8 On-Line Learning

This chapter presents an introduction to on-line learning, an important area with a rich literature and multiple connections with game theory and optimization that is increasingly influencing the theoretical and algorithmic advances in machine learning. In addition to the intriguing novel learning theory questions that they raise, on-line learning algorithms are particularly attractive in modern applications since they provide an efficient solution for large-scale problems.

These algorithms process one sample at a time with an update per iteration that is often computationally cheap and easy to implement. As a result, they are typically significantly more efficient both in time and space and more practical than batch algorithms, when processing modern data sets of several million or billion points. They are also typically easy to implement. Moreover, on-line algorithms do not require any distributional assumption; their analysis assumes an adversarial scenario. This makes them applicable in a variety of scenarios where the sample points are not drawn i.i.d. or according to a fixed distribution.

We first introduce the general scenario of on-line learning, then present and analyze several key algorithms for on-line learning with expert advice, including the deterministic and randomized weighted majority algorithms for the zero-one loss and an extension of these algorithms for convex losses. We also describe and analyze two standard on-line algorithms for linear classification, the Perceptron and Winnow algorithms, as well as some extensions. While on-line learning algorithms are designed for an adversarial scenario, they can be used, under some assumptions, to derive accurate predictors for a distributional scenario. We derive learning guarantees for this on-line to batch conversion. Finally, we briefly point out the connection of on-line learning with game theory by describing its use to derive a simple proof of von Neumann's minimax theorem.

8.1 Introduction

The learning framework for on-line algorithms is in stark contrast to the PAC learning or stochastic models discussed up to this point. First, instead of learning from a training set and then testing on a test set, the on-line learning scenario mixes the training and test phases. Second, PAC learning follows the key assumption that the distribution over data points is fixed over time, both for training and test points, and that points are sampled in an i.i.d. fashion. Under this assumption, the natural goal is to learn a hypothesis with a small expected loss or generalization error. In contrast, with on-line learning, *no distributional assumption* is made, and thus there is no notion of generalization. Instead, the performance of on-line learning algorithms is measured using a *mistake model* and the notion of *regret*. To derive guarantees in this model, theoretical analyses are based on a worst-case or adversarial assumption.

The general on-line setting involves T rounds. At the tth round, the algorithm receives an instance $x_t \in \mathcal{X}$ and makes a prediction $\widehat{y}_t \in \mathcal{Y}$. It then receives the true label $y_t \in \mathcal{Y}$ and incurs a loss $L(\widehat{y}_t, y_t)$, where $L\colon \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$ is a loss function. More generally, the prediction domain for the algorithm may be $\mathcal{Y}' \neq \mathcal{Y}$ and the loss function defined over $\mathcal{Y}' \times \mathcal{Y}$. For classification problems, we often have $\mathcal{Y} = \{0, 1\}$ and $L(y, y') = |y' - y|$, while for regression $\mathcal{Y} \subseteq \mathbb{R}$ and typically $L(y, y') = (y' - y)^2$. The objective in the on-line setting is to minimize the cumulative loss: $\sum_{t=1}^{T} L(\widehat{y}_t, y_t)$ over T rounds.

8.2 Prediction with expert advice

We first discuss the setting of online learning with expert advice, and the associated notion of *regret*. In this setting, at the tth round, in addition to receiving $x_t \in \mathcal{X}$, the algorithm also receives *advice* $y_{t,i} \in \mathcal{Y}$, $i \in [N]$, from N experts. Following the general framework of on-line algorithms, it then makes a prediction, receives the true label, and incurs a loss. After T rounds, the algorithm has incurred a cumulative loss. The objective in this setting is to minimize the *regret* R_T, also called *external regret*, which compares the cumulative loss of the algorithm to that of the best expert in hindsight after T rounds:

$$R_T = \sum_{t=1}^{T} L(\widehat{y}_t, y_t) - \min_{i=1}^{N} \sum_{t=1}^{T} L(\widehat{y}_{t,i}, y_t). \tag{8.1}$$

Figure 8.1
Weather forecast: an example of a prediction problem based on expert advice.

This problem arises in a variety of different domains and applications. Figure 8.1 illustrates the problem of predicting the weather using several forecasting sources as experts.

8.2.1 Mistake bounds and Halving algorithm

Here, we assume that the loss function is the standard zero-one loss used in classification. To analyze the expert advice setting, we first consider the realizable case, that is the setting where at least one of the experts makes no errors. As such, we discuss the *mistake bound model*, which asks the simple question "How many mistakes before we learn a particular concept?" Since we are in the realizable case, after some number of rounds T, we will learn the concept and no longer make errors in subsequent rounds. For any fixed concept c, we define the maximum number of mistakes a learning algorithm $\mathcal{A}$ makes as

$$M_{\mathcal{A}}(c) = \max_{x_1,\ldots,x_T} |\text{mistakes}(\mathcal{A}, c)|. \tag{8.2}$$

Further, for any concept in a concept class $\mathcal{C}$, the maximum number of mistakes a learning algorithm makes is

$$M_{\mathcal{A}}(\mathcal{C}) = \max_{c \in \mathcal{C}} M_{\mathcal{A}}(c). \tag{8.3}$$

Our goal in this setting is to derive *mistake bounds*, that is, a bound M on $M_{\mathcal{A}}(\mathcal{C})$. We will first do this for the Halving algorithm, an elegant and simple algorithm for which we can guarantee surprisingly favorable mistake bounds. At each round, the Halving algorithm makes its prediction by taking the majority vote over all *active* experts. After any incorrect prediction, it deactivates all experts that gave faulty advice. Initially, all experts are active, and by the time the algorithm has converged to the correct concept, the active set contains only those experts that are consistent with the target concept. The pseudocode for this algorithm is shown in figure 8.2. We also present straightforward mistake bounds in theorems 8.1 and 8.2, where the former deals with finite hypothesis sets and the latter relates mistake bounds to VC-dimension. Note that the hypothesis complexity term in theorem 8.1 is identical to the corresponding complexity term in the PAC model bound of theorem 2.5.

HALVING($\mathcal{H}$)

$\mathcal{H}_1 \leftarrow \mathcal{H}$
for $t \leftarrow 1$ **to** T **do**
  RECEIVE(x_t)
  $\widehat{y}_t \leftarrow$ MAJORITYVOTE($\mathcal{H}_t, x_t$)
  RECEIVE(y_t)
  if $(\widehat{y}_t \neq y_t)$ **then**
    $\mathcal{H}_{t+1} \leftarrow \{c \in \mathcal{H}_t : c(x_t) = y_t\}$
  else $\mathcal{H}_{t+1} \leftarrow \mathcal{H}_t$
return $\mathcal{H}_{T+1}$

Figure 8.2
Halving algorithm.

Theorem 8.1 *Let $\mathcal{H}$ be a finite hypothesis set. Then*

$$M_{\text{HALVING}}(\mathcal{H}) \leq \log_2 |\mathcal{H}|. \tag{8.4}$$

Proof: Since at each round the algorithm makes predictions using majority vote from the active set, at each mistake, the active set is reduced by at least half. Hence, after $\log_2 |\mathcal{H}|$ mistakes, there can only remain one active hypothesis, and since we are in the realizable case, this hypothesis must coincide with the target concept. □

Theorem 8.2 *Let $opt(\mathcal{H})$ be the optimal mistake bound for $\mathcal{H}$. Then,*

$$\text{VCdim}(\mathcal{H}) \leq opt(\mathcal{H}) \leq M_{\text{HALVING}}(\mathcal{H}) \leq \log_2 |\mathcal{H}|. \tag{8.5}$$

Proof: The second inequality is true by definition and the third inequality holds based on theorem 8.1. To prove the first inequality, we let $d = \text{VCdim}(\mathcal{H})$. Then there exists a shattered set of d points, for which we can form a complete binary tree of the mistakes with height d, and we can choose labels at each round of learning to ensure that d mistakes are made. Note that this adversarial argument is valid since the on-line setting makes no statistical assumptions about the data. □

```
WEIGHTED-MAJORITY(N)
 1  for i ← 1 to N do
 2      w_{1,i} ← 1
 3  for t ← 1 to T do
 4      RECEIVE(x_t)
 5      if Σ_{i: y_{t,i}=1} w_{t,i} ≥ Σ_{i: y_{t,i}=0} w_{t,i} then
 6          ŷ_t ← 1
 7      else ŷ_t ← 0
 8      RECEIVE(y_t)
 9      if (ŷ_t ≠ y_t) then
10          for i ← 1 to N do
11              if (y_{t,i} ≠ y_t) then
12                  w_{t+1,i} ← β w_{t,i}
13              else w_{t+1,i} ← w_{t,i}
14  return w_{T+1}
```

Figure 8.3
Weighted majority algorithm, $y_t, y_{t,i} \in \{0, 1\}$.

8.2.2 Weighted majority algorithm

In the previous section, we focused on the realizable setting in which the Halving algorithm simply discarded experts after a single mistake. We now move to the non-realizable setting and use a more general and less extreme algorithm, the Weighted Majority (WM) algorithm, that *weights* the importance of experts as a function of their mistake rate. The WM algorithm begins with uniform weights over all N experts. At each round, it generates predictions using a weighted majority vote. After receiving the true label, the algorithm then reduces the weight of each incorrect expert by a factor of $\beta \in [0, 1)$. Note that this algorithm reduces to the Halving algorithm when $\beta = 0$. The pseudocode for the WM algorithm is shown in figure 8.3.

Since we are not in the realizable setting, the mistake bounds of theorem 8.1 cannot apply. However, the following theorem presents a bound on the number of mistakes m_T made by the WM algorithm after $T \geq 1$ rounds of on-line learning as a function of the number of mistakes made by the best expert, that is the expert

who achieves the smallest number of mistakes for the sequence $y_1, \ldots, y_T$. Let us emphasize that this is the best expert *in hindsight*.

Theorem 8.3 *Fix $\beta \in (0,1)$. Let m_T be the number of mistakes made by algorithm WM after $T \geq 1$ rounds, and m_T^* be the number of mistakes made by the best of the N experts. Then, the following inequality holds:*

$$m_T \leq \frac{\log N + m_T^* \log \frac{1}{\beta}}{\log \frac{2}{1+\beta}}. \tag{8.6}$$

Proof: To prove this theorem, we first introduce a potential function. We then derive upper and lower bounds for this function, and combine them to obtain our result. This potential function method is a general proof technique that we will use throughout this chapter.

For any $t \geq 1$, we define our potential function as $W_t = \sum_{i=1}^{N} w_{t,i}$. Since predictions are generated using weighted majority vote, if the algorithm makes an error at round t, this implies that

$$W_{t+1} \leq \big[1/2 + (1/2)\beta\big] W_t = \left[\frac{1+\beta}{2}\right] W_t. \tag{8.7}$$

Since $W_1 = N$ and m_T mistakes are made after T rounds, we thus have the following upper bound:

$$W_T \leq \left[\frac{1+\beta}{2}\right]^{m_T} N. \tag{8.8}$$

Next, since the weights are all non-negative, it is clear that for any expert i, $W_T \geq w_{T,i} = \beta^{m_{T,i}}$, where $m_{T,i}$ is the number of mistakes made by the ith expert after T rounds. Applying this lower bound to the best expert and combining it with the upper bound in (8.8) gives us:

$$\begin{aligned} \beta^{m_T^*} &\leq W_T \leq \left[\frac{1+\beta}{2}\right]^{m_T} N \\ \Rightarrow\ & m_T^* \log \beta \leq \log N + m_T \log \left[\frac{1+\beta}{2}\right] \\ \Rightarrow\ & m_T \log \left[\frac{2}{1+\beta}\right] \leq \log N + m_T^* \log \frac{1}{\beta}, \end{aligned}$$

which concludes the proof. □

Thus, the theorem guarantees a bound of the following form for algorithm WM:

$$m_T \leq O(\log N) + \text{constant} \times |\text{mistakes of best expert}|.$$

Since the first term varies only logarithmically as a function of N, the theorem guarantees that the number of mistakes is roughly a constant times that of the best expert in hindsight. This is a remarkable result, especially because it requires no

assumption about the sequence of points and labels generated. In particular, the sequence could be chosen adversarially. In the realizable case where $m_T^* = 0$, the bound reduces to $m_T \leq O(\log N)$ as for the Halving algorithm.

8.2.3 Randomized weighted majority algorithm

In spite of the guarantees just discussed, the WM algorithm admits a drawback that affects all deterministic algorithms in the case of the zero-one loss: no deterministic algorithm can achieve a regret $R_T = o(T)$ over all sequences. Clearly, for any deterministic algorithm $\mathcal{A}$ and any $t \in [T]$, we can adversarially select y_t to be 1 if the algorithm predicts 0, and choose it to be 0 otherwise. Thus, $\mathcal{A}$ errs at every point of such a sequence and its cumulative mistake is $m_T = T$. Assume for example that $N = 2$ and that one expert always predicts 0, the other one always 1. The error of the best expert over that sequence (and in fact any sequence of that length) is then at most $m_T^* \leq T/2$. Thus, for that sequence, we have

$$R_T = m_T - m_T^* \geq T/2,$$

which shows that $R_T = o(T)$ cannot be achieved in general. Note that this does not contradict the bound proven in the previous section, since for any $\beta \in (0, 1)$, $\frac{\log \frac{1}{\beta}}{\log \frac{2}{1+\beta}} \geq 2$. As we shall see in the next section, this negative result does not hold for any loss that is convex with respect to one of its arguments. But for the zero-one loss, this leads us to consider randomized algorithms instead.

In the randomized scenario of on-line learning, we assume that a set $\mathcal{A} = \{1, \ldots, N\}$ of N actions is available. At each round $t \in [T]$, an on-line algorithm $\mathcal{A}$ selects a distribution $\mathbf{p}_t$ over the set of actions, receives a loss vector $\mathbf{l}_t$, whose ith component $l_{t,i} \in [0, 1]$ is the loss associated with action i, and incurs the expected loss $L_t = \sum_{i=1}^N p_{t,i}\, l_{t,i}$. The total loss incurred by the algorithm over T rounds is $\mathcal{L}_T = \sum_{t=1}^T L_t$. The total loss associated to action i is $\mathcal{L}_{T,i} = \sum_{t=1}^T l_{t,i}$. The minimal loss of a single action is denoted by $\mathcal{L}_T^{\min} = \min_{i \in \mathcal{A}} \mathcal{L}_{T,i}$. The regret R_T of the algorithm after T rounds is then typically defined by the difference of the loss of the algorithm and that of the best single action:[11]

$$R_T = \mathcal{L}_T - \mathcal{L}_T^{\min}.$$

Here, we consider specifically the case of zero-one losses and assume that $l_{t,i} \in \{0, 1\}$ for all $t \in [T]$ and $i \in \mathcal{A}$.

[11] Alternative definitions of the regret with comparison classes different from the set of single actions can be considered.

```
RANDOMIZED-WEIGHTED-MAJORITY (N)
 1  for i ← 1 to N do
 2       w_{1,i} ← 1
 3       p_{1,i} ← 1/N
 4  for t ← 1 to T do
 5       RECEIVE(l_t)
 6       for i ← 1 to N do
 7            if (l_{t,i} = 1) then
 8                 w_{t+1,i} ← β w_{t,i}
 9            else w_{t+1,i} ← w_{t,i}
10       W_{t+1} ← Σ_{i=1}^{N} w_{t+1,i}
11       for i ← 1 to N do
12            p_{t+1,i} ← w_{t+1,i} / W_{t+1}
13  return w_{T+1}
```

Figure 8.4
Randomized weighted majority algorithm.

The WM algorithm admits a straightforward randomized version, the randomized weighted majority (RWM) algorithm. The pseudocode of this algorithm is given in figure 8.4. The algorithm updates the weight $w_{t,i}$ of expert i as in the case of the WM algorithm by multiplying it by β. The following theorem gives a strong guarantee on the regret R_T of the RWM algorithm, showing that it is in $O(\sqrt{T \log N})$.

Theorem 8.4 *Fix $\beta \in [1/2, 1)$. Then, for any $T \geq 1$, the loss of algorithm RWM on any sequence can be bounded as follows:*

$$\mathcal{L}_T \leq \frac{\log N}{1-\beta} + (2-\beta)\mathcal{L}_T^{\min}. \tag{8.9}$$

In particular, for $\beta = \max\{1/2, 1 - \sqrt{(\log N)/T}\}$, the loss can be bounded as:

$$\mathcal{L}_T \leq \mathcal{L}_T^{\min} + 2\sqrt{T \log N}. \tag{8.10}$$

Proof: As in the proof of theorem 8.3, we derive upper and lower bounds for the potential function $W_t = \sum_{i=1}^{N} w_{t,i}$, $t \in [T]$, and combine these bounds to obtain

the result. By definition of the algorithm, for any $t \in [T]$, W_{t+1} can be expressed as follows in terms of W_t:

$$\begin{aligned} W_{t+1} = \sum_{i:\, l_{t,i}=0} w_{t,i} + \beta \sum_{i:\, l_{t.i}=1} w_{t,i} &= W_t + (\beta - 1) \sum_{i:\, l_{t,i}=1} w_{t,i} \\ &= W_t + (\beta - 1) W_t \sum_{i:\, l_{t,i}=1} p_{t,i} \\ &= W_t + (\beta - 1) W_t L_t \\ &= W_t (1 - (1 - \beta) L_t). \end{aligned}$$

Thus, since $W_1 = N$, it follows that $W_{T+1} = N \prod_{t=1}^{T} (1 - (1 - \beta) L_t)$. On the other hand, the following lower bound clearly holds: $W_{T+1} \geq \max_{i \in [N]} w_{T+1,i} = \beta^{\mathcal{L}_T^{\min}}$. This leads to the following inequality and series of derivations after taking the log and using the inequalities $\log(1 - x) \leq -x$ valid for all $x < 1$, and $-\log(1 - x) \leq x + x^2$ valid for all $x \in [0, 1/2]$:

$$\begin{aligned} \beta^{\mathcal{L}_T^{\min}} \leq N \prod_{t=1}^{T} (1 - (1 - \beta) L_t) &\implies \mathcal{L}_T^{\min} \log \beta \leq \log N + \sum_{t=1}^{T} \log(1 - (1 - \beta) L_t) \\ &\implies \mathcal{L}_T^{\min} \log \beta \leq \log N - (1 - \beta) \sum_{t=1}^{T} L_t \\ &\implies \mathcal{L}_T^{\min} \log \beta \leq \log N - (1 - \beta) \mathcal{L}_T \\ &\implies \mathcal{L}_T \leq \frac{\log N}{1 - \beta} - \frac{\log \beta}{1 - \beta} \mathcal{L}_T^{\min} \\ &\implies \mathcal{L}_T \leq \frac{\log N}{1 - \beta} - \frac{\log(1 - (1 - \beta))}{1 - \beta} \mathcal{L}_T^{\min} \\ &\implies \mathcal{L}_T \leq \frac{\log N}{1 - \beta} + (2 - \beta) \mathcal{L}_T^{\min}. \end{aligned}$$

This shows the first statement. Since $\mathcal{L}_T^{\min} \leq T$, this also implies

$$\mathcal{L}_T \leq \frac{\log N}{1 - \beta} + (1 - \beta) T + \mathcal{L}_T^{\min}. \tag{8.11}$$

Differentiating the upper bound with respect to β and setting it to zero gives $\frac{\log N}{(1-\beta)^2} - T = 0$, that is $\beta = 1 - \sqrt{(\log N)/T} < 1$. Thus, if $1 - \sqrt{(\log N)/T} \geq 1/2$, $\beta_0 = 1 - \sqrt{(\log N)/T}$ is the minimizing value of β, otherwise the boundary value $\beta_0 = 1/2$ is the optimal value. The second statement follows by replacing β with β_0 in (8.11). □

The bound (8.10) assumes that the algorithm additionally receives as a parameter the number of rounds T. As we shall see in the next section, however, there exists a general *doubling trick* that can be used to relax this requirement at the price of a

small constant factor increase. Inequality 8.10 can be written directly in terms of the regret R_T of the RWM algorithm:

$$R_T \leq 2\sqrt{T \log N}. \tag{8.12}$$

Thus, for N constant, the regret verifies $R_T = O(\sqrt{T})$ and the *average regret* or *regret per round* R_T/T decreases as $O(1/\sqrt{T})$. These results are optimal, as shown by the following theorem.

Theorem 8.5 *Let $N = 2$. There exists a stochastic sequence of losses for which the regret of any on-line learning algorithm verifies $\mathbb{E}[R_T] \geq \sqrt{T/8}$.*

Proof: For any $t \in [T]$, let the vector of losses $\mathbf{l}_t$ take the values $\mathbf{l}_{01} = (0,1)^\top$ and $\mathbf{l}_{10} = (1,0)^\top$ with equal probability. Then, the expected loss of any randomized algorithm $\mathcal{A}$ is

$$\mathbb{E}[\mathcal{L}_T] = \mathbb{E}\Big[\sum_{t=1}^{T} \mathbf{p}_t \cdot \mathbf{l}_t\Big] = \sum_{t=1}^{T} \mathbf{p}_t \cdot \mathbb{E}[\mathbf{l}_t] = \sum_{t=1}^{T} \frac{1}{2} p_{t,1} + \frac{1}{2}(1 - p_{t,1}) = T/2,$$

where we denoted by $\mathbf{p}_t$ the distribution selected by $\mathcal{A}$ at round t. By definition, $\mathcal{L}_T^{\min}$ can be written as follows:

$$\mathcal{L}_T^{\min} = \min\{\mathcal{L}_{T,1}, \mathcal{L}_{T,2}\} = \frac{1}{2}(\mathcal{L}_{T,1} + \mathcal{L}_{T,2} - |\mathcal{L}_{T,1} - \mathcal{L}_{T,2}|) = T/2 - |\mathcal{L}_{T,1} - T/2|,$$

using the fact that $\mathcal{L}_{T,1} + \mathcal{L}_{T,2} = T$. Thus, the expected regret of $\mathcal{A}$ is

$$\mathbb{E}[R_T] = \mathbb{E}[\mathcal{L}_T] - \mathbb{E}[\mathcal{L}_T^{\min}] = \mathbb{E}[|\mathcal{L}_{T,1} - T/2|].$$

Let σ_t, $t \in [T]$, denote Rademacher variables taking values in $\{-1, +1\}$, then $\mathcal{L}_{T,1}$ can be rewritten as $\mathcal{L}_{T,1} = \sum_{t=1}^{T} \frac{1+\sigma_t}{2} = T/2 + \frac{1}{2}\sum_{t=1}^{T} \sigma_t$. Thus, introducing scalars $x_t = 1/2$, $t \in [T]$, by the Khintchine-Kahane inequality, (D.24) we have:

$$\mathbb{E}[R_T] = \mathbb{E}\Big[|\sum_{t=1}^{T} \sigma_t x_t|\Big] \geq \sqrt{\frac{1}{2}\sum_{t=1}^{T} x_t^2} = \sqrt{T/8},$$

which concludes the proof. □

More generally, for $T \geq N$, a lower bound of $R_T = \Omega(\sqrt{T \log N})$ can be proven for the regret of any algorithm.

8.2.4 Exponential weighted average algorithm

The WM algorithm can be extended to other loss functions L taking values in $[0,1]$. The Exponential Weighted Average algorithm presented here can be viewed as that extension for the case where L is convex in its first argument. Note that this algorithm is deterministic and yet, as we shall see, admits a very favorable regret

EXPONENTIAL-WEIGHTED-AVERAGE (N)

for $i \leftarrow 1$ **to** N **do**
    $w_{1,i} \leftarrow 1$
for $t \leftarrow 1$ **to** T **do**
    RECEIVE(x_t)
    $\widehat{y}_t \leftarrow \frac{\sum_{i=1}^N w_{t,i} y_{t,i}}{\sum_{i=1}^N w_{t,i}}$
    RECEIVE(y_t)
    for $i \leftarrow 1$ **to** N **do**
        $w_{t+1,i} \leftarrow w_{t,i}\, e^{-\eta L(\widehat{y}_{t,i}, y_t)}$
return $\mathbf{w}_{T+1}$

Figure 8.5
Exponential weighted average, $L(\widehat{y}_{t,i}, y_t) \in [0,1]$.

guarantee. Figure 8.5 gives its pseudocode. At round $t \in [T]$, the algorithm's prediction is

$$\widehat{y}_t = \frac{\sum_{i=1}^N w_{t,i} y_{t,i}}{\sum_{i=1}^N w_{t,i}}, \tag{8.13}$$

where $y_{t,i}$ is the prediction by expert i and $w_{t,i}$ the weight assigned by the algorithm to that expert. Initially, all weights are set to one. The algorithm then updates the weights at the end of round t according to the following rule:

$$w_{t+1,i} \leftarrow w_{t,i}\, e^{-\eta L(\widehat{y}_{t,i}, y_t)} = e^{-\eta L_{t,i}}, \tag{8.14}$$

where $L_{t,i}$ is the total loss incurred by expert i after t rounds. Note that this algorithm, as well as the others presented in this chapter, are simple, since they do not require keeping track of the losses incurred by each expert at all previous rounds but only of their cumulative performance. Furthermore, this property is also computationally advantageous. The following theorem presents a regret bound for this algorithm.

Theorem 8.6 *Assume that the loss function L is convex in its first argument and takes values in $[0,1]$. Then, for any $\eta > 0$ and any sequence $y_1, \ldots, y_T \in \mathcal{Y}$, the regret of the Exponential Weighted Average algorithm after T rounds satisfies*

$$R_T \leq \frac{\log N}{\eta} + \frac{\eta T}{8}. \tag{8.15}$$

In particular, for $\eta = \sqrt{8 \log N / T}$*, the regret is bounded as*

$$R_T \leq \sqrt{(T/2) \log N}. \tag{8.16}$$

Proof: We apply the same potential function analysis as in previous proofs but using as potential $\Phi_t = \log \sum_{i=1}^N w_{t,i}$, $t \in [T]$. Let $\mathbf{p}_t$ denote the distribution over $\{1, \ldots, N\}$ with $p_{t,i} = \frac{w_{t,i}}{\sum_{i=1}^N w_{t,i}}$. To derive an upper bound on Φ_t, we first examine the difference of two consecutive potential values:

$$\Phi_{t+1} - \Phi_t = \log \frac{\sum_{i=1}^N w_{t,i}\, e^{-\eta L(\widehat{y}_{t,i}, y_t)}}{\sum_{i=1}^N w_{t,i}} = \log \big(\mathop{\mathbb{E}}_{\mathbf{p}_t} [e^{\eta X}] \big),$$

with $X = -L(\widehat{y}_{t,i}, y_t) \in [-1, 0]$. To upper bound the expression appearing in the right-hand side, we apply Hoeffding's lemma (lemma D.1) to the centered random variable $X - \mathbb{E}_{\mathbf{p}_t}[X]$, then Jensen's inequality (theorem B.20) using the convexity of L with respect to its first argument:

$$\begin{aligned}
\Phi_{t+1} - \Phi_t &= \log \big(\mathop{\mathbb{E}}_{\mathbf{p}_t} \big[e^{\eta(X - \mathbb{E}[X]) + \eta\, \mathbb{E}[X]} \big] \big) \\
&\leq \frac{\eta^2}{8} + \eta \mathop{\mathbb{E}}_{\mathbf{p}_t}[X] = \frac{\eta^2}{8} - \eta \mathop{\mathbb{E}}_{\mathbf{p}_t}[L(\widehat{y}_{t,i}, y_t)] && \text{(Hoeffding's lemma)} \\
&\leq -\eta L\big(\mathop{\mathbb{E}}_{\mathbf{p}_t}[\widehat{y}_{t,i}], y_t \big) + \frac{\eta^2}{8} && \text{(convexity of first arg. of } L\text{)} \\
&= -\eta L(\widehat{y}_t, y_t) + \frac{\eta^2}{8}.
\end{aligned}$$

Summing up these inequalities yields the following upper bound:

$$\Phi_{T+1} - \Phi_1 \leq -\eta \sum_{t=1}^T L(\widehat{y}_t, y_t) + \frac{\eta^2 T}{8}. \tag{8.17}$$

We obtain a lower bound for the same quantity as follows:

$$\Phi_{T+1} - \Phi_1 = \log \sum_{i=1}^N e^{-\eta L_{T,i}} - \log N \geq \log \max_{i=1}^N e^{-\eta L_{T,i}} - \log N = -\eta \min_{i=1}^N L_{T,i} - \log N.$$

Combining the upper and lower bounds yields:

$$\begin{aligned}
&-\eta \min_{i=1}^N L_{T,i} - \log N \leq -\eta \sum_{t=1}^T L(\widehat{y}_t, y_t) + \frac{\eta^2 T}{8} \\
\implies &\sum_{t=1}^T L(\widehat{y}_t, y_t) - \min_{i=1}^N L_{T,i} \leq \frac{\log N}{\eta} + \frac{\eta T}{8},
\end{aligned}$$

and concludes the proof. □

The optimal choice of η in theorem 8.6 requires knowledge of the horizon T, which is an apparent disadvantage of this analysis. However, we can use a standard *doubling trick* to eliminate this requirement, at the price of a small constant factor. This consists of dividing time into periods $[2^k, 2^{k+1}-1]$ of length 2^k with $k = 0, \ldots, n$ and $T \geq 2^n - 1$, and then choosing $\eta_k = \sqrt{\frac{8 \log N}{2^k}}$ in each period. The following theorem presents a regret bound when using the doubling trick to select η. A more general method consists of interpreting η as a function of time, i.e., $\eta_t = \sqrt{(8 \log N)/t}$, which can lead to a further constant factor improvement over the regret bound of the following theorem.

Theorem 8.7 *Assume that the loss function L is convex in its first argument and takes values in $[0, 1]$. Then, for any $T \geq 1$ and any sequence $y_1, \ldots, y_T \in \mathcal{Y}$, the regret of the Exponential Weighted Average algorithm after T rounds is bounded as follows:*

$$R_T \leq \frac{\sqrt{2}}{\sqrt{2}-1} \sqrt{(T/2) \log N} + \sqrt{\log N/2}. \tag{8.18}$$

Proof: Let $T \geq 1$ and let $\mathcal{I}_k = [2^k, 2^{k+1} - 1]$, for $k \in [0, n]$, with $n = \lfloor \log(T+1) \rfloor$. Let $L_{\mathcal{I}_k}$ denote the loss incurred in the interval $\mathcal{I}_k$. By theorem 8.6 (8.16), for any $k \in \{0, \ldots, n\}$, we have

$$L_{\mathcal{I}_k} - \min_{i=1}^{N} L_{\mathcal{I}_k, i} \leq \sqrt{2^k/2 \, \log N}. \tag{8.19}$$

Thus, we can bound the total loss incurred by the algorithm after T rounds as:

$$\begin{aligned} L_T = \sum_{k=0}^{n} L_{\mathcal{I}_k} &\leq \sum_{k=0}^{n} \min_{i=1}^{N} L_{\mathcal{I}_k, i} + \sum_{k=0}^{n} \sqrt{2^k \, (\log N)/2} \\ &\leq \min_{i=1}^{N} L_{T,i} + \sqrt{(\log N)/2} \cdot \sum_{k=0}^{n} 2^{\frac{k}{2}}, \end{aligned} \tag{8.20}$$

where the second inequality follows from the super-additivity of min, that is $\min_i X_i + \min_i Y_i \leq \min_i (X_i + Y_i)$ for any sequences $(X_i)_i$ and $(Y_i)_i$, which implies $\sum_{k=0}^{n} \min_{i=1}^{N} L_{\mathcal{I}_k, i} \leq \min_{i=1}^{N} \sum_{k=0}^{n} L_{\mathcal{I}_k, i}$. The geometric sum appearing in the right-hand side of (8.20) can be expressed as follows:

$$\sum_{k=0}^{n} 2^{\frac{k}{2}} = \frac{2^{(n+1)/2} - 1}{\sqrt{2} - 1} \leq \frac{\sqrt{2}\sqrt{T+1} - 1}{\sqrt{2} - 1} \leq \frac{\sqrt{2}(\sqrt{T} + 1) - 1}{\sqrt{2} - 1} = \frac{\sqrt{2}\sqrt{T}}{\sqrt{2} - 1} + 1.$$

Plugging back into (8.20) and rearranging terms yields (8.18). □

The $O(\sqrt{T})$ dependency on T presented in this bound cannot be improved for general loss functions.

```
PERCEPTRON(w_0)
1  w_1 ← w_0        ▷ typically w_0 = 0
2  for t ← 1 to T do
3      RECEIVE(x_t)
4      ŷ_t ← sgn(w_t · x_t)
5      RECEIVE(y_t)
6      if (ŷ_t ≠ y_t) then
7          w_{t+1} ← w_t + y_t x_t    ▷ more generally η y_t x_t, η > 0.
8      else w_{t+1} ← w_t
9  return w_{T+1}
```

Figure 8.6
Perceptron algorithm.

8.3 Linear classification

This section presents two well-known on-line learning algorithms for linear classification: the Perceptron and Winnow algorithms.

8.3.1 Perceptron algorithm

The Perceptron algorithm is one of the earliest machine learning algorithms. It is an on-line linear classification algorithm. Thus, it learns a decision function based on a hyperplane by processing training points one at a time. Figure 8.6 gives its pseudocode.

The algorithm maintains a weight vector $\mathbf{w}_t \in \mathbb{R}^N$ defining the hyperplane learned, starting with an arbitrary vector $\mathbf{w}_0$. At each round $t \in [T]$, it predicts the label of the point $\mathbf{x}_t \in \mathbb{R}^N$ received, using the current vector $\mathbf{w}_t$ (line 4). When the prediction made does not match the correct label (lines 6-7), it updates $\mathbf{w}_t$ by adding $y_t\mathbf{x}_t$. More generally, when a learning rate $\eta > 0$ is used, the vector added is $\eta y_t\mathbf{x}_t$. This update can be partially motivated by examining the inner product of the current weight vector with $y_t\mathbf{x}_t$, whose sign determines the classification of $\mathbf{x}_t$. Just before an update, $\mathbf{x}_t$ is misclassified and thus $y_t\mathbf{w}_t \cdot \mathbf{x}_t$ is negative; afterward, $y_t\mathbf{w}_{t+1} \cdot \mathbf{x}_t = y_t\mathbf{w}_t \cdot \mathbf{x}_t + \eta\|\mathbf{x}_t\|^2$, thus, the update corrects the weight vector in the direction of making the inner product $y_t\mathbf{w}_t \cdot \mathbf{x}_t$ positive by augmenting it with the quantity $\eta\|\mathbf{x}_t\|^2 > 0$.

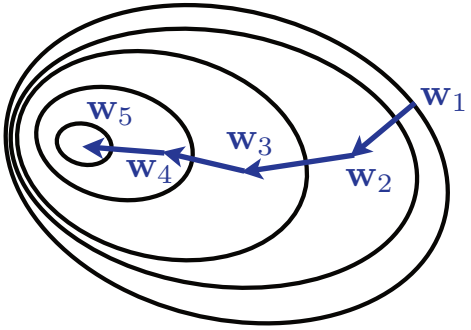

Figure 8.7
An example path followed by the iterative stochastic gradient descent technique. Each inner contour indicates a region of lower elevation.

The Perceptron algorithm can be shown in fact to seek a weight vector $\mathbf{w}$ minimizing an objective function F precisely based on the quantities $(-y_t\mathbf{w}\cdot\mathbf{x}_t)$, $t\in[T]$. Since $(-y_t\mathbf{w}\cdot\mathbf{x}_t)$ is positive when $\mathbf{x}_t$ is misclassified by $\mathbf{w}$, F is defined for all $\mathbf{w}\in\mathbb{R}^N$ by

$$F(\mathbf{w})=\frac{1}{T}\sum_{t=1}^{T}\max\Big(0,-y_t(\mathbf{w}\cdot\mathbf{x}_t)\Big)=\mathop{\mathbb{E}}_{\mathbf{x}\sim\widehat{\mathcal{D}}}[\widetilde{F}(\mathbf{w},\mathbf{x})],\tag{8.21}$$

where $\widetilde{F}(\mathbf{w},\mathbf{x})=\max\big(0,-f(\mathbf{x})(\mathbf{w}\cdot\mathbf{x})\big)$ with $f(\mathbf{x})$ denoting the label of $\mathbf{x}$, and $\widehat{\mathcal{D}}$ is the empirical distribution associated with the sample $(\mathbf{x}_1,\ldots,\mathbf{x}_T)$. For any $t\in[T]$, $\mathbf{w}\mapsto-y_t(\mathbf{w}\cdot\mathbf{x}_t)$ is linear and thus convex. Since the max operator preserves convexity, this shows that F is convex. However, F is not differentiable. Nevertheless, the Perceptron algorithm coincides with the application of the *stochastic subgradient descent* technique to F.

The stochastic (or on-line) subgradient descent technique examines one point $\mathbf{x}_t$ at a time. Note, the function $\widetilde{F}(\cdot,\mathbf{x}_t)$ is non-differentiable for any $\mathbf{w}_t$ where $\mathbf{w}_t\cdot\mathbf{x}_t=0$. In such a case any subgradient of $\widetilde{F}$, i.e. any vector in the convex hull of $\mathbf{0}$ and $-y_t\mathbf{x}_t$, may be used for the update step (see B.4.1). Choosing the subgradient $-y_t\mathbf{x}_t$, we arrive at the following general update for each point $\mathbf{x}_t$:

$$\mathbf{w}_{t+1}\leftarrow\begin{cases}\mathbf{w}_t-\eta\nabla_{\mathbf{w}}\widetilde{F}(\mathbf{w}_t,\mathbf{x}_t) & \text{if } \mathbf{w}_t\cdot\mathbf{x}_t\neq 0\\ \mathbf{w}_t+\eta y_t\mathbf{x}_t & \text{otherwise,}\end{cases}\tag{8.22}$$

where $\eta>0$ is a learning rate parameter. Figure 8.7 illustrates an example path the gradient descent follows. In the specific case we are considering, $\mathbf{w}\mapsto\widetilde{F}(\mathbf{w},\mathbf{x}_t)$ is differentiable at any $\mathbf{w}$ such that $y_t(\mathbf{w}\cdot\mathbf{x}_t)\neq 0$ with $\nabla_{\mathbf{w}}\widetilde{F}(\mathbf{w},\mathbf{x}_t)=-y\mathbf{x}_t$ if $y_t(\mathbf{w}\cdot\mathbf{x}_t)<0$ and $\nabla_{\mathbf{w}}\widetilde{F}(\mathbf{w},\mathbf{x}_t)=0$ if $y_t(\mathbf{w}\cdot\mathbf{x}_t)>0$. Thus, the stochastic gradient

descent update becomes

$$\mathbf{w}_{t+1} \leftarrow \begin{cases} \mathbf{w}_t + \eta y_t \mathbf{x}_t & \text{if } y_t(\mathbf{w}_t \cdot \mathbf{x}_t) \leq 0; \\ \mathbf{w}_t & \text{if } y_t(\mathbf{w}_t \cdot \mathbf{x}_t) > 0, \end{cases} \tag{8.23}$$

which coincides exactly with the update of the Perceptron algorithm.

The following theorem gives a margin-based upper bound on the number of mistakes or updates made by the Perceptron algorithm when processing a sequence of T points that can be linearly separated by a hyperplane with margin $\rho > 0$.

Theorem 8.8 *Let $\mathbf{x}_1, \ldots, \mathbf{x}_T \in \mathbb{R}^N$ be a sequence of T points with $\|\mathbf{x}_t\| \leq r$ for all $t \in [T]$, for some $r > 0$. Assume that there exist $\rho > 0$ and $\mathbf{v} \in \mathbb{R}^N$ such that for all $t \in [T]$, $\rho \leq \frac{y_t(\mathbf{v} \cdot \mathbf{x}_t)}{\|\mathbf{v}\|}$. Then, the number of updates made by the Perceptron algorithm when processing $\mathbf{x}_1, \ldots, \mathbf{x}_T$ is bounded by r^2/ρ^2.*

Proof: Let $\mathcal{I}$ be the subset of the T rounds at which there is an update, and let M be the total number of updates, i.e., $|\mathcal{I}| = M$. Summing up the assumption inequalities yields:

$$\begin{aligned} M\rho \leq \frac{\mathbf{v} \cdot \sum_{t \in \mathcal{I}} y_t \mathbf{x}_t}{\|\mathbf{v}\|} &\leq \Big\| \sum_{t \in \mathcal{I}} y_t \mathbf{x}_t \Big\| && \text{(Cauchy-Schwarz inequality)} \\ &= \Big\| \sum_{t \in \mathcal{I}} (\mathbf{w}_{t+1} - \mathbf{w}_t) \Big\| && \text{(definition of updates)} \\ &= \|\mathbf{w}_{T+1}\| && \text{(telescoping sum, } \mathbf{w}_0 = 0) \\ &= \sqrt{\sum_{t \in \mathcal{I}} \|\mathbf{w}_{t+1}\|^2 - \|\mathbf{w}_t\|^2} && \text{(telescoping sum, } \mathbf{w}_0 = 0) \\ &= \sqrt{\sum_{t \in \mathcal{I}} \|\mathbf{w}_t + y_t \mathbf{x}_t\|^2 - \|\mathbf{w}_t\|^2} && \text{(definition of updates)} \\ &= \sqrt{\sum_{t \in \mathcal{I}} 2 \underbrace{y_t \mathbf{w}_t \cdot \mathbf{x}_t}_{\leq 0} + \|\mathbf{x}_t\|^2} \\ &\leq \sqrt{\sum_{t \in \mathcal{I}} \|\mathbf{x}_t\|^2} \leq \sqrt{M r^2}. \end{aligned}$$

Comparing the left- and right-hand sides gives $\sqrt{M} \leq r/\rho$, that is, $M \leq r^2/\rho^2$. □

By definition of the algorithm, the weight vector $\mathbf{w}_T$ after processing T points is a linear combination of the vectors $\mathbf{x}_t$ at which an update was made: $\mathbf{w}_T = \sum_{t \in \mathcal{I}} y_t \mathbf{x}_t$. Thus, as in the case of SVMs, these vectors can be referred to as *support vectors* for the Perceptron algorithm.

The bound of theorem 8.8 is remarkable, since it depends only on the normalized margin ρ/r and not on the dimension N of the space. This bound can be shown to be tight, that is the number of updates can be equal to r^2/ρ^2 in some instances (see exercise 8.3 to show the upper bound is tight).

The theorem required no assumption about the sequence of points $\mathbf{x}_1, \ldots, \mathbf{x}_T$. A standard setting for the application of the Perceptron algorithm is one where a finite sample S of size $m < T$ is available and where the algorithm makes multiple passes over these m points. The result of the theorem implies that when S is linearly separable, the Perceptron algorithm converges after a finite number of updates and thus passes. For a small margin ρ, the convergence of the algorithm can be quite slow, however. In fact, for some samples, regardless of the order in which the points in S are processed, the number of updates made by the algorithm is in $\Omega(2^N)$ (see exercise 8.1). Of course, if S is not linearly separable, the Perceptron algorithm does not converge. In practice, it is stopped after some number of passes over S.

There are many variants of the standard Perceptron algorithm which are used in practice and have been theoretically analyzed. One notable example is the *voted Perceptron algorithm*, which predicts according to the rule $\operatorname{sgn}\left((\sum_{t \in \mathcal{I}} c_t \mathbf{w}_t) \cdot \mathbf{x}\right)$, where c_t is a weight proportional to the number of iterations that $\mathbf{w}_t$ survives, i.e., the number of iterations between $\mathbf{w}_t$ and $\mathbf{w}_{t+1}$.

For the following theorem, we consider the case where the Perceptron algorithm is trained via multiple passes till convergence over a finite sample that is linearly separable. In view of theorem 8.8, convergence occurs after a finite number of updates.

For a linearly separable sample S, we denote by r_S the radius of the smallest origin-centered sphere containing all points in S and by ρ_S the largest margin of a separating hyperplane for S. We also denote by $M(S)$ the number of updates made by the algorithm after training over S.

Theorem 8.9 *Assume that the data is linearly separable. Let h_S be the hypothesis returned by the Perceptron algorithm after training over a sample S of size m drawn according to some distribution $\mathcal{D}$. Then, the expected error of h_S is bounded as follows:*

$$\mathop{\mathbb{E}}_{S \sim \mathcal{D}^m}[R(h_S)] \leq \mathop{\mathbb{E}}_{S \sim \mathcal{D}^{m+1}}\left[\frac{\min\left(M(S), r_S^2/\rho_S^2\right)}{m+1}\right].$$

Proof: Let S be a linearly separable sample of size $m+1$ drawn i.i.d. according to $\mathcal{D}$ and let $\mathbf{x}$ be a point in S. If $h_{S-\{\mathbf{x}\}}$ misclassifies $\mathbf{x}$, then $\mathbf{x}$ must be a support vector for h_S. Thus, the leave-one-out error of the Perceptron algorithm on sample

S is at most $\frac{M(S)}{m+1}$. The result then follows lemma 5.3, which relates the expected leave-one-out error to the expected error, along with the upper bound on $M(S)$ given by theorem 8.8. □

This result can be compared with a similar one given for the SVM algorithm (with no offset) in the following theorem, which is an extension of theorem 5.4. We denote by $N_{\mathrm{SV}}(S)$ the number of support vectors that define the hypothesis h_S returned by SVMs when trained on a sample S.

Theorem 8.10 *Assume that the data is linearly separable. Let h_S be the hypothesis returned by SVMs used with no offset ($b = 0$) after training over a sample S of size m drawn according to some distribution $\mathcal{D}$. Then, the expected error of h_S is bounded as follows:*

$$\mathop{\mathbb{E}}_{S\sim\mathcal{D}^m}[R(h_S)] \leq \mathop{\mathbb{E}}_{S\sim\mathcal{D}^{m+1}}\left[\frac{\min\left(N_{SV}(S), r_S^2/\rho_S^2\right)}{m+1}\right].$$

Proof: The fact that the expected error can be upper bounded by the average fraction of support vectors ($N_{\mathrm{SV}}(S)/(m+1)$) was already shown by theorem 5.4. Thus, it suffices to show that it is also upper bounded by the expected value of $(r_S^2/\rho_S^2)/(m+1)$. To do so, we will bound the leave-one-out error of the SVM algorithm for a sample S of size $m+1$ by $(r_S^2/\rho_S^2)/(m+1)$. The result will then follow by lemma 5.3, which relates the expected leave-one-out error to the expected error.

Let $S = (\mathbf{x}_1, \ldots, \mathbf{x}_{m+1})$ be a linearly separable sample drawn i.i.d. according to $\mathcal{D}$ and let $\mathbf{x}$ be a point in S that is misclassified by $h_{S-\{\mathbf{x}\}}$. We will analyze the case where $\mathbf{x} = \mathbf{x}_{m+1}$, the analysis of other cases is similar. We denote by S' the sample $(\mathbf{x}_1, \ldots, \mathbf{x}_m)$.

For any $q \in [m+1]$, let G_q denote the function defined over $\mathbb{R}^q$ by $G_q \colon \boldsymbol{\alpha} \mapsto \sum_{i=1}^q \alpha_i - \frac{1}{2}\sum_{i,j=1}^q \alpha_i\alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j)$. Then, G_{m+1} is the objective function of the dual optimization problem for SVMs associated to the sample S and G_m the one for the sample S'. Let $\boldsymbol{\alpha} \in \mathbb{R}^{m+1}$ denote a solution of the dual SVM problem $\max_{\boldsymbol{\alpha}\geq 0} G_{m+1}(\boldsymbol{\alpha})$ and $\boldsymbol{\alpha}' \in \mathbb{R}^{m+1}$ the vector such that $(\alpha'_1, \ldots, \alpha'_m)^\top \in \mathbb{R}^m$ is a solution of $\max_{\boldsymbol{\alpha}\geq 0} G_m(\boldsymbol{\alpha})$ and $\alpha'_{m+1} = 0$. Let $\mathbf{e}_{m+1}$ denote the $(m+1)$th unit vector in $\mathbb{R}^{m+1}$. By definition of $\boldsymbol{\alpha}$ and $\boldsymbol{\alpha}'$ as maximizers, $\max_{\beta\geq 0} G_{m+1}(\boldsymbol{\alpha}' + \beta\mathbf{e}_{m+1}) \leq G_{m+1}(\boldsymbol{\alpha})$ and $G_{m+1}(\boldsymbol{\alpha} - \alpha_{m+1}\mathbf{e}_{m+1}) \leq G_m(\boldsymbol{\alpha}')$. Thus, the quantity $A = G_{m+1}(\boldsymbol{\alpha}) - G_m(\boldsymbol{\alpha}')$ admits the following lower and upper bounds:

$$\max_{\beta\geq 0} G_{m+1}(\boldsymbol{\alpha}' + \beta\mathbf{e}_{m+1}) - G_m(\boldsymbol{\alpha}') \leq A \leq G_{m+1}(\boldsymbol{\alpha}) - G_{m+1}(\boldsymbol{\alpha} - \alpha_{m+1}\mathbf{e}_{m+1}).$$

Let $\mathbf{w} = \sum_{i=1}^{m+1} y_i\alpha_i\mathbf{x}_i$ denote the weight vector returned by SVMs for the sample S. Since $h_{S'}$ misclassifies $\mathbf{x}_{m+1}$, $\mathbf{x}_{m+1}$ must be a support vector for h_S, thus

$y_{m+1}\mathbf{w}\cdot\mathbf{x}_{m+1}=1$. In view of that, the upper bound can be rewritten as follows:

$$\begin{aligned}
&G_{m+1}(\boldsymbol{\alpha})-G_{m+1}(\boldsymbol{\alpha}-\alpha_{m+1}\mathbf{e}_{m+1})\\
&\qquad=\alpha_{m+1}-\sum_{i=1}^{m+1}(y_i\alpha_i\mathbf{x}_i)\cdot(y_{m+1}\alpha_{m+1}\mathbf{x}_{m+1})+\frac{1}{2}\alpha_{m+1}^2\|\mathbf{x}_{m+1}\|^2\\
&\qquad=\alpha_{m+1}(1-y_{m+1}\mathbf{w}\cdot\mathbf{x}_{m+1})+\frac{1}{2}\alpha_{m+1}^2\|\mathbf{x}_{m+1}\|^2\\
&\qquad=\frac{1}{2}\alpha_{m+1}^2\|\mathbf{x}_{m+1}\|^2.
\end{aligned}$$

Similarly, let $\mathbf{w}'=\sum_{i=1}^{m}y_i\alpha_i'\mathbf{x}_i$. Then, for any $\beta\geq 0$, the quantity maximized in the lower bound can be written as

$$\begin{aligned}
&G_{m+1}(\boldsymbol{\alpha}'+\beta\mathbf{e}_{m+1})-G_m(\boldsymbol{\alpha}')\\
&\qquad=\beta\big(1-y_{m+1}(\mathbf{w}'+\beta y_{m+1}\mathbf{x}_{m+1})\cdot\mathbf{x}_{m+1}\big)+\frac{1}{2}\beta^2\|\mathbf{x}_{m+1}\|^2\\
&\qquad=\beta(1-y_{m+1}\mathbf{w}'\cdot\mathbf{x}_{m+1})-\frac{1}{2}\beta^2\|\mathbf{x}_{m+1}\|^2.
\end{aligned}$$

The right-hand side is maximized for the following value of β: $\frac{1-y_{m+1}\mathbf{w}'\cdot\mathbf{x}_{m+1}}{\|\mathbf{x}_{m+1}\|^2}$. Plugging in this value in the right-hand side gives $\frac{1}{2}\frac{(1-y_{m+1}\mathbf{w}'\cdot\mathbf{x}_{m+1})^2}{\|\mathbf{x}_{m+1}\|^2}$. Thus,

$$A\geq\frac{1}{2}\frac{(1-y_{m+1}\mathbf{w}'\cdot\mathbf{x}_{m+1})^2}{\|\mathbf{x}_{m+1}\|^2}\geq\frac{1}{2\|\mathbf{x}_{m+1}\|^2},$$

using the fact that $y_{m+1}\mathbf{w}'\cdot\mathbf{x}_{m+1}<0$, since $\mathbf{x}_{m+1}$ is misclassified by $\mathbf{w}'$. Comparing this lower bound on A with the upper bound previously derived leads to $\frac{1}{2\|\mathbf{x}_{m+1}\|^2}\leq\frac{1}{2}\alpha_{m+1}^2\|\mathbf{x}_{m+1}\|^2$, that is

$$\alpha_{m+1}\geq\frac{1}{\|\mathbf{x}_{m+1}\|^2}\geq\frac{1}{r_S^2}.$$

The analysis carried out in the case $\mathbf{x}=\mathbf{x}_{m+1}$ holds similarly for any $\mathbf{x}_i$ in S that is misclassified by $h_{S-\{\mathbf{x}_i\}}$. Let $\mathcal{I}$ denote the set of such indices i. Then, we can write:

$$\sum_{i\in\mathcal{I}}\alpha_i\geq\frac{|\mathcal{I}|}{r_S^2}.$$

By (5.19), the following simple expression holds for the margin: $\sum_{i=1}^{m+1}\alpha_i=1/\rho_S^2$. Using this identity leads to

$$|\mathcal{I}|\leq r_S^2\sum_{i\in\mathcal{I}}\alpha_i\leq r_S^2\sum_{i=1}^{m+1}\alpha_i=\frac{r_S^2}{\rho_S^2}.$$

Since by definition $|\mathcal{I}|$ is the total number of leave-one-out errors, this concludes the proof. □

Thus, the guarantees given by theorem 8.9 and theorem 8.10 in the separable case have a similar form. These bounds do not seem sufficient to distinguish the effectiveness of the SVM and Perceptron algorithms. Note, however, that while the same margin quantity ρ_S appears in both bounds, the radius r_S can be replaced by a finer quantity that is different for the two algorithms: in both cases, instead of the radius of the sphere containing all sample points, r_S can be replaced by the radius of the sphere containing the support vectors, as can be seen straightforwardly from the proof of the theorems. Thus, the position of the support vectors in the case of SVMs can provide a more favorable guarantee than that of the support vectors (update vectors) for the Perceptron algorithm. Finally, the guarantees given by these theorems are somewhat weak. These are not high probability bounds, they hold only for the expected error of the hypotheses returned by the algorithms and in particular provide no information about the variance of their error.

The following two theorems give bounds on the number of updates or mistakes made by the Perceptron algorithm in the more general scenario of a non-linearly separable sample in terms of the ρ-Hinge losses of an arbitrary weight vector $\mathbf{v}$.

Theorem 8.11 *Let $\mathcal{I}$ denote the set of indices $t \in [T]$ at which the Perceptron algorithm makes an update when processing a sequence $\mathbf{x}_1, \ldots, \mathbf{x}_T$ with $\|\mathbf{x}_t\| \leq r$ for some $r > 0$. Then, the number of updates $M = |\mathcal{I}|$ made by the algorithm can be bounded as follows:*

$$M \leq \inf_{\rho>0, \|\mathbf{v}\|_2 \leq 1} \left\lceil \frac{\frac{r}{\rho} + \sqrt{\frac{r^2}{\rho^2} + 4\|\mathbf{l}_\rho\|_1}}{2} \right\rceil^2 \leq \inf_{\rho>0, \|v\|_2 \leq 1} \left(\frac{r}{\rho} + \sqrt{\|\mathbf{l}_\rho\|_1} \right)^2 ,$$

where $\mathbf{l}_\rho = (l_t)_{t \in \mathcal{I}}$ with $l_t = \max\left\{0, 1 - \frac{y_t(\mathbf{v} \cdot \mathbf{x}_t)}{\rho}\right\}$.

Proof: Fix $\rho > 0$ and $\mathbf{v}$ with $\|\mathbf{v}\|_2 = 1$. By definition of l_t, for any t, we have $1 - \frac{y_t(\mathbf{v} \cdot \mathbf{x}_t)}{\rho} \leq l_t$. Summing up these inequalities over all $t \in \mathcal{I}$ yields

$$\begin{aligned} M &\leq \sum_{t \in \mathcal{I}} l_t + \sum_{t \in \mathcal{I}} \frac{y_t(\mathbf{v} \cdot \mathbf{x}_t)}{\rho} \\ &= \|\mathbf{l}_\rho\|_1 + \sum_{t \in \mathcal{I}} \frac{y_t(\mathbf{v} \cdot \mathbf{x}_t)}{\rho} \leq \|\mathbf{l}_\rho\|_1 + \frac{\sqrt{Mr^2}}{\rho} , \end{aligned} \tag{8.24}$$

where the last inequality holds by the bound shown in the proof of the separable case (theorem 8.8): $\frac{\mathbf{v} \cdot \sum_{t \in \mathcal{I}} y_t \mathbf{x}_t}{\|\mathbf{v}\|} \leq \sqrt{Mr^2}$. Now, solving the resulting second-degree inequality $M \leq \|\mathbf{l}_\rho\|_1 + \frac{\sqrt{Mr^2}}{\rho}$ gives $\sqrt{M} \leq \frac{1}{2}\left(\frac{r}{\rho} + \sqrt{\frac{r^2}{\rho^2} + 4\|\mathbf{l}_\rho\|_1}\right)$, which proves the first inequality. The second inequality follows from the sub-additivity of the square-root function. □

Theorem 8.12 *Let $\mathcal{I}$ denote the set of indices $t \in [T]$ at which the Perceptron algorithm makes an update when processing a sequence $\mathbf{x}_1, \ldots, \mathbf{x}_T$ with $\|\mathbf{x}_t\| \leq r$ for some $r > 0$. Then, the number of updates $M = |\mathcal{I}|$ made by the algorithm can be bounded as follows:*

$$M \leq \inf_{\rho > 0, \|v\|_2 \leq 1} \left(\frac{r}{\rho} + \sqrt{\|\mathbf{l}_\rho\|_2} \right)^2 ,$$

where $\mathbf{l}_\rho = (l_t)_{t \in \mathcal{I}}$ with $l_t = \max\left\{0, 1 - \frac{y_t(\mathbf{v} \cdot \mathbf{x}_t)}{\rho}\right\}$.

Proof: Fix $\rho > 0$ and $\mathbf{v}$ with $\|\mathbf{v}\|_2 = 1$. Starting with line (8.24) of theorem 8.11 and using $\|\mathbf{l}_\rho\|_1 \leq \sqrt{M}\|\mathbf{l}_\rho\|_2$, which holds by the Cauchy-Schwarz inequality, give

$$M \leq \|\mathbf{l}_\rho\|_1 + \frac{\sqrt{Mr^2}}{\rho} \leq \sqrt{M}\|\mathbf{l}_\rho\|_2 + \frac{\sqrt{Mr^2}}{\rho} .$$

This implies $\sqrt{M} \leq \|\mathbf{l}_\rho\|_2 + \frac{\sqrt{r^2}}{\rho}$ and proves the statement. □

These bounds strictly generalize the bounds given in the separable case (theorem 8.8) since in that case the vector $\mathbf{v}$ can be chosen to be that of a maximum-margin hyperplane with no Hinge loss at any point. The main difference between the two bounds is the L_1-norm of the vector of Hinge losses in Theorem 8.11 versus the L_2-norm in Theorem 8.12. Note that, since the L_2-norm bound follows from upper bounding inequality (8.24), which is equivalent to the first inequality of Theorem 8.11, the first L_1-norm bound of Theorem 8.11 is always tighter than the L_2-norm bound of Theorem 8.12.

The Perceptron algorithm can be generalized, as in the case of SVMs, to define a linear separation in a high-dimensional space. It admits an equivalent dual form, the dual Perceptron algorithm, which is presented in figure 8.8. The dual Perceptron algorithm maintains a vector $\boldsymbol{\alpha} \in \mathbb{R}^T$ of coefficients assigned to each point $\mathbf{x}_t$, $t \in [T]$. The label of a point $\mathbf{x}_t$ is predicted according to the rule $\operatorname{sgn}(\mathbf{w} \cdot \mathbf{x}_t)$, where $\mathbf{w} = \sum_{s=1}^T \alpha_s y_s \mathbf{x}_s$. The coefficient α_t is incremented by one when this prediction does not match the correct label. Thus, an update for $\mathbf{x}_t$ is equivalent to augmenting the weight vector $\mathbf{w}$ with $y_t \mathbf{x}_t$, which shows that the dual algorithm matches exactly the standard Perceptron algorithm. The dual Perceptron algorithm can be written solely in terms of inner products between training instances. Thus, as in the case of SVMs, instead of the inner product between points in the input space, an arbitrary PDS kernel can be used, which leads to the kernel Perceptron algorithm detailed in figure 8.9. The kernel Perceptron algorithm and its average variant, i.e., voted Perceptron with uniform weights c_t, are commonly used algorithms in a variety of applications.

```
DUALPERCEPTRON(α_0)
1  α ← α_0        ▷ typically α_0 = 0
2  for t ← 1 to T do
3      RECEIVE(x_t)
4      ŷ_t ← sgn(Σ_{s=1}^{T} α_s y_s (x_s · x_t))
5      RECEIVE(y_t)
6      if (ŷ_t ≠ y_t) then
7          α_t ← α_t + 1
8      else α_t ← α_t
9  return α
```

Figure 8.8
Dual Perceptron algorithm.

8.3.2 Winnow algorithm

This section presents an alternative on-line linear classification algorithm, the *Winnow algorithm*. Thus, it learns a weight vector defining a separating hyperplane by sequentially processing the training points. As suggested by the name, the algorithm is particularly well suited to cases where a relatively small number of dimensions or experts can be used to define an accurate weight vector. Many of the other dimensions may then be irrelevant.

The Winnow algorithm is similar to the Perceptron algorithm, but, instead of the additive update of the weight vector in the Perceptron case, Winnow's update is multiplicative. The pseudocode of the algorithm is given in figure 8.10. The algorithm takes as input a learning parameter $\eta > 0$. It maintains a non-negative weight vector $\mathbf{w}_t$ with components summing to one ($\|\mathbf{w}_t\|_1 = 1$) starting with the uniform weight vector (line 1). At each round $t \in [T]$, if the prediction does not match the correct label (line 6), each component $w_{t,i}$, $i \in [N]$, is updated by multiplying it by $\exp(\eta y_t x_{t,i})$ and dividing by the normalization factor Z_t to ensure that the weights sum to one (lines 7–9). Thus, if the label y_t and $x_{t,i}$ share the same sign, then $w_{t,i}$ is increased, while, in the opposite case, it is significantly decreased.

The Winnow algorithm is closely related to the WM algorithm: when $\mathbf{x}_{t,i} \in \{-1, +1\}$, $\text{sgn}(\mathbf{w}_t \cdot \mathbf{x}_t)$ coincides with the majority vote, since multiplying the weight of correct or incorrect experts by e^{η} or $e^{-\eta}$ is equivalent to multiplying the weight

KernelPerceptron($\boldsymbol{\alpha}_0$)

$\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha}_0$ ▷ typically $\boldsymbol{\alpha}_0 = \mathbf{0}$
for $t \leftarrow 1$ **to** T **do**
  Receive(x_t)
  $\widehat{y}_t \leftarrow \operatorname{sgn}(\sum_{s=1}^{T} \alpha_s y_s K(x_s, x_t))$
  Receive(y_t)
  if $(\widehat{y}_t \neq y_t)$ **then**
    $\alpha_t \leftarrow \alpha_t + 1$
  else $\alpha_t \leftarrow \alpha_t$
return $\boldsymbol{\alpha}$

Figure 8.9
Kernel Perceptron algorithm for PDS kernel K.

of incorrect ones by $\beta = e^{-2\eta}$. The multiplicative update rule of Winnow is of course also similar to that of AdaBoost.

The following theorem gives a mistake bound for the Winnow algorithm in the separable case, which is similar in form to the bound of theorem 8.8 for the Perceptron algorithm.

Theorem 8.13 *Let $\mathbf{x}_1, \ldots, \mathbf{x}_T \in \mathbb{R}^N$ be a sequence of T points with $\|x_t\|_\infty \leq r_\infty$ for all $t \in [T]$, for some $r_\infty > 0$. Assume that there exist $\mathbf{v} \in \mathbb{R}^N$, $\mathbf{v} \geq 0$, and $\rho_\infty > 0$ such that for all $t \in [T]$, $\rho_\infty \leq \frac{y_t(\mathbf{v} \cdot \mathbf{x}_t)}{\|\mathbf{v}\|_1}$. Then, for $\eta = \frac{\rho_\infty}{r_\infty^2}$, the number of updates made by the Winnow algorithm when processing $\mathbf{x}_1, \ldots, \mathbf{x}_T$ is upper bounded by $2\,(r_\infty^2/\rho_\infty^2) \log N$.*

Proof: Let $\mathcal{I} \subseteq [T]$ be the set of iterations at which there is an update, and let M be the total number of updates, i.e., $|\mathcal{I}| = M$. The potential function Φ_t, $t \in [T]$, used for this proof is the relative entropy of the distribution defined by the normalized weights $v_i/\|\mathbf{v}\|_1 \geq 0$, $i \in [N]$, and the one defined by the components of the weight vector $w_{t,i}$, $i \in [N]$:

$$\Phi_t = \sum_{i=1}^{N} \frac{v_i}{\|\mathbf{v}\|_1} \log \frac{v_i/\|\mathbf{v}\|_1}{w_{t,i}}\,.$$

To derive an upper bound on Φ_t, we analyze the difference of the potential functions at two consecutive rounds. For all $t \in \mathcal{I}$, this difference can be expressed and

WINNOW(η)

$w_1 \leftarrow \mathbf{1}/N$
for $t \leftarrow 1$ **to** T **do**
  RECEIVE($\mathbf{x}_t$)
  $\widehat{y}_t \leftarrow \text{sgn}(\mathbf{w}_t \cdot \mathbf{x}_t)$
  RECEIVE(y_t)
  if $(\widehat{y}_t \neq y_t)$ **then**
    $Z_t \leftarrow \sum_{i=1}^{N} w_{t,i} \exp(\eta y_t x_{t,i})$
    for $i \leftarrow 1$ **to** N **do**
      $w_{t+1,i} \leftarrow \frac{w_{t,i} \exp(\eta y_t x_{t,i})}{Z_t}$
  else $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t$
return $\mathbf{w}_{T+1}$

Figure 8.10
Winnow algorithm, with $y_t \in \{-1, +1\}$ for all $t \in [T]$.

bounded as follows:

$$\begin{aligned}
\Phi_{t+1} - \Phi_t &= \sum_{i=1}^{N} \frac{v_i}{\|\mathbf{v}\|_1} \log \frac{w_{t,i}}{w_{t+1,i}} \\
&= \sum_{i=1}^{N} \frac{v_i}{\|\mathbf{v}\|_1} \log \frac{Z_t}{\exp(\eta y_t x_{t,i})} \\
&= \log Z_t - \eta \sum_{i=1}^{N} \frac{v_i}{\|\mathbf{v}\|_1} y_t x_{t,i} \\
&\leq \log \Big[\sum_{i=1}^{N} w_{t,i} \exp(\eta y_t x_{t,i}) \Big] - \eta \rho_\infty \\
&= \log \mathop{\mathbb{E}}_{i \sim \mathbf{w}_t} \big[\exp(\eta y_t x_{t,i}) \big] - \eta \rho_\infty \\
&= \log \mathop{\mathbb{E}}_{i \sim \mathbf{w}_t} \big[\exp(\eta y_t x_{t,i} - \eta y_t \mathbf{w}_t \cdot x_t + \eta y_t \mathbf{w}_t \cdot x_t) \big] - \eta \rho_\infty \\
&\leq \log \big[\exp(\eta^2 (2r_\infty)^2 / 8) \big] + \underbrace{\eta y_t (\mathbf{w}_t \cdot x_t)}_{\leq 0} - \eta \rho_\infty \\
&\leq \eta^2 r_\infty^2 / 2 - \eta \rho_\infty.
\end{aligned}$$

The first inequality follows the definition of ρ_∞. The subsequent equality rewrites the summation as an expectation over the distribution defined by $\mathbf{w}_t$. The next inequality uses Hoeffding's lemma (lemma D.1) and the last one the fact that there has been an update at t, which implies $y_t(\mathbf{w}_t \cdot x_t) \leq 0$. Summing up these inequalities over all $t \in \mathcal{I}$ yields:

$$\Phi_{T+1} - \Phi_1 \leq M(\eta^2 r_\infty^2/2 - \eta\rho_\infty).$$

Next, we derive a lower bound by noting that

$$\Phi_1 = \sum_{i=1}^{N} \frac{v_i}{\|\mathbf{v}\|_1} \log \frac{v_i/\|\mathbf{v}\|_1}{1/N} = \log N + \sum_{i=1}^{N} \frac{v_i}{\|\mathbf{v}\|_1} \log \frac{v_i}{\|\mathbf{v}\|_1} \leq \log N\,.$$

Additionally, since the relative entropy is always non-negative, we have $\Phi_{T+1} \geq 0$. This yields the following lower bound:

$$\Phi_{T+1} - \Phi_1 \geq 0 - \log N = -\log N\,.$$

Combining the upper and lower bounds we see that $-\log N \leq M(\eta^2 r_\infty^2/2 - \eta\rho_\infty)$. Setting $\eta = \frac{\rho_\infty}{r_\infty^2}$ yields the statement of the theorem. □

The margin-based mistake bounds of theorem 8.8 and theorem 8.13 for the Perceptron and Winnow algorithms have a similar form, but they are based on different norms. For both algorithms, the norm $\|\cdot\|_p$ used for the input vectors $\mathbf{x}_t$, $t \in [T]$, is the dual of the norm $\|\cdot\|_q$ used for the margin vector $\mathbf{v}$, that is p and q are conjugate: $1/p + 1/q = 1$: in the case of the Perceptron algorithm $p = q = 2$, while for Winnow $p = \infty$ and $q = 1$.

These bounds imply different types of guarantees. The bound for Winnow is favorable when a sparse set of the experts $i \in [N]$ can predict well. For example, if $\mathbf{v} = \mathbf{e}_1$ where $\mathbf{e}_1$ is the unit vector along the first axis in $\mathbb{R}^N$ and if $\mathbf{x}_t \in \{-1, +1\}^N$ for all t, then the upper bound on the number of mistakes given for Winnow by theorem 8.13 is only $2 \log N$, while the upper bound of theorem 8.8 for the Perceptron algorithm is N. The guarantee for the Perceptron algorithm is more favorable in the opposite situation, where sparse solutions are not effective.

8.4 On-line to batch conversion

The previous sections presented several algorithms for the scenario of on-line learning, including the Perceptron and Winnow algorithms, and analyzed their behavior within the mistake model, where no assumption is made about the way the training sequence is generated. Can these algorithms be used to derive hypotheses with small generalization error in the standard stochastic setting? How can the interme-

diate hypotheses they generate be combined to form an accurate predictor? These are the questions addressed in this section.

Let $\mathcal{H}$ be a hypothesis of functions mapping $\mathcal{X}$ to $\mathcal{Y}'$, and let $L\colon \mathcal{Y}' \times \mathcal{Y} \to \mathbb{R}_+$ be a bounded loss function, that is $L \leq M$ for some $M \geq 0$. We assume a standard supervised learning setting where a labeled sample $S = ((x_1, y_1), \ldots, (x_T, y_T)) \in (\mathcal{X} \times \mathcal{Y})^T$ is drawn i.i.d. according to some fixed but unknown distribution $\mathcal{D}$. The sample is sequentially processed by an on-line learning algorithm $\mathcal{A}$. The algorithm starts with an initial hypothesis $h_1 \in \mathcal{H}$ and generates a new hypothesis $h_{t+1} \in \mathcal{H}$, after processing pair (x_t, y_t), $t \in [m]$. The regret of the algorithm is defined as before by

$$R_T = \sum_{t=1}^{T} L(h_t(x_t), y_t) - \min_{h \in \mathcal{H}} \sum_{t=1}^{T} L(h(x_t), y_t). \tag{8.25}$$

The generalization error of a hypothesis $h \in \mathcal{H}$ is its expected loss $R(h) = \mathbb{E}_{(x,y)\sim\mathcal{D}}[L(h(x), y)]$.

The following lemma gives a bound on the average of the generalization errors of the hypotheses generated by $\mathcal{A}$ in terms of its average loss $\frac{1}{T}\sum_{t=1}^{T} L(h_t(x_t), y_t)$.

Lemma 8.14 *Let $S = ((x_1, y_1), \ldots, (x_T, y_T)) \in (\mathcal{X} \times \mathcal{Y})^T$ be a labeled sample drawn i.i.d. according to $\mathcal{D}$, L a loss bounded by M and $h_1, \ldots, h_T$ the sequence of hypotheses generated by an on-line algorithm $\mathcal{A}$ sequentially processing S. Then, for any $\delta > 0$, with probability at least $1 - \delta$, the following holds:*

$$\frac{1}{T}\sum_{t=1}^{T} R(h_t) \leq \frac{1}{T}\sum_{t=1}^{T} L(h_t(x_t), y_t) + M\sqrt{\frac{2\log\frac{1}{\delta}}{T}}. \tag{8.26}$$

Proof: For any $t \in [T]$, let V_t be the random variable defined by $V_t = R(h_t) - L(h_t(x_t), y_t)$. Observe that for any $t \in [T]$,

$$\mathbb{E}[V_t | x_1, \ldots, x_{t-1}] = R(h_t) - \mathbb{E}[L(h_t(x_t), y_t) | h_t] = R(h_t) - R(h_t) = 0.$$

Since the loss is bounded by M, V_t takes values in the interval $[-M, +M]$ for all $t \in [T]$. Thus, by Azuma's inequality (theorem D.7), $\mathbb{P}[\frac{1}{T}\sum_{t=1}^{T} V_t \geq \epsilon] \leq \exp(-2T\epsilon^2/(2M)^2))$. Setting the right-hand side to be equal to $\delta > 0$ yields the statement of the lemma. □

When the loss function is convex with respect to its first argument, the lemma can be used to derive a bound on the generalization error of the average of the hypotheses generated by $\mathcal{A}$, $\frac{1}{T}\sum_{t=1}^{T} h_t$, in terms of the average loss of $\mathcal{A}$ on S, or in terms of the regret R_T and the infimum error of hypotheses in $\mathcal{H}$.

Theorem 8.15 *Let $S = ((x_1, y_1), \ldots, (x_T, y_T)) \in (\mathcal{X} \times \mathcal{Y})^T$ be a labeled sample drawn i.i.d. according to $\mathcal{D}$, L a loss bounded by M and convex with respect to its first argument, and $h_1, \ldots, h_T$ the sequence of hypotheses generated by an on-line algorithm $\mathcal{A}$ sequentially processing S. Then, for any $\delta > 0$, with probability at least*

$1-\delta$, *each of the following holds:*

$$R\left(\frac{1}{T}\sum_{t=1}^{T} h_t\right) \le \frac{1}{T}\sum_{t=1}^{T} L(h_t(x_t), y_t) + M\sqrt{\frac{2\log\frac{1}{\delta}}{T}} \tag{8.27}$$

$$R\left(\frac{1}{T}\sum_{t=1}^{T} h_t\right) \le \inf_{h\in\mathcal{H}} R(h) + \frac{R_T}{T} + 2M\sqrt{\frac{2\log\frac{2}{\delta}}{T}}. \tag{8.28}$$

Proof: By the convexity of L with respect to its first argument, for any $(x, y) \in \mathcal{X}\times\mathcal{Y}$, we have $L(\frac{1}{T}\sum_{t=1}^{T} h_t(x), y) \le \frac{1}{T}\sum_{t=1}^{T} L(h_t(x), y)$. Taking the expectation gives $R(\frac{1}{T}\sum_{t=1}^{T} h_t) \le \frac{1}{T}\sum_{t=1}^{T} R(h_t)$. The first inequality then follows by lemma 8.14. Thus, by definition of the regret R_T, for any $\delta > 0$, the following holds with probability at least $1-\delta/2$:

$$\begin{aligned} R\left(\frac{1}{T}\sum_{t=1}^{T} h_t\right) &\le \frac{1}{T}\sum_{t=1}^{T} L(h_t(x_t), y_t) + M\sqrt{\frac{2\log\frac{2}{\delta}}{T}} \\ &\le \min_{h\in\mathcal{H}} \frac{1}{T}\sum_{t=1}^{T} L(h(x_t), y_t) + \frac{R_T}{T} + M\sqrt{\frac{2\log\frac{2}{\delta}}{T}}. \end{aligned}$$

By definition of $\inf_{h\in\mathcal{H}} R(h)$, for any $\epsilon > 0$, there exists $h^* \in \mathcal{H}$ with $R(h^*) \le \inf_{h\in\mathcal{H}} R(h) + \epsilon$. By Hoeffding's inequality, for any $\delta > 0$, with probability at least $1-\delta/2$, $\frac{1}{T}\sum_{t=1}^{T} L(h^*(x_t), y_t) \le R(h^*) + M\sqrt{\frac{2\log\frac{2}{\delta}}{T}}$. Thus, for any $\epsilon > 0$, by the union bound, the following holds with probability at least $1-\delta$:

$$\begin{aligned} R\left(\frac{1}{T}\sum_{t=1}^{T} h_t\right) &\le \frac{1}{T}\sum_{t=1}^{T} L(h^*(x_t), y_t) + \frac{R_T}{T} + M\sqrt{\frac{2\log\frac{2}{\delta}}{T}} \\ &\le R(h^*) + M\sqrt{\frac{2\log\frac{2}{\delta}}{T}} + \frac{R_T}{T} + M\sqrt{\frac{2\log\frac{2}{\delta}}{T}} \\ &= R(h^*) + \frac{R_T}{T} + 2M\sqrt{\frac{2\log\frac{2}{\delta}}{T}} \\ &\le \inf_{h\in\mathcal{H}} R(h) + \epsilon + \frac{R_T}{T} + 2M\sqrt{\frac{2\log\frac{2}{\delta}}{T}}. \end{aligned}$$

Since this inequality holds for all $\epsilon > 0$, it implies the second statement of the theorem. □

The theorem can be applied to a variety of on-line regret minimization algorithms, for example when $R_T/T = O(1/\sqrt{T})$. In particular, we can apply the theorem to the exponential weighted average algorithm. Assuming that the loss L is bounded

by $M = 1$ and that the number of rounds T is known to the algorithm, we can use the regret bound of theorem 8.6. The doubling trick (used in theorem 8.7) can be used to derive a similar bound if T is not known in advance. Thus, for any $\delta > 0$, with probability at least $1 - \delta$, the following holds for the generalization error of the average of the hypotheses generated by exponential weighted average:

$$R\left(\frac{1}{T}\sum_{t=1}^{T} h_t\right) \leq \inf_{h \in \mathcal{H}} R(h) + \sqrt{\frac{\log N}{2T}} + 2\sqrt{\frac{2\log \frac{2}{\delta}}{T}},$$

where N is the number of experts, or the dimension of the weight vectors.

8.5 Game-theoretic connection

The existence of regret minimization algorithms can be used to give a simple proof of von Neumann's theorem. For any $m \geq 1$, we will denote by Δ_m the set of all distributions over $\{1, \ldots, m\}$, that is $\Delta_m = \{\mathbf{p} \in \mathbb{R}^m \colon \mathbf{p} \geq 0 \wedge \|\mathbf{p}\|_1 = 1\}$.

Theorem 8.16 (Von Neumann's minimax theorem) *Let $m, n \geq 1$. Then, for any two-person zero-sum game defined by matrix $\mathbf{M} \in \mathbb{R}^{m \times n}$,*

$$\min_{\mathbf{p} \in \Delta_m} \max_{\mathbf{q} \in \Delta_n} \mathbf{p}^\top \mathbf{M} \mathbf{q} = \max_{\mathbf{q} \in \Delta_n} \min_{\mathbf{p} \in \Delta_m} \mathbf{p}^\top \mathbf{M} \mathbf{q}. \tag{8.29}$$

Proof: The inequality $\max_{\mathbf{q}} \min_{\mathbf{p}} \mathbf{p}^\top \mathbf{M} \mathbf{q} \leq \min_{\mathbf{p}} \max_{\mathbf{q}} \mathbf{p}^\top \mathbf{M} \mathbf{q}$ is straightforward, since by definition of min, for all $\mathbf{p} \in \Delta_m, \mathbf{q} \in \Delta_n$, we have $\min_{\mathbf{p}} \mathbf{p}^\top \mathbf{M} \mathbf{q} \leq \mathbf{p}^\top \mathbf{M} \mathbf{q}$. Taking the maximum over $\mathbf{q}$ of both sides gives: $\max_{\mathbf{q}} \min_{\mathbf{p}} \mathbf{p}^\top \mathbf{M} \mathbf{q} \leq \max_{\mathbf{q}} \mathbf{p}^\top \mathbf{M} \mathbf{q}$ for all $\mathbf{p}$, subsequently taking the minimum over $\mathbf{p}$ proves the inequality.[12]

To show the reverse inequality, consider an on-line learning setting where at each round $t \in [T]$, algorithm $\mathcal{A}$ returns $\mathbf{p}_t$ and incurs loss $\mathbf{M}\mathbf{q}_t$. We can assume that $\mathbf{q}_t$ is selected in the optimal adversarial way, that is $\mathbf{q}_t \in \operatorname{argmax}_{q \in \Delta_m} \mathbf{p}_t^\top \mathbf{M} \mathbf{q}$, and that $\mathcal{A}$ is a regret minimization algorithm, that is $R_T/T \to 0$, where $R_T = \sum_{t=1}^{T} \mathbf{p}_t^\top \mathbf{M} \mathbf{q}_t - \min_{\mathbf{p} \in \Delta_m} \sum_{t=1}^{T} \mathbf{p}^\top \mathbf{M} \mathbf{q}_t$. Then, the following holds:

$$\min_{\mathbf{p} \in \Delta_m} \max_{\mathbf{q} \in \Delta_n} \mathbf{p}^\top \mathbf{M} \mathbf{q} \leq \max_{\mathbf{q}} \left(\frac{1}{T}\sum_{t=1}^{T} \mathbf{p}_t\right)^\top \mathbf{M} \mathbf{q} \leq \frac{1}{T}\sum_{t=1}^{T} \max_{\mathbf{q}} \mathbf{p}_t^\top \mathbf{M} \mathbf{q} = \frac{1}{T}\sum_{t=1}^{T} \mathbf{p}_t^\top \mathbf{M} \mathbf{q}_t.$$

[12] More generally, the maxmin is always upper bounded by the minmax for any function or two arguments and any constraint sets, following the same proof.

By definition of regret, the right-hand side can be expressed and bounded as follows:

$$\frac{1}{T}\sum_{t=1}^{T}\mathbf{p}_t^\top\mathbf{M}\mathbf{q}_t = \min_{\mathbf{p}\in\Delta_m}\frac{1}{T}\sum_{t=1}^{T}\mathbf{p}^\top\mathbf{M}\mathbf{q}_t + \frac{R_T}{T} = \min_{\mathbf{p}\in\Delta_m}\mathbf{p}^\top\mathbf{M}\Big(\frac{1}{T}\sum_{t=1}^{T}\mathbf{q}_t\Big) + \frac{R_T}{T}$$
$$\leq \max_{\mathbf{q}\in\Delta_n}\min_{\mathbf{p}\in\Delta_m}\mathbf{p}^\top\mathbf{M}\mathbf{q} + \frac{R_T}{T}.$$

This implies that the following bound holds for the minmax for all $T \geq 1$:

$$\min_{\mathbf{p}\in\Delta_m}\max_{\mathbf{q}\in\Delta_n}\mathbf{p}^\top\mathbf{M}\mathbf{q} \leq \max_{\mathbf{q}\in\Delta_n}\min_{\mathbf{p}\in\Delta_m}\mathbf{p}^\top\mathbf{M}\mathbf{q} + \frac{R_T}{T}$$

Since $\lim_{T\to+\infty}\frac{R_T}{T} = 0$, this shows that $\min_{\mathbf{p}}\max_{\mathbf{q}}\mathbf{p}^\top\mathbf{M}\mathbf{q} \leq \max_{\mathbf{q}}\min_{\mathbf{p}}\mathbf{p}^\top\mathbf{M}\mathbf{q}$.□

8.6 Chapter notes

Algorithms for regret minimization were initiated with the pioneering work of Hannan [1957] who gave an algorithm whose regret decreases as $O(\sqrt{T})$ as a function of T but whose dependency on N is linear. The weighted majority algorithm and the randomized weighted majority algorithm, whose regret is only logarithmic in N, are due to Littlestone and Warmuth [1989]. The exponential weighted average algorithm and its analysis, which can be viewed as an extension of the WM algorithm to convex non-zero-one losses is due to the same authors [Littlestone and Warmuth, 1989, 1994]. The analysis we presented follows Cesa-Bianchi [1999] and Cesa-Bianchi and Lugosi [2006]. The doubling trick technique appears in Vovk [1990] and Cesa-Bianchi et al. [1997]. The algorithm of exercise 8.7 and the analysis leading to a second-order bound on the regret are due to Cesa-Bianchi et al. [2005]. The lower bound presented in theorem 8.5 is from Blum and Mansour [2007].

While the regret bounds presented are logarithmic in the number of the experts N, when N is exponential in the size of the input problem, the computational complexity of an expert algorithm could be exponential. For example, in the online shortest paths problem, N is the number of paths between two vertices of a directed graph. However, several computationally efficient algorithms have been presented for broad classes of such problems by exploiting their structure [Takimoto and Warmuth, 2002, Kalai and Vempala, 2003, Zinkevich, 2003].

The notion of regret (or *external regret*) presented in this chapter can be generalized to that of *internal regret* or even *swap regret*, by comparing the loss of the algorithm not just to that of the best expert in retrospect, but to that of any modification of the actions taken by the algorithm by replacing each occurrence of some specific action with another one (internal regret), or even replacing actions via an arbitrary mapping (swap regret) [Foster and Vohra, 1997, Hart and Mas-Colell, 2000, Lehrer, 2003]. Several algorithms for low internal regret have been given

[Foster and Vohra, 1997, 1998, 1999, Hart and Mas-Colell, 2000, Cesa-Bianchi and Lugosi, 2001, Stoltz and Lugosi, 2003], including a conversion of low external regret to low swap regret by Blum and Mansour [2005].

The Perceptron algorithm was introduced by Rosenblatt [1958]. The algorithm raised a number of reactions, in particular by Minsky and Papert [1969], who objected that the algorithm could not be used to recognize the XOR function. Of course, the kernel Perceptron algorithm already given by Aizerman et al. [1964] could straightforwardly succeed to do so using second-degree polynomial kernels. The margin bound for the Perceptron algorithm was proven by Novikoff [1962] and is one of the first results in learning theory. We presented two extensions of Novikoff's result which hold in the more general non-separable case: Theorem 8.12 due to Freund and Schapire [1999a] and Theorem 8.11 due to Mohri and Rostamizadeh [2013]. Our proof of Theorem 8.12 is significantly more concise that the original proof given by Freund and Schapire [1999a] and shows that the bound of Theorem 8.11 is always tighter than that of Theorem 8.12. See [Mohri and Rostamizadeh, 2013] for other more general data-dependent upper bounds on the number of updates made by the Perceptron algorithm in the non-separable case. The leave-one-out analysis for SVMs is described by Vapnik [1998]. The Winnow algorithm was introduced by Littlestone [1987].

The analysis of the on-line to batch conversion and exercises 8.10 and 8.11 are from Cesa-Bianchi et al. [2001, 2004] (see also Littlestone [1989]). Von Neumann's minimax theorem admits a number of different generalizations. See Sion [1958] for a generalization to quasi-concave-convex functions semi-continuous in each argument and the references therein. The simple proof of von Neumann's theorem presented here is entirely based on learning-related techniques. A proof of a more general version using multiplicative updates was presented by Freund and Schapire [1999b].

On-line learning is a very broad and fast-growing research area in machine learning. The material presented in this chapter should be viewed only as an introduction to the topic, but the proofs and techniques presented should indicate the flavor of most results in this area. For a more comprehensive presentation of on-line learning and related game theory algorithms and techniques, the reader could consult the book of Cesa-Bianchi and Lugosi [2006].

8.7 Exercises

8.1 Perceptron lower bound. Let S be a labeled sample of m points in $\mathbb{R}^N$ with

$$x_i = (\underbrace{(-1)^i, \ldots, (-1)^i, (-1)^{i+1}}_{i \text{ first components}}, 0, \ldots, 0) \quad \text{and} \quad y_i = (-1)^{i+1}. \tag{8.30}$$

```
ON-LINE-SVM(w_0)
1  w_1 ← w_0        ▷ typically w_0 = 0
2  for t ← 1 to T do
3       RECEIVE(x_t, y_t)
4       if y_t(w_t · x_t) < 1 then
5            w_{t+1} ← w_t − η(w_t − C y_t x_t)
6       elseif y_t(w_t · x_t) > 1 then
7            w_{t+1} ← w_t − η w_t
8       else w_{t+1} ← w_t
9  return w_{T+1}
```

Figure 8.11
On-line SVM algorithm.

Show that the Perceptron algorithm makes $\Omega(2^N)$ updates before finding a separating hyperplane, regardless of the order in which it receives the points.

8.2 Generalized mistake bound. Theorem 8.8 presents a margin bound on the maximum number of updates for the Perceptron algorithm for the special case $\eta = 1$. Consider now the general Perceptron update $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \eta y_t \mathbf{x}_t$, where $\eta > 0$. Prove a bound on the maximum number of mistakes. How does η affect the bound?

8.3 Sparse instances. Suppose each input vector $\mathbf{x}_t$, $t \in [T]$, coincides with the tth unit vector of $\mathbb{R}^T$. How many updates are required for the Perceptron algorithm to converge? Show that the number of updates matches the margin bound of theorem 8.8.

8.4 Tightness of lower bound. Is the lower bound of theorem 8.5 tight? Explain why or show a counter-example.

8.5 On-line SVM algorithm. Consider the algorithm described in figure 8.11. Show that this algorithm corresponds to the stochastic gradient descent technique applied to the SVM problem (5.24) with hinge loss and no offset (i.e., fix $p = 1$ and $b = 0$).

```
MarginPerceptron()
1  w_1 ← 0
2  for t ← 1 to T do
3       Receive(x_t)
4       Receive(y_t)
5       if ((w_t = 0) or (y_t w_t · x_t / ||w_t|| < ρ/2)) then
6            w_{t+1} ← w_t + y_t x_t
7       else w_{t+1} ← w_t
8  return w_{T+1}
```

Figure 8.12
Margin Perceptron algorithm.

8.6 Margin Perceptron. Given a training sample S that is linearly separable with a maximum margin $\rho > 0$, theorem 8.8 states that the Perceptron algorithm run cyclically over S is guaranteed to converge after at most R^2/ρ^2 updates, where R is the radius of the sphere containing the sample points. However, this theorem does not guarantee that the hyperplane solution of the Perceptron algorithm achieves a margin close to ρ. Suppose we modify the Perceptron algorithm to ensure that the margin of the hyperplane solution is at least $\rho/2$. In particular, consider the algorithm described in figure 8.12. In this problem we show that this algorithm converges after at most $16R^2/\rho^2$ updates. Let $\mathcal{I}$ denote the set of times $t \in [T]$ at which the algorithm makes an update and let $M = |\mathcal{I}|$ be the total number of updates.

(a) Using an analysis similar to the one given for the Perceptron algorithm, show that $M\rho \leq \|\mathbf{w}_{T+1}\|$. Conclude that if $\|\mathbf{w}_{T+1}\| < \frac{4R^2}{\rho}$, then $M < 4R^2/\rho^2$. (For the remainder of this problem, we will assume that $\|\mathbf{w}_{T+1}\| \geq \frac{4R^2}{\rho}$.)

(b) Show that for any $t \in \mathcal{I}$ (including $t = 0$), the following holds:

$$\|\mathbf{w}_{t+1}\|^2 \leq (\|\mathbf{w}_t\| + \rho/2)^2 + R^2.$$

(c) From (b), infer that for any $t \in \mathcal{I}$ we have

$$\|\mathbf{w}_{t+1}\| \leq \|\mathbf{w}_t\| + \rho/2 + \frac{R^2}{\|\mathbf{w}_t\| + \|\mathbf{w}_{t+1}\| + \rho/2}.$$

(d) Using the inequality from (c), show that for any $t \in \mathcal{I}$ such that either $\|\mathbf{w}_t\| \geq \frac{4R^2}{\rho}$ or $\|\mathbf{w}_{t+1}\| \geq \frac{4R^2}{\rho}$, we have

$$\|\mathbf{w}_{t+1}\| \leq \|\mathbf{w}_t\| + \frac{3}{4}\rho.$$

(e) Show that $\|\mathbf{w}_1\| \leq R \leq 4R^2/\rho$. Since by assumption we have $\|\mathbf{w}_{T+1}\| \geq \frac{4R^2}{\rho}$, conclude that there must exist a largest time $t_0 \in \mathcal{I}$ such that $\|\mathbf{w}_{t_0}\| \leq \frac{4R^2}{\rho}$ and $\|\mathbf{w}_{t_0+1}\| \geq \frac{4R^2}{\rho}$.

(f) Show that $\|\mathbf{w}_{T+1}\| \leq \|\mathbf{w}_{t_0}\| + \frac{3}{4}M\rho$. Conclude that $M \leq 16R^2/\rho^2$.

8.7 Second-order regret bound. Consider the randomized algorithm that differs from the RWM algorithm only by the weight update, i.e., $w_{t+1,i} \leftarrow (1-(1-\beta)l_{t,i})w_{t,i}$, $t \in [T]$, which is applied to all $i \in [N]$ with $1/2 \leq \beta < 1$. This algorithm can be used in a more general setting than RWM since the losses $l_{t,i}$ are only assumed to be in $[0,1]$. The objective of this problem is to show that a similar upper bound can be shown for the regret.

(a) Use the same potential W_t as for the RWM algorithm and derive a simple upper bound for $\log W_{T+1}$:

$$\log W_{T+1} \leq \log N - (1-\beta)\mathcal{L}_T\,.$$

(*Hint*: Use the inequality $\log(1-x) \leq -x$ for $x \in [0, 1/2]$.)

(b) Prove the following lower bound for the potential for all $i \in [N]$:

$$\log W_{T+1} \geq -(1-\beta)\mathcal{L}_{T,i} - (1-\beta)^2 \sum_{t=1}^{T} l_{t,i}^2\,.$$

(*Hint*: Use the inequality $\log(1-x) \geq -x - x^2$, which is valid for all $x \in [0, 1/2]$.)

(c) Use upper and lower bounds to derive the following regret bound for the algorithm: $R_T \leq 2\sqrt{T \log N}$.

8.8 Polynomial weighted algorithm. The objective of this problem is to show how another regret minimization algorithm can be defined and studied. Let L be a loss function convex in its first argument and taking values in $[0, M]$.

We will assume $N > e^2$ and then for any expert $i \in [N]$, we denote by $r_{t,i}$ the instantaneous regret of that expert at time $t \in [T]$, $r_{t,i} = L(\widehat{y}_t, y_t) - L(y_{t,i}, y_t)$,

and by $R_{t,i}$ its cumulative regret up to time t: $R_{t,i} = \sum_{s=1}^{t} r_{t,i}$. For convenience, we also define $R_{0,i} = 0$ for all $i \in [N]$. For any $x \in \mathbb{R}$, $(x)_+$ denotes $\max(x, 0)$, that is the positive part of x, and for $\mathbf{x} = (x_1, \ldots, x_N)^\top \in \mathbb{R}^N$, $(\mathbf{x})_+ = ((x_1)_+, \ldots, (x_N)_+)^\top$.

Let $\alpha > 2$ and consider the algorithm that predicts at round $t \in [T]$ according to $\widehat{y}_t = \frac{\sum_{i=1}^{n} w_{t,i} y_{t,i}}{\sum_{i=1}^{n} w_{t,i}}$, with the weight $w_{t,i}$ defined based on the αth power of the regret up to time $(t-1)$: $w_{t,i} = (R_{t-1,i})_+^{\alpha-1}$. The potential function we use to analyze the algorithm is based on the function Φ defined over $\mathbb{R}^N$ by $\Phi \colon \mathbf{x} \mapsto \|(\mathbf{x})_+\|_\alpha^2 = \left[\sum_{i=1}^{N} (x_i)_+^\alpha\right]^{\frac{2}{\alpha}}$.

(a) Show that Φ is twice differentiable over $\mathbb{R}^N - B$, where B is defined as follows:

$$B = \{\mathbf{u} \in \mathbb{R}^N \colon (\mathbf{u})_+ = 0\}.$$

(b) For any $t \in [T]$, let $\mathbf{r}_t$ denote the vector of instantaneous regrets, $\mathbf{r}_t = (r_{t,1}, \ldots, r_{t,N})^\top$, and similarly $\mathbf{R}_t = (R_{t,1}, \ldots, R_{t,N})^\top$. We define the potential function as $\Phi(\mathbf{R}_t) = \|(\mathbf{R}_t)_+\|_\alpha^2$. Compute $\nabla\Phi(\mathbf{R}_{t-1})$ for $\mathbf{R}_{t-1} \notin B$ and show that $\nabla\Phi(\mathbf{R}_{t-1}) \cdot \mathbf{r}_t \leq 0$ (*Hint*: use the convexity of the loss with respect to the first argument).

(c) Prove the inequality $\mathbf{r}^\top[\nabla^2\Phi(\mathbf{u})]\mathbf{r} \leq 2(\alpha-1)\|\mathbf{r}\|_\alpha^2$ valid for all $\mathbf{r} \in \mathbb{R}^N$ and $\mathbf{u} \in \mathbb{R}^N - B$ (*Hint*: write the Hessian $\nabla^2\Phi(\mathbf{u})$ as a sum of a diagonal matrix and a positive semidefinite matrix multiplied by $(2-\alpha)$. Also, use Hölder's inequality generalizing Cauchy-Schwarz: for any $p > 1$ and $q > 1$ with $\frac{1}{p} + \frac{1}{q} = 1$ and $\mathbf{u}, \mathbf{v} \in \mathbb{R}^N$, $|\mathbf{u} \cdot \mathbf{v}| \leq \|\mathbf{u}\|_p \|\mathbf{v}\|_q$).

(d) Using the answers to the two previous questions and Taylor's formula, show that for all $t \geq 1$, $\Phi(\mathbf{R}_t) - \Phi(\mathbf{R}_{t-1}) \leq (\alpha-1)\|\mathbf{r}_t\|_\alpha^2$, if $\gamma\mathbf{R}_{t-1} + (1-\gamma)\mathbf{R}_t \notin B$ for all $\gamma \in [0, 1]$.

(e) Suppose there exists $\gamma \in [0, 1]$ such that $(1-\gamma)\mathbf{R}_{t-1} + \gamma\mathbf{R}_t \in B$. Show that $\Phi(\mathbf{R}_t) \leq (\alpha-1)\|\mathbf{r}_t\|_\alpha^2$.

(f) Using the two previous questions, derive an upper bound on $\Phi(\mathbf{R}_T)$ expressed in terms of T, N, and M.

(g) Show that $\Phi(\mathbf{R}_T)$ admits as a lower bound the square of the regret R_T of the algorithm.

(h) Using the two previous questions give an upper bound on the regret R_T. For what value of α is the bound the most favorable? Give a simple expression of the upper bound on the regret for a suitable approximation of that optimal value.

8.9 General inequality. In this exercise we generalize the result of exercise 8.7 by using a more general inequality: $\log(1-x) \geq -x - \frac{x^2}{\alpha}$ for some $0 < \alpha < 2$.

(a) First prove that the inequality is true for $x \in [0, 1 - \frac{\alpha}{2}]$. What does this imply about the valid range of β?

(b) Give a generalized version of the regret bound derived in exercise 8.7 in terms of α, which shows:

$$R_T \leq \frac{\log N}{1-\beta} + \frac{1-\beta}{\alpha} T\,.$$

What is the optimal choice of β and the resulting bound in this case?

(c) Explain how α may act as a regularization parameter. What is the optimal choice of α?

8.10 On-line to batch — non-convex loss.

The on-line to batch result of theorem 8.15 heavily relies on the fact that the loss is convex in order to provide a generalization guarantee for the uniformly averaged hypothesis $\frac{1}{T}\sum_{i=1}^{T} h_i$. For general losses, instead of using the averaged hypothesis we will use a different strategy and try to estimate the best single base hypothesis and show the expected loss of this hypothesis is bounded.

Let m_i denote the cumulative loss of hypothesis h_i on the points $(x_i, \ldots, x_T)$, that is $m_i = \sum_{t=i}^{T} L(h_i(x_t), y_t)$. Then we define the *penalized risk estimate* of hypothesis h_i as,

$$\frac{m_i}{T-i+1} + c_\delta(T-i+1) \quad \text{where} \quad c_\delta(x) = \sqrt{\frac{1}{2x}\log\frac{T(T+1)}{\delta}}\,.$$

The term c_δ penalizes the empirical error when the test sample is small. Define $\widehat{h} = h_{i^*}$ where $i^* = \operatorname{argmin}_i m_i/(T-i+1) + c_\delta(T-i+1)$. We will then show under the same conditions of theorem 8.15 (with $M = 1$ for simplicity), but without requiring the convexity of L, that the following holds with probability at least $1-\delta$:

$$R(\widehat{h}) \leq \frac{1}{T}\sum_{i=1}^{T} L(h_i(x_i), y_i) + 6\sqrt{\frac{1}{T}\log\frac{2(T+1)}{\delta}}\,. \tag{8.31}$$

(a) Prove the following inequality:

$$\min_{i\in[T]}(R(h_i) + 2c_\delta(T-i+1)) \leq \frac{1}{T}\sum_{i=1}^{T} R(h_i) + 4\sqrt{\frac{1}{T}\log\frac{T+1}{\delta}}\,.$$

(b) Use part (a) to show that with probability at least $1-\delta$,

$$\min_{i\in[T]}(R(h_i)+2c_\delta(T-i+1)) < \sum_{i=1}^{T} L(h_i(x_i),y_i)+\sqrt{\frac{2}{T}\log\frac{1}{\delta}}+4\sqrt{\frac{1}{T}\log\frac{T+1}{\delta}}\,.$$

(c) By design, the definition of c_δ ensures that with probability at least $1-\delta$

$$R(\widehat{h}) \leq \min_{i\in[T]}(R(h_i)+2c_\delta(T-i+1))\,.$$

Use this property to complete the proof of (8.31).

8.11 On-line to batch — kernel Perceptron margin bound. In this problem, we give a margin-based generalization guarantee for the kernel Perceptron algorithm. Let $h_1,\ldots,h_T$ be the sequence of hypotheses generated by the kernel Perceptron algorithm and let $\widehat{h}$ be defined as in exercise 8.10. Finally, let L denote the zero-one loss. We now wish to more precisely bound the generalization error of $\widehat{h}$ in this setting.

(a) First, show that

$$\sum_{i=1}^{T} L(h_i(x_i),y_i) \leq \inf_{h\in\mathbb{H}:\|h\|\leq 1}\sum_{i=1}^{T}\max\left(0,1-\frac{y_i h(x_i)}{\rho}\right)+\frac{1}{\rho}\sqrt{\sum_{i\in I}K(x_i,x_i)},$$

where I is the set of indices where the kernel Perceptron makes an update and where δ and ρ are defined as in theorem 8.12.

(b) Now, use the result of exercise 8.10 to derive a generalization guarantee for $\widehat{h}$ in the case of kernel Perceptron, which states that for any $0<\delta\leq 1$, the following holds with probability at least $1-\delta$:

$$R(\widehat{h}) \leq \inf_{h\in\mathbb{H}:\|h\|\leq 1}\widehat{R}_{S,\rho}(h)+\frac{1}{\rho T}\sqrt{\sum_{i\in I}K(x_i,x_i)}+6\sqrt{\frac{1}{T}\log\frac{2(T+1)}{\delta}},$$

where $\widehat{R}_{S,\rho}(h)=\frac{1}{T}\sum_{i=1}^{T}\max\left(0,1-\frac{y_i h(x_i)}{\rho}\right)$. Compare this result with the margin bounds for kernel-based hypotheses given by corollary 6.13.