21 Online Learning

In this chapter we describe a different model of learning, which is called *online* learning. Previously, we studied the PAC learning model, in which the learner first receives a batch of training examples, uses the training set to learn a hypothesis, and only when learning is completed uses the learned hypothesis for predicting the label of new examples. In our papayas learning problem, this means that we should first buy a bunch of papayas and taste them all. Then, we use all of this information to learn a prediction rule that determines the taste of new papayas. In contrast, in online learning there is no separation between a training phase and a prediction phase. Instead, each time we buy a papaya, it is first considered a *test* example since we should predict whether it is going to taste good. Then, after taking a bite from the papaya, we know the true label, and the same papaya can be used as a *training* example that can help us improve our prediction mechanism for future papayas.

Concretely, online learning takes place in a sequence of consecutive rounds. On each online round, the learner first receives an instance (the learner buys a papaya and knows its shape and color, which form the instance). Then, the learner is required to predict a label (is the papaya tasty?). At the end of the round, the learner obtains the correct label (he tastes the papaya and then knows whether it is tasty or not). Finally, the learner uses this information to improve his future predictions.

To analyze online learning, we follow a similar route to our study of PAC learning. We start with online binary classification problems. We consider both the realizable case, in which we assume, as prior knowledge, that all the labels are generated by some hypothesis from a given hypothesis class, and the unrealizable case, which corresponds to the agnostic PAC learning model. In particular, we present an important algorithm called *Weighted-Majority*. Next, we study online learning problems in which the loss function is convex. Finally, we present the *Perceptron* algorithm as an example of the use of surrogate convex loss functions in the online learning model.

21.1 Online Classification in the Realizable Case

Online learning is performed in a sequence of consecutive rounds, where at round t the learner is given an instance, $\mathbf{x}_t$, taken from an instance domain $\mathcal{X}$, and is required to provide its label. We denote the predicted label by p_t. After predicting the label, the correct label, $y_t \in \{0, 1\}$, is revealed to the learner. The learner's goal is to make as few prediction mistakes as possible during this process. The learner tries to deduce information from previous rounds so as to improve its predictions on future rounds.

Clearly, learning is hopeless if there is no correlation between past and present rounds. Previously in the book, we studied the PAC model in which we assume that past and present examples are sampled i.i.d. from the same distribution source. In the online learning model we 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 e-mail filtering). Naturally, an adversary can make the number of prediction mistakes of our online learning algorithm arbitrarily large. For example, the adversary can present the same instance on each online round, wait for the learner's prediction, and provide the opposite label as the correct label.

To make nontrivial statements we must further restrict the problem. The realizability assumption is one possible natural restriction. In the realizable case, we assume that all the labels are generated by some hypothesis, $h^\star : \mathcal{X} \to \mathcal{Y}$. Furthermore, $h^\star$ is taken from a hypothesis class $\mathcal{H}$, which is known to the learner. This is analogous to the PAC learning model we studied in Chapter 3. With this restriction on the sequence, the learner should make as few mistakes as possible, assuming that both $h^\star$ and the sequence of instances 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 instances 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. Formally:

DEFINITION 21.1 (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 $M_A(\mathcal{H}) \leq B < \infty$.

Our goal is to study which hypothesis classes are learnable in the online model, and in particular to find good learning algorithms for a given hypothesis class.

Remark 21.1 Throughout this section and the next, we ignore the computa-

tional aspect of learning, and do not restrict the algorithms to be efficient. In Section 21.3 and Section 21.4 we study efficient online learning algorithms.

To simplify the presentation, we start with the case of a finite hypothesis class, namely, $|\mathcal{H}| < \infty$.

In PAC learning, we identified ERM as a good learning algorithm, in the sense that if $\mathcal{H}$ is learnable then it is learnable by the rule $\text{ERM}_{\mathcal{H}}$. A natural learning rule for online learning is to use (at any online round) any ERM hypothesis, namely, 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 label $y_t = h^\star(\mathbf{x}_t)$
  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 the following:

COROLLARY 21.2 *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 (see Exercise 1). Therefore, 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.

`Halving`

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

THEOREM 21.3 *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 this inequality we conclude our proof. □

Of course, `Halving`'s mistake bound is much better than `Consistent`'s mistake bound. We already see that online learning is different from PAC learning—while in PAC, any ERM hypothesis is good, in online learning choosing an arbitrary ERM hypothesis is far from being optimal.

21.1.1 Online Learnability

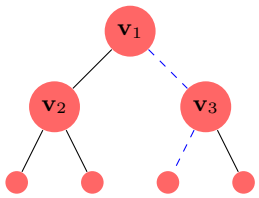
We next take a more general approach, and aim at characterizing online learnability. In particular, we target the following question: What is the optimal online learning algorithm for a given hypothesis class $\mathcal{H}$?

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

To motivate the definition of Ldim it is convenient to view the online learning process as a game between two players: the learner versus 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 it should 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 $t \in [T]$.

A strategy for an adversarial environment can be formally described as a binary tree, as follows. Each node of the tree is associated with an instance from $\mathcal{X}$. Initially, the environment presents to the learner the instance associated with the root of the tree. Then, if the learner predicts $p_t = 1$ the environment will declare that this is a wrong prediction (i.e., $y_t = 0$) and will traverse to the right child of the current node. If the learner predicts $p_t = 0$ then the environment will set $y_t = 1$ and will traverse to the left child. This process will continue and at each round, the environment will present the instance associated with the current node.

Formally, consider 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 $2^{T+1} - 1$ nodes in such a tree, and we attach an instance to each node. Let $\mathbf{v}_1, \ldots, \mathbf{v}_{2^{T+1}-1}$ 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



	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

Figure 21.1 An illustration of a shattered tree of depth 2. The dashed 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).

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 preceding strategy for the environment succeeds only if for every $(y_1, \ldots, y_T)$ 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 21.4 ($\mathcal{H}$ Shattered Tree) A shattered tree of depth d is a sequence of instances $\mathbf{v}_1, \ldots, \mathbf{v}_{2^d-1}$ in $\mathcal{X}$ such that for every labeling $(y_1, \ldots, y_d) \in \{0,1\}^d$ 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 21.1.

DEFINITION 21.5 (Littlestone's Dimension (Ldim)) $\mathrm{Ldim}(\mathcal{H})$ is the maximal integer T such that there exists a shattered tree of depth T, which is shattered by $\mathcal{H}$.

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

LEMMA 21.6 *No algorithm can have a mistake bound strictly smaller than* $\mathrm{Ldim}(\mathcal{H})$*; namely, for every algorithm, A, we have* $M_A(\mathcal{H}) \geq \mathrm{Ldim}(\mathcal{H})$.

Proof Let $T = \mathrm{Ldim}(\mathcal{H})$ 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. □

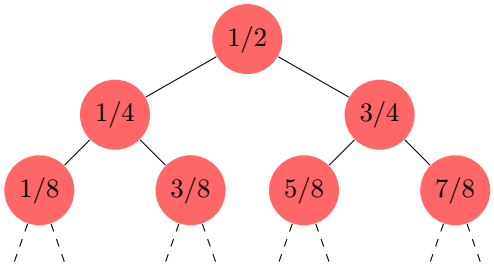
Let us now give several examples.

Example 21.2 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, $\mathrm{Ldim}(\mathcal{H}) \leq \log_2(|\mathcal{H}|)$. Another way to conclude this inequality is by combining Lemma 21.6 with Theorem 21.3.

Example 21.3 Let $\mathcal{X} = \{1, \ldots, d\}$ and $\mathcal{H} = \{h_1, \ldots, h_d\}$ where $h_j(x) = 1$ iff

$x = j$. Then, it is easy to show that $\text{Ldim}(\mathcal{H}) = 1$ while $|\mathcal{H}| = d$ can be arbitrarily large. Therefore, this example shows that $\text{Ldim}(\mathcal{H})$ can be significantly smaller than $\log_2(|\mathcal{H}|)$.

Example 21.4 Let $\mathcal{X} = [0,1]$ and $\mathcal{H} = \{x \mapsto \mathbb{1}_{[x<a]} : a \in [0,1]\}$; namely, $\mathcal{H}$ is the class of thresholds on the interval $[0,1]$. Then, $\text{Ldim}(\mathcal{H}) = \infty$. To see this, consider the tree



This tree is shattered by $\mathcal{H}$. And, because of the density of the reals, this tree can be made arbitrarily deep.

Lemma 21.6 states that $\text{Ldim}(\mathcal{H})$ 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 made 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` partitions 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 in the following 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 $p_t = \text{argmax}_{r \in \{0,1\}} \text{Ldim}(V_t^{(r)})$
    (in case of a tie predict $p_t = 1$)
  receive true label 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 preceding algorithm.

LEMMA 21.7 *SOA enjoys the mistake bound* $M_{\texttt{SOA}}(\mathcal{H}) \leq \text{Ldim}(\mathcal{H})$.

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

Combining Lemma 21.7 and Lemma 21.6 we obtain:

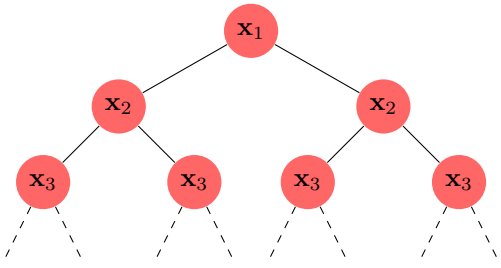
COROLLARY 21.8 *Let $\mathcal{H}$ be any hypothesis class. Then, the standard optimal algorithm enjoys the mistake bound $M_{\texttt{SOA}}(\mathcal{H}) = \text{Ldim}(\mathcal{H})$ and no other algorithm can have $M_A(\mathcal{H}) < \text{Ldim}(\mathcal{H})$.*

Comparison to VC Dimension

In the PAC learning model, learnability is characterized by the VC dimension of the class $\mathcal{H}$. Recall that the VC dimension of a class $\mathcal{H}$ 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 21.9 *For any class $\mathcal{H}$, $\text{VCdim}(\mathcal{H}) \leq \text{Ldim}(\mathcal{H})$, and there are classes for which strict inequality holds. Furthermore, the gap can be arbitrarily larger.*

Proof We first prove that $\text{VCdim}(\mathcal{H}) \leq \text{Ldim}(\mathcal{H})$. Suppose $\text{VCdim}(\mathcal{H}) = d$ 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 following illustration:



Now, the definition of a shattered set 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 21.4 has VC dimension of 1 whereas its Littlestone dimension is infinite. □

21.2 Online Classification in the Unrealizable Case

In the previous section we studied online learnability in the realizable case. We now consider the unrealizable case. Similarly to the agnostic PAC model, we no longer assume that all labels 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 $h \in \mathcal{H}$. Formally, the regret of an algorithm A relative to h when running on a sequence of T examples is defined as

$$\text{Regret}_A(h, T) = \sup_{(x_1,y_1),\ldots,(x_T,y_T)} \left[\sum_{t=1}^{T} |p_t - y_t| - \sum_{t=1}^{T} |h(x_t) - y_t| \right], \tag{21.1}$$

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

$$\text{Regret}_A(\mathcal{H}, T) = \sup_{h \in \mathcal{H}} \text{Regret}_A(h, T). \tag{21.2}$$

We restate the learner's goal as having the lowest possible regret relative to $\mathcal{H}$. An interesting question is whether we can derive an algorithm with low regret, meaning that $\text{Regret}_A(\mathcal{H}, T)$ grows sublinearly with the number of rounds, T, which implies that the difference between the *error rate* of the learner and the best hypothesis in $\mathcal{H}$ tends to zero as T goes to infinity.

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 be equal to T, by simply waiting for the learner's prediction and then providing the opposite label as the true label. In contrast, for any sequence of true labels, $y_1, \ldots, y_T$, 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 sublinear in T. This impossibility result is attributed to Cover (Cover 1965).

To sidestep Cover's impossibility result, we must further restrict the power of the adversarial environment. We do so by allowing the learner to randomize his predictions. Of course, this by itself does not circumvent Cover's impossibility result, since in deriving this result we assumed nothing about 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 coin flips of previous rounds, but it does not know the actual value of the random coin flips used by the learner on round t. With this (mild) change of game, we analyze the *expected* number of mistakes of the algorithm, where the 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 of the learner being in $\{0, 1\}$ we allow them to be in $[0, 1]$, and interpret $p_t \in [0, 1]$ as the probability to predict the label 1 on round t.

With this assumption it is possible to derive a low regret algorithm. In particular, we will prove the following theorem.

THEOREM 21.10 *For every hypothesis class $\mathcal{H}$, there exists an algorithm for online classification, whose predictions come from $[0, 1]$, that enjoys the regret bound*

$$\forall h \in \mathcal{H}, \quad \sum_{t=1}^{T} |p_t - y_t| - \sum_{t=1}^{T} |h(\mathbf{x}_t) - y_t| \leq \sqrt{2 \min\{\log(|\mathcal{H}|), \operatorname{Ldim}(\mathcal{H}) \log(eT)\}\, T}.$$

Furthermore, no algorithm can achieve an expected regret bound smaller than $\Omega\left(\sqrt{\operatorname{Ldim}(\mathcal{H})\, T}\right)$.

We will provide a constructive proof of the upper bound part of the preceding theorem. The proof of the lower bound part can be found in (Ben-David, Pal, & Shalev-Shwartz 2009).

The proof of Theorem 21.10 relies on the *Weighted-Majority* algorithm for learning with expert advice. This algorithm is important by itself and we dedicate the next subsection to it.

21.2.1 Weighted-Majority

Weighted-majority is an algorithm for the problem of *prediction with expert advice*. In this online learning problem, on round t the learner has to choose the advice of d given experts. We also allow the learner to randomize his choice by defining a distribution over the d experts, that is, picking a vector $\mathbf{w}^{(t)} \in [0, 1]^d$, with $\sum_i w_i^{(t)} = 1$, and choosing the ith expert with probability $w_i^{(t)}$. After the learner chooses an expert, it receives a vector of costs, $\mathbf{v}_t \in [0, 1]^d$, where $v_{t,i}$ is the cost of following the advice of the ith expert. If the learner's predictions are randomized, then its loss is defined to be the averaged cost, namely, $\sum_i w_i^{(t)} v_{t,i} = \langle \mathbf{w}^{(t)}, \mathbf{v}_t \rangle$. The algorithm assumes that the number of rounds T is given. In Exercise 4 we show how to get rid of this dependence using the *doubling trick*.

`Weighted-Majority`

input: number of experts, d ; number of rounds, T
parameter: $\eta = \sqrt{2\,\log(d)/T}$
initialize: $\tilde{\mathbf{w}}^{(1)} = (1, \ldots, 1)$
for $t = 1, 2, \ldots$
  set $\mathbf{w}^{(t)} = \tilde{\mathbf{w}}^{(t)}/Z_t$ where $Z_t = \sum_i \tilde{w}_i^{(t)}$
  choose expert i at random according to $\mathbb{P}[i] = w_i^{(t)}$
  receive costs of all experts $\mathbf{v}_t \in [0,1]^d$
  pay cost $\langle \mathbf{w}^{(t)}, \mathbf{v}_t \rangle$
  update rule $\forall i,\ \tilde{w}_i^{(t+1)} = \tilde{w}_i^{(t)} e^{-\eta v_{t,i}}$

The following theorem is key for analyzing the regret bound of Weighted-Majority.

THEOREM 21.11 *Assuming that $T > 2\log(d)$, the* `Weighted-Majority` *algorithm enjoys the bound*

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

Proof We have:

$$\log \frac{Z_{t+1}}{Z_t} = \log \sum_i \frac{\tilde{w}_i^{(t)}}{Z_t}\, e^{-\eta v_{t,i}} = \log \sum_i w_i^{(t)}\, e^{-\eta v_{t,i}}.$$

Using the inequality $e^{-a} \leq 1 - a + a^2/2$, which holds for all $a \in (0,1)$, and the fact that $\sum_i w_i^{(t)} = 1$, we obtain

$$\begin{aligned} \log \frac{Z_{t+1}}{Z_t} &\leq \log \sum_i w_i^{(t)} \left(1 - \eta v_{t,i} + \eta^2 v_{t,i}^2/2\right) \\ &= \log\big(1 - \underbrace{\sum_i w_i^{(t)} \left(\eta v_{t,i} - \eta^2 v_{t,i}^2/2\right)}_{\stackrel{\text{def}}{=} b}\big). \end{aligned}$$

Next, note that $b \in (0,1)$. Therefore, taking log of the two sides of the inequality $1 - b \leq e^{-b}$ we obtain the inequality $\log(1-b) \leq -b$, which holds for all $b \leq 1$, and obtain

$$\begin{aligned} \log \frac{Z_{t+1}}{Z_t} &\leq -\sum_i w_i^{(t)} \left(\eta v_{t,i} - \eta^2 v_{t,i}^2/2\right) \\ &= -\eta \left\langle \mathbf{w}^{(t)}, \mathbf{v}_t \right\rangle + \eta^2 \sum_i w_i^{(t)} v_{t,i}^2/2 \\ &\leq -\eta \left\langle \mathbf{w}^{(t)}, \mathbf{v}_t \right\rangle + \eta^2/2. \end{aligned}$$

Summing this inequality over t we get

$$\log(Z_{T+1}) - \log(Z_1) = \sum_{t=1}^{T} \log \frac{Z_{t+1}}{Z_t} \leq -\eta \sum_{t=1}^{T} \langle \mathbf{w}^{(t)}, \mathbf{v}_t \rangle + \frac{T\eta^2}{2}. \tag{21.3}$$

Next, we lower bound Z_{T+1}. For each i, we can rewrite $\tilde{w}_i^{(T+1)} = e^{-\eta \sum_t v_{t,i}}$ and we get that

$$\log Z_{T+1} = \log \left(\sum_i e^{-\eta \sum_t v_{t,i}} \right) \geq \log \left(\max_i e^{-\eta \sum_t v_{t,i}} \right) = -\eta \min_i \sum_t v_{t,i}.$$

Combining the preceding with Equation (21.3) and using the fact that $\log(Z_1) = \log(d)$ we get that

$$-\eta \min_i \sum_t v_{t,i} - \log(d) \;\leq\; -\eta \sum_{t=1}^{T} \langle \mathbf{w}^{(t)}, \mathbf{v}_t \rangle + \frac{T\eta^2}{2},$$

which can be rearranged as follows:

$$\sum_{t=1}^{T} \langle \mathbf{w}^{(t)}, \mathbf{v}_t \rangle - \min_i \sum_t v_{t,i} \;\leq\; \frac{\log(d)}{\eta} + \frac{\eta T}{2}.$$

Plugging the value of η into the equation concludes our proof. □

Proof of Theorem 21.10

Equipped with the `Weighted-Majority` algorithm and Theorem 21.11, we are ready to prove Theorem 21.10. We start with the simpler case, in which $\mathcal{H}$ is a finite class, and let us write $\mathcal{H} = \{h_1, \ldots, h_d\}$. In this case, we can refer to each hypothesis, h_i, as an expert, whose advice is to predict $h_i(\mathbf{x}_t)$, and whose cost is $v_{t,i} = |h_i(\mathbf{x}_t) - y_t|$. The prediction of the algorithm will therefore be $p_t = \sum_i w_i^{(t)} h_i(\mathbf{x}_t) \in [0,1]$, and the loss is

$$|p_t - y_t| = \left| \sum_{i=1}^{d} w_i^{(t)} h_i(\mathbf{x}_t) - y_t \right| = \left| \sum_{i=1}^{d} w_i^{(t)} (h_i(\mathbf{x}_t) - y_t) \right|.$$

Now, if $y_t = 1$, then for all i, $h_i(\mathbf{x}_t) - y_t \leq 0$. Therefore, the above equals to $\sum_i w_i^{(t)} |h_i(\mathbf{x}_t) - y_t|$. If $y_t = 0$ then for all i, $h_i(\mathbf{x}_t) - y_t \geq 0$, and the above also equals $\sum_i w_i^{(t)} |h_i(\mathbf{x}_t) - y_t|$. All in all, we have shown that

$$|p_t - y_t| = \sum_{i=1}^{d} w_i^{(t)} |h_i(\mathbf{x}_t) - y_t| = \langle \mathbf{w}^{(t)}, \mathbf{v}_t \rangle.$$

Furthermore, for each i, $\sum_t v_{t,i}$ is exactly the number of mistakes hypothesis h_i makes. Applying Theorem 21.11 we obtain

COROLLARY 21.12 *Let $\mathcal{H}$ be a finite hypothesis class. There exists an algorithm for online classification, whose predictions come from $[0,1]$, that enjoys the regret bound*

$$\sum_{t=1}^{T} |p_t - y_t| - \min_{h \in \mathcal{H}} \sum_{t=1}^{T} |h(\mathbf{x}_t) - y_t| \;\leq\; \sqrt{2\,\log(|\mathcal{H}|)\,T}.$$

Next, we consider the case of a general hypothesis class. Previously, we constructed an expert for each individual hypothesis. However, if $\mathcal{H}$ is infinite this leads to a vacuous bound. 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 and, 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 $L \leq \text{Ldim}(\mathcal{H})$ 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 $\tilde{y}_t = \text{argmax}_r \text{Ldim}\left(V_t^{(r)}\right)$
    (in case of a tie set $\tilde{y}_t = 0$)
  if $t \in \{i_1, i_2, \ldots, i_L\}$
      predict $\hat{y}_t = 1 - \tilde{y}_t$
  else
      predict $\hat{y}_t = \tilde{y}_t$
  update $V_{t+1} = V_t^{(\hat{y}_t)}$

Note that each such expert can give us predictions at every round t while only observing the instances $\mathbf{x}_1, \ldots, \mathbf{x}_t$. Our generic online learning algorithm is now an application of the `Weighted-Majority` algorithm with these experts.

To analyze the algorithm we first note that the number of experts is

$$d = \sum_{L=0}^{\text{Ldim}(\mathcal{H})} \binom{T}{L}. \tag{21.4}$$

It can be shown that when $T \geq \text{Ldim}(\mathcal{H}) + 2$, the right-hand side of the equation is bounded by $(eT/\text{Ldim}(\mathcal{H}))^{\text{Ldim}(\mathcal{H})}$ (the proof can be found in Lemma A.5).

Theorem 21.11 tells us that the expected number of mistakes of `Weighted-Majority` is at most the number of mistakes of the best expert plus $\sqrt{2\log(d)\,T}$. We will next show that the number of mistakes of the best expert is at most the number of mistakes of the best hypothesis in $\mathcal{H}$. 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 21.13 *Let $\mathcal{H}$ be any hypothesis class with $\text{Ldim}(\mathcal{H}) < \infty$. Let $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T$ be any sequence of instances. For any $h \in \mathcal{H}$, there exists $L \leq \text{Ldim}(\mathcal{H})$ and indices $1 \leq i_1 < i_2 < \cdots < i_L \leq T$ such that when running $\text{Expert}(i_1, i_2, \ldots, i_L)$ 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 $\text{Ldim}(\mathcal{H})$ 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 $\text{Expert}(i_1, i_2, \ldots, i_L)$ running on the sequence $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T$. By construction, the set V_t maintained by $\text{Expert}(i_1, i_2, \ldots, i_L)$ equals the set V_t maintained by SOA when running on the sequence $(\mathbf{x}_1, h(\mathbf{x}_1)), \ldots, (\mathbf{x}_T, h(\mathbf{x}_T))$. The predictions of SOA differ from the predictions of h if and only if the round is in $\{i_1, i_2, \ldots, i_L\}$. Since $\text{Expert}(i_1, i_2, \ldots, i_L)$ predicts exactly like SOA if t is not in $\{i_1, i_2, \ldots, i_L\}$ and the opposite of SOAs' predictions if t is in $\{i_1, i_2, \ldots, i_L\}$, we conclude that the predictions of the expert are always the same as the predictions of h. □

The previous 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 21.14 *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 $\text{Ldim}(\mathcal{H}) < \infty$. There exists $L \leq \text{Ldim}(\mathcal{H})$ and indices $1 \leq i_1 < i_2 < \cdots < i_L \leq T$, such that $\text{Expert}(i_1, i_2, \ldots, i_L)$ 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.

Together with Theorem 21.11, the upper bound part of Theorem 21.10 is proven.

21.3 Online Convex Optimization

In Chapter 12 we studied convex learning problems and showed learnability results for these problems in the agnostic PAC learning framework. In this section we show that similar learnability results hold for convex problems in the online learning framework. In particular, we consider the following problem.

Online Convex Optimization

definitions:
hypothesis class $\mathcal{H}$; domain Z ; loss function $\ell : \mathcal{H} \times Z \to \mathbb{R}$
assumptions:
$\mathcal{H}$ is convex
$\forall z \in Z$, $\ell(\cdot, z)$ is a convex function
for $t = 1, 2, \ldots, T$
learner predicts a vector $\mathbf{w}^{(t)} \in \mathcal{H}$
environment responds with $z_t \in Z$
learner suffers loss $\ell(\mathbf{w}^{(t)}, z_t)$

As in the online classification problem, we analyze the *regret* of the algorithm. Recall that the regret of an online algorithm with respect to a competing hypothesis, which here will be some vector $\mathbf{w}^\star \in \mathcal{H}$, is defined as

$$\text{Regret}_A(\mathbf{w}^\star, T) = \sum_{t=1}^{T} \ell(\mathbf{w}^{(t)}, z_t) - \sum_{t=1}^{T} \ell(\mathbf{w}^\star, z_t). \tag{21.5}$$

As before, the regret of the algorithm relative to a set of competing vectors, $\mathcal{H}$, is defined as

$$\text{Regret}_A(\mathcal{H}, T) = \sup_{\mathbf{w}^\star \in \mathcal{H}} \text{Regret}_A(\mathbf{w}^\star, T).$$

In Chapter 14 we have shown that Stochastic Gradient Descent solves convex learning problems in the agnostic PAC model. We now show that a very similar algorithm, Online Gradient Descent, solves online convex learning problems.

Online Gradient Descent

parameter: $\eta > 0$
initialize: $\mathbf{w}^{(1)} = \mathbf{0}$
for $t = 1, 2, \ldots, T$
predict $\mathbf{w}^{(t)}$
receive z_t and let $f_t(\cdot) = \ell(\cdot, z_t)$
choose $\mathbf{v}_t \in \partial f_t(\mathbf{w}^{(t)})$
update:
1. $\mathbf{w}^{(t+\frac{1}{2})} = \mathbf{w}^{(t)} - \eta \mathbf{v}_t$
2. $\mathbf{w}^{(t+1)} = \text{argmin}_{\mathbf{w} \in \mathcal{H}} \|\mathbf{w} - \mathbf{w}^{(t+\frac{1}{2})}\|$

THEOREM 21.15 *The Online Gradient Descent algorithm enjoys the following regret bound for every* $\mathbf{w}^\star \in \mathcal{H}$*,*

$$\text{Regret}_A(\mathbf{w}^\star, T) \leq \frac{\|\mathbf{w}^\star\|^2}{2\eta} + \frac{\eta}{2}\sum_{t=1}^{T}\|\mathbf{v}_t\|^2.$$

If we further assume that f_t *is* ρ*-Lipschitz for all* t*, then setting* $\eta = 1/\sqrt{T}$ *yields*

$$\text{Regret}_A(\mathbf{w}^\star, T) \leq \frac{1}{2}(\|\mathbf{w}^\star\|^2 + \rho^2)\sqrt{T}.$$

If we further assume that $\mathcal{H}$ *is* B*-bounded and we set* $\eta = \frac{B}{\rho\sqrt{T}}$ *then*

$$\text{Regret}_A(\mathcal{H}, T) \leq B\,\rho\,\sqrt{T}.$$

Proof The analysis is similar to the analysis of Stochastic Gradient Descent with projections. Using the projection lemma, the definition of $\mathbf{w}^{(t+\frac{1}{2})}$, and the definition of subgradients, we have that for every t,

$$\begin{aligned}
&\|\mathbf{w}^{(t+1)} - \mathbf{w}^\star\|^2 - \|\mathbf{w}^{(t)} - \mathbf{w}^\star\|^2 \\
&\quad = \|\mathbf{w}^{(t+1)} - \mathbf{w}^\star\|^2 - \|\mathbf{w}^{(t+\frac{1}{2})} - \mathbf{w}^\star\|^2 + \|\mathbf{w}^{(t+\frac{1}{2})} - \mathbf{w}^\star\|^2 - \|\mathbf{w}^{(t)} - \mathbf{w}^\star\|^2 \\
&\quad \leq \|\mathbf{w}^{(t+\frac{1}{2})} - \mathbf{w}^\star\|^2 - \|\mathbf{w}^{(t)} - \mathbf{w}^\star\|^2 \\
&\quad = \|\mathbf{w}^{(t)} - \eta\mathbf{v}_t - \mathbf{w}^\star\|^2 - \|\mathbf{w}^{(t)} - \mathbf{w}^\star\|^2 \\
&\quad = -2\eta\langle \mathbf{w}^{(t)} - \mathbf{w}^\star, \mathbf{v}_t\rangle + \eta^2\|\mathbf{v}_t\|^2 \\
&\quad \leq -2\eta(f_t(\mathbf{w}^{(t)}) - f_t(\mathbf{w}^\star)) + \eta^2\|\mathbf{v}_t\|^2.
\end{aligned}$$

Summing over t and observing that the left-hand side is a telescopic sum we obtain that

$$\|\mathbf{w}^{(T+1)} - \mathbf{w}^\star\|^2 - \|\mathbf{w}^{(1)} - \mathbf{w}^\star\|^2 \leq -2\eta\sum_{t=1}^{T}(f_t(\mathbf{w}^{(t)}) - f_t(\mathbf{w}^\star)) + \eta^2\sum_{t=1}^{T}\|\mathbf{v}_t\|^2.$$

Rearranging the inequality and using the fact that $\mathbf{w}^{(1)} = \mathbf{0}$, we get that

$$\begin{aligned}
\sum_{t=1}^{T}(f_t(\mathbf{w}^{(t)}) - f_t(\mathbf{w}^\star)) &\leq \frac{\|\mathbf{w}^{(1)} - \mathbf{w}^\star\|^2 - \|\mathbf{w}^{(T+1)} - \mathbf{w}^\star\|^2}{2\eta} + \frac{\eta}{2}\sum_{t=1}^{T}\|\mathbf{v}_t\|^2 \\
&\leq \frac{\|\mathbf{w}^\star\|^2}{2\eta} + \frac{\eta}{2}\sum_{t=1}^{T}\|\mathbf{v}_t\|^2.
\end{aligned}$$

This proves the first bound in the theorem. The second bound follows from the assumption that f_t is ρ-Lipschitz, which implies that $\|\mathbf{v}_t\| \leq \rho$. □

21.4 The Online Perceptron Algorithm

The Perceptron is a classic online learning algorithm for binary classification with the hypothesis class of homogenous halfspaces, namely, $\mathcal{H} = \{\mathbf{x} \mapsto \text{sign}(\langle \mathbf{w}, \mathbf{x}\rangle) :$

$\mathbf{w} \in \mathbb{R}^d\}$. In Section 9.1.2 we have presented the batch version of the Perceptron, which aims to solve the ERM problem with respect to $\mathcal{H}$. We now present an online version of the Perceptron algorithm.

Let $\mathcal{X} = \mathbb{R}^d$, $\mathcal{Y} = \{-1, 1\}$. 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 Section 21.1 we characterized the optimal algorithm and showed that the best achievable mistake bound depends on the Littlestone dimension of the class. We show later that if $d \geq 2$ then $\text{Ldim}(\mathcal{H}) = \infty$, which implies that we have no hope of making few prediction mistakes. Indeed, consider the tree for which $\mathbf{v}_1 = (\frac{1}{2}, 1, 0, \ldots, 0)$, $\mathbf{v}_2 = (\frac{1}{4}, 1, 0, \ldots, 0)$, $\mathbf{v}_3 = (\frac{3}{4}, 1, 0, \ldots, 0)$, etc. Because of 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 algorithm relies on the technique of *surrogate convex losses* (see Section 12.3). This is also closely related to the notion of *margin* we studied in Chapter 15.

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 y. Therefore, we can write the $0-1$ loss function as follows

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

On rounds on which the algorithm makes a prediction mistake, we shall use the hinge-loss as a surrogate convex loss function

$$f_t(\mathbf{w}) = \max\{0, 1 - y_t \langle \mathbf{w}, \mathbf{x}_t \rangle\}.$$

The hinge-loss 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, $f_t(\mathbf{w}^{(t)}) = \ell(\mathbf{w}^{(t)}, (\mathbf{x}_t, y_t)) = 0$.

Remark 21.5 In Section 12.3 we used the same surrogate loss function for all the examples. In the online model, we allow the surrogate to depend on the specific round. It can even depend on $\mathbf{w}^{(t)}$. Our ability to use a round specific surrogate stems from the worst-case type of analysis we employ in online learning.

Let us now run the `Online Gradient Descent` algorithm on the sequence of functions, $f_1, \ldots, f_T$, with the hypothesis class being all vectors in $\mathbb{R}^d$ (hence, the projection step is vacuous). Recall that the algorithm initializes $\mathbf{w}^{(1)} = \mathbf{0}$ and its update rule is

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \mathbf{v}_t$$

for some $\mathbf{v}_t \in \partial f_t(\mathbf{w}^{(t)})$. 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 $\mathbf{v}_t = \mathbf{0}$. Otherwise, it is easy to verify that $\mathbf{v}_t = -y_t \mathbf{x}_t$ 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

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

This 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 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 Online Gradient Descent given in the previous section. In our case, the subgradient of f_t we use in the Perceptron is $\mathbf{v}_t = -\mathbb{1}_{[y_t \langle \mathbf{w}^{(t)}, \mathbf{x}_t \rangle \leq 0]} \, y_t \, \mathbf{x}_t$. Indeed, the Perceptron's update is $\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \mathbf{v}_t$, and as discussed before this is equivalent to $\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \mathbf{v}_t$ for every $\eta > 0$. Therefore, Theorem 21.15 tells us that

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

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

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

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

$$|\mathcal{M}| - R \|\mathbf{w}^\star\| \sqrt{|\mathcal{M}|} - \sum_{t=1}^{T} f_t(\mathbf{w}^\star) \leq 0. \tag{21.6}$$

This inequality implies

THEOREM 21.16 *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* $f_t(\mathbf{w}) = \mathbb{1}_{[t \in \mathcal{M}]} [1 - y_t \langle \mathbf{w}, \mathbf{x}_t \rangle]_+$. *Then, for every* $\mathbf{w}^\star$

$$|\mathcal{M}| \leq \sum_t f_t(\mathbf{w}^\star) + R \|\mathbf{w}^\star\| \sqrt{\sum_t f_t(\mathbf{w}^\star)} + R^2 \|\mathbf{w}^\star\|^2 \ .$$

In particular, if there exists $\mathbf{w}^\star$ *such that* $y_t \langle \mathbf{w}^\star, \mathbf{x}_t \rangle \geq 1$ *for all* t *then*

$$|\mathcal{M}| \leq R^2 \|\mathbf{w}^\star\|^2 .$$

Proof The theorem follows from Equation (21.6) and the following claim: Given $x, b, c \in \mathbb{R}_+$, the inequality $x - b\sqrt{x} - c \leq 0$ implies that $x \leq c + b^2 + b\sqrt{c}$. 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 21.16 is called *separability with large margin* (see Chapter 15). That is, there exists $\mathbf{w}^\star$ 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 $\gamma = 1/\|\mathbf{w}^\star\|$ and the bound becomes $(R/\gamma)^2$.

When the separability assumption does not hold, the bound involves the term $[1 - y_t \langle \mathbf{w}^\star, \mathbf{x}_t \rangle]_+$ 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 $\mathbf{w}^\star$ that makes zero errors on the sequence but the Perceptron will make many errors. Indeed, this is a direct consequence of the fact that $\text{Ldim}(\mathcal{H}) = \infty$. The way we sidestep this impossibility result is by assuming more on the sequence of examples – the bound in Theorem 21.16 will be meaningful only if the cumulative surrogate loss, $\sum_t f_t(\mathbf{w}^\star)$ is not excessively large.

21.5 Summary

In this chapter we have studied the online learning model. Many of the results we derived for the PAC learning model have an analog in the online model. First, we have shown that a combinatorial dimension, the Littlestone dimension, characterizes online learnability. To show this, we introduced the SOA algorithm (for the realizable case) and the Weighted-Majority algorithm (for the unrealizable case). We have also studied online convex optimization and have shown that online gradient descent is a successful online learner whenever the loss function is convex and Lipschitz. Finally, we presented the online Perceptron algorithm as a combination of online gradient descent and the concept of surrogate convex loss functions.

21.6 Bibliographic Remarks

The Standard Optimal Algorithm was derived by the seminal work of Littlestone (1988). A generalization to the nonrealizable case, as well as other variants like margin-based Littlestone's dimension, were derived in (Ben-David et al. 2009). Characterizations of online learnability beyond classification have been obtained in (Abernethy, Bartlett, Rakhlin & Tewari 2008, Rakhlin, Sridharan & Tewari 2010, Daniely et al. 2011). The Weighted-Majority algorithm is due to (Littlestone & Warmuth 1994) and (Vovk 1990).

The term "online convex programming" was introduced by Zinkevich (2003) but this setting was introduced some years earlier by Gordon (1999). The Perceptron dates back to Rosenblatt (Rosenblatt 1958). An analysis for the realizable case (with margin assumptions) appears in (Agmon 1954, Minsky & Papert 1969). Freund and Schapire (Freund & Schapire 1999) 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 (Gentile 2003).

For additional information we refer the reader to Cesa-Bianchi & Lugosi (2006) and Shalev-Shwartz (2011).

21.7 Exercises

1. Find a hypothesis class $\mathcal{H}$ and a sequence of examples on which `Consistent` makes $|\mathcal{H}| - 1$ mistakes.
2. Find a hypothesis class $\mathcal{H}$ and a sequence of examples on which the mistake bound of the *Halving* algorithm is tight.
3. Let $d \geq 2$, $\mathcal{X} = \{1, \ldots, d\}$ and let $\mathcal{H} = \{h_j : j \in [d]\}$, where $h_j(x) = \mathbb{1}_{[x=j]}$. Calculate $M_{\text{Halving}}(\mathcal{H})$ (i.e., derive lower and upper bounds on $M_{\text{Halving}}(\mathcal{H})$, and prove that they are equal).
4. **The Doubling Trick:**

   In Theorem 21.15, the parameter η depends on the time horizon T. In this exercise we 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 in the following, 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 $t = 2^m, \ldots, 2^{m+1} - 1$

Show that if the regret of A on each period of 2^m rounds is at most $\alpha\sqrt{2^m}$, then the total regret is at most

$$\frac{\sqrt{2}}{\sqrt{2}-1}\,\alpha\,\sqrt{T}.$$

5. **Online-to-batch Conversions:** In this exercise we demonstrate how a successful online learning algorithm can be used to derive a successful PAC learner as well.

   Consider a PAC learning problem for binary classification parameterized by an instance domain, $\mathcal{X}$, and a hypothesis class, $\mathcal{H}$. Suppose that there exists an online learning algorithm, A, which enjoys a mistake bound $M_A(\mathcal{H}) < \infty$. Consider running this algorithm on a sequence of T examples which are sampled i.i.d. from a distribution $\mathcal{D}$ over the instance space $\mathcal{X}$, and are labeled by some $h^\star \in \mathcal{H}$. Suppose that for every round t, the prediction of the algorithm is based on a hypothesis $h_t : \mathcal{X} \to \{0,1\}$. Show that

   $$\mathbb{E}[L_{\mathcal{D}}(h_r)] \leq \frac{M_A(\mathcal{H})}{T},$$

   where the expectation is over the random choice of the instances as well as a random choice of r according to the uniform distribution over $[T]$.
   Hint: Use similar arguments to the ones appearing in the proof of Theorem 14.8.