{u})) \le R(\mathbf{u}) - R(\mathbf{w}_1) + \sum_{t=1}^{T} (f_t(\mathbf{w}_t) - f_t(\mathbf{w}_{t+1})).$$

Proof. Observe that running FoReL on $f_1, \ldots, f_T$ is equivalent to running FTL on $f_0, f_1, \ldots, f_T$ where $f_0 = R$. Using Lemma 2.1 we obtain

$$\sum_{t=0}^{T} (f_t(\mathbf{w}_t) - f_t(\mathbf{u})) \le \sum_{t=0}^{T} (f_t(\mathbf{w}_t) - f_t(\mathbf{w}_{t+1})).$$

Rearranging the above and using $f_0 = R$ we conclude our proof. □

zero term = regularization,

Based on the above lemma we can easily derive a regret bound for online linear optimization with the regularizer $R(\mathbf{w}) = \frac{1}{2\eta}\|\mathbf{w}\|_2^2$.

Theorem 2.4. Consider running FoReL on a sequence of linear functions, $f_t(\mathbf{w}) = \langle \mathbf{w}, \mathbf{z}_t \rangle$ for all t, with $S = \mathbb{R}^d$, and with the regularizer $R(\mathbf{w}) = \frac{1}{2\eta}\|\mathbf{w}\|_2^2$, which yields the predictions given in Equation (2.2). Then, for all $\mathbf{u}$ we have

$$\mathrm{Regret}_T(\mathbf{u}) \le \frac{1}{2\eta}\|\mathbf{u}\|_2^2 + \eta \sum_{t=1}^{T} \|\mathbf{z}_t\|_2^2.$$

In particular, consider the set $U = \{\mathbf{u} : \|\mathbf{u}\| \le B\}$ and let L be such that $\frac{1}{T}\sum_{t=1}^{T}\|\mathbf{z}_t\|_2^2 \le L^2$, then by setting $\eta = \frac{B}{L\sqrt{2T}}$ we obtain

$$\mathrm{Regret}_T(U) \ \le \ BL\sqrt{2T}.$$

Proof. Using Lemma 2.3 and Equation (2.2),

$$\begin{aligned}
\mathrm{Regret}_T(\mathbf{u}) &\le R(\mathbf{u}) - R(\mathbf{w}_1) + \sum_{t=1}^{T}(f_t(\mathbf{w}_t) - f_t(\mathbf{w}_{t+1})) \\
&= \frac{1}{2\eta}\|\mathbf{u}\|_2^2 + \sum_{t=1}^{T}\langle \mathbf{w}_t - \mathbf{w}_{t+1}, \mathbf{z}_t \rangle \\
&= \frac{1}{2\eta}\|\mathbf{u}\|_2^2 + \eta\sum_{t=1}^{T}\|\mathbf{z}_t\|_2^2. \qquad \square
\end{aligned}$$

The parameter η in the above theorem depends on the time horizon T. It is possible to derive a similar result without using the time horizon. In the next subsection we show a generic way (although not always optimal) to get rid of the dependence on the time horizon.

We see that the Euclidean regularization function guarantees low regret for linear functions with bounded ℓ_2-norm because it stabilizes the predictions. We shall later generalize the above result in two aspects. First, we allow any sequence of Lipschitz functions (rather than linear functions with bounded norm). Second, we consider other regularization functions which guarantee stability in other scenarios.

2.3.1 The Doubling Trick

In Theorem 2.4, the parameter η depends on the time horizon T. We now show how to get rid of this dependence by a simple trick.

Consider an algorithm that enjoys a regret bound of the form $\alpha\sqrt{T}$, but its parameters require the knowledge of T. The doubling trick, described below, enables us to convert such an algorithm into an algorithm that does not need to know the time horizon. The idea is to divide the time into periods of increasing size and run the original algorithm on each period.

```
The Doubling Trick

input: algorithm $A$ whose parameters depend on the time horizon
for $m = 0, 1, 2, \ldots$
  run $A$ on the $2^m$ rounds $t = 2^m, \ldots, 2^{m+1} - 1$
```

The regret of A on each period of 2^m rounds is at most $\alpha\sqrt{2^m}$. Therefore, the total regret is at most

$$\begin{aligned}\sum_{m=1}^{\lceil \log_2(T)\rceil} \alpha\sqrt{2^m} &= \alpha \sum_{m=1}^{\lceil \log_2(T)\rceil} (\sqrt{2})^m \\ &= \alpha \frac{1 - \sqrt{2}^{\lceil \log_2(T)\rceil + 1}}{1 - \sqrt{2}} \\ &\le \alpha \frac{1 - \sqrt{2T}}{1 - \sqrt{2}} \\ &\le \frac{\sqrt{2}}{\sqrt{2} - 1} \alpha\sqrt{T}.\end{aligned}$$

That is, we obtain that the regret is worse by a constant multiplicative factor.

2.4 Online Gradient Descent: Linearization of Convex Functions

In the previous section we introduced the FoReL approach and analyzed it for the case of linear functions, $S = \mathbb{R}^d$, and Euclidean regularization.

We now generalize this result by deriving a simple reduction from convex functions to linear functions.

To do so, we use an important property of convex functions, which is in fact an alternative characterization of convexity, as given by the following lemma.

Lemma 2.5. Let S be a convex set. A function $f : S \to \mathbb{R}$ is convex iff for all $\mathbf{w} \in S$ there exists $\mathbf{z}$ such that

$$\forall \mathbf{u} \in S, \quad f(\mathbf{u}) \geq f(\mathbf{w}) + \langle \mathbf{u} - \mathbf{w}, \mathbf{z} \rangle. \tag{2.3}$$

In words, convexity is characterized by the existence of tangents that lie below the function. The proof of this lemma can be found in many convex analysis textbooks (e.g., [9]).

Definition 2.3 (sub-gradients). A vector $\mathbf{z}$ that satisfies Equation (2.3) is called a *sub-gradient* of f at $\mathbf{w}$. The set of sub-gradients of f at $\mathbf{w}$ is denoted $\partial f(\mathbf{w})$. Furthermore, if f is differentiable at $\mathbf{w}$ then $\partial f(\mathbf{w})$ contains a single element — the gradient of f at $\mathbf{w}$, $\nabla f(\mathbf{w})$.

An illustration of sub-gradients is given in Figure 2.1.

Getting back to online convex optimization, for each round t, there exists $\mathbf{z}_t$ such that for all $\mathbf{u}$,

$$f_t(\mathbf{w}_t) - f_t(\mathbf{u}) \leq \langle \mathbf{w}_t - \mathbf{u}, \mathbf{z}_t \rangle.$$

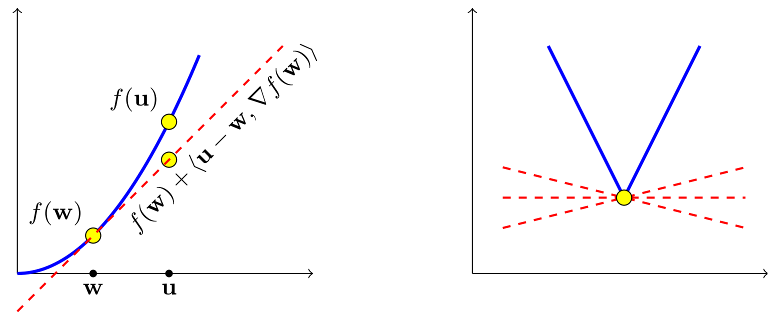

$f(\mathbf{u})$
$f(\mathbf{w}) + \langle \mathbf{u} - \mathbf{w}, \nabla f(\mathbf{w}) \rangle$
$f(\mathbf{w})$
$\mathbf{w}$
$\mathbf{u}$

Fig. 2.1 Left: The right-hand side of Equation (2.3) is the tangent of f at $\mathbf{w}$. For a convex function, the tangent lower bounds f. Right: Illustration of several sub-gradients of a non-differentiable convex function.

It follows that for any sequence of convex functions $f_1,\ldots,f_T$ and vectors $\mathbf{w}_1,\ldots,\mathbf{w}_T$, if for all t, $\mathbf{z}_t \in \partial f_t(\mathbf{w}_t)$ (namely, it is a sub-gradient) then

$$\sum_{t=1}^{T}(f_t(\mathbf{w}_t) - f_t(\mathbf{u})) \leq \sum_{t=1}^{T}(\langle \mathbf{w}_t, \mathbf{z}_t\rangle - \langle \mathbf{u}, \mathbf{z}_t\rangle). \tag{2.4}$$

In words, the regret of a procedure that generates the vectors $\mathbf{w}_1,\ldots,\mathbf{w}_T$ for the sequence of linear functions upper bounds the regret with respect to the convex functions $f_1,\ldots,f_T$.

Note that in this construction, $\mathbf{z}_t$ depends on $\mathbf{w}_t$. As mentioned previously, this does not pose any problem since we allow the adversarial environment to base its loss function on the vector predicted by the learner.

Combining the above observation with the FoReL procedure with Euclidean regularization (see Equation (2.2)) yields the Online Gradient Descent algorithm:

```
Online Gradient Descent (OGD)

parameter: $\eta > 0$
initialize: $\mathbf{w}_1 = \mathbf{0}$
update rule: $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta\mathbf{z}_t$   where $\mathbf{z}_t \in \partial f_t(\mathbf{w}_t)$
```

To analyze OGD, we combine Equation (2.4) with the analysis of FoReL for linear functions given in Theorem 2.4, to get that

$$\mathrm{Regret}_T(\mathbf{u}) \leq \frac{1}{2\eta}\|\mathbf{u}\|_2^2 + \eta\sum_{t=1}^{T}\|\mathbf{z}_t\|_2^2. \tag{2.5}$$

This regret bound depends on the norms of the sub-gradients of the vectors produced by the algorithm, and is therefore not satisfactory. To derive a more concrete bound, we must assure that the norms of sub-gradients will not be excessively large. One way to do this is by assuming that the functions are Lipschitz. The following lemma relates norms of sub-gradients to Lipschitzness of f_t.

Lemma 2.6. Let $f : S \to \mathbb{R}$ be a convex function. Then, f is L-Lipschitz over S with respect to a norm $\|\cdot\|$ iff for all $\mathbf{w} \in S$ and $\mathbf{z} \in \partial f(\mathbf{w})$ we have that $\|\mathbf{z}\|_\star \le L$, where $\|\cdot\|_\star$ is the dual norm.

Proof. Assume that f is Lipschitz. Choose some $\mathbf{w} \in S, \mathbf{z} \in \partial f(\mathbf{w})$. Let $\mathbf{u}$ be such that $\mathbf{u} - \mathbf{w} = \operatorname{argmax}_{\mathbf{v}:\|\mathbf{v}\|=1} \langle \mathbf{v}, \mathbf{z} \rangle$. Therefore, $\langle \mathbf{u} - \mathbf{w}, \mathbf{z} \rangle = \|\mathbf{z}\|_\star$. From the definition of the sub-gradient,

$$f(\mathbf{u}) - f(\mathbf{w}) \ge \langle \mathbf{z}, \mathbf{u} - \mathbf{w} \rangle = \|\mathbf{z}\|_\star.$$

On the other hand, from the Lipschitzness of f we have

$$L = L\|\mathbf{u} - \mathbf{w}\| \ge f(\mathbf{u}) - f(\mathbf{w}).$$

Combining the above two inequalities we conclude that $\|\mathbf{z}\|_\star \le L$. For the other direction, since $\mathbf{z} \in \partial f(\mathbf{w})$ we also have

$$f(\mathbf{w}) - f(\mathbf{u}) \le \langle \mathbf{z}, \mathbf{w} - \mathbf{u} \rangle.$$

Combining the above with Equation (1.3) we obtain

$$f(\mathbf{w}) - f(\mathbf{u}) \le \|\mathbf{z}\|_\star \|\mathbf{w} - \mathbf{u}\| \le L\,\|\mathbf{w} - \mathbf{u}\|,$$

hence f is L-Lipschitz. □

Therefore, the term $\sum_{t=1}^T \|\mathbf{z}_t\|_2^2$ given in Equation (2.5) can be bounded by $\sum_{t=1}^T L_t^2$, where L_t is the Lipschitz constant of f_t. We conclude:

Corollary 2.7. Assume that OGD is run on a sequence $f_1, \ldots, f_T$ of convex functions. Then, for all $\mathbf{u}$ we have

$$\mathrm{Regret}_T(\mathbf{u}) \le \frac{1}{2\eta}\|\mathbf{u}\|_2^2 + \eta \sum_{t=1}^T \|\mathbf{z}_t\|_2^2.$$

If we further assume that each f_t is L_t-Lipschitz with respect to $\|\cdot\|_2$, and let L be such that $\frac{1}{T}\sum_{t=1}^T L_t^2 \le L^2$. Then, for all $\mathbf{u}$, the regret of OGD satisfies

$$\mathrm{Regret}_T(\mathbf{u}) \le \frac{1}{2\eta}\|\mathbf{u}\|_2^2 + \eta T L^2.$$

In particular, if $U = \{\mathbf{u} : \|\mathbf{u}\|_2 \leq B\}$ and $\eta = \frac{B}{L\sqrt{2T}}$ then

$$\text{Regret}_T(U) \ \leq \ BL\sqrt{2T}.$$

Let us now discuss the consequences of Corollary 2.7, starting with the online linear regression problem (Example 2.1). Recall that for this example, $f_t(\mathbf{w}) = |\langle \mathbf{w}, \mathbf{x}_t \rangle - y_t|$, where $\mathbf{x}_t$ comes from a set A. If the set A is contained in a ball of ℓ_2 radius L, then f_t is L-Lipschitz with respect to the ℓ_2 norm. We therefore obtain a regret bound of $BL\sqrt{2T}$ which holds for all competing vectors $\mathbf{u}$ with $\|\mathbf{u}\|_2 \leq B$.

For the problem of prediction with expert advice (see Section 2.1.1), we cannot apply the OGD framework as it does not guarantee that $\mathbf{w}_t$ will always be in the probability simplex. In the next section we describe the FoReL framework with other regularization functions. In particular, we present regularization functions appropriate for the problem of prediction with expert advice.

2.5 Strongly Convex Regularizers

So far we applied FoReL with the Euclidean regularization function. As mentioned at the end of the previous section, this regularization cannot be used for the learning with expert advice problem. In this section we consider other regularization functions and underscore strong convexity as an important property of regularization functions that yields meaningful regret bounds.

2.5.1 Strong Convexity

Intuitively, a function is strongly convex if it grows faster than a linear function. To give a precise definition, recall that for a convex function f, at any point $\mathbf{w}$ we can find a linear function (the “tangent”) which equals to f at $\mathbf{w}$ and does not exceed f at any other point (see Lemma 2.5). A function is strongly convex if f is strictly above the tangent, and the difference can be quantified as follows:

Definition 2.4. A function $f : S \to \mathbb{R}$ is σ-strongly-convex over S with respect to a norm $\|\cdot\|$ if for any $\mathbf{w} \in S$ we have

$$\forall \mathbf{z} \in \partial f(\mathbf{w}), \quad \forall \mathbf{u} \in S, \quad f(\mathbf{u}) \geq f(\mathbf{w}) + \langle \mathbf{z}, \mathbf{u} - \mathbf{w} \rangle + \frac{\sigma}{2}\|\mathbf{u} - \mathbf{w}\|^2.$$

A graphical illustration is given in Figure 2.2.

An important property of strong convexity that we use is the following:

Lemma 2.8. Let S be a nonempty convex set. Let $f : S \to \mathbb{R}$ be a σ-strongly-convex function over S with respect to a norm $\|\cdot\|$. Let $\mathbf{w} = \mathrm{argmin}_{\mathbf{v} \in S} f(\mathbf{v})$. Then, for all $\mathbf{u} \in S$

$$f(\mathbf{u}) - f(\mathbf{w}) \geq \frac{\sigma}{2}\|\mathbf{u} - \mathbf{w}\|^2.$$

Proof. To give intuition, assume first that f is differentiable and $\mathbf{w}$ is in the interior of S. Then, $\nabla f(\mathbf{w}) = \mathbf{0}$ and therefore, by the definition of strong convexity we have

$$\forall \mathbf{u} \in S, \;\; f(\mathbf{u}) - f(\mathbf{w}) \geq \langle \nabla f(\mathbf{w}), \mathbf{u} - \mathbf{w} \rangle + \frac{\sigma}{2}\|\mathbf{u} - \mathbf{w}\|^2 = \frac{\sigma}{2}\|\mathbf{u} - \mathbf{w}\|^2,$$

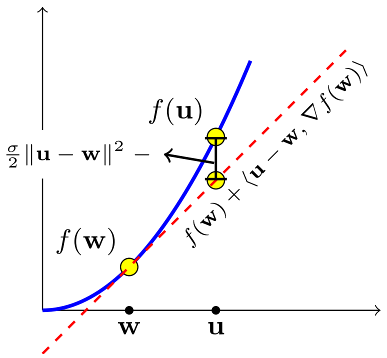

$f(\mathbf{u})$
$\frac{\sigma}{2}\|\mathbf{u} - \mathbf{w}\|^2$ −
$f(\mathbf{w})$
$f(\mathbf{w}) + \langle \mathbf{u} - \mathbf{w}, \nabla f(\mathbf{w}) \rangle$
$\mathbf{w}$
$\mathbf{u}$

Fig. 2.2 Illustrating strong-convexity: the distance between f and its tangent at $\mathbf{w}$ is at least $\frac{\sigma}{2}\|\mathbf{u} - \mathbf{w}\|^2$.

as required. Even if $\mathbf{w}$ is on the boundary of S we still have that for all $\mathbf{u} \in S$, $\langle \nabla f(\mathbf{w}), \mathbf{u} - \mathbf{w} \rangle \geq 0$ (otherwise, $\mathbf{w}$ would not have been optimal since we can make a small step in the direction $\mathbf{u} - \mathbf{w}$ and decrease the value of f). So, the desired inequality still holds. Finally, to give the formal proof for a non-differentiable f, let $g : \mathbb{R}^d \to \mathbb{R} \cup \{\infty\}$ be such that $g(\mathbf{w}) = f(\mathbf{w})$ if $\mathbf{w} \in S$ and $g(\mathbf{w}) = \infty$ otherwise. We can therefore rewrite $\mathbf{w} = \operatorname{argmin}_{\mathbf{v}} g(\mathbf{v})$. Since g is a proper[1] convex function we have that $\mathbf{0} \in \partial g(\mathbf{w})$. The inequality follows by using the strong-convexity of g. □

If R is twice differentiable, then it is easy to verify that a sufficient condition for strong convexity of R is that for all $\mathbf{w}, \mathbf{x}$, $\langle \nabla^2 R(\mathbf{w})\mathbf{x}, \mathbf{x} \rangle \geq \sigma \|\mathbf{x}\|^2$, where $\nabla^2 R(\mathbf{w})$ is the Hessian matrix of R at $\mathbf{w}$, namely, the matrix of second-order partial derivatives of R at $\mathbf{w}$ [39, Lemma 14].

Example 2.4 (Euclidean regularization). The function $R(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|_2^2$ is 1-strongly-convex with respect to the ℓ_2 norm over $\mathbb{R}^d$. To see this, simply note that the Hessian of R at any $\mathbf{w}$ is the identity matrix.

Example 2.5 (Entropic regularization). The function $R(\mathbf{w}) = \sum_{i=1}^d w[i] \log(w[i])$ is $\frac{1}{B}$-strongly-convex with respect to the ℓ_1 norm over the set $S = \{\mathbf{w} \in \mathbb{R}^d : \mathbf{w} > \mathbf{0} \wedge \|\mathbf{w}\|_1 \leq B\}$. In particular, R is 1-strongly-convex over the probability simplex, which is the positive vectors whose elements sum to 1.

To see this note that

$$\begin{aligned}\langle \nabla^2 R(\mathbf{w})\mathbf{x}, \mathbf{x} \rangle &= \sum_i \frac{x[i]^2}{w[i]} = \frac{1}{\|\mathbf{w}\|_1} \left(\sum_i w[i] \right) \left(\sum_i \frac{x[i]^2}{w[i]} \right) \\ &\geq \frac{1}{\|\mathbf{w}\|_1} \left(\sum_i \sqrt{w[i]} \frac{|x[i]|}{\sqrt{w[i]}} \right)^2 = \frac{\|\mathbf{x}\|_1^2}{\|\mathbf{w}\|_1}, \qquad (2.6)\end{aligned}$$

where the inequality follows from Cauchy–Schwartz inequality.

[1] A convex function is proper if it never receives the value $-\infty$ and it receives a finite value at least once. For such functions, an optimality condition for $\mathbf{w}$ being a minimum is that the zero vector is a sub-gradient of the function at $\mathbf{w}$.

Additional useful properties are given in the following lemma, whose proof follows directly from the definition of strong convexity.

Lemma 2.9. If R is 1-strongly convex over S with respect to some norm then σR is σ-strongly-convex over S with respect to the same norm. In addition, if S' is a convex subset of S, then R is 1-strongly convex over S' as well.

2.5.2 Analyzing FoReL with Strongly Convex Regularizers

We now analyze FoReL with strongly convex regularizers. Recall the regret bound given in Lemma 2.3:

$$\sum_{t=1}^{T}(f_t(\mathbf{w}_t) - f_t(\mathbf{u})) \leq R(\mathbf{u}) - R(\mathbf{w}_1) + \sum_{t=1}^{T}(f_t(\mathbf{w}_t) - f_t(\mathbf{w}_{t+1})).$$

If f_t is L-Lipschitz with respect to a norm $\|\cdot\|$ then

$$f_t(\mathbf{w}_t) - f_t(\mathbf{w}_{t+1}) \leq L\,\|\mathbf{w}_t - \mathbf{w}_{t+1}\|.$$

Therefore, we need to ensure that $\|\mathbf{w}_t - \mathbf{w}_{t+1}\|$ is small. The following lemma shows that if the regularization function $R(\mathbf{w})$ is strongly convex with respect to the same norm, then $\mathbf{w}_t$ will be close to $\mathbf{w}_{t+1}$.

Lemma 2.10. Let $R : S \to \mathbb{R}$ be a σ-strongly-convex function over S with respect to a norm $\|\cdot\|$. Let $\mathbf{w}_1, \mathbf{w}_2, \ldots$ be the predictions of the FoReL algorithm. Then, for all t, if f_t is L_t-Lipschitz with respect to $\|\cdot\|$ then

$$f_t(\mathbf{w}_t) - f_t(\mathbf{w}_{t+1}) \leq L_t\|\mathbf{w}_t - \mathbf{w}_{t+1}\| \leq \frac{L_t^2}{\sigma}.$$

Proof. For all t let $F_t(\mathbf{w}) = \sum_{i=1}^{t-1} f_i(\mathbf{w}) + R(\mathbf{w})$ and note that the FoReL rule is $\mathbf{w}_t = \operatorname{argmin}_{\mathbf{w}\in S} F_t(\mathbf{w})$. Note also that F_t is σ-strongly-convex since the addition of a convex function to a strongly convex function keeps the strong convexity property. Therefore, Lemma 2.8 implies that:

$$F_t(\mathbf{w}_{t+1}) \geq F_t(\mathbf{w}_t) + \frac{\sigma}{2}\|\mathbf{w}_t - \mathbf{w}_{t+1}\|^2.$$

Repeating the same argument for F_{t+1} and its minimizer $\mathbf{w}_{t+1}$ we get

$$F_{t+1}(\mathbf{w}_t) \geq F_{t+1}(\mathbf{w}_{t+1}) + \frac{\sigma}{2}\|\mathbf{w}_t - \mathbf{w}_{t+1}\|^2.$$

Summing the above two inequalities and rearranging we obtain

$$\sigma\|\mathbf{w}_t - \mathbf{w}_{t+1}\|^2 \leq f_t(\mathbf{w}_t) - f_t(\mathbf{w}_{t+1}). \tag{2.7}$$

Next, using the Lipschitzness of f_t we get that

$$f_t(\mathbf{w}_t) - f_t(\mathbf{w}_{t+1}) \leq L_t\|\mathbf{w}_t - \mathbf{w}_{t+1}\|.$$

Combining with Equation (2.7) and rearranging we get that $\|\mathbf{w}_t - \mathbf{w}_{t+1}\| \leq L/\sigma$ and together with the above we conclude our proof. □

Combining the above Lemma with Lemma 2.3 we obtain

Theorem 2.11. Let $f_1, \ldots, f_T$ be a sequence of convex functions such that f_t is L_t-Lipschitz with respect to some norm $\|\cdot\|$. Let L be such that $\frac{1}{T}\sum_{t=1}^{T} L_t^2 \leq L^2$. Assume that FoReL is run on the sequence with a regularization function which is σ-strongly-convex with respect to the same norm. Then, for all $\mathbf{u} \in S$,

$$\mathrm{Regret}_T(\mathbf{u}) \leq R(\mathbf{u}) - \min_{\mathbf{v} \in S} R(\mathbf{v}) + TL^2/\sigma.$$

2.5.3 Derived Bounds

We now derive concrete bounds from Theorem 2.11. We start with the simplest case of Euclidean regularization, which is 1-strongly convex over $\mathbb{R}^d$, hence the following corollary follows.

Corollary 2.12. Let $f_1, \ldots, f_T$ be a sequence of convex functions such that f_t is L_t-Lipschitz with respect to $\|\cdot\|_2$. Let L be such that $\frac{1}{T}\sum_{t=1}^{T} L_t^2 \leq L^2$. Assume that FoReL is run on the sequence with the regularization function $R(\mathbf{w}) = \frac{1}{2\eta}\|\mathbf{w}\|_2^2$. Then, for all $\mathbf{u}$,

$$\mathrm{Regret}_T(\mathbf{u}) \leq \frac{1}{2\eta}\|\mathbf{u}\|_2^2 + \eta T L^2.$$

In particular, if $U = \{\mathbf{u} : \|\mathbf{u}\|_2 \leq B\}$ and $\eta = \frac{B}{L\sqrt{2T}}$ then

$$\text{Regret}_T(U) \leq BL\sqrt{2T}.$$

Observe that the bound we obtained is identical to the bound of Online-Gradient-Descent given in Corollary 2.7.

As mentioned in the previous section, the Euclidean regularization cannot be applied to the problem of prediction with expert advice since it does not enforce $\mathbf{w}_t$ to be in the probability simplex. A simple solution is to enforce the constraint $\mathbf{w}_t \in S$ by setting $R(\mathbf{w}) = \infty$ whenever $\mathbf{w} \notin S$. In light of Lemma 2.9, the resulting regularization function remains strongly convex on S and we obtain the following corollary.

Corollary 2.13. Assume that the conditions of Corollary 2.12 hold. Let S be a convex set and consider running FoReL with the regularization function

$$R(\mathbf{w}) = \begin{cases} \dfrac{1}{2\eta}\|\mathbf{w}\|_2^2 & \text{if } \mathbf{w} \in S \\ \infty & \text{if } \mathbf{w} \notin S \end{cases}$$

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

$$\text{Regret}_T(\mathbf{u}) \leq \frac{1}{2\eta}\|\mathbf{u}\|_2^2 + \eta T L^2.$$

In particular, if $B \geq \max_{\mathbf{u}\in S}\|\mathbf{u}\|_2$ and $\eta = \frac{B}{L\sqrt{2T}}$ then

$$\text{Regret}_T(S) \;\leq\; BL\sqrt{2T}.$$

We can apply the regularization function given in the above corollary to the problem of prediction with expert advice. In this case, S is the probability simplex and $\mathbf{x}_t \in [0,1]^d$. Hence, we can set $B = 1$ and $L = \sqrt{d}$ which leads to the regret bound $\sqrt{2dT}$. We next show another regularization function which leads to a regret bound of $\sqrt{2\log(d)T}$. The improvement is based on the following corollary of Theorem 2.11.

Corollary 2.14. Let $f_1,\ldots,f_T$ be a sequence of convex functions such that f_t is L_t-Lipschitz with respect to $\|\cdot\|_1$. Let L be such that $\frac{1}{T}\sum_{t=1}^{T} L_t^2 \le L^2$. Assume that FoReL is run on the sequence with the regularization function $R(\mathbf{w}) = \frac{1}{\eta}\sum_i w[i]\log(w[i])$ and with the set $S = \{\mathbf{w} : \|\mathbf{w}\|_1 = B \wedge \mathbf{w} > \mathbf{0}\} \subset \mathbb{R}^d$. Then,

$$\mathrm{Regret}_T(S) \le \frac{B\log(d)}{\eta} + \eta B T L^2.$$

In particular, setting $\eta = \frac{\sqrt{\log d}}{L\sqrt{2T}}$ yields

$$\mathrm{Regret}_T(S) \le BL\sqrt{2\log(d)T}.$$

The Entropic regularization used in the above corollary is strongly convex with respect to the ℓ_1 norm, and therefore the Lipschitzness requirement of the loss functions is also with respect to the ℓ_1-norm. For linear functions, $f_t(\mathbf{w}) = \langle \mathbf{w}, \mathbf{x}_t\rangle$, we have by Holder inequality that,

$$|f_t(\mathbf{w}) - f_t(\mathbf{u})| = |\langle \mathbf{w} - \mathbf{u}, \mathbf{x}_t\rangle| \le \|\mathbf{w} - \mathbf{u}\|_1 \|\mathbf{x}_t\|_\infty.$$

Therefore, the Lipschitz parameter grows with the ℓ_∞ norm of $\mathbf{x}_t$ rather than the ℓ_2 norm of $\mathbf{x}_t$. Applying this to the problem of prediction with expert advice (with $B = 1$ and $L = 1$), we obtain the regret bound of $\sqrt{2\log(d)T}$.

More generally, it is interesting to compare the two bounds given in Corollaries 2.13 and 2.14. Apart from the extra $\log(d)$ factor that appears in Corollary 2.14, both bounds look similar. However, the parameters B, L have different meanings in the two bounds. In Corollary 2.12, the parameter B imposes an ℓ_2 constraint on $\mathbf{u}$ and the parameter L captures Lipschitzness of the loss functions with respect to the ℓ_2 norm. In contrast, in Corollary 2.14 the parameter B imposes an ℓ_1 constraint on $\mathbf{u}$ (which is stronger than an ℓ_2 constraint) while the parameter L captures Lipschitzness with respect to the ℓ_1 norm (which is weaker than Lipschitzness with respect to the ℓ_2 norm). Therefore, the choice of the regularization function should depend on the Lipschitzness of the loss functions and on prior assumptions on the set of competing vectors (does a competing vector have a small ℓ_1 norm or only a small ℓ_2 norm). In the prediction with expert advice, the competing vector would be a singleton, for which both the ℓ_2 and ℓ_1 norms are 1. On the other hand, the gap between Lipschitzness with respect to ℓ_2 norm (which was $\sqrt{d}$) and Lipschitzness with respect to ℓ_1 norm (which was 1) is large. Therefore, we should prefer the Entropic regularization for the problem of prediction with expert advice.

2.6 Online Mirror Descent

In the previous section we analyzed the FoReL approach in the presence of a general regularization. A possible disadvantage of the FoReL approach is that it requires solving an optimization problem at each online round. In this section we derive and analyze the family of Online Mirror Descent algorithms from the FoReL framework. We will show that Online Mirror Descent achieves the same regret bound as FoReL but the update step is much simpler.

The starting point is to apply FoReL on a sequence of linear functions in which $f_t(\mathbf{w}) = \langle \mathbf{w}, \mathbf{z}_t \rangle$ with some regularization function $R(\mathbf{w})$. Throughout, we assume that $R(\mathbf{w}) = \infty$ for $\mathbf{w} \notin S$. Using the notation $\mathbf{z}_{1:t} = \sum_{i=1}^{t} \mathbf{z}_i$ we can rewrite the prediction of FoReL as follows:

$$\begin{aligned}\mathbf{w}_{t+1} &= \underset{\mathbf{w}}{\operatorname{argmin}}\, R(\mathbf{w}) + \sum_{i=1}^{t} \langle \mathbf{w}, \mathbf{z}_t \rangle \\ &= \underset{\mathbf{w}}{\operatorname{argmin}}\, R(\mathbf{w}) + \langle \mathbf{w}, \mathbf{z}_{1:t} \rangle \\ &= \underset{\mathbf{w}}{\operatorname{argmax}} \langle \mathbf{w}, -\mathbf{z}_{1:t} \rangle - R(\mathbf{w}).\end{aligned}$$

Letting

$$g(\boldsymbol{\theta}) = \underset{\mathbf{w}}{\operatorname{argmax}}\ \langle \mathbf{w}, \boldsymbol{\theta} \rangle - R(\mathbf{w}), \tag{2.8}$$

we can rewrite the FoReL prediction based on the following recursive update rule:

1. $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \mathbf{z}_t$
2. $\mathbf{w}_{t+1} = g(\boldsymbol{\theta}_{t+1})$

Now, if f_t is convex but nonlinear, we can use the same technique we used for deriving the Online Gradient Descent algorithm and use sub-gradients of f_t at $\mathbf{w}_t$ to linearize the problem. That is, letting $\mathbf{z}_t$ be a sub-gradient of f_t at $\mathbf{w}_t$ we have that for all $\mathbf{u}$

$$f_t(\mathbf{w}_t) - f_t(\mathbf{u}) \leq \langle \mathbf{w}_t, \mathbf{z}_t \rangle - \langle \mathbf{u}, \mathbf{z}_t \rangle.$$

Summing over t we obtain that the regret with respect to the nonlinear loss functions is upper bounded by the regret with respect to the linear functions. This yields the Online Mirror Descent framework.

```
Online Mirror Descent (OMD)

parameter: a link function $g : \mathbb{R}^d \to S$
initialize: $\boldsymbol{\theta}_1 = \mathbf{0}$
for $t = 1, 2, \ldots$
  predict $\mathbf{w}_t = g(\boldsymbol{\theta}_t)$
  update $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \mathbf{z}_t$ where $\mathbf{z}_t \in \partial f_t(\mathbf{w}_t)$
```

Clearly, Online Gradient Descent is a special case of Online Mirror Descent that is obtained by setting $S = \mathbb{R}^d$ and $g(\boldsymbol{\theta}) = \eta\boldsymbol{\theta}$, for some $\eta > 0$. When g is nonlinear we obtain that the vector $\boldsymbol{\theta}$ is updated by subtracting the gradient out of it, but the actual prediction is “mirrored” or “linked” to the set S via the function g. Hence the name Online Mirror Descent, and this is why g is often referred to as a link function.

Before we derive concrete algorithms from the OMD framework we give a generic bound for the OMD family based on our analysis of the FoReL rule.

Theorem 2.15. Let R be a $(1/\eta)$-strongly-convex function over S with respect to a norm $\|\cdot\|$. Assume that OMD is run on the sequence with a link function

$$g(\boldsymbol{\theta}) = \operatorname*{argmax}_{\mathbf{w} \in S}(\langle \mathbf{w}, \boldsymbol{\theta} \rangle - R(\mathbf{w})). \tag{2.9}$$

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

$$\mathrm{Regret}_T(\mathbf{u}) \leq R(\mathbf{u}) - \min_{\mathbf{v} \in S} R(\mathbf{v}) + \eta \sum_{t=1}^{T} \|\mathbf{z}_t\|_\star^2,$$

where $\|\cdot\|_\star$ is the dual norm. Furthermore, if f_t is L_t-Lipschitz with respect to $\|\cdot\|$, then we can further upper bound $\|\mathbf{z}_t\|_\star \leq L_t$.

Proof. As we have shown previously,

$$\sum_{t=1}^{T} (f_t(\mathbf{w}_t) - f_t(\mathbf{u})) \leq \sum_{t=1}^{T} \langle \mathbf{w}_t - \mathbf{u}, \mathbf{z}_t \rangle,$$

and the OMD algorithm is equivalent to running FoReL on the sequence of linear functions with the regularization $R(\mathbf{w})$. The theorem now follows directly from Theorem 2.11 and Lemma 2.6. □

2.6.1 Derived Algorithms

We now derive additional algorithms from the OMD framework.

The first algorithm we derive is often called normalized Exponentiated Gradient. In this algorithm, $S = \{\mathbf{w} : \|\mathbf{w}\|_1 = 1 \wedge \mathbf{w} \geq \mathbf{0}\}$ is the probability simplex and $g : \mathbb{R}^d \to \mathbb{R}^d$ is the vector valued function whose ith component is the function

$$g_i(\boldsymbol{\theta}) = \frac{e^{\eta \theta[i]}}{\sum_j e^{\eta \theta[j]}}. \tag{2.10}$$

Therefore,

$$\begin{aligned} w_{t+1}[i] &= \frac{e^{\eta \theta_{t+1}[i]}}{\sum_j e^{\eta \theta_{t+1}[j]}} = \frac{e^{\eta \theta_{t+1}[i]}}{\sum_j e^{\eta \theta_{t+1}[j]}} \cdot \frac{\sum_k e^{\eta \theta_t[k]}}{\sum_k e^{\eta \theta_t[k]}} \\ &= \frac{e^{\eta \theta_t[i]} e^{-\eta z_t[i]}}{\sum_j e^{\eta \theta_t[j]} e^{-\eta z_t[j]}} \cdot \frac{\sum_k e^{\eta \theta_t[k]}}{\sum_k e^{\eta \theta_t[k]}} \\ &= \frac{w_t[i] e^{-\eta z_t[i]}}{\sum_j w_t[j] e^{-\eta z_t[j]}}. \end{aligned}$$

```
Normalized Exponentiated Gradient
(normalized-EG)

parameter: $\eta > 0$
initialize: $\mathbf{w}_1 = (1/d, \ldots, 1/d)$
update rule $\forall i,\ w_{t+1}[i] = \frac{w_t[i] e^{-\eta z_t[i]}}{\sum_j w_t[j] e^{-\eta z_t[j]}}$ where $\mathbf{z}_t \in \partial f_t(\mathbf{w}_t)$
```

To analyze the normalized-EG algorithm, we rely on Theorem 2.15. Let $R(\mathbf{w}) = \frac{1}{\eta}\sum_i w[i]\log(w[i])$ be the entropic regularization, let S be the probability simplex, and recall that R is $(1/\eta)$-strongly convex over S with respect to the ℓ_1 norm. Using the technique of Lagrange multipliers, it is easy to verify that the solution to the optimization problem given in Equation (2.9) is the value of the link function given in Equation (2.10). Therefore, Theorem 2.15 yields:

Corollary 2.16. The normalized EG algorithm enjoys the regret bound given in Corollary 2.14 (with $B = 1$).

Next, we derive an algorithm which is called Online Gradient Descent with Lazy Projections. To derive this algorithm, let S be a convex set and define

$$g(\boldsymbol{\theta}) = \operatorname*{argmin}_{\mathbf{w} \in S} \|\mathbf{w} - \eta\boldsymbol{\theta}\|_2.$$

That is, $g(\boldsymbol{\theta})$ returns the point in S which is closest to $\eta\boldsymbol{\theta}$.

```
Online Gradient Descent with Lazy Projections

parameters: $\eta > 0$ and a convex set $S$
initialize: $\boldsymbol{\theta}_1 = \mathbf{0}$
for $t = 1, 2, \ldots, T$
  $\mathbf{w}_t = \operatorname{argmin}_{\mathbf{w} \in S} \|\mathbf{w} - \eta\boldsymbol{\theta}_t\|_2$
  $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \mathbf{z}_t$ where $\mathbf{z}_t \in \partial f_t(\mathbf{w}_t)$
```

To analyze the above algorithm we consider the Euclidean regularization function $R(\mathbf{w}) = \frac{1}{2\eta}\|\mathbf{w}\|_2^2$, which is $(1/\eta)$-strongly convex over S

with respect to the ℓ_2 norm. We have that

$$\begin{aligned}\operatorname*{argmax}_{\mathbf{w}\in S}(\langle \mathbf{w},\boldsymbol{\theta}\rangle - R(\mathbf{w})) &= \operatorname*{argmax}_{\mathbf{w}\in S}(\langle \mathbf{w},\eta\boldsymbol{\theta}\rangle - \frac{1}{2}\|\mathbf{w}\|_2^2) \\ &= \operatorname*{argmin}_{\mathbf{w}\in S}\frac{1}{2}\|\mathbf{w}-\eta\boldsymbol{\theta}\|_2^2 = \operatorname*{argmin}_{\mathbf{w}\in S}\|\mathbf{w}-\eta\boldsymbol{\theta}\|_2.\end{aligned}$$

We therefore conclude that:

Corollary 2.17. Online Gradient Descent with Lazy Projections enjoys the same regret bound given in Corollary 2.13.

Finally, we derive the p-norm algorithm, in which $S = \mathbb{R}^d$ and

$$g_i(\boldsymbol{\theta}) = \eta\,\frac{\operatorname{sign}(\theta[i])|\theta[i]|^{p-1}}{\|\boldsymbol{\theta}\|_p^{p-2}},$$

where $p \geq 2$ is a parameter and

$$\|\boldsymbol{\theta}\|_p = \left(\sum_{i=1}^{d}|\theta[i]|^p\right)^{1/p}.$$

```
                    $p$-norm

parameters: $\eta > 0$ and $p > 2$
initialize: $\boldsymbol{\theta}_1 = \mathbf{0}$
for $t = 1,2,\ldots,T$
  $\forall i,\ w_{t,i} = \eta\frac{\operatorname{sign}(\theta_t[i])|\theta_t[i]|^{p-1}}{\|\boldsymbol{\theta}_t\|_p^{p-2}}$
  $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \mathbf{z}_t$ where $\mathbf{z}_t \in \partial f_t(\mathbf{w}_t)$
```

To analyze the p-norm algorithm consider the regularization function $R(\mathbf{w}) = \frac{1}{2\eta(q-1)}\|\mathbf{w}\|_q^2$, where $q = p/(p-1)$. It is possible to show that if $q \in (1,2]$ then G is $(1/\eta)$-strongly convex over $\mathbb{R}^d$ with respect to the ℓ_q norm (see for example [39, Lemma 17]). It is also possible to verify that $g(\boldsymbol{\theta}) = \operatorname{argmax}_{\mathbf{w}}\langle \mathbf{w},\boldsymbol{\theta}\rangle - R(\mathbf{w})$. We therefore conclude:

Corollary 2.18. Let $f_1,\ldots,f_T$ be a sequence of convex functions such that f_t is L_t-Lipschitz over $\mathbb{R}^d$ with respect to $\|\cdot\|_q$. Let L be such that $\frac{1}{T}\sum_{t=1}^{T} L_t^2 \le L^2$. Then, for all $\mathbf{u}$, the regret of the p-norm algorithm satisfies

$$\text{Regret}_T(\mathbf{u}) \le \frac{1}{2\eta(q-1)}\|\mathbf{w}\|_q^2 + \eta T L^2.$$

In particular, if $U = \{\mathbf{u} : \|\mathbf{u}\|_q \le B\}$ and $\eta = \frac{B}{L\sqrt{2T/(q-1)}}$ then

$$\text{Regret}_T(U) \le BL\sqrt{\frac{2T}{q-1}}.$$

When $q = 2$ the link function becomes $g(\boldsymbol{\theta}) = \eta\boldsymbol{\theta}$ and the p-norm algorithm boils down to the Online Gradient Descent algorithm. When q is close to 1 it is not very hard to see that the p-norm algorithm behaves like the Entropic regularization. In particular, when $p = \log(d)$ we can obtain a regret bound similar to the regret bound of the EG algorithm. Intermediate values of q enables us to interpolate between the properties of the Entropic and Euclidean regularizations.

2.7 The Language of Duality

In the previous sections we relied on the FoReL framework for deriving online learning algorithms. In this section we present a different proof technique that relies on duality. There are several reasons to consider this different approach. First, in some cases, it is easier to derive tighter bounds based on the duality approach. In particular, we will tighten the regret bounds we derived for the OMD framework by a factor of $\sqrt{2}$ and we will also derive tighter bounds that involve the so-called local norms rather than fixed norms. Second, it may become convenient for developing new algorithms. Last, many previous papers on online learning uses the language of duality, so the reader may find this section useful for understanding the previous literature.

We start by giving some background on Fenchel conjugacy and duality. For more details, see for example [9].

2.7.1 Fenchel Conjugacy

There are two equivalent representations of a convex function. Either as pairs $(x, f(x))$ or as the set of tangents of f, namely pairs of the form

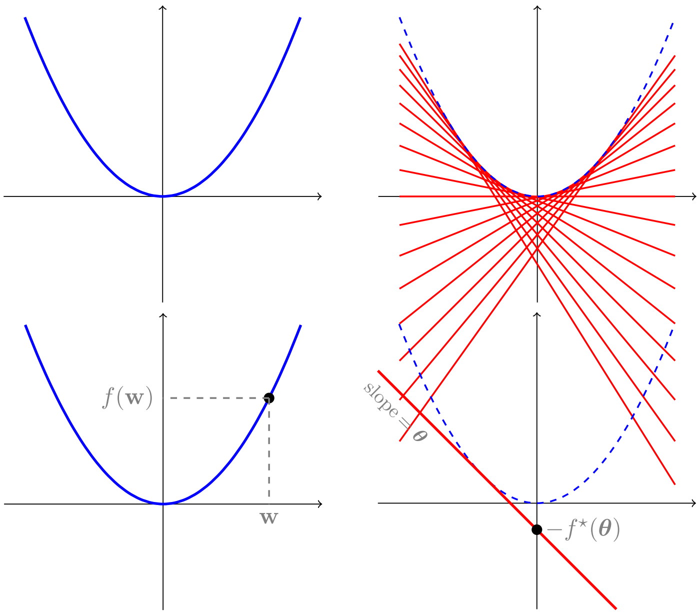

$f(\mathbf{w})$
$\mathbf{w}$
slope $= \boldsymbol{\theta}$
$-f^\star(\boldsymbol{\theta})$

Fig. 2.3 Illustrating Fenchel Conjugacy. We can represent a function as a set of points (**left**) or as a set of tangents (**right**).

(slope,intersection-with-y-axis). See Figure 2.3 for an illustration. The function that relates slopes of tangents to their intersection with the y axis is called the Fenchel conjugate of f, and is formally defined as

$$f^\star(\boldsymbol{\theta}) = \max_{\mathbf{u}} \langle \mathbf{u}, \boldsymbol{\theta} \rangle - f(\mathbf{u}).$$

It is possible to show that $f = (f^\star)^\star$ if and only if f is a closed[2] and convex function (see [9, Theorem 4.2.1]). From now on, we always assume that our functions are closed and convex.

The definition of Fenchel conjugacy immediately implies **Fenchel–Young inequality**:

$$\forall \mathbf{u}, \quad f^\star(\boldsymbol{\theta}) \geq \langle \mathbf{u}, \boldsymbol{\theta} \rangle - f(\mathbf{u}). \tag{2.11}$$

It is possible to show that equality holds if $\mathbf{u}$ is a sub-gradient of $f^\star$ at $\boldsymbol{\theta}$ and in particular, if $f^\star$ is differentiable, equality holds when $\mathbf{u} = \nabla f^\star(\boldsymbol{\theta})$.

[2] f is closed if its epigraph, i.e., the set $\{(\mathbf{w}, y) : f(\mathbf{w}) \leq y\}$, is a closed set.

Table 2.1. Example of Fenchel conjugate pairs.

$f(\mathbf{w})$	$f^\star(\boldsymbol{\theta})$	Comments
$\frac{1}{2}\|\mathbf{w}\|_2^2$	$\frac{1}{2}\|\boldsymbol{\theta}\|_2^2$	
$\frac{1}{2}\|\mathbf{w}\|_q^2$	$\frac{1}{2}\|\boldsymbol{\theta}\|_p^2$	where $\frac{1}{p}+\frac{1}{q}=1$
$\sum_i w[i]\log(w[i]) + I_{\{\mathbf{v}\geq 0:\|\mathbf{v}\|_1=1\}}(\mathbf{w})$	$\log(\sum_i e^{\theta[i]})$	(normalized Entropy)
$\sum_i w[i](\log(w[i])-1)$	$\sum_i e^{\theta[i]}$	(un-normalized Entropy)
$\frac{1}{\eta}g(\mathbf{w})$	$\frac{1}{\eta}g^\star(\eta\boldsymbol{\theta})$	where $\eta>0$
$g(\mathbf{w})+\langle\mathbf{w},\mathbf{x}\rangle$	$g^\star(\boldsymbol{\theta}-\mathbf{x})$	

Table 2.1 lists several Fenchel conjugate pairs. Recall that given a set S, we use the notation

$$I_S(\mathbf{w}) = \begin{cases} 0 & \text{if } \mathbf{w}\in S \\ \infty & \text{if } \mathbf{w}\notin S \end{cases}$$

2.7.2 Bregman Divergences and the Strong/Smooth Duality

A differentiable function R defines a Bregman divergence between two vectors as follows:

$$D_R(\mathbf{w}\|\mathbf{u}) = R(\mathbf{w}) - (R(\mathbf{u}) + \langle\nabla R(\mathbf{u}),\mathbf{w}-\mathbf{u}\rangle). \tag{2.12}$$

That is, the Bregman divergence is the difference, at the point $\mathbf{w}$, between R and its linearization around $\mathbf{u}$. When R is convex the Bregman divergence is always non-negative. However, it is not a metric measure because it is not symmetric and also does not satisfy the triangle inequality. An illustration is given in Figure 2.4.

When $R(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|_2^2$ the Bregman divergence becomes $D_R(\mathbf{w}\|\mathbf{u}) = \frac{1}{2}\|\mathbf{w}-\mathbf{u}\|_2^2$. When $R(\mathbf{w}) = \sum_i w[i]\log(w[i])$ the Bregman divergence between two vectors in the probability simplex becomes the Kullback–Leibler divergence, $D_R(\mathbf{w}\|\mathbf{u}) = \sum_i w[i]\log\frac{w[i]}{u[i]}$.

Recall the definition of strong-convexity (Definition 2.4). If R is differentiable, we can rewrite the σ-strong-convexity requirement as

$$D_R(\mathbf{w}\|\mathbf{u}) \geq \frac{\sigma}{2}\|\mathbf{w}-\mathbf{u}\|^2.$$

A related property is *strong-smoothness*.

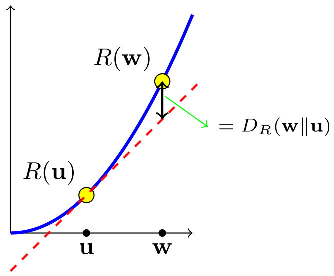


Fig. 2.4 Illustrating the Bregman divergence.

Definition 2.5. A function R is σ-strongly-smooth with respect to a norm $\|\cdot\|$ if it is differentiable and for all $\mathbf{u}, \mathbf{w}$ we have

$$D_R(\mathbf{w}\|\mathbf{u}) \leq \frac{\sigma}{2}\|\mathbf{w} - \mathbf{u}\|^2 .$$

Not surprisingly, strong convexity and strong smoothness are dual properties.

Lemma 2.19. (Strong/Smooth Duality) Assume that R is a closed and convex function. Then R is β-strongly convex with respect to a norm $\|\cdot\|$ if and only if $R^\star$ is $\frac{1}{\beta}$-strongly smooth with respect to the dual norm $\|\cdot\|_\star$.

The proof can be found, for instance, in [44] (In particular, see Corollary 3.5.11 on p. 217 and Remark 3.5.3 on p. 218). The above lemma implies in particular that if R is strongly convex then $R^\star$ is differentiable. Based on Section 2.7.1, this also implies that

$$\nabla R^\star(\boldsymbol{\theta}) = \underset{\mathbf{w}}{\operatorname{argmax}} \left(\langle \mathbf{w}, \boldsymbol{\theta} \rangle - R(\mathbf{w})\right). \tag{2.13}$$

2.7.3 Analyzing OMD using Duality

Recall that the OMD rule is

$$\mathbf{w}_t = g(\boldsymbol{\theta}_t) = g(-\mathbf{z}_{1:t-1}),$$

where the link function g is (see Equation (2.8) in Section 2.6)

$$g(\boldsymbol{\theta}) = \underset{\mathbf{w}}{\operatorname{argmax}}(\langle \mathbf{w}, \boldsymbol{\theta} \rangle - R(\mathbf{w})).$$

Based on Equation (2.13) we can also rewrite $g(\boldsymbol{\theta}) = \nabla R^\star(\boldsymbol{\theta})$.

To analyze OMD, we first use the following lemma.

Lemma 2.20. Suppose that OMD is run with a link function $g = \nabla R^\star$. Then, its regret is upper bounded by

$$\sum_{t=1}^{T} \langle \mathbf{w}_t - \mathbf{u}, \mathbf{z}_t \rangle \leq R(\mathbf{u}) - R(\mathbf{w}_1) + \sum_{t=1}^{T} D_{R^\star}(-\mathbf{z}_{1:t} \| - \mathbf{z}_{1:t-1}).$$

Furthermore, equality holds for the vector $\mathbf{u}$ that minimizes $R(\mathbf{u}) + \sum_t \langle \mathbf{u}, \mathbf{z}_t \rangle$.

Proof. First, using Fenchel–Young inequality we have

$$R(\mathbf{u}) + \sum_{t=1}^{T} \langle \mathbf{u}, \mathbf{z}_t \rangle = R(\mathbf{u}) - \langle \mathbf{u}, -\mathbf{z}_{1:T} \rangle \geq -R^\star(-\mathbf{z}_{1:T}),$$

where equality holds for the vector $\mathbf{u}$ that maximizes $\langle \mathbf{u}, -\mathbf{z}_{1:T} \rangle - R(\mathbf{u})$ hence minimizes $R(\mathbf{u}) + \langle \mathbf{u}, \mathbf{z}_{1:T} \rangle$. Second, using the fact that $\mathbf{w}_t = \nabla R^\star(-\mathbf{z}_{1:t-1})$ and the definition of the Bregman divergence, we can rewrite the right-hand side as

$$\begin{aligned} -R^\star(-\mathbf{z}_{1:T}) &= -R^\star(\mathbf{0}) - \sum_{t=1}^{T} (R^\star(-\mathbf{z}_{1:t}) - R^\star(-\mathbf{z}_{1:t-1})) \\ &= -R^\star(\mathbf{0}) + \sum_{t=1}^{T} (\langle \mathbf{w}_t, \mathbf{z}_t \rangle - D_{R^\star}(-\mathbf{z}_{1:t} \| - \mathbf{z}_{1:t-1})). \end{aligned} \tag{2.14}$$

Note that $R^\star(\mathbf{0}) = \max_{\mathbf{w}} \langle \mathbf{0}, \mathbf{w} \rangle - R(\mathbf{w}) = -\min_{\mathbf{w}} R(\mathbf{w}) = -R(\mathbf{w}_1)$. Combining all the above we conclude our proof. □

It is interesting to compare the above lemma to Lemma 2.3, which for linear functions yields the regret bound

$$\sum_{t=1}^{T} \langle \mathbf{w}_t - \mathbf{u}, \mathbf{z}_t \rangle \leq R(\mathbf{u}) - R(\mathbf{w}_1) + \sum_{t=1}^{T} \langle \mathbf{w}_t - \mathbf{w}_{t+1}, \mathbf{z}_t \rangle.$$

In both bounds we have a stability term but the way we measure it is different.

We can easily derive concrete bounds from Lemma 2.20 if R is strongly convex.

Theorem 2.21. Let R be a $(1/\eta)$-strongly convex with respect to a norm $\|\cdot\|$ and suppose the OMD algorithm is run with the link function $g = \nabla R^\star$. Then,

$$\sum_{t=1}^{T} \langle \mathbf{w}_t - \mathbf{u}, \mathbf{z}_t \rangle \leq R(\mathbf{u}) - R(\mathbf{w}_1) + \frac{\eta}{2} \sum_{t=1}^{T} \|\mathbf{z}_t\|_\star^2.$$

Proof. The proof follows directly from Lemma 2.20 and the strong/smooth duality given in Lemma 2.19. □

The reader can easily obtain the bounds we derived in Section 2.6 using the above theorem. Consider for example the case in which the regularization function is $R(\mathbf{w}) = \frac{1}{2\eta}\|\mathbf{w}\|_2^2$, which is $(1/\eta)$-strongly convex with respect to the Euclidean norm. Since the Euclidean norm is dual to itself, we obtain the bound

$$\sum_{t=1}^{T} \langle \mathbf{w}_t - \mathbf{u}, \mathbf{z}_t \rangle \leq \frac{1}{2\eta}\|\mathbf{u}\|_2^2 + \frac{\eta}{2} \sum_{t=1}^{T} \|\mathbf{z}_t\|_2^2. \tag{2.15}$$

Letting $L^2 = \frac{1}{T}\sum_t \|\mathbf{z}_t\|_2^2$ and setting $\eta = \frac{B}{L\sqrt{T}}$ then for all $\mathbf{u}$ with $\|\mathbf{u}\|_2 \leq B$ we obtain the bound

$$\sum_{t=1}^{T} \langle \mathbf{w}_t - \mathbf{u}, \mathbf{z}_t \rangle \leq BL\sqrt{T}.$$

Comparing this bound to the bound we derived in Corollary 2.7, we observe that we have obtained a factor of $\sqrt{2}$ improvement in the regret bound.

2.7.4 Other Proof Techniques

In the previous subsection we used Fenchel–Young inequality to derive bounds for OMD, which is equivalent to FoReL when the loss functions are linear. It is possible to extend this proof technique, based on Fenchel duality, and to derive a larger family of online convex optimization algorithms. The basic idea is to derive the Fenchel dual of the optimization problem $\min_{\mathbf{w}} R(\mathbf{w}) + \sum_t f_t(\mathbf{w})$ and to construct an online learning algorithm by incrementally solving the dual problem. We refer the reader to [39] for more details.

Another popular approach is to derive regret bounds by monitoring the Bregman divergence $D_R(\mathbf{w}_t \| \mathbf{u})$, where $\mathbf{u}$ is the competing vector. A detailed description of this approach can be found in [12, 34] and is therefore omitted from this survey. It is important to note that the analysis using the Bregman divergence potential requires that R will be a Legendre function (see a precise definition in [12, p. 294]). In particular, Legendre functions guarantee the property that ∇R and $\nabla R^\star$ are inverse mappings. We do not require that R is Legendre and in particular, the normalized entropy regularization function, $R(w) = \sum_i w[i] \log(w[i]) + I_S(\mathbf{w})$, where S is the probability simplex, is not a Legendre function. Therefore, when analyzing the normalized EG algorithm using Bregman divergences we need to use the unnormalized Entropy as a regularization function and to include an additional Bregman projection step on the simplex, which leads to the desired normalization. To the best of our knowledge, the two proof techniques lead to the same regret bounds.

2.8 Bounds with Local Norms

Consider running the normalized EG algorithm, namely, running FoReL on linear loss functions with the normalized entropy $R(\mathbf{w}) = \frac{1}{\eta}\sum_i w[i]\log(w[i]) + I_S(\mathbf{w})$, where $S = \{\mathbf{w} \geq 0 : \|\mathbf{w}\|_1 = 1\}$. Previously, we have derived the regret bound,

$$\sum_{t=1}^{T} \langle \mathbf{w}_t - \mathbf{u}, \mathbf{z}_t \rangle \leq \frac{\log(d)}{\eta} + \eta \sum_{t=1}^{T} \|\mathbf{z}_t\|_\infty^2.$$

We now derive a refined bound for the normalized EG algorithm, in which each term $\|\mathbf{z}_t\|_\infty^2$ is replaced by a term $\sum_i w_t[i] z_t[i]^2$. Since $\mathbf{w}_t$ is in the probability simplex, we clearly have that $\sum_i w_t[i] z_t[i]^2 \leq \|\mathbf{z}_t\|_\infty^2$.

In fact, we can rewrite $\sum_i w_t[i] z_t[i]^2$ as a local norm $\|\mathbf{z}_t\|_t^2$ where

$$\|\mathbf{z}\|_t = \sqrt{\sum_i w_t[i] z[i]^2}.$$

Note that this is indeed a valid norm. In the next sections we will show cases in which the refined local-norm bounds lead to much tighter regret bounds.

Theorem 2.22. Assume that the normalized EG algorithm is run on a sequence of linear loss functions such that for all t, i we have $\eta z_t[i] \geq -1$. Then,

$$\sum_{t=1}^{T} \langle \mathbf{w}_t - \mathbf{u}, \mathbf{z}_t \rangle \leq \frac{\log(d)}{\eta} + \eta \sum_{t=1}^{T} \sum_i w_t[i] z_t[i]^2.$$

Proof. Using Lemma 2.20, it suffices to show that

$$D_{R^\star}(-\mathbf{z}_{1:t} \| - \mathbf{z}_{1:t-1}) \leq \eta \sum_i w_t[i] z_t[i]^2,$$

where, based on Table 2.1, the conjugate function is

$$R^\star(\boldsymbol{\theta}) = \frac{1}{\eta} \log\left(\sum_i e^{\eta \theta[i]}\right).$$

Indeed,

$$\begin{aligned} D_{R^\star}(-\mathbf{z}_{1:t} \| - \mathbf{z}_{1:t-1}) &= R^\star(-\mathbf{z}_{1:t}) - R^\star(-\mathbf{z}_{1:t-1}) + \langle \mathbf{w}_t, \mathbf{z}_t \rangle && (2.16) \\ &= \frac{1}{\eta} \log\left(\frac{\sum_i e^{-\eta z_{1:t}[i]}}{\sum_i e^{-\eta z_{1:t-1}[i]}}\right) + \langle \mathbf{w}_t, \mathbf{z}_t \rangle && (2.17) \\ &= \frac{1}{\eta} \log\left(\sum_i w_t[i] e^{-\eta z_t[i]}\right) + \langle \mathbf{w}_t, \mathbf{z}_t \rangle. && (2.18) \end{aligned}$$

Using the inequality $e^{-a} \leq 1 - a + a^2$ which holds for all $a \geq -1$ (and hence holds by the assumptions of the theorem) we obtain

$$D_{R^\star}(-\mathbf{z}_{1:t} \| - \mathbf{z}_{1:t-1}) \leq \frac{1}{\eta} \log\left(\sum_i w_t[i] (1 - \eta z_t[i] + \eta^2 z_t[i]^2)\right) + \langle \mathbf{w}_t, \mathbf{z}_t \rangle.$$

Next, we use the fact that $\sum_i w_t[i] = 1$ and the inequality $\log(1-a) \le -a$, which holds for all $a \le 1$ we obtain

$$\begin{aligned} D_{R^\star}(-\mathbf{z}_{1:t}\| - \mathbf{z}_{1:t-1}) &\le \frac{1}{\eta}\sum_i w_t[i](-\eta z_t[i] + \eta^2 z_t[i]^2) + \langle \mathbf{w}_t, \mathbf{z}_t \rangle \\ &= \eta \sum_i w_t[i] z_t[i]^2. \end{aligned} \qquad \square$$

Next, we describe another variant of the EG algorithm in which we do not normalize the weights to the simplex on each round. We show a similar regret bounds for this variant as well.

```
          Unnormalized Exponentiated Gradient
                  (unnormalized-EG)

parameters: $\eta, \lambda > 0$
initialize: $\mathbf{w}_1 = (\lambda, \ldots, \lambda)$
update rule $\forall i,\ w_{t+1}[i] = w_t[i] e^{-\eta z_t[i]}$
```

The following theorem provides a regret bound with local-norms for the unnormalized EG algorithm.

Theorem 2.23. Assume that the unnormalized EG algorithm is run on a sequence of linear loss functions such that for all t, i we have $\eta z_t[i] \ge -1$. Then, for all $\mathbf{u} \ge \mathbf{0}$,

$$\sum_{t=1}^{T} \langle \mathbf{w}_t - \mathbf{u}, \mathbf{z}_t \rangle \le \frac{d\lambda + \sum_{i=1}^{d} u[i] \log(u[i]/(e\,\lambda))}{\eta} + \eta \sum_{t=1}^{T} \sum_{i=1}^{d} w_t[i] z_t[i]^2.$$

In particular, setting $\lambda = 1/d$ yields

$$\begin{aligned} \sum_{t=1}^{T} \langle \mathbf{w}_t - \mathbf{u}, \mathbf{z}_t \rangle \le{}& \frac{1 + (\log(d) - 1)\|\mathbf{u}\|_1 + \sum_{i=1}^{d} u[i] \log(u[i])}{\eta} \\ &+ \eta \sum_{t=1}^{T} \sum_{i=1}^{d} w_t[i] z_t[i]^2. \end{aligned}$$

Proof. To analyze the unnormalized-EG algorithm, we first note that it is equivalent to running FoReL on the sequence of linear loss functions with the following unnormalized entropy regularization function:

$$R(\mathbf{w}) = \frac{1}{\eta}\sum_i w[i](\log(w[i]) - 1 - \log(\lambda)).$$

Using Lemma 2.20, it suffices to show that

$$D_{R^\star}(-\mathbf{z}_{1:t} \| - \mathbf{z}_{1:t-1}) \le \eta \sum_i w_t[i] z_t[i]^2,$$

where, based on Table 2.1, the conjugate function is

$$R^\star(\boldsymbol{\theta}) = \frac{\lambda}{\eta}\sum_i e^{\eta\theta[i]}.$$

We have:

$$\begin{aligned} D_{R^\star}(-\mathbf{z}_{1:t} \| - \mathbf{z}_{1:t-1}) &= R^\star(-\mathbf{z}_{1:t}) - R^\star(-\mathbf{z}_{1:t-1}) + \langle \mathbf{w}_t, \mathbf{z}_t \rangle \\ &= \frac{\lambda}{\eta}\sum_i e^{-\eta z_{1:t-1}[i]}(e^{-\eta z_t[i]} - 1) + \langle \mathbf{w}_t, \mathbf{z}_t \rangle \\ &= \frac{1}{\eta}\sum_i w_t[i](e^{-\eta z_t[i]} - 1) + \langle \mathbf{w}_t, \mathbf{z}_t \rangle \\ &\le \eta \sum_i w_t[i] z_t[i]^2, \end{aligned}$$

where in the last inequality we used the inequality $e^{-a} \le 1 - a + a^2$ which holds for all $a \ge -1$ (and hence holds by the assumptions of the lemma). □

2.9 Bibliographic Remarks

The term "online convex programming" was introduced by Zinkevich [46] but this setting was introduced some years earlier by Gordon [20]. This model is also closely related to the model of relative loss bounds presented by Kivinen and Warmuth [26, 27, 28]. Our presentation of relative mistake bounds follows the works of Littlestone [31], and Kivinen and Warmuth [28].

Zinkevich presented and analyzed the online gradient descent (with projections or with lazy projections). The name “Follow the Leader” is due to [25]. Analysis of “Follow the Regularized Leader” was given in [40, 39] using Fenchel duality as part of the analysis of a larger algorithmic framework. The more direct analysis we give here is adapted from [34, 22]. The duality based analysis we present is due to [39, 24]. Similar ideas in a more limited context have been introduced in [21].

The EG update is due to [26]. See also [6, 28]. The p-norm algorithm was developed by Gentile and Littlestone [19]. Local norms were introduced in [2]. See also [34].

3

Online Classification

In this section we return to the problem of online classification, in which the target domain is $\mathcal{Y} = \{0,1\}$. We already mentioned this setting in Section 1.2, where we introduced the `Consistent` and `Halving` algorithms. These algorithms rely on two simplifying assumptions: realizability of the problem and finiteness of $\mathcal{H}$. While trying to relax the realizability assumption, we presented Cover's impossibility result, which shows that even if $|\mathcal{H}| = 2$, no algorithm can have low regret in the unrealizable case. We sidestepped the impossibility result by allowing the learner to randomize his predictions (i.e., the predictions domain is allowed to be $D = [0,1]$). As discussed in Section 2.1.1, this can be thought of as a convexification of the problem. Based on this convexification, we start the section by deriving the `Weighted Majority` algorithm for prediction with expert advice. We will see that this is a specific instance of the `normalized EG` algorithm discussed in the previous section. The analysis of Weighted Majority implies online learnability of finite hypothesis classes.

Next, in Section 3.2, we study the fundamental question of online learnability, namely, what is an optimal algorithm for a given hypothesis class. We shall characterize learnability for the general case, namely, without assuming neither realizability nor finiteness of $\mathcal{H}$.

Finally, in Section 3.3, we study the important and practically relevant hypothesis class of halfspaces with margin and derive the classic Perceptron and Winnow algorithms from the online convex optimization framework.

3.1 Finite Hypothesis Class and Experts Advice

Consider the problem of classification, where on round t the learner receives $\mathbf{x}_t \in \mathcal{X}$, predicts $p_t \in [0,1]$, receives $y_t \in \{0,1\}$, and pays $|p_t - y_t|$. The goal is to have low regret with respect to a finite hypothesis class $\mathcal{H} = \{h_1,\ldots,h_d\}$. As we hinted in Section 2.1.1, this problem can be reduced to the problem of prediction with expert advice as follows. Let $S = \{\mathbf{w} \in \mathbb{R}_+^d : \|\mathbf{w}\|_1 = 1\}$ be the probability simplex. The learner will maintain a weight vector $\mathbf{w}_t \in S$ and will predict the label 1 with probability $p_t = \sum_{i=1}^d w_t[i] h_i(\mathbf{x}_t)$. The loss he will pay can therefore be rewritten as

$$|p_t - y_t| = \left| \sum_{i=1}^d w_t[i] h_i(\mathbf{x}_t) - y_t \right| = \sum_{i=1}^d w_t[i] |h_i(\mathbf{x}_t) - y_t|,$$

where the last equality follows because both y_t and $h_i(\mathbf{x}_t)$ are in $\{0,1\}$. Letting $\mathbf{z}_t = (|h_1(\mathbf{x}_t) - y_t|,\ldots,|h_d(\mathbf{x}_t) - y_t|)$ we obtain that the loss is $\langle \mathbf{w}_t, \mathbf{z}_t \rangle$. This is a special case of the prediction with expert advice problem, where $\mathbf{z}_t \in [0,1]^d$ is the costs vector of the different experts. Applying the normalized-EG algorithm we obtain an algorithm which is often called Weighted Majority.

```
                    Weighted Majority

parameter: $\eta \in (0,1)$
initialize: $\mathbf{w}_1 = (1/d,\ldots,1/d)$
for $t = 1,2,\ldots$
  choose $i \sim \mathbf{w}_t$ and predict according to the advice of the $i$'th
  expert receive costs of all experts $\mathbf{z}_t \in [0,1]^d$
  update rule $\forall i,\ w_{t+1}[i] = \frac{w_t[i] e^{-\eta z_t[i]}}{\sum_j w_t[j] e^{-\eta z_t[j]}}$
```

The analysis of `Weighted Majority` follows directly from the analysis of the `normalized-EG` algorithm given in the previous section and is given in the following theorem.

Theorem 3.1. The Weighted Majority algorithm enjoys the bounds

1. $$\sum_{t=1}^{T} \langle \mathbf{w}_t, \mathbf{z}_t \rangle \leq \min_{i \in [d]} \sum_{t=1}^{T} z_t[i] + \frac{\log(d)}{\eta} + \eta T$$
2. $$\sum_{t=1}^{T} \langle \mathbf{w}_t, \mathbf{z}_t \rangle \leq \frac{1}{1-\eta} \left(\min_{i \in [d]} \sum_{t=1}^{T} z_t[i] + \frac{\log(d)}{\eta} \right)$$

In particular, setting $\eta = \sqrt{\log(d)/T}$ in the first bound we obtain

$$\sum_{t=1}^{T} \langle \mathbf{w}_t, \mathbf{z}_t \rangle \leq \min_{i \in [d]} \sum_{t=1}^{T} z_t[i] + 2\sqrt{\log(d)T},$$

and setting $\eta = 1/2$ in the second bound we obtain

$$\sum_{t=1}^{T} \langle \mathbf{w}_t, \mathbf{z}_t \rangle \leq 2 \min_{i \in [d]} \sum_{t=1}^{T} z_t[i] + 4 \log(d).$$

Proof. We rely on the analysis of the normalized-EG algorithm given in Theorem 2.22. Since $\mathbf{z}_t \in [0,1]^d$ and $\eta > 0$ the conditions of the theorem holds and we obtain that

$$\sum_{t=1}^{T} \langle \mathbf{w}_t - \mathbf{u}, \mathbf{z}_t \rangle \leq \frac{\log(d)}{\eta} + \eta \sum_{t=1}^{T} \sum_{i} w_t[i] z_t[i]^2.$$

Since $\mathbf{w}_t \in S$ and $\mathbf{z}_t \in [0,1]^d$ the right-most term is at most ηT. Furthermore, the bound holds for all $\mathbf{u}$ in the probability simplex and in particular for all singletons, i.e., vectors of the form $\mathbf{u} = (0, \ldots, 0, 1, 0, \ldots, 0)$. Rearranging, we obtain,

$$\sum_{t=1}^{T} \langle \mathbf{w}_t, \mathbf{z}_t \rangle \leq \min_{i \in [d]} \sum_{t=1}^{T} z_t[i] + \frac{\log(d)}{\eta} + \eta T.$$

To derive the second bound, observe that $w_t[i], z_t[i] \in [0,1]$ and therefore $w_t[i] z_t[i]^2 \leq w_t[i] z_t[i]$, which yields

$$\sum_{t=1}^{T} \langle \mathbf{w}_t - \mathbf{u}, \mathbf{z}_t \rangle \leq \frac{\log(d)}{\eta} + \eta \sum_{t=1}^{T} \langle \mathbf{w}_t, \mathbf{z}_t \rangle .$$

Rearranging, we obtain that

$$\sum_{t=1}^{T} \langle \mathbf{w}_t, \mathbf{z}_t \rangle \leq \frac{1}{1-\eta} \left(\min_{i \in [d]} \sum_{t=1}^{T} z_t[i] + \frac{\log(d)}{\eta} \right). \qquad \square$$

In particular, in the realizable case when $\min_{i \in [d]} \sum_{t=1}^{T} z_t[i] = 0$ we obtain that `Weighted Majority` with $\eta = 1/2$ enjoys the bound

$$\sum_{t=1}^{T} \langle \mathbf{w}_t, \mathbf{z}_t \rangle \leq 4 \log(d).$$

This is similar to the bound we derived for `Halving` in Section 1.2.

3.2 Learnability and the Standard Optimal Algorithm

So far, we focused on finite hypothesis classes. In this section we take a more general approach, and aim at characterizing online learnability. In particular, we target the following question: what is the optimal online learning algorithm for a given class $\mathcal{H}$?

We start with the realizable case, where we assume that all target labels are generated by some $h^\star \in \mathcal{H}$, namely, $y_t = h^\star(x_t)$ for all t. Later, in Section 3.2.1 we generalize the results to the unrealizable case. In the realizable case, we study the best achievable mistake bound, formally defined below.

Definition 3.1 (Mistake bounds, online learnability). Let $\mathcal{H}$ be a hypothesis class and let A be an online learning algorithm. Given any sequence $S = (x_1, h^\star(y_1)), \ldots, (x_T, h^\star(y_T))$, where T is any integer and $h^\star \in \mathcal{H}$, let $M_A(S)$ be the number of mistakes A makes on the sequence S. We denote by $M_A(\mathcal{H})$ the supremum of $M_A(S)$ over all sequences of the above form. A bound of the form $M_A(\mathcal{H}) \leq B < \infty$ is called a *mistake bound.* We say that a hypothesis class $\mathcal{H}$ is online learnable if there exists an algorithm A for which $M_A(\mathcal{H}) \leq B < \infty$.

We present a dimension of hypothesis classes that characterizes the best possible achievable mistake bound. This measure was proposed by Nick Littlestone and we therefore refer to it as $\mathrm{Ldim}(\mathcal{H})$.

To motivate the definition of Ldim it is convenient to view the online learning process as a game between two players: the learner vs. the environment. On round t of the game, the environment picks an instance $\mathbf{x}_t$, the learner predicts a label $p_t \in \{0,1\}$, and finally the environment outputs the true label, $y_t \in \{0,1\}$. Suppose that the environment wants to make the learner err on the first T rounds of the game. Then, it must output $y_t = 1 - p_t$, and the only question is how to choose the instances $\mathbf{x}_t$ in such a way that ensures that for some $h^\star \in \mathcal{H}$ we have $y_t = h^\star(\mathbf{x}_t)$ for all $t \in [T] = \{1,\dots,T\}$.

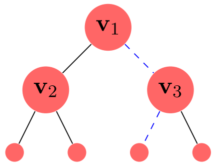
It makes sense to assume that the environment should pick $\mathbf{x}_t$ based on the previous predictions of the learner, $p_1,\dots,p_{t-1}$. Since in our case we have $y_t = 1 - p_t$ we can also say that $\mathbf{x}_t$ is a function of $y_1,\dots,y_{t-1}$. We can represent this dependence using a complete binary tree of depth T (we define the depth of the tree as the number of edges in a path from the root to a leaf). We have $2^T - 1$ nodes in such a tree, and we attach an instance to each node. Let $\mathbf{v}_1,\dots,\mathbf{v}_{2^T-1}$ be these instances. We start from the root of the tree, and set $\mathbf{x}_1 = \mathbf{v}_1$. At round t, we set $\mathbf{x}_t = \mathbf{v}_{i_t}$, where i_t is the current node. At the end of round t, we go to the left child of i_t if $y_t = 0$ or to the right child if $y_t = 1$. That is, $i_{t+1} = 2i_t + y_t$. Unraveling the recursion we obtain $i_t = 2^{t-1} + \sum_{j=1}^{t-1} y_j\, 2^{t-1-j}$.

The above strategy for the environment succeeds only if for any $(y_1,\dots,y_T)$ there exists $h \in \mathcal{H}$ such that $y_t = h(\mathbf{x}_t)$ for all $t \in [T]$. This leads to the following definition.

Definition 3.2 ($\mathcal{H}$ Shattered tree). A shattered tree of depth d is a sequence of instances $\mathbf{v}_1,\dots,\mathbf{v}_{2^d-1}$ in $\mathcal{X}$ such that for all labeling $(y_1,\dots,y_d) \in \{0,1\}^d$ there exists $h \in \mathcal{H}$ such that for all $t \in [d]$ we have $h(\mathbf{v}_{i_t}) = y_t$, where $i_t = 2^{t-1} + \sum_{j=1}^{t-1} y_j\, 2^{t-1-j}$.

An illustration of a shattered tree of depth 2 is given in Figure 3.1.

Definition 3.3 (Littlestone's dimension (Ldim)). $\mathrm{Ldim}(\mathcal{H})$ is the maximal integer T such that there exist a shattered tree of depth T.



	h_1	h_2	h_3	h_4
$\mathbf{v}_1$	0	0	1	1
$\mathbf{v}_2$	0	1	$*$	$*$
$\mathbf{v}_3$	$*$	$*$	0	1

Fig. 3.1 An illustration of a shattered tree of depth 2. The *dashed blue* path corresponds to the sequence of examples $((\mathbf{v}_1,1),(\mathbf{v}_3,0))$. The tree is shattered by $\mathcal{H} = \{h_1,h_2,h_3,h_4\}$, where the predictions of each hypothesis in $\mathcal{H}$ on the instances $\mathbf{v}_1,\mathbf{v}_2,\mathbf{v}_3$ is given in the table (the "$*$" mark means that $h_j(\mathbf{v}_i)$ can be either 1 or 0).

The definition of Ldim and the discussion above immediately imply the following:

Lemma 3.2. No algorithm can have a mistake bound strictly smaller than Ldim($\mathcal{H}$), namely, $\forall A,\ M_A(\mathcal{H}) \geq \text{Ldim}(\mathcal{H})$.

Proof. Let $T = \text{Ldim}(\mathcal{H})$and let $\mathbf{v}_1,\ldots,\mathbf{v}_{2^T-1}$ be a sequence that satisfies the requirements in the definition of Ldim. If the environment sets $\mathbf{x}_t = \mathbf{v}_{i_t}$ and $y_t = 1 - p_t$ for all $t \in [T]$, then the learner makes T mistakes while the definition of Ldim implies that there exists a hypothesis $h \in \mathcal{H}$ such that $y_t = h(\mathbf{x}_t)$ for all t. □

Let us now give several examples.

Example 3.1. Let $\mathcal{H}$ be a finite hypothesis class. Clearly, any tree that is shattered by $\mathcal{H}$ has depth of at most $\log_2(|\mathcal{H}|)$. Therefore, $\text{Ldim}(\mathcal{H}) \leq \log_2(|\mathcal{H}|)$. Another way to conclude this inequality is by combining Lemma 3.2 with Theorem 1.2.

Example 3.2. Let $\mathcal{X} = \{1,\ldots,d\}$ and $\mathcal{H} = \{h_1,\ldots,h_d\}$, where $h_d(x) = 1$ iff $x = d$. Then, it is easy to show that $\text{Ldim}(\mathcal{H}) = 1$ while $|\mathcal{H}| = d$ can be arbitrarily large.

Example 3.3. Let $\mathcal{X} = [0,1]$ and $\mathcal{H} = \{x \mapsto \mathbf{1}_{[x>a]} : a \in [0,1]\}$, namely, $\mathcal{H}$ is the class of thresholds on the segment $[0,1]$. Then, $\text{Ldim}(\mathcal{H}) = \infty$.

To see this, consider the tree for which $v_1 = \frac{1}{2}, v_2 = \frac{1}{4}, v_3 = \frac{3}{4}, \ldots$. Due to the density of the reals, this tree is shattered by $\mathcal{H}$.

Example 3.4. Let $X = \{\mathbf{x} \in \{0,1\}^* : \|\mathbf{x}\|_0 \leq r\}$ and $\mathcal{H} = \{\mathbf{x} \mapsto \mathbf{1}_{[\langle \mathbf{w},\mathbf{x}\rangle > 0.5]} : \|\mathbf{w}\|_0 \leq k\}$. The size of $\mathcal{H}$ is infinite. Nevertheless, $\mathrm{Ldim}(\mathcal{H}) \leq r\,k$. The proof uses the fact that $M_{\texttt{Perceptron}}(\mathcal{H}) \leq r\,k$, where the Perceptron algorithm will be described in the next section.

Lemma 3.2 states that $\mathrm{Ldim}(\mathcal{H})$ lower bounds the mistake bound of any algorithm. Interestingly, there is a standard algorithm whose mistake bound matches this lower bound. The algorithm is similar to the `Halving` algorithm. Recall that the prediction of `Halving` is according to a majority vote of the hypotheses which are consistent with previous examples. We denoted this set by V_t. Put another way, `Halving` partition V_t into two sets: $V_t^+ = \{h \in V_t : h(\mathbf{x}_t) = 1\}$ and $V_t^- = \{h \in V_t : h(\mathbf{x}_t) = 0\}$. It then predicts according to the larger of the two groups. The rationale behind this prediction is that whenever `Halving` makes a mistake it ends up with $|V_{t+1}| \leq 0.5\,|V_t|$.

The optimal algorithm we present below uses the same idea, but instead of predicting according to the larger class, it predicts according to the class with larger Ldim.

```
Standard Optimal Algorithm (SOA)

input: A hypothesis class $\mathcal{H}$
initialize: $V_1 = \mathcal{H}$
for $t = 1,2,\ldots$
  receive $\mathbf{x}_t$
  for $r \in \{0,1\}$ let $V_t^{(r)} = \{h \in V_t : h(\mathbf{x}_t) = r\}$
  predict $p_t = \mathrm{argmax}_{r \in \{0,1\}} \mathrm{Ldim}(V_t^{(r)})$
    (in case of a tie predict $p_t = 1$)
  receive true answer $y_t$
  update $V_{t+1} = \{h \in V_t : h(\mathbf{x}_t) = y_t\}$
```

The following lemma formally establishes the optimality of the above algorithm.

Lemma 3.3. SOA enjoys the mistake bound $M_{\texttt{SOA}}(\mathcal{H}) \le \mathrm{Ldim}(\mathcal{H})$.

Proof. It suffices to prove that whenever the algorithm makes a prediction mistake we have $\mathrm{Ldim}(V_{t+1}) \le \mathrm{Ldim}(V_t) - 1$. We prove this claim by assuming the contrary, that is, $\mathrm{Ldim}(V_{t+1}) = \mathrm{Ldim}(V_t)$. If this holds true, then the definition of p_t implies that $\mathrm{Ldim}(V_t^{(r)}) = \mathrm{Ldim}(V_t)$ for both $r = 1$ and $r = 0$. But, then we can construct a shattered tree of depth $\mathrm{Ldim}(V_t) + 1$ for the class V_t, which leads to the desired contradiction. □

Combining Lemma 3.3 and Lemma 3.2 we obtain:

Corollary 3.4. Let $\mathcal{H}$ be any hypothesis class. Then, the standard optimal algorithm enjoys the mistake bound $M_{\texttt{SOA}}(\mathcal{H}) = \mathrm{Ldim}(\mathcal{H})$ and no other algorithm can have $M_A(\mathcal{H}) < \mathrm{Ldim}(\mathcal{H})$.

Comparison to VC dimension In the PAC learning model of Valiant, learnability is characterized by the Vapnik–Chervonenkis (VC) dimension of the class $\mathcal{H}$. To remind the reader, the VC dimension of a class $\mathcal{H}$, denoted $\mathrm{VCdim}(\mathcal{H})$, is the maximal number d such that there are instances $\mathbf{x}_1,\ldots,\mathbf{x}_d$ that are shattered by $\mathcal{H}$. That is, for any sequence of labels $(y_1,\ldots,y_d) \in \{0,1\}^d$ there exists a hypothesis $h \in \mathcal{H}$ that gives exactly this sequence of labels. The following theorem relates the VC dimension to the Littlestone dimension.

Theorem 3.5. For any class $\mathcal{H}$, $\mathrm{VCdim}(\mathcal{H}) \le \mathrm{Ldim}(\mathcal{H})$, and there are classes for which strict inequality holds. Furthermore, the gap can be arbitrarily large.

Proof. We first prove that $\mathrm{VCdim}(\mathcal{H}) \le \mathrm{Ldim}(\mathcal{H})$. Suppose $\mathrm{VCdim}(\mathcal{H}) = d$ and let $\mathbf{x}_1,\ldots,\mathbf{x}_d$ be a shattered set. We now construct a complete binary tree of instances $\mathbf{v}_1,\ldots,\mathbf{v}_{2^d-1}$, where all nodes at depth i are set to be $\mathbf{x}_i$ (see the illustration in Figure 3.2). Now,

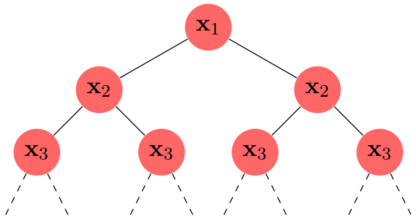


Fig. 3.2 How to construct a shattered tree from a shattered sequence $\mathbf{x}_1, \ldots, \mathbf{x}_d$.

the definition of shattered sample clearly implies that we got a valid shattered tree of depth d, and we conclude that $\text{VCdim}(\mathcal{H}) \leq \text{Ldim}(\mathcal{H})$. To show that the gap can be arbitrarily large simply note that the class given in Example 3.3 has VC dimension of 1 whereas its Littlestone dimension is infinite. □

3.2.1 The Unrealizable Case

In the previous section we have shown that Littlestone's dimension exactly characterizes the achievable mistake bounds in the realizable case. We now show that the same dimension characterizes online learnability in the unrealizable case as well. Specifically, we will prove the following.

Theorem 3.6. For any hypothesis class $\mathcal{H}$, there exists an online learning algorithm such that for any $h \in \mathcal{H}$ and any sequence of T examples we have

$$\sum_{t=1}^{T} |p_t - y_t| - \sum_{t=1}^{T} |h(\mathbf{x}_t) - y_t| \leq O(\sqrt{\text{Ldim}(\mathcal{H}) \ln(T) T}),$$

where p_t is the learner's prediction on round t. Furthermore, no algorithm can achieve an expected regret bound smaller than $\Omega(\sqrt{\text{Ldim}(\mathcal{H}) T})$.

Recall that in the unrealizable case, to sidestep Cover's impossibility result, the learner is allowed to make randomized predictions and we analyze his expected regret. We will construct a generic online algorithm that has the expected regret bound $\sqrt{\text{Ldim}(\mathcal{H}) \log(T) T}$.

Finally, we provide an almost matching lower bound on the achievable regret.

Our starting point is the `Weighted Majority` algorithm, whose regret depends on $\log(d)$. In the case of a finite hypothesis class, we let each hypothesis be an expert. In the unrealizable case, the main idea is to construct a set of experts in a more sophisticated way. The challenge is how to define a set of experts that on one hand is not excessively large while on the other hand contains experts that give accurate predictions.

We construct the set of experts so that for each hypothesis $h \in \mathcal{H}$ and every sequence of instances, $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T$, there exists at least one expert in the set which behaves exactly as h on these instances. For each $L \leq \mathrm{Ldim}(\mathcal{H})$ and each sequence $1 \leq i_1 < i_2 < \cdots < i_L \leq T$ we define an expert. The expert simulates the game between SOA (presented in the previous section) and the environment on the sequence of instances $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T$ assuming that SOA makes a mistake precisely in rounds $i_1, i_2, \ldots, i_L$. The expert is defined by the following algorithm.

```
                    Expert($i_1, i_2, \ldots, i_L$)

input A hypothesis class $\mathcal{H}$ ; Indices $i_1 < i_2 < \cdots < i_L$
initialize: $V_1 = \mathcal{H}$
for $t = 1, 2, \ldots, T$
  receive $\mathbf{x}_t$
  for $r \in \{0, 1\}$ let $V_t^{(r)} = \{h \in V_t : h(\mathbf{x}_t) = r\}$
  define $\tilde{y}_t = \mathrm{argmax}_r \, \mathrm{Ldim}\left(V_t^{(r)}\right)$
    (in case of a tie set $\tilde{y}_t = 0$)
  if $t \in \{i_1, i_2, \ldots, i_L\}$
        predict $\hat{y}_t = \neg \tilde{y}_t$
  else
        predict $\hat{y}_t = \tilde{y}_t$
  update $V_{t+1} = V_t^{(\hat{y}_t)}$
```

The following key lemma shows that, on any sequence of instances, for each hypothesis $h \in \mathcal{H}$ there exists an expert with the same behavior.

Lemma 3.7. Let $\mathcal{H}$ be any hypothesis class with $\mathrm{Ldim}(\mathcal{H}) < \infty$. Let $\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_T$ be any sequence of instances. For any $h \in \mathcal{H}$, there exists $L \le \mathrm{Ldim}(\mathcal{H})$ and indices $1 \le i_1 < i_2 < \cdots < i_L \le T$ such that when running $\mathrm{Expert}(i_1,i_2,\ldots,i_L)$ on the sequence $\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_T$, the expert predicts $h(\mathbf{x}_t)$ on each online round $t = 1,2,\ldots,T$.

Proof. Fix $h \in \mathcal{H}$ and the sequence $\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_T$. We must construct L and the indices $i_1,i_2,\ldots,i_L$. Consider running SOA on the input $(\mathbf{x}_1,h(\mathbf{x}_1))$, $(\mathbf{x}_2,h(\mathbf{x}_2)),\ldots,(\mathbf{x}_T,h(\mathbf{x}_T))$. SOA makes at most $\mathrm{Ldim}(\mathcal{H})$ mistakes on such input. We define L to be the number of mistakes made by SOA and we define $\{i_1,i_2,\ldots,i_L\}$ to be the set of rounds in which SOA made the mistakes.

Now, consider the $\mathrm{Expert}(i_1,i_2,\ldots,i_L)$ running on the sequence $\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_T$. By construction, the set V_t maintained by $\mathrm{Expert}(i_1,i_2,\ldots,i_L)$ equals to the set V_t maintained by SOA when running on the sequence $(\mathbf{x}_1,h(\mathbf{x}_1)),\ldots,(\mathbf{x}_T,h(\mathbf{x}_T))$. Since the predictions of SOA differ from the predictions of h if and only if the round is in $\{i_1,i_2,\ldots,i_L\}$, we conclude that the predictions of $\mathrm{Expert}(i_1,i_2,\ldots,i_L)$ are always the same as the predictions of h. □

The above lemma holds in particular for the hypothesis in $\mathcal{H}$ that makes the least number of mistakes on the sequence of examples, and we therefore obtain the following:

Corollary 3.8. Let $(\mathbf{x}_1,y_1),(\mathbf{x}_2,y_2),\ldots,(\mathbf{x}_T,y_T)$ be a sequence of examples and let $\mathcal{H}$ be a hypothesis class with $\mathrm{Ldim}(\mathcal{H}) < \infty$. There exists $L \le \mathrm{Ldim}(\mathcal{H})$ and indices $1 \le i_1 < i_2 < \cdots < i_L \le T$, such that $\mathrm{Expert}(i_1,i_2,\ldots,i_L)$ makes at most as many mistakes as the best $h \in \mathcal{H}$ does. Namely,

$$\min_{h\in\mathcal{H}} \sum_{t=1}^{T} |h(\mathbf{x}_t) - y_t|$$

mistakes on the sequence of examples.

Our generic online learning algorithm is now an application of the `Weighted Majority` algorithm with the constructed experts. To analyze the algorithm we combine Corollary 3.8 with the upper bound on the number of experts,

$$d = \sum_{L=0}^{\mathrm{Ldim}(\mathcal{H})} \binom{T}{L} \leq \left(eT/\mathrm{Ldim}(\mathcal{H})\right)^{\mathrm{Ldim}(\mathcal{H})}, \tag{3.1}$$

and with Theorem 3.1. This proves the upper bound part of Theorem 3.6. The proof of the lower bound part can be found in [7].

3.3 Perceptron and Winnow

In this section we describe two classic online learning algorithms for binary classification with the hypothesis class of halfspaces. Throughout this section, it is more convenient to let the labels set be $\mathcal{Y} = \{-1,1\}$ instead of $\mathcal{Y} = \{0,1\}$. Each halfspace 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 prediction is according to whether the point falls on one side of the line or on the other side. See Figure 3.3 for an illustration.

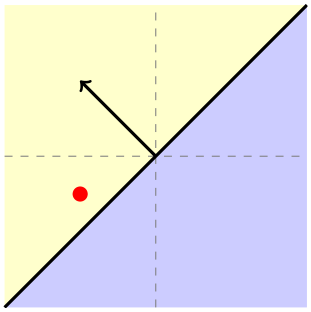


Fig. 3.3 An illustration of linear separators in the plane ($\mathbb{R}^2$). The *solid black line* separates the plane into two regions. The *circled point* represents an input vector which is labeled 1 by the linear separator. The arrow designates a weight vector that represents the hypothesis.

On round t, the learner receives a vector $\mathbf{x}_t \in \mathbb{R}^d$. The learner maintains a weight vector $\mathbf{w}_t \in \mathbb{R}^d$ and predicts $p_t = \text{sign}(\langle \mathbf{w}_t, \mathbf{x}_t \rangle)$. Then, it receives $y_t \in \mathcal{Y}$ and pays 1 if $p_t \neq y_t$ and 0 otherwise.

The goal of the learner is to make as few prediction mistakes as possible. In the previous section we characterized the optimal algorithm and showed that the best achievable regret bound depends on the Littlestone dimension of the class. In our case, the class of halfspaces is the class $\mathcal{H} = \{\mathbf{x} \mapsto \text{sign}(\langle \mathbf{w}, \mathbf{x} \rangle) : \mathbf{w} \in \mathbb{R}^d\}$. We show below that if $d \geq 2$ then $\text{Ldim}(\mathcal{H}) = \infty$, which implies that we have no hope to make few prediction mistakes. Indeed, consider the tree for which $v_1 = (\frac{1}{2}, 1, 0, \ldots, 0)$, $v_2 = (\frac{1}{4}, 1, 0, \ldots, 0)$, $v_3 = (\frac{3}{4}, 1, 0, \ldots, 0)$, etc. Due to the density of the reals, this tree is shattered by the subset of $\mathcal{H}$ which contains all hypotheses that are parametrized by $\mathbf{w}$ of the form $\mathbf{w} = (-1, a, 0, \ldots, 0)$, for $a \in [0,1]$. We conclude that indeed $\text{Ldim}(\mathcal{H}) = \infty$.

To sidestep this impossibility result, the Perceptron and Winnow algorithms rely on the technique of *surrogate convex losses* we discussed in Section 2.1.2. We now derive these algorithms.

3.3.1 Perceptron

A weight vector $\mathbf{w}$ makes a mistake on an example $(\mathbf{x}, y)$ whenever the sign of $\langle \mathbf{w}, \mathbf{x} \rangle$ does not equal to y. Therefore, we can write the 0–1 loss function as follows:

$$\ell(\mathbf{w}, (\mathbf{x}, y)) = \mathbf{1}_{[y\langle \mathbf{w}, \mathbf{x} \rangle \leq 0]}.$$

On rounds on which the algorithm makes a prediction mistake, we shall define the following surrogate convex loss function

$$f_t(\mathbf{w}) = [1 - y_t \langle \mathbf{w}, \mathbf{x}_t \rangle]_+ ,$$

where $[a]_+ = \max\{a, 0\}$ is the hinge function. This loss function is often called the *hinge-loss*. It satisfies the two conditions:

- f_t is a convex function
- For all $\mathbf{w}$, $f_t(\mathbf{w}) \geq \ell(\mathbf{w}, (\mathbf{x}_t, y_t))$. In particular, this holds for $\mathbf{w}_t$.

On rounds on which the algorithm is correct, we shall define $f_t(\mathbf{w}) = 0$. Clearly, f_t is convex in this case as well. Furthermore, $f_t(\mathbf{w}_t) = \ell(\mathbf{w}_t, (\mathbf{x}_t, y_t)) = 0$.

Let us now run the `Online Gradient Descent` (OGD) algorithm on the sequence of functions. Recall that OGD initializes $\mathbf{w}_1 = \mathbf{0}$ and its update rule

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \mathbf{z}_t$$

for some $\mathbf{z}_t \in \partial f_t(\mathbf{w}_t)$. In our case, if $y_t\langle \mathbf{w}_t, \mathbf{x}_t\rangle > 0$ then f_t is the zero function and we can take $\mathbf{z}_t = \mathbf{0}$. Otherwise, it is easy to verify that $\mathbf{z}_t = -y_t\mathbf{x}_t$ is in $\partial f_t(\mathbf{w}_t)$. We therefore obtain the update rule

$$\mathbf{w}_{t+1} = \begin{cases} \mathbf{w}_t & \text{if } y_t\langle \mathbf{w}_t, \mathbf{x}_t\rangle > 0 \\ \mathbf{w}_t + \eta y_t \mathbf{x}_t & \text{otherwise} \end{cases}$$

Denote by $\mathcal{M}$ the set of rounds in which $\text{sign}(\langle \mathbf{w}_t, \mathbf{x}_t\rangle) \neq y_t$. Note that on round t, the prediction of the Perceptron can be rewritten as,

$$p_t = \text{sign}(\langle \mathbf{w}_t, \mathbf{x}_t\rangle) = \text{sign}\left(\sum_{i \in \mathcal{M}: i < t} y_i \langle \mathbf{x}_i, \mathbf{x}_t\rangle\right).$$

The above form implies that the predictions of the Perceptron algorithm and the set $\mathcal{M}$ do not depend on the actual value of η as long as $\eta > 0$. We have therefore obtained the well known Perceptron algorithm.

```
Perceptron

initialize: $\mathbf{w}_1 = \mathbf{0}$
for $t = 1, 2, \ldots, T$
  receive $\mathbf{x}_t$
  predict $p_t = \text{sign}(\langle \mathbf{w}_t, \mathbf{x}_t\rangle)$
  if $y_t\langle \mathbf{w}_t, \mathbf{x}_t\rangle \leq 0$
    $\mathbf{w}_{t+1} = \mathbf{w}_t + y_t\mathbf{x}_t$
  else
    $\mathbf{w}_{t+1} = \mathbf{w}_t$
```

To analyze the Perceptron, we rely on the analysis of OGD given in the previous section. In particular, we rely on Corollary 2.7 and its improved version is given in Equation (2.15). In our case, the sub-gradients of f_t we use in the Perceptron are $\mathbf{z}_t = -\mathbf{1}_{[y_t\langle\mathbf{w}_t,\mathbf{x}_t\rangle\le 0]}\, y_t\,\mathbf{x}_t$. Indeed, the Perceptron's update is $\mathbf{w}_{t+1} = \mathbf{w}_t - \mathbf{z}_t$, and as discussed before this is equivalent to $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta\mathbf{z}_t$ for any $\eta > 0$. Therefore, the bound given in Equation (2.15) tells us that

$$\sum_{t=1}^{T} f_t(\mathbf{w}_t) - \sum_{t=1}^{T} f_t(\mathbf{u}) \le \frac{1}{2\eta}\|\mathbf{u}\|_2^2 + \frac{\eta}{2}\sum_{t=1}^{T}\|\mathbf{z}_t\|_2^2.$$

Since $f_t(\mathbf{w}_t)$ is a surrogate for the 0–1 loss we know that $\sum_{t=1}^{T} f_t(\mathbf{w}_t) \ge |\mathcal{M}|$. Denote $R = \max_t \|\mathbf{x}_t\|$, then we obtain

$$|\mathcal{M}| - \sum_{t=1}^{T} f_t(\mathbf{u}) \le \frac{1}{2\eta}\|\mathbf{u}\|_2^2 + \frac{\eta}{2}|\mathcal{M}|R^2$$

Setting $\eta = \frac{\|\mathbf{u}\|}{R\sqrt{|\mathcal{M}|}}$ and rearranging, we obtain

$$|\mathcal{M}| - R\|\mathbf{u}\|\sqrt{|\mathcal{M}|} - \sum_{t=1}^{T} f_t(\mathbf{u}) \le 0. \tag{3.2}$$

This inequality implies:

Theorem 3.9. Suppose that the Perceptron algorithm runs on a sequence $(\mathbf{x}_1,y_1),\ldots,(\mathbf{x}_T,y_T)$ and let $R = \max_t \|\mathbf{x}_t\|$. Let $\mathcal{M}$ be the rounds on which the Perceptron errs and let $f_t(\mathbf{w}) = \mathbf{1}_{[t\in\mathcal{M}]}\,[1 - y_t\langle\mathbf{w},\mathbf{x}_t\rangle]_+$. Then, for any $\mathbf{u}$

$$|\mathcal{M}| \;\le\; \sum_t f_t(\mathbf{u}) + R\|\mathbf{u}\|\sqrt{\sum_t f_t(\mathbf{u})} + R^2\,\|\mathbf{u}\|^2.$$

In particular, if there exists $\mathbf{u}$ such that $y_t\langle\mathbf{u},\mathbf{x}_t\rangle \ge 1$ for all t then

$$|\mathcal{M}| \le R^2\|\mathbf{u}\|^2.$$

Proof. The theorem follows from Equation (3.2) and the following claim: Given $x,b,c \in \mathbb{R}_+$, the inequality $x - b\sqrt{x} - c \le 0$ implies that $x \le c + b^2 + b\sqrt{c}$. The last claim can be easily derived by analyzing the roots of the convex parabola $Q(y) = y^2 - by - c$. □

The last assumption of Theorem 3.9 is called separability with large margin. That is, there exists $\mathbf{u}$ that not only satisfies that the point $\mathbf{x}_t$ lies on the correct side of the halfspace, it also guarantees that $\mathbf{x}_t$ is not too close to the decision boundary. More specifically, the distance from $\mathbf{x}_t$ to the decision boundary is at least $\gamma = 1/\|\mathbf{u}\|$ and the bound becomes $(R/\gamma)^2$. This classic mistake bound of the Perceptron appears in [3, 33].

When the separability assumption does not hold, the bound involves the terms $[1 - y_t\langle\mathbf{u},\mathbf{x}_t\rangle]_+$ which measures how much the separability with margin requirement is violated.

As a last remark we note that there can be cases in which there exists some $\mathbf{u}$ that makes zero errors on the sequence but the Perceptron will make many errors. Indeed, this is a direct consequence of the fact that $\mathrm{Ldim}(\mathcal{H}) = \infty$. The way we sidestep this impossibility result is by assuming more on the sequence of examples — the bound in Theorem 3.9 will be meaningful only if $\sum_t f_t(\mathbf{u})$ is not excessively large.

3.3.2 Winnow

Winnow is an online classification algorithm originally proposed for learning the class of k monotone disjunctive Boolean functions. Namely, $\mathcal{X} = \{0,1\}^d$ and a k monotone disjunction hypothesis takes the form $x[i_1] \vee \ldots \vee x[i_k]$, where $\{i_1,\ldots,i_k\} \subset [d]$. This can be written as a halfspace as follows. Let $\mathbf{w} \in \{0,1\}^d$ be a vector with exactly k elements that equal 1 (we call these elements the relevant variables). Then, $\langle\mathbf{w},\mathbf{x}\rangle$ will be at least 1 if one of the relevant variables is turned on in $\mathbf{x}$. Otherwise, $\langle\mathbf{w},\mathbf{x}\rangle$ is 0. So, $\mathrm{sign}(\langle\mathbf{w},\mathbf{x}\rangle - 1/2)$ behaves exactly like a k monotone disjunction and our hypothesis class is:

$$\mathcal{H} = \{\mathbf{x} \mapsto \mathrm{sign}(\langle\mathbf{w},\mathbf{x}\rangle - 1/2) : \mathbf{w} \in \{0,1\}^d, \|\mathbf{w}\|_1 = k\}.$$

In order to learn $\mathcal{H}$ we need to convexify the learning problem. The first step is to convexify the domain of $\mathbf{w}$ by simply enlarging it to be $S = \mathbb{R}_+^d$. The second step is to construct a surrogate convex loss function, similarly to the one we used for deriving the Perceptron. A weight vector $\mathbf{w}$ errs on $(\mathbf{x},y)$ if $\mathrm{sign}(\langle\mathbf{w},\mathbf{x}\rangle - 1/2) \neq y$, or equivalently if $y(2\langle\mathbf{w},\mathbf{x}\rangle - 1) \le 0$. Therefore, the original 0–1 loss function is

$$\ell(\mathbf{w},(\mathbf{x},y)) = \mathbf{1}_{[y(2\langle\mathbf{w},\mathbf{x}\rangle-1)\le 0]}.$$

On rounds on which the algorithm errs, we define the hinge-loss surrogate

$$f_t(\mathbf{w}) = [1 - y_t(2\langle \mathbf{w}, \mathbf{x}_t\rangle - 1)]_+ .$$

If the algorithm does not err we set $f_t(\mathbf{w}) = 0$. This function satisfies:

- f_t is convex.
- f_t is a surrogate: $f_t(\mathbf{w}_t) \geq \ell(\mathbf{w}_t, (\mathbf{x}_t, y_t))$.
- In the realizable case, there exists $\mathbf{u} \in \{0,1\}^d$ with $\|\mathbf{u}\|_1 = k$ such that $f_t(\mathbf{u}) = 0$ for all t.

Before we derive the Winnow algorithm, we note that we can use the Perceptron algorithm for learning this problem as follows. Denote $\phi(\mathbf{x})$ to be the vector $[2\mathbf{x}, -1]$, namely, we concatenate the constant -1 to the vector $2\mathbf{x}$. Then, the prediction can be performed according to $\text{sign}(\langle \mathbf{w}, \phi(\mathbf{x})\rangle)$ and the loss $f_t(\mathbf{w})$ becomes the hinge-loss. In the realizable case, this yields the mistake bound $R^2\|\mathbf{u}\|_2^2 = 4(d+1)k$. We shall see that the mistake bound of Winnow is $8k\log(d)$. That is, we obtain an exponential improvement in terms of the dependence on the dimension.

Winnow is a specialization of the unnormalized-EG algorithm, given in Section 2.8, to the aforementioned surrogate loss. We set the parameter λ of the unnormalized-EG algorithm to be $1/d$.

```
                         Winnow

parameter: $\eta > 0$
initialize: $\mathbf{w}_1 = (1/d, \ldots, 1/d)$
for $t = 1, 2, \ldots, T$
  receive $\mathbf{x}_t$
  predict $p_t = \text{sign}(2\langle \mathbf{w}_t, \mathbf{x}_t\rangle - 1)$
  if $y_t(2\langle \mathbf{w}_t, \mathbf{x}_t\rangle - 1) \leq 0$
    $\forall i,\ w_{t+1}[i] = w_t[i]e^{-\eta 2 y_t x_t[i]}$
  else
    $\mathbf{w}_{t+1} = \mathbf{w}_t$
```

Theorem 3.10. Suppose that the Winnow algorithm runs with a parameter $\eta \le 1/2$ on a sequence $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)$, where $\mathbf{x}_t \in \{0,1\}^d$ for all t. Let $\mathcal{M}$ be the rounds on which Winnow errs and let $f_t(\mathbf{w}) = \mathbf{1}_{[t \in \mathcal{M}]} \, [1 - y_t(2\langle \mathbf{w}, \mathbf{x}_t \rangle - 1)]_+$. Then, for any $\mathbf{u} \in \{0,1\}^d$ such that $\|\mathbf{u}\|_1 = k$ it holds that

$$|\mathcal{M}| \le \sum_{t=1}^{T} f_t(\mathbf{w}_t) \le \frac{1}{1 - 2\eta} \left(\sum_{t=1}^{T} f_t(\mathbf{u}) + \frac{k \log(d)}{\eta} \right).$$

In particular, if there exists such $\mathbf{u}$ for which $y_t(2\langle \mathbf{u}, \mathbf{x}_t \rangle - 1) \ge 1$ for all t then we can set $\eta = 1/4$ and obtain

$$|\mathcal{M}| \le 8\, k \log(d).$$

Proof. Let $\mathcal{M}$ be the set of rounds on which Winnow errs, i.e., $\mathcal{M} = \{t : y_t(2\langle \mathbf{w}_t, \mathbf{x}_t \rangle - 1) \le 0\}$. Winnow is derived from the unnormalized-EG algorithm with

$$\mathbf{z}_t = \begin{cases} 2 y_t \mathbf{x}_t & \text{if } t \in \mathcal{M} \\ \mathbf{0} & \text{if } t \notin \mathcal{M} \end{cases}$$

Using Theorem 2.23 (with $\lambda = 1/d$) we have

$$\sum_{t=1}^{T} \langle \mathbf{w}_t - \mathbf{u}, \mathbf{z}_t \rangle \le \frac{1 + \sum_i u[i] \log(d u[i]/e)}{\eta} + \eta \sum_{t=1}^{T} \sum_i w_t[i] z_t[i]^2.$$

In our case, $\mathbf{u} \in \{0,1\}^d$ and $\|\mathbf{u}\|_1 = k$ so

$$1 + \sum_i u[i] \log(d u[i]/e) \le 1 + k \log(d/e) \le k \log(d).$$

Plugging this in the above and using the fact that $\mathbf{z}_t$ is a sub-gradient of f_t at $\mathbf{w}_t$ we obtain:

$$\sum_{t=1}^{T} (f_t(\mathbf{w}_t) - f_t(\mathbf{u})) \le \sum_{t=1}^{T} \langle \mathbf{w}_t - \mathbf{u}, \mathbf{z}_t \rangle \le \frac{k \log(d)}{\eta} + \eta \sum_{t=1}^{T} \sum_i w_t[i] z_t[i]^2. \tag{3.3}$$

We next show that for all t,

$$\sum_i w_t[i] z_t[i]^2 \leq 2 f_t(\mathbf{w}_t). \tag{3.4}$$

To do so, it is convenient to consider three cases:

- If $t \notin \mathcal{M}$ then $f_t(\mathbf{w}_t) = 0$ and $\mathbf{z}_t = \mathbf{0}$. Hence, Equation (3.4) holds.
- If $t \in \mathcal{M}$ and $y_t = 1$ then the left-hand side of Equation (3.4) becomes $4\langle \mathbf{w}_t, \mathbf{x}_t \rangle$. This is because $\mathbf{x}_t \in \{0,1\}^d$ so $x_t[i]^2 = x_t[i]$. But since $t \in \mathcal{M}$ we also know that $2\langle \mathbf{w}_t, \mathbf{x}_t \rangle \leq 1$ and so $\sum_i w_t[i] z_t[i]^2 \leq 2$. On the other hand, by the surrogate property of f_t and the fact that $t \in \mathcal{M}$ we know that $f_t(\mathbf{w}_t) \geq 1$, which yields Equation (3.4).
- If $t \in \mathcal{M}$ and $y_t = -1$ then the left-hand side of Equation (3.4) becomes again $4\langle \mathbf{w}_t, \mathbf{x}_t \rangle$. However, now we have that $f_t(\mathbf{w}_t) = 2\langle \mathbf{w}_t, \mathbf{x}_t \rangle$, which yields Equation (3.4).

Combining Equation (3.4) with Equation (3.3) and rearranging terms we obtain

$$\sum_{t=1}^{T} f_t(\mathbf{w}_t) \leq \frac{1}{1 - 2\eta} \left(\sum_{t=1}^{T} f_t(\mathbf{u}) + \frac{k \log(d)}{\eta} \right).$$

Combining the above with the surrogate property we conclude our proof. □

3.4 Bibliographic Remarks

The Weighted Majority algorithm is due to [32] and [43]. The Standard Optimal Algorithm was derived by the seminal work of Littlestone [29]. A generalization to the nonrealizable case as well as other variants like margin-based Littlestone's dimension was derived in [7]. Characterizations of online learnability beyond classification has been obtained in [1, 35].

The Perceptron dates back to Rosenblatt [37]. An analysis for the realizable case (with margin assumptions) appears in [3, 33]. Freund and Schapire [17] presented an analysis for the unrealizable case with a squared-hinge-loss based on a reduction to the realizable case. A direct analysis for the unrealizable case with the hinge-loss was given by Gentile [18]. Winnow was invented and analyzed in the realizable case by Littlestone [29]. An analysis for the unrealizable case was carried out in [5].

4

Limited Feedback (Bandits)

In Section 2 we studied the general framework of online convex optimization and in particular, derived the family of online mirror descent algorithms. To apply this algorithm, it is required to find a sub-gradient of the loss function at the end of each round. In this section we study online learning problems where the learner knows the value of the loss function at the predicted vector but he doesn't know the value of the loss function at other points.

We first show that to apply the online mirror descent framework it suffices to know how to calculate an estimate of the gradient. We next show how this observation leads to a low regret algorithm for a famous problem called "the multi-armed bandit problem." This problem is similar to prediction with expert advice but at the end of each round the learner only observes the cost of the expert he picked and does not observe the costs of the rest of the experts.

Finally, we discuss the general problem of online convex optimization without gradient information.

4.1 Online Mirror Descent with Estimated Gradients

Recall the Online Mirror Descent (OMD) algorithm we described in Section 2. Now suppose that instead of setting $\mathbf{z}_t$ to be a sub-gradient of f_t at $\mathbf{w}_t$, we shall set $\mathbf{z}_t$ to be a random vector with $\mathbb{E}[\mathbf{z}_t] \in \partial f_t(\mathbf{w}_t)$.

```
Online Mirror Descent with Estimated
            Gradients

parameter: a link function $g : \mathbb{R}^d \to S$
initialize: $\boldsymbol{\theta}_1 = \mathbf{0}$
for $t = 1, 2, \ldots$
  predict $\mathbf{w}_t = g(\boldsymbol{\theta}_t)$
  pick $\mathbf{z}_t$ at random such that $\mathbb{E}[\mathbf{z}_t | \mathbf{z}_{t-1}, \ldots, \mathbf{z}_1] \in \partial f_t(\mathbf{w}_t)$
  update $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \mathbf{z}_t$
```

The following theorem tells us how to extend previous regret bounds we derived for OMD to the case of estimated sub-gradients.

Theorem 4.1. Suppose that the `Online Mirror Descent with Estimated Gradients` is run on a sequence of loss functions, $f_1, \ldots, f_T$. Suppose that the estimated sub-gradients are chosen such that with probability 1 we have

$$\sum_{t=1}^{T} \langle \mathbf{w}_t - \mathbf{u}, \mathbf{z}_t \rangle \leq B(\mathbf{u}) + \sum_{t=1}^{T} \|\mathbf{z}_t\|_t^2,$$

where B is some function and for all round t the norm $\|\cdot\|_t$ may depend on $\mathbf{w}_t$. Then,

$$\mathbb{E}\left[\sum_{t=1}^{T} (f_t(\mathbf{w}_t) - f_t(\mathbf{u}))\right] \leq B(\mathbf{u}) + \sum_{t=1}^{T} \mathbb{E}[\|\mathbf{z}_t\|_t^2],$$

where expectation is with respect to the randomness in choosing $\mathbf{z}_1, \ldots, \mathbf{z}_T$.

Proof. Taking expectation of both sides of the first inequality with respect to the randomness in choosing $\mathbf{z}_t$ we obtain that

$$\mathbb{E}\left[\sum_{t=1}^{T} \langle \mathbf{w}_t - \mathbf{u}, \mathbf{z}_t \rangle\right] \leq B(\mathbf{u}) + \sum_{t=1}^{T} \mathbb{E}[\|\mathbf{z}_t\|_t^2].$$

At each round, let $\mathbf{v}_t = \mathbb{E}[\mathbf{z}_t | \mathbf{z}_{t-1}, \ldots, \mathbf{z}_1]$. By the law of total probability we obtain that

$$\mathbb{E}\left[\sum_{t=1}^{T} \langle \mathbf{w}_t - \mathbf{u}, \mathbf{z}_t \rangle\right] = \mathbb{E}\left[\sum_{t=1}^{T} \langle \mathbf{w}_t - \mathbf{u}, \mathbf{v}_t \rangle\right].$$

Since we assume that $\mathbf{v}_t \in \partial f_t(\mathbf{w}_t)$ we know that

$$\langle \mathbf{w}_t - \mathbf{u}, \mathbf{v}_t \rangle \geq f_t(\mathbf{w}_t) - f_t(\mathbf{u}).$$

Combining all the above we conclude our proof. □

The above theorem tells us that as long as we can find $\mathbf{z}_1, \ldots, \mathbf{z}_T$ which on one hand are unbiased estimators of sub-gradients and on the other hand has bounded norms, we can still obtain a valid regret bound. In the next section we demonstrate how to construct such estimates for a specific problem and in Section 4.3 we derive a more general approach.

4.2 The Multi-armed Bandit Problem

In the multi-armed bandit problem, there are d arms, and on each online round the learner should choose one of the arms, denoted p_t, where the chosen arm can be a random variable. Then, it receives a cost of choosing this arm, $y_t[p_t] \in [0,1]$. The vector $\mathbf{y}_t \in [0,1]^d$ associates a cost for each of the arms, but the learner only gets to see the cost of the arm it pulls. Nothing is assumed about the sequence of vectors $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_T$.

This problem is similar to prediction with expert advice. The only difference is that the learner does not get to see the cost of experts he didn't choose. The goal of the learner is to have low regret for not always pulling the best arm,

$$\mathbb{E}\left[\sum_{t=1}^{T} y_t[p_t]\right] - \min_i \sum_{t=1}^{T} y_t[i],$$

where the expectation is over the learner's own randomness.

This problem nicely captures the exploration–exploitation tradeoff. On one hand, we would like to pull the arm which, based on previous rounds, we believe has the lowest cost. On the other hand, maybe it is better to explore the arms and find another arm with a smaller cost.

To approach the multi-armed bandit problem we use the OMD with estimated gradients method derived in the previous section. As in the Weighted Majority algorithm for prediction with expert advice, we let S be the probability simplex and the loss functions be $f_t(\mathbf{w}) = \langle \mathbf{w}, \mathbf{y}_t \rangle$. The learner picks an arm according to $\mathbb{P}[p_t = i] = w_t[i]$ and therefore $f_t(\mathbf{w}_t)$ is the expected cost of the chosen arm. The gradient of the loss function is $\mathbf{y}_t$. However, we do not know the value of all elements of $\mathbf{y}_t$, we only get to see the value $y_t[p_t]$. To estimate the gradient, we define the random vector $\mathbf{z}_t$ as follows:

$$z_t[j] = \begin{cases} y_t[j]/w_t[j] & \text{if } j = p_t \\ 0 & \text{else} \end{cases} .$$

To emphasize the dependence of $\mathbf{z}_t$ on p_t we will sometimes use the notation $\mathbf{z}_t^{(p_t)}$. We indeed have that $\mathbf{z}_t^{(p_t)}$ is an unbiased estimate of the gradient because

$$\mathbb{E}[z_t^{(p_t)}[j] | \mathbf{z}_{t-1}, \ldots, \mathbf{z}_1] = \sum_{i=1}^{d} \mathbb{P}[p_t = i] z_t^{(i)}[j] = w_t[j] \frac{y_t[j]}{w_t[j]} = y_t[j].$$

We update $\mathbf{w}_t$ using the update rule of the normalized EG algorithm. The resulting algorithm is given below.

```
Multi-Armed Bandit Algorithm

parameter: $\eta \in (0,1)$
initialize: $\mathbf{w}_1 = (1/d, \ldots, 1/d)$
for $t = 1, 2, \ldots$
  choose $p_t \sim \mathbf{w}_t$ and pull the $p_t$'th arm
  receive cost of the arm $y_t[p_t] \in [0,1]$
  update
    $\tilde{w}[p_t] = w_t[p_t] e^{-\eta y_t[p_t]/w_t[p_t]}$
    for $i \neq p_t$, $\tilde{w}[i] = w_t[i]$
    $\forall i,\ w_{t+1}[i] = \frac{\tilde{w}[i]}{\sum_j \tilde{w}[j]}$
```

To analyze the algorithm we combining Theorem 4.1 with Theorem 2.22. The conditions of Theorem 2.22 hold here because $\mathbf{z}_t$ is a non-negative vector. We therefore obtain that

$$\sum_{t=1}^{T} \langle \mathbf{w}_t - \mathbf{u}, \mathbf{z}_t \rangle \leq \frac{\log(d)}{\eta} + \eta \sum_{t=1}^{T} \sum_i w_t[i] z_t[i]^2.$$

So Theorem 4.1 gives us that

$$\mathbb{E}\left[\sum_{t=1}^{T} (f_t(\mathbf{w}_t) - f_t(\mathbf{u}))\right] \leq \frac{\log(d)}{\eta} + \eta \sum_{t=1}^{T} \mathbb{E}\left[\sum_i w_t[i] z_t[i]^2\right].$$

The last term can be bounded as follows:

$$\begin{aligned} \mathbb{E}\left[\sum_i w_t[i] z_t^{(p_t)}[i]^2 \mid \mathbf{z}_{t-1}, \ldots, \mathbf{z}_1\right] &= \sum_j \mathbb{P}[p_t = j] \sum_i w_t[i] z_t^{(j)}[i]^2 \\ &= \sum_j w_t[j] w_t[j] (y_t[j]/w_t[j])^2 \\ &= \sum_j y_t[j]^2 \leq d. \end{aligned}$$

Overall, we have obtained the following:

Corollary 4.2. The multi-armed bandit algorithm enjoys the bound

$$\mathbb{E}\left[\sum_{t=1}^{T} y_t[p_t]\right] \;\leq\; \min_i \sum_{t=1}^{T} y_t[i] + \frac{\log d}{\eta} + \eta d T.$$

In particular, setting $\eta = \sqrt{\log(d)/(dT)}$ we obtain the regret bound of $2\sqrt{d \log(d) T}$.

Comparing the above bound to the bound we derived for the Weighted Majority algorithm we observe an additional factor of d, which intuitively stems from the fact that here we only receive $1/d$ of the feedback the Weighted Majority algorithm receives. It is possible to rigorously show that the dependence on d is unavoidable and that the bound we derived is essentially tight.

4.3 Gradient Descent Without a Gradient

In this section we consider the general online convex optimization problem, where we only have a black-box access to the loss functions, and thus cannot calculate sub-gradients directly. The multi-armed bandit problem discussed in the previous section was a special case, where the loss function was linear. However, in the multi-armed bandit problem we had the additional constraint that we can only ask the value of the loss function at singletons vectors. Here we allow the learner to predict any vector[1] but he only receives the evaluation of the loss function on the vector he picked.

In Section 2 we derived several gradient-based algorithms for online convex optimization. How can we utilize these algorithms when we can only receive the evaluation of the loss function on the predicted vector? As in the multi-armed bandit problem, the main idea is to derive a one-shot estimate to the gradient. Below we present a method due to Flaxman, Kalai, and McMahan [16], which derives a one-shot estimate to the gradient.

4.3.1 A One-Shot Gradient Estimate

Let U_{b} be the uniform distribution over the unit Euclidean ball and let U_{sp} be the uniform distribution over the unit Euclidean sphere. Given $\delta > 0$ we define a smoothed version of f as follows:

$$\hat{f}(\mathbf{w}) = \mathop{\mathbb{E}}_{\mathbf{v} \sim U_{\mathrm{b}}}[f(\mathbf{w} + \delta \mathbf{v})]. \tag{4.1}$$

As we will show below, the advantage of $\hat{f}$ is that it is differentiable and we can estimate its gradient using a single oracle call to f. But, before that, we show that $\hat{f}$ is similar to f.

Lemma 4.3. Let f be an L-Lipschitz function and let $\hat{f}$ be as defined in Equation (4.1). Then, $|\hat{f}(\mathbf{w}) - f(\mathbf{w})| \le L\delta$.

[1] We make another simplification, for the sake of simplicity, that the learner does not have to predict an element from S. In some applications, like the multi-armed bandit problem, the learner must predict an element from S and then a more careful analysis is required. See for example [16].

Proof. By the Lipschitzness, $|f(\mathbf{w}) - f(\mathbf{w} + \delta\mathbf{v})| \leq L\delta\|\mathbf{v}\|$. Using the fact that $\|\mathbf{v}\| \leq 1$ we obtain the desired result. □

We next show that the gradient of $\hat{f}$ can be estimated using a single oracle call to f.

Lemma 4.4. The function $\hat{f}$ is differentiable and we have

$$\mathop{\mathbb{E}}_{\mathbf{v}\sim U_{\mathrm{sp}}}\left[\frac{d}{\delta} f(\mathbf{w} + \delta\mathbf{v})\mathbf{v}\right] = \nabla\hat{f}(\mathbf{w}).$$

Proof. We prove the theorem for the case $d = 1$. Given a 1-dimensional function f, let F be the antiderivative of f, namely, $F'(w) = f(w)$. By the fundmental theorem of calculus we have

$$\int_a^b f(w)dw = F(b) - F(a).$$

It follows that

$$\int_{-\delta}^{\delta} f(w + t)dt = F(w + \delta) - F(w - \delta).$$

Note that in the 1-dimensional case we have that v is distributed uniformly over $[-1,1]$ and

$$\hat{f}(w) = \mathbb{E}[f(w + \delta v)] = \frac{\int_{-\delta}^{\delta} f(w + t)dt}{2\delta} = \frac{F(w + \delta) - F(w - \delta)}{2\delta}.$$

It follows that

$$\hat{f}'(w) = \frac{f(w + \delta) - f(w - \delta)}{2\delta} = \frac{\mathbb{E}_{v\sim U_{\mathrm{sp}}}[f(w + \delta v)v]}{\delta}.$$

This concludes the proof for the case $d = 1$. The proof for $d > 1$ follows similarly using Stoke's theorem and can be found in [16]. □

4.3.2 The Resulting Algorithm

The algorithm is based on `gradient descent with lazy projections`, we described in Section 2.6.

```
Bandit Online Gradient Descent

parameters: $\eta, \delta > 0$ and a convex set $S \subset \mathbb{R}^d$
initialize: $\boldsymbol{\theta}_1 = \mathbf{0}$
for $t = 1, 2, \ldots, T$
  let $\mathbf{w}_t = \operatorname{argmin}_{\mathbf{w} \in S} \|\mathbf{w} - \eta \boldsymbol{\theta}_t\|_2$
  pick $\mathbf{v}_t \sim U_{\mathrm{sp}}$
  predict $\mathbf{w}_t + \delta \mathbf{v}_t$ and receive $f_t(\mathbf{w}_t + \delta \mathbf{v}_t)$
  set $\mathbf{z}_t = \frac{d}{\delta} f_t(\mathbf{w}_t + \delta \mathbf{v}_t) \mathbf{v}_t$
  update $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \mathbf{z}_t$
```

Let us now analyze the regret of this algorithm. First, applying Corollary 2.17 we obtain that for all $\mathbf{u} \in S$,

$$\sum_t \langle \mathbf{z}_t, \mathbf{w}_t - \mathbf{u} \rangle \leq \frac{1}{2\eta} \|\mathbf{u}\|_2^2 + \eta \sum_{t=1}^T \|\mathbf{z}_t\|^2.$$

Taking expectation, using Lemma 4.4, and using the fact that $\mathbf{v}_t$ is in the unit sphere, we obtain

$$\mathbb{E}\left[\sum_t (\hat{f}_t(\mathbf{w}_t) - \hat{f}_t(\mathbf{u}))\right] \leq \frac{1}{2\eta} \|\mathbf{u}\|_2^2 + \eta \sum_{t=1}^T \frac{d^2}{\delta^2} \mathbb{E}[f_t(\mathbf{w}_t + \delta \mathbf{v}_t)^2]. \qquad (4.2)$$

Let $B = \max_{\mathbf{u} \in S} \|\mathbf{u}\|$ and $F = \max_{\mathbf{u} \in S, t} f_t(\mathbf{u})$. Then, by the Lipschitzness of f_t we have $f_t(\mathbf{w}_t + \delta \mathbf{v}_t) \leq f_t(\mathbf{w}_t) + L\delta \|\mathbf{v}_t\| \leq F + L\delta$. Combining with Equation (4.2) yields

$$\mathbb{E}\left[\sum_t (\hat{f}_t(\mathbf{w}_t) - \hat{f}_t(\mathbf{u}))\right] \leq \frac{B^2}{2\eta} + \eta T d^2 (F/\delta + L)^2. \qquad (4.3)$$

To derive a concrete bound out of the above we need to relate the regret with respect to $\hat{f}_t$ to the regret with respect to f_t. We do this using Lemma 4.3, which implies

$$f_t(\mathbf{w}_t + \delta \mathbf{v}_t) - f_t(\mathbf{u}) \leq f_t(\mathbf{w}_t) - f_t(\mathbf{u}) + L\delta \leq \hat{f}_t(\mathbf{w}_t) - \hat{f}_t(\mathbf{u}) + 3L\delta.$$

Combining the above with Equation (4.3) yields

Corollary 4.5. Consider running the `Bandit Online Convex Optimization` algorithm on a sequence $f_1, \ldots, f_T$ of L-Lipschitz functions. Let S be a convex set and define $B = \max_{\mathbf{u} \in S} \|\mathbf{u}\|$ and $F = \max_{\mathbf{u} \in S, t \in [T]} f_t(\mathbf{u})$. Then, for all $\mathbf{u} \in S$ we have

$$\mathbb{E}\left[\sum_t (f_t(\mathbf{w}_t + \delta \mathbf{v}_t) - f_t(\mathbf{u}))\right] \leq 3L\delta T + \frac{B^2}{2\eta} + \eta T d^2 (F/\delta + L)^2.$$

In particular, setting $\eta = \frac{B}{d(F/\delta + L)\sqrt{2T}}$, $\delta = \sqrt{\frac{BdF}{3L}} T^{-1/4}$ we obtain that the regret is bounded by $O(\sqrt{BdFL}T^{3/4})$.

Comparing this bound to the full information bound given in Corollary 2.17, we note two main difference. First, the dependence on T is $T^{3/4}$ as opposed to $T^{1/2}$ in the full information case. Second, the regret depends on the dimensionality. While the dependence on the dimensionality is indeed tight, it is not known if the worse dependence on T is tight.

4.4 Bibliographic Remarks

Multi-armed bandit problems were originally studied in a stochastic setting [36]. The adversarial multi-armed bandit problem was studied in [4].

The algorithm we described for bandit online convex optimization is a variant of the algorithm given in [16]. Better algorithms have been derived for the specific case of *linear* functions. In particular, [15] gave a non efficient algorithm whose regret is $O(\text{poly}(d)\sqrt{T})$, and later on, [2] derived an efficient algorithm with a similar regret bound using self-concordant regularizers.

5

Online-to-Batch Conversions

In this section we discuss conversions from online learning to stochastic learning. In particular, we consider the following general model of stochastic learning.

Definition 5.1. (Vapnik's General Setting of Learning) Let S be a hypothesis class and let Ψ be an example domain set. There is a loss function, $c : S \times \Psi \to \mathbb{R}$, which gets a hypothesis and an example and returns the cost of using the hypothesis on the example. Let Q be an unknown distribution over Ψ and define[1] $C(\mathbf{w}) = \mathbb{E}_{\psi \sim Q}[c(\mathbf{w}, \psi)]$. The goal of the learner is to approximately minimize C over S. The learner does not know Q, but can get independent samples from Q. We denote by $A(\psi_1, \ldots, \psi_T)$ the output of the learning algorithm when receiving T samples from Q. We say that a learning algorithm ϵ-learns S using T

[1] Technically, in the above definition, for every $\mathbf{w} \in S$, we view the function $c(\mathbf{w}, \cdot) : \Psi \to \mathbb{R}^+$ as a random variable, and define $C(\mathbf{w})$ to be the expected value of this random variable. For that, we need to require that the function $c(\mathbf{w}, \cdot)$ is measurable. Formally, we assume that there is a σ-algebra of subsets of Ψ, over which the probability Q is defined, and that the pre-image of every initial segment in $\mathbb{R}^+$ is in this σ-algebra.

examples if

$$\mathbb{E}[C(A(\psi_1,\ldots,\psi_T))] \le \min_{\mathbf{w}\in S} C(\mathbf{w}) + \epsilon,$$

where expectation[2] is over the random choice of $\psi_1,\ldots,\psi_T$.

This model is also closely related to the problem of Stochastic Optimization or more specifically Stochastic Approximation (for more details see for example [41]).

Maybe the most simple algorithm for solving a stochastic learning problem is by sampling T examples, $\psi_1,\ldots,\psi_T$, and then returning a hypothesis which minimizes the average cost over these examples. This is called empirical risk minimization (ERM):

$$\mathrm{ERM}_S(\psi_1,\ldots,\psi_T) \;\in\; \operatorname*{argmin}_{\mathbf{w}\in S} \frac{1}{T}\sum_{t=1}^{T} c(\mathbf{w},\psi_t).$$

Our goal is to demonstrate how online learning can sometimes yield an alternative algorithm for solving stochastic learning problems.

The basic idea is as follows. First, we run the online learner on a sequence of loss functions, where $f_t(\mathbf{w}) = c(\mathbf{w},\psi_t)$. This produces a sequence of predictions $\mathbf{w}_1,\ldots,\mathbf{w}_T$. Second, we produce a single vector $\bar{\mathbf{w}}$ based on $\mathbf{w}_1,\ldots,\mathbf{w}_T$. There are many ways to produce $\bar{\mathbf{w}}$ and here we consider two simple approaches: averaging or randomization. This yields the following skeleton.

```
Online-To-Batch Conversion

 parameters: a set $S$; a cost function $c(\cdot,\cdot)$ ;
   an online learning algorithm $A$
input: $\psi_1,\ldots,\psi_T$ are independently sampled from distribution $Q$ over $\Psi$
for $t = 1,\ldots,T$
  let $\mathbf{w}_t$ be the prediction of $A$
  provide the loss function $f_t(\mathbf{w}) = c(\mathbf{w},\psi_t)$ to $A$
output: produce $\bar{\mathbf{w}}$ from $\mathbf{w}_1,\ldots,\mathbf{w}_T$. For example:
  AVERAGING: $\bar{\mathbf{w}} = \frac{1}{T}\sum_{t=1}^{T}\mathbf{w}_t$
  RANDOMIZATION: $\bar{\mathbf{w}} = \mathbf{w}_r$,
    where $r$ is chosen uniformly at random from $[T]$
```

[2] It is common to require that the inequality will hold with high probability. By a simple amplification argument it is possible to convert a guarantee on the expected value to a guarantee that holds with high probability. Therefore, for the sake of simplicity, we only require success in expectation.

The following theorem shows that the average online loss of the online learner upper bounds the cost of $\bar{\mathbf{w}}$.

Theorem 5.1. Let $\psi_1,\ldots,\psi_T$ be a sequence of independent random variables, each of which is distributed according to a distribution Q over Ψ. For the ONLINE-TO-BATCH CONVERSION WITH RANDOMIZATION, we have that

$$\mathbb{E}\left[C(\bar{\mathbf{w}})\right] = \mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T} f_t(\mathbf{w}_t)\right],$$

where expectation is with respect to the random choice of $\psi_1,\ldots,\psi_T$ and r. If we further assume that C is a convex function, then for ONLINE-TO-BATCH CONVERSION WITH AVERAGING we have that

$$\mathbb{E}\left[C(\bar{\mathbf{w}})\right] \le \mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T} f_t(\mathbf{w}_t)\right].$$

Proof. We first show that

$$\mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T} C(\mathbf{w}_t)\right] = \mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T} c(\mathbf{w}_t,\psi_t)\right]. \tag{5.1}$$

Using the linearity of expectation we have,

$$\mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T} c(\mathbf{w}_t,\psi_t)\right] = \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[c(\mathbf{w}_t,\psi_t)]. \tag{5.2}$$

Recall that the law of total expectation implies that for any two random variables R_1,R_2, and a function f, $\mathbb{E}_{R_1}[f(R_1)] = \mathbb{E}_{R_2}\mathbb{E}_{R_1}[f(R_1)|R_2]$. Since $\mathbf{w}_t$ only depends on $\psi_1,\ldots,\psi_{t-1}$, we can set $R_1 = \psi_1,\ldots,\psi_t$ and $R_2 = \psi_1,\ldots,\psi_{t-1}$ to get that

$$\mathbb{E}[c(\mathbf{w}_t,\psi_t)] = \mathbb{E}[C(\mathbf{w}_t)].$$

Combining the above with Equation (5.2) yields Equation (5.1). Now, for the randomization technique we have that $C(\bar{\mathbf{w}}) = \mathbb{E}_r[C(\mathbf{w}_r)] = \frac{1}{T}\sum_{t=1}^{T} C(\mathbf{w}_t)$, which concludes our proof for the case of the randomization technique. For the averaging technique, since we assume that C is convex we can apply Jensen's inequality to get that

$$C(\bar{\mathbf{w}}) = C\left(\frac{1}{T}\sum_{t=1}^{T}\mathbf{w}_t\right) \leq \frac{1}{T}\sum_{t=1}^{T}C(\mathbf{w}_t). \qquad \square$$

Clearly, $\mathbb{E}[\frac{1}{T}\sum_{t=1}^{T}c(\mathbf{u},\psi_t)] = C(\mathbf{u})$. Therefore,

Corollary 5.2. Assume that the conditions of Theorem 5.1 hold and let $\mathbf{u}$ be any vector, then

$$\mathbb{E}[C(\bar{\mathbf{w}})] - C(\mathbf{u}) \leq \mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T}(f_t(\mathbf{w}_t) - f_t(\mathbf{u}))\right].$$

The right-hand side is the expected regret of the online algorithm (divided by T). Therefore, if we have an online learning algorithm that is guaranteed to have a low regret with respect to functions of the form $f(\mathbf{w}) = c(\mathbf{w},\psi)$ relative to a set S, we can be assured that the cost of $\bar{\mathbf{w}}$ is close to the optimal cost.

To demonstrate the usage of this approach, we describe the applicability of online mirror descent to stochastic learning of problems in which c is convex with respect to its first argument.

```
Stochastic Online Mirror Descent

parameters: a link function $g : \mathbb{R}^d \to S$; number of rounds $T$
input: sampler from distribution $Q$ over $\Psi$; cost function $c(\cdot,\cdot)$
initialize: $\boldsymbol{\theta}_1 = \mathbf{0}$
for $t = 1,\ldots,T$
  predict $\mathbf{w}_t = g(\boldsymbol{\theta}_t)$
  pick $\psi_t \sim Q$ and let $f_t(\mathbf{w}) = c(\mathbf{w},\psi_t)$
  let $\mathbf{z}_t \in \partial f_t(\mathbf{w}_t)$
  update $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \mathbf{z}_t$
output: $\bar{\mathbf{w}} = \frac{1}{T}\sum_{t=1}^{T}\mathbf{w}_t$
```

5.1 Bibliographic Remarks

Littlestone [30] initiated the study of online to batch conversions. Freund and Schapire [17] demonstrated that using the Perceptron algorithm along with the voting online-to-batch technique [23] yields an efficient replacement for Support Vector Machines [14, 38, 42]. Concentration bounds on the risk of the ensemble that rely on the online loss were derived by Zhang [45] and Cesa-Bianchi et al. [10, 11].

Acknowledgments

The author would like to thank the anonymous reviewers for their helpful comments.

References

[1] J. Abernethy, P. L. Bartlett, A. Rakhlin, and A. Tewari, “Optimal strategies and minimax lower bounds for online convex games,” in *Proceedings of the Annual Conference on Computational Learning Theory*, 2008.

[2] J. Abernethy, E. Hazan, and A. Rakhlin, “Competing in the dark: An efficient algorithm for bandit linear optimization,” in *Proceedings of the Annual Conference on Learning Theory (COLT)*, no. 3, 2008.

[3] S. Agmon, “The relaxation method for linear inequalities,” *Canadian Journal of Mathematics*, vol. 6, no. 3, pp. 382–392, 1954.

[4] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire, “The nonstochastic multiarmed bandit problem,” *SICOMP: SIAM Journal on Computing*, vol. 32, no. 1, pp. 48–77, 2003.

[5] P. Auer and M. Warmuth, “Tracking the best disjunction,” *Machine Learning*, vol. 32, no. 2, pp. 127–150, 1998.

[6] K. Azoury and M. Warmuth, “Relative loss bounds for on-line density estimation with the exponential family of distributions,” *Machine Learning*, vol. 43, no. 3, pp. 211–246, 2001.

[7] S. Ben-David, D. Pal, and S. Shalev-Shwartz, “Agnostic online learning,” in *Proceedings of the Annual Conference on Learning Theory (COLT)*, 2009.

[8] A. Blum, “On-line algorithms in machine learning,” *Online Algorithms*, pp. 306–325, 1998.

[9] J. Borwein and A. Lewis, *Convex Analysis and Nonlinear Optimization*. Springer, 2006.

[10] N. Cesa-Bianchi, A. Conconi, and C. Gentile, “On the generalization ability of on-line learning algorithms,” *IEEE Transactions on Information Theory*, vol. 50, no. 9, pp. 2050–2057, 2004.

[11] N. Cesa-Bianchi and C. Gentile, "Improved risk tail bounds for on-line algorithms," *Neural Information Proceesing Systems (NIPS)*, 2006.

[12] N. Cesa-Bianchi and G. Lugosi, *Prediction, Learning, and Games.* Cambridge University Press, 2006.

[13] T. M. Cover, "Behavior of sequential predictors of binary sequences," in *Transactions on Prague Conference on Information Theory Statistical Decision Functions, Random Processes*, pp. 263–272, 1965.

[14] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines.* Cambridge University Press, 2000.

[15] V. Dani, T. Hayes, and S. M. Kakade, "The price of bandit information for online optimization," *Advances in Neural Information Processing Systems*, vol. 20, pp. 345–352, 2008.

[16] A. D. Flaxman, A. T. Kalai, and H. B. McMahan, "Online convex optimization in the bandit setting: Gradient descent without a gradient," in *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, Society for Industrial and Applied Mathematics, pp. 385–394, 2005.

[17] Y. Freund and R. E. Schapire, "Large margin classification using the perceptron algorithm," *Machine Learning*, vol. 37, no. 3, pp. 277–296, 1999.

[18] C. Gentile, "The robustness of the p-norm algorithms," *Machine Learning*, vol. 53, no. 3, pp. 265–299, 2003.

[19] C. Gentile and N. Littlestone, "The robustness of the p-norm algorithms," in *Proceedings of the Annual Conference on Learning Theory (COLT)*, 1999.

[20] G. Gordon, "Regret bounds for prediction problems," in *Proceedings of the Annual Conference on Learning Theory (COLT)*, 1999.

[21] A. J. Grove, N. Littlestone, and D. Schuurmans, "General convergence results for linear discriminant updates," *Machine Learning*, vol. 43, no. 3, pp. 173–210, 2001.

[22] E. Hazan, *The Convex Optimization Approach to Regret Minimization.* 2009.

[23] D. P. Helmbold and M. Warmuth, "On weak learning," *Journal of Computer and System Sciences*, vol. 50, pp. 551–573, 1995.

[24] S. Kakade, S. Shalev-Shwartz, and A. Tewari, "Regularization techniques for learning with matrices," *Journal of Machine Learning Research*, 2012 (To appear).

[25] A. Kalai and S. Vempala, "Efficient algorithms for online decision problems," *Journal of Computer and System Sciences*, vol. 71, no. 3, pp. 291–307, 2005.

[26] J. Kivinen and M. Warmuth, "Exponentiated gradient versus gradient descent for linear predictors," *Information and Computation*, vol. 132, no. 1, pp. 1–64, 1997.

[27] J. Kivinen and M. K. Warmuth, "Additive versus exponentiated gradient updates for linear prediction," *Symposium on Theory of Computing* (*STOC*), See also Technical Report UCSC-CRL-94-16, University of California, Santa Cruz, Computer Research Laboratory, pp. 209–218, 1995.

[28] J. Kivinen and M. Warmuth, "Relative loss bounds for multidimensional regression problems," *Machine Learning*, vol. 45, no. 3, pp. 301–329, 2001.

[29] N. Littlestone, "Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm," *Machine Learning*, vol. 2, pp. 285–318, 1988.

[30] N. Littlestone, “From on-line to batch learning,” in *Proceedings of the Annual Conference on Learning Theory (COLT)*, pp. 269–284, July 1989.

[31] N. Littlestone, “Mistake bounds and logarithmic linear-threshold learning algorithms,” Ph.D. Thesis, University of California at Santa Cruz, 1990.

[32] N. Littlestone and M. K. Warmuth, “The weighted majority algorithm,” *Information and Computation*, vol. 108, pp. 212–261, 1994.

[33] M. Minsky and S. Papert, *Perceptrons: An Introduction to Computational Geometry.* The MIT Press, 1969.

[34] A. Rakhlin, *Lecture Notes on Online Learning.* draft, 2009.

[35] A. Rakhlin, K. Sridharan, and A. Tewari, “Online learning: Random averages, combinatorial parameters, and learnability,” in *Neural Information Processing Systems (NIPS)*, 2010.

[36] H. Robbins, “Some aspects of the sequential design of experiments,” *Bulletin American Mathematical Society*, vol. 55, pp. 527–535, 1952.

[37] F. Rosenblatt, “The perceptron: A probabilistic model for information storage and organization in the brain,” *Psychological Review*, vol. 65, pp. 386–407, 1958. (Reprinted in Neurocomputing, (MIT Press, 1988)).

[38] B. Schölkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond.* MIT Press, 2002.

[39] S. Shalev-Shwartz, “Online learning: Theory, algorithms, and applications,” Ph.D. Thesis, The Hebrew University, 2007.

[40] S. Shalev-Shwartz and Y. Singer, “A primal-dual perspective of online learning algorithms,” *Machine Learning Journal*, vol. 69, no. 2, pp. 115–142, 2007.

[41] J. C. Spall, *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*, vol. 64. LibreDigital, 2003.

[42] V. N. Vapnik, *Statistical Learning Theory.* Wiley, 1998.

[43] Volodimir G. Vovk, “Aggregating Strategies,” in *Proceedings of the Annual Conference on Learning Theory (COLT)*, pp. 371–383, 1990.

[44] C. Zalinescu, *Convex Analysis in General Vector Spaces.* River Edge, NJ: World Scientific Publishing Co. Inc., 2002.

[45] T. Zhang, “Data dependent concentration bounds for sequential prediction algorithms,” in *Proceedings of the Annual Conference on Learning Theory (COLT)*, pp. 173–187, 2005.

[46] M. Zinkevich, “Online convex programming and generalized infinitesimal gradient ascent,” in *International Conference on Machine Learning*, 2003.