7 Boosting

Ensemble methods are general techniques in machine learning for combining several predictors to create a more accurate one. This chapter studies an important family of ensemble methods known as *boosting*, and more specifically the *AdaBoost* algorithm. This algorithm has been shown to be very effective in practice in some scenarios and is based on a rich theoretical analysis. We first introduce AdaBoost, show how it can rapidly reduce the empirical error as a function of the number of rounds of boosting, and point out its relationship with some known algorithms. Next, we present a theoretical analysis of the generalization properties of AdaBoost based on the VC-dimension of its hypothesis set and then based on the notion of margin. The margin theory developed in this context can be applied to other similar ensemble algorithms. A game-theoretic interpretation of AdaBoost further helps analyzing its properties and revealing the equivalence between the weak learning assumption and a separability condition. We end with a discussion of AdaBoost's benefits and drawbacks.

7.1 Introduction

It is often difficult, for a non-trivial learning task, to directly devise an accurate algorithm satisfying the strong PAC-learning requirements of chapter 2. But, there can be more hope for finding simple predictors guaranteed only to perform slightly better than random. The following gives a formal definition of such *weak learners*. As in the PAC-learning chapter, we let n be a number such that the computational cost of representing any element $x \in \mathcal{X}$ is at most $O(n)$ and denote by $\mathrm{size}(c)$ the maximal cost of the computational representation of $c \in \mathcal{C}$.

Definition 7.1 (Weak learning) *A concept class $\mathcal{C}$ is said to be* weakly PAC-learnable *if there exists an algorithm $\mathcal{A}$, $\gamma > 0$, and a polynomial function $poly(\cdot, \cdot, \cdot)$ such that for any $\delta > 0$, for all distributions $\mathcal{D}$ on $\mathcal{X}$ and for any target concept $c \in \mathcal{C}$,*

```
AdaBoost($S = ((x_1, y_1), \ldots, (x_m, y_m))$)
for $i \leftarrow 1$ to $m$ do
    $\mathcal{D}_1(i) \leftarrow \frac{1}{m}$
for $t \leftarrow 1$ to $T$ do
    $h_t \leftarrow$ base classifier in $\mathcal{H}$ with small error $\epsilon_t = \mathbb{P}_{i \sim \mathcal{D}_t}\big[h_t(x_i) \neq y_i\big]$
    $\alpha_t \leftarrow \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$
    $Z_t \leftarrow 2\big[\epsilon_t(1-\epsilon_t)\big]^{\frac{1}{2}}$    ▷ normalization factor
    for $i \leftarrow 1$ to $m$ do
        $\mathcal{D}_{t+1}(i) \leftarrow \frac{\mathcal{D}_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$
$f \leftarrow \sum_{t=1}^{T} \alpha_t h_t$
return $f$
```

Figure 7.1
AdaBoost algorithm for a base classifier set $\mathcal{H} \subseteq \{-1, +1\}^{\mathcal{X}}$.

the following holds for any sample size $m \geq poly(1/\delta, n, size(c))$:

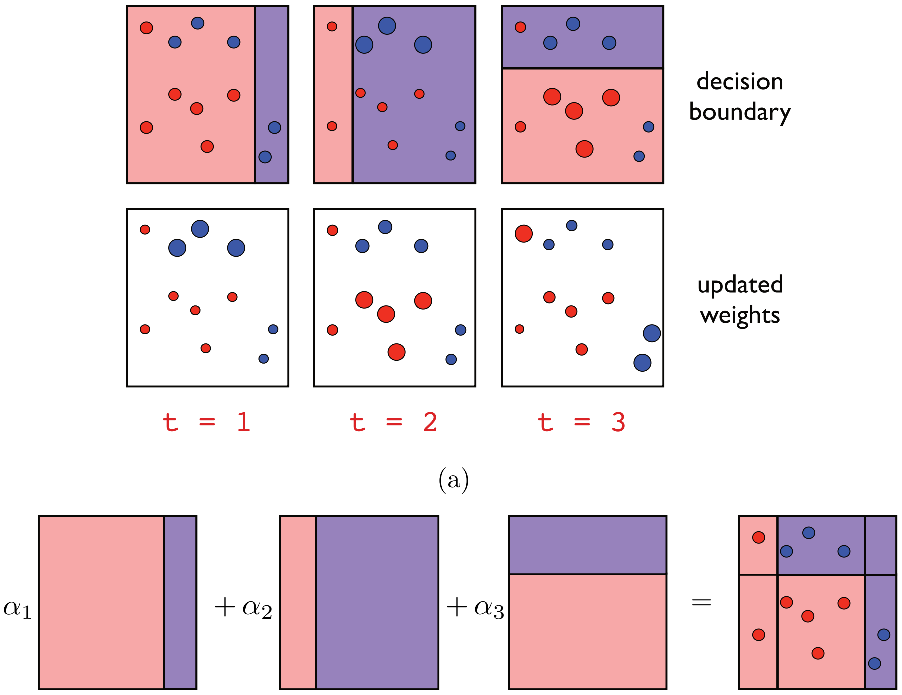
$$\underset{S \sim \mathcal{D}^m}{\mathbb{P}}\left[R(h_S) \leq \frac{1}{2} - \gamma\right] \geq 1 - \delta\,, \tag{7.1}$$

where h_S is the hypothesis returned by algorithm $\mathcal{A}$ when trained on sample S. When such an algorithm $\mathcal{A}$ exists, it is called a weak learning algorithm *for $\mathcal{C}$ or a* weak learner*. The hypotheses returned by a weak learning algorithm are called* base classifiers*.*

The key idea behind boosting techniques is to use a weak learning algorithm to build a *strong learner*, that is, an accurate PAC-learning algorithm. To do so, boosting techniques use an ensemble method: they combine different base classifiers returned by a weak learner to create a more accurate predictor. But which base classifiers should be used and how should they be combined? The next section addresses these questions by describing in detail one of the most prevalent and successful boosting algorithms, AdaBoost.

7.2 AdaBoost

We denote by $\mathcal{H}$ the hypothesis set out of which the base classifiers are selected, which we will sometimes refer to as the *base classifier set*. Figure 7.1 gives the



(a)

(b)

Figure 7.2
Example of AdaBoost with axis-aligned hyperplanes as base classifiers. (a) The top row shows decision boundaries at each boosting round. The bottom row shows how weights are updated at each round, with incorrectly (resp., correctly) points given increased (resp., decreased) weights. (b) Visualization of final classifier, constructed as a non-negative linear combination of base classifiers.

pseudocode of AdaBoost in the case where the base classifiers are functions mapping from $\mathcal{X}$ to $\{-1, +1\}$, thus $\mathcal{H} \subseteq \{-1, +1\}^{\mathcal{X}}$.

The algorithm takes as input a labeled sample $S = ((x_1, y_1), \ldots, (x_m, y_m))$, with $(x_i, y_i) \in \mathcal{X} \times \{-1, +1\}$ for all $i \in [m]$, and maintains a distribution over the indices $\{1, \ldots, m\}$. Initially (lines 1-2), the distribution is uniform ($\mathcal{D}_1$). At each *round of boosting*, that is each iteration $t \in [T]$ of the loop 3–8, a new base classifier $h_t \in \mathcal{H}$ is selected that minimizes the error on the training sample weighted by the distribution $\mathcal{D}_t$:

$$h_t \in \operatorname*{argmin}_{h \in \mathcal{H}} \underset{i \sim \mathcal{D}_t}{\mathbb{P}} \big[h(x_i) \neq y_i \big] = \operatorname*{argmin}_{h \in \mathcal{H}} \sum_{i=1}^{m} \mathcal{D}_t(i) 1_{h(x_i) \neq y_i}.$$

Z_t is simply a normalization factor to ensure that the weights $\mathcal{D}_{t+1}(i)$ sum to one. The precise reason for the definition of the coefficient α_t will become clear later. For now, observe that if ϵ_t, the error of the base classifier, is less than $\frac{1}{2}$, then $\frac{1-\epsilon_t}{\epsilon_t} > 1$ and α_t is positive ($\alpha_t > 0$). Thus, the new distribution $\mathcal{D}_{t+1}$ is defined from $\mathcal{D}_t$ by substantially increasing the weight on i if point x_i is incorrectly classified ($y_i h_t(x_i) < 0$), and, on the contrary, decreasing it if x_i is correctly classified. This has the effect of focusing more on the points incorrectly classified at the next round of boosting, less on those correctly classified by h_t.

After T rounds of boosting, the classifier returned by AdaBoost is based on the sign of function f, which is a non-negative linear combination of the base classifiers h_t. The weight α_t assigned to h_t in that sum is a logarithmic function of the ratio of the accuracy $1 - \epsilon_t$ and error ϵ_t of h_t. Thus, more accurate base classifiers are assigned a larger weight in that sum. Figure 7.2 illustrates the AdaBoost algorithm. The size of the points represents the distribution weight assigned to them at each boosting round.

For any $t \in [T]$, we will denote by f_t the linear combination of the base classifiers after t rounds of boosting: $f_t = \sum_{s=1}^{t} \alpha_s h_s$. In particular, we have $f_T = f$. The distribution $\mathcal{D}_{t+1}$ can be expressed in terms of f_t and the normalization factors Z_s, $s \in [t]$, as follows:

$$\forall i \in [m], \quad \mathcal{D}_{t+1}(i) = \frac{e^{-y_i f_t(x_i)}}{m \prod_{s=1}^{t} Z_s}\,. \tag{7.2}$$

We will make use of this identity several times in the proofs of the following sections. It can be shown straightforwardly by repeatedly expanding the definition of the distribution over the point x_i:

$$\begin{aligned}\mathcal{D}_{t+1}(i) = \frac{\mathcal{D}_t(i) e^{-\alpha_t y_i h_t(x_i)}}{Z_t} &= \frac{\mathcal{D}_{t-1}(i) e^{-\alpha_{t-1} y_i h_{t-1}(x_i)} e^{-\alpha_t y_i h_t(x_i)}}{Z_{t-1} Z_t} \\ &= \frac{e^{-y_i \sum_{s=1}^{t} \alpha_s h_s(x_i)}}{m \prod_{s=1}^{t} Z_s}.\end{aligned}$$

The AdaBoost algorithm can be generalized in several ways:

- Instead of a hypothesis with minimal weighted error, h_t can be more generally the base classifier returned by a weak learning algorithm trained on $\mathcal{D}_t$;
- The range of the base classifiers could be $[-1, +1]$, or more generally a bounded subset of $\mathbb{R}$. The coefficients α_t can then be different and may not even admit a closed form. In general, they are chosen to minimize an upper bound on the empirical error, as discussed in the next section. Of course, in that general case, the hypotheses h_t are not binary *classifiers*, but their sign could define the label, and their magnitude could be interpreted as a measure of confidence.

In rest of this chapter, the range of the base classifiers in $\mathcal{H}$ will be assumed to be included in $[-1, +1]$. We now further analyze the properties of AdaBoost and discuss its typical use in practice.

7.2.1 Bound on the empirical error

We first show that the empirical error of AdaBoost decreases exponentially fast as a function of the number of rounds of boosting.

Theorem 7.2 *The empirical error of the classifier returned by AdaBoost verifies:*

$$\widehat{R}_S(f) \leq \exp\left[-2\sum_{t=1}^{T}\Big(\frac{1}{2}-\epsilon_t\Big)^2\right]. \tag{7.3}$$

Furthermore, if for all $t \in [T]$, $\gamma \leq \big(\frac{1}{2}-\epsilon_t\big)$, then

$$\widehat{R}_S(f) \leq \exp(-2\gamma^2 T)\,. \tag{7.4}$$

Proof: Using the general inequality $1_{u\leq 0} \leq \exp(-u)$ valid for all $u \in \mathbb{R}$ and identity 7.2, we can write:

$$\widehat{R}_S(f) = \frac{1}{m}\sum_{i=1}^{m} 1_{y_i f(x_i)\leq 0} \leq \frac{1}{m}\sum_{i=1}^{m} e^{-y_i f(x_i)} = \frac{1}{m}\sum_{i=1}^{m}\left[m\prod_{t=1}^{T} Z_t\right]\mathcal{D}_{T+1}(i) = \prod_{t=1}^{T} Z_t.$$

Since for all $t \in [T]$, Z_t is a normalization factor, it can be expressed in terms of ϵ_t by:

$$\begin{aligned}
Z_t = \sum_{i=1}^{m}\mathcal{D}_t(i)e^{-\alpha_t y_i h_t(x_i)} &= \sum_{i:y_ih_t(x_i)=+1}\mathcal{D}_t(i)e^{-\alpha_t} + \sum_{i:y_ih_t(x_i)=-1}\mathcal{D}_t(i)e^{\alpha_t}\\
&= (1-\epsilon_t)e^{-\alpha_t} + \epsilon_t e^{\alpha_t}\\
&= (1-\epsilon_t)\sqrt{\frac{\epsilon_t}{1-\epsilon_t}} + \epsilon_t\sqrt{\frac{1-\epsilon_t}{\epsilon_t}} = 2\sqrt{\epsilon_t(1-\epsilon_t)}\,.
\end{aligned}$$

Thus, the product of the normalization factors can be expressed and upper bounded as follows:

$$\begin{aligned}
\prod_{t=1}^{T} Z_t = \prod_{t=1}^{T} 2\sqrt{\epsilon_t(1-\epsilon_t)} = \prod_{t=1}^{T}\sqrt{1-4\big(\tfrac{1}{2}-\epsilon_t\big)^2} &\leq \prod_{t=1}^{T}\exp\Big[-2\big(\tfrac{1}{2}-\epsilon_t\big)^2\Big]\\
&= \exp\Big[-2\sum_{t=1}^{T}\big(\tfrac{1}{2}-\epsilon_t\big)^2\Big],
\end{aligned}$$

where the inequality follows from the inequality $1-x \leq e^{-x}$ valid for all $x \in \mathbb{R}$. □

Note that the value of γ, which is known as the *edge*, and the accuracy of the base classifiers do not need to be known to the algorithm. The algorithm adapts to their

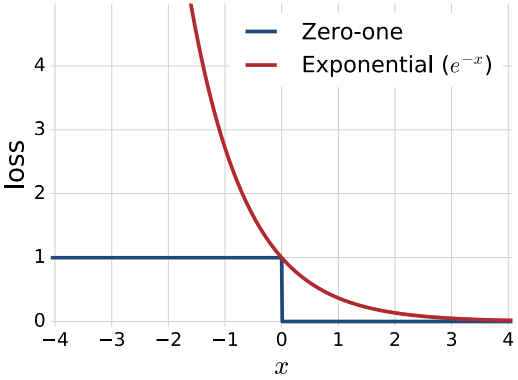


Figure 7.3
Visualization of the zero-one loss (blue) and the convex and differentiable upper bound on the zero-one loss (red) that is optimized by AdaBoost.

accuracy and defines a solution based on these values. This is the source of the extended name of AdaBoost: *adaptive boosting*.

The proof of theorem 7.2 reveals several other important properties. First, observe that α_t is the minimizer of the function $\varphi\colon \alpha \mapsto (1-\epsilon_t)e^{-\alpha} + \epsilon_t e^{\alpha}$. Indeed, φ is convex and differentiable, and setting its derivative to zero yields:

$$\varphi'(\alpha) = -(1-\epsilon_t)e^{-\alpha} + \epsilon_t e^{\alpha} = 0 \Leftrightarrow (1-\epsilon_t)e^{-\alpha} = \epsilon_t e^{\alpha} \Leftrightarrow \alpha = \frac{1}{2}\log\frac{1-\epsilon_t}{\epsilon_t}. \quad (7.5)$$

Thus, α_t is chosen to minimize $Z_t = \varphi(\alpha_t)$ and, in light of the bound $\widehat{R}_S(f) \leq \prod_{t=1}^{T} Z_t$ shown in the proof, these coefficients are selected to minimize an upper bound on the empirical error. In fact, for base classifiers whose range is $[-1,+1]$ or $\mathbb{R}$, α_t can be chosen in a similar fashion to minimize Z_t, and this is the way AdaBoost is extended to these more general cases.

Observe also that the equality $(1-\epsilon_t)e^{-\alpha_t} = \epsilon_t e^{\alpha_t}$ just shown in (7.5) implies that at each iteration, AdaBoost assigns equal distribution mass to correctly and incorrectly classified instances, since $(1-\epsilon_t)e^{-\alpha_t}$ is the total distribution assigned to correctly classified points and $\epsilon_t e^{\alpha_t}$ that of incorrectly classified ones. This may seem to contradict the fact that AdaBoost increases the weights of incorrectly classified points and decreases that of others, but there is in fact no inconsistency: the reason is that there are always fewer incorrectly classified points, since the base classifier's accuracy is better than random.

7.2.2 Relationship with coordinate descent

AdaBoost was originally designed to address the theoretical question of whether a weak learning algorithm could be used to derive a strong learning one. Here, we will show that it coincides in fact with a very simple algorithm, which consists of applying a general coordinate descent technique to a convex and differentiable objective function.

For simplicity, in this section, we assume that the base classifier set $\mathcal{H}$ is finite, with cardinality N: $\mathcal{H} = \{h_1, \ldots, h_N\}$. An ensemble function f such as the one returned by AdaBoost can then be written as $f = \sum_{j=1}^N \bar{\alpha}_j h_j$, with $\bar{\alpha}_j \geq 0$. Given a labeled sample $S = ((x_1, y_1), \ldots, (x_m, y_m))$, let F be the objective function defined for all $\bar{\boldsymbol{\alpha}} = (\bar{\alpha}_1, \ldots, \bar{\alpha}_N) \in \mathbb{R}^N$ by

$$F(\bar{\boldsymbol{\alpha}}) = \frac{1}{m} \sum_{i=1}^m e^{-y_i f(x_i)} = \frac{1}{m} \sum_{i=1}^m e^{-y_i \sum_{j=1}^N \bar{\alpha}_j h_j(x_i)}. \tag{7.6}$$

Since the exponential loss $u \mapsto e^{-u}$ is an upper bound on the zero-one loss $u \mapsto 1_{u \leq 0}$ (see figure 7.3), F is an upper bound on the empirical error:

$$\widehat{R}_S(f) = \frac{1}{m} \sum_{i=1}^m 1_{y_i f(x_i) \leq 0} \leq \frac{1}{m} \sum_{i=1}^m e^{-y_i f(x_i)}. \tag{7.7}$$

F is a convex function of $\bar{\boldsymbol{\alpha}}$ since it is a sum of convex functions, each obtained by composition of the (convex) exponential function with an affine function of $\bar{\boldsymbol{\alpha}}$. F is also differentiable since the exponential function is differentiable. We will show that F is the objective function minimized by AdaBoost.

Different convex optimization techniques can be used to minimize F. Here, we will use a variant of the coordinate descent technique. Coordinate descent is applied over T rounds. Let $\bar{\boldsymbol{\alpha}}_0 = \mathbf{0}$ and let $\bar{\boldsymbol{\alpha}}_t$ denote the parameter vector at the end of iteration t. At each round $t \in [T]$, a direction $\mathbf{e}_k$ corresponding to the kth coordinate of $\bar{\boldsymbol{\alpha}}$ in $\mathbb{R}^N$ is selected, as well as a step size η along that direction. $\bar{\boldsymbol{\alpha}}_t$ is obtained from $\bar{\boldsymbol{\alpha}}_{t-1}$ according to the update $\bar{\boldsymbol{\alpha}}_t = \bar{\boldsymbol{\alpha}}_{t-1} + \eta \mathbf{e}_k$, where η is the step size chosen along the direction $\mathbf{e}_k$. Observe that if we denote by $\bar{g}_t$ the ensemble function defined by $\bar{\boldsymbol{\alpha}}_t$, that is $\bar{g}_t = \sum_{j=1}^N \bar{\alpha}_{t,j} h_j$, then the coordinate descent update coincides with the update $\bar{g}_t = \bar{g}_{t-1} + \eta h_k$, which is also the AdaBoost update. Thus, since both algorithms start with $\bar{g}_0 = 0$, to show that AdaBoost coincides with coordinate descent applied to F, it suffices to show at every iteration t, coordinate descent selects the same base hypothesis h_k and step η as AdaBoost. We will assume by induction that this holds up to iteration $t-1$, which implies the equality $\bar{g}_{t-1} = f_{t-1}$, and will show then that it also holds at iteration t.

The variant of coordinate descent we consider here consists of selecting, at each iteration, the maximum descent direction, that is the direction $\mathbf{e}_k$ along which the derivative of F is the largest in absolute value, and of selecting the best step along that direction, that is of choosing η to minimize $F(\bar{\boldsymbol{\alpha}}_{t-1} + \eta \mathbf{e}_k)$. To give the expressions of the direction and the step at each iteration, we first introduce similar quantities to those appearing in the analysis of the boosting algorithm. For any $t \in [T]$, we define a distribution $\bar{\mathcal{D}}_t$ over the indices $\{1, \ldots, m\}$ as follows:

$$\bar{\mathcal{D}}_t(i) = \frac{e^{-y_i \sum_{j=1}^N \bar{\alpha}_{t-1,j} h_j(x_i)}}{\bar{Z}_t} = \frac{e^{-y_i \bar{g}_{t-1}(x_i)}}{\bar{Z}_t},$$

where $\bar{Z}_t$ is the normalization factor $\bar{Z}_t = \sum_{i=1}^m e^{-y_i \sum_{j=1}^N \bar{\alpha}_{t-1,j} h_j(x_i)}$. Observe that, since $\bar{g}_{t-1} = f_{t-1}$, $\bar{\mathcal{D}}_t$ coincides with $\mathcal{D}_t$. We also define for any base hypothesis h_j, $j \in [N]$, its expected error $\bar{\epsilon}_{t,j}$ with respect to the distribution $\bar{\mathcal{D}}_t$:

$$\bar{\epsilon}_{t,j} = \mathop{\mathbb{E}}_{i \sim \bar{\mathcal{D}}_t} \left[1_{y_i h_j(x_i) \leq 0} \right].$$

The directional derivative of F at $\bar{\boldsymbol{\alpha}}_{t-1}$ along $\mathbf{e}_k$ is denoted by $F'(\bar{\boldsymbol{\alpha}}_{t-1}, \mathbf{e}_k)$ and defined by

$$F'(\bar{\boldsymbol{\alpha}}_{t-1}, \mathbf{e}_k) = \lim_{\eta \to 0} \frac{F(\bar{\boldsymbol{\alpha}}_{t-1} + \eta \mathbf{e}_k) - F(\bar{\boldsymbol{\alpha}}_{t-1})}{\eta}.$$

Since $F(\bar{\boldsymbol{\alpha}}_{t-1} + \eta \mathbf{e}_k) = \sum_{i=1}^m e^{-y_i \sum_{j=1}^N \bar{\alpha}_{t-1,j} h_j(x_i) - \eta y_i h_k(x_i)}$, the directional derivative along $\mathbf{e}_k$ can be expressed as follows:

$$\begin{aligned}
F'(\bar{\boldsymbol{\alpha}}_{t-1}, \mathbf{e}_k) &= -\frac{1}{m} \sum_{i=1}^m y_i h_k(x_i) e^{-y_i \sum_{j=1}^N \bar{\alpha}_{t-1,j} h_j(x_i)} \\
&= -\frac{1}{m} \sum_{i=1}^m y_i h_k(x_i) \bar{\mathcal{D}}_t(i) \bar{Z}_t \\
&= -\left[\sum_{i=1}^m \bar{\mathcal{D}}_t(i) 1_{y_i h_k(x_i) = +1} - \sum_{i=1}^m \bar{\mathcal{D}}_t(i) 1_{y_i h_k(x_i) = -1} \right] \frac{\bar{Z}_t}{m} \\
&= -\left[(1 - \bar{\epsilon}_{t,k}) - \bar{\epsilon}_{t,k} \right] \frac{\bar{Z}_t}{m} = \left[2\bar{\epsilon}_{t,k} - 1 \right] \frac{\bar{Z}_t}{m}.
\end{aligned}$$

Since $\frac{\bar{Z}_t}{m}$ does not depend on k, the maximum descent direction k is the one minimizing $\bar{\epsilon}_{t,k}$. Thus, the hypothesis h_k selected by coordinate descent at iteration t is the one with the smallest expected error on the sample S, where the expectation is taken with respect to $\bar{\mathcal{D}}_t = \mathcal{D}_t$. This matches exactly the choice made by AdaBoost at the tth round.

The step size η is selected to minimize the function along the direction $\mathbf{e}_k$ chosen: $\operatorname{argmin}_\eta F(\bar{\boldsymbol{\alpha}}_{t-1} + \eta\, \mathbf{e}_k)$. Since $F(\bar{\boldsymbol{\alpha}}_{t-1} + \eta\, \mathbf{e}_k)$ is a convex function of η, to find the

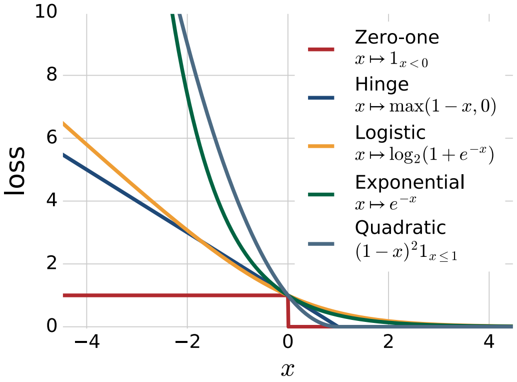


Figure 7.4
Examples of several convex upper bounds on the zero-one loss.

minimum, it suffices to set its derivative to zero:

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

This proves that the step size chosen by coordinate descent coincides with the weight α_t assigned by AdaBoost to the classifier chosen in the tth round. Thus, coordinate descent applied to exponential objective F precisely coincides with AdaBoost and F can be viewed as the objective function that AdaBoost seeks to minimize.

In light of this relationship, one may wish to consider similar applications of coordinate descent to other convex and differentiable functions of $\bar{\boldsymbol{\alpha}}$ upper-bounding the zero-one loss. In particular, the *logistic loss* $x \mapsto \log_2(1+e^{-x})$ is convex and differentiable and upper bounds the zero-one loss. Figure 7.4 shows other examples of alternative convex loss functions upper-bounding the zero-one loss. Using the logistic loss, instead of the exponential loss used by AdaBoost, leads to an objective that coincides with *logistic regression*.

7.2.3 Practical use

Here, we briefly describe the standard practical use of AdaBoost. An important requirement for the algorithm is the choice of the base classifiers or that of the weak learner. The family of base classifiers typically used with AdaBoost in practice is that of *decision trees*, which are equivalent to hierarchical partitions of the space (see chapter 9, section 9.3.3). Among decision trees, those of depth one, also known as *stumps*, are by far the most frequently used base classifiers.

Boosting stumps are threshold functions associated to a single feature. Thus, a stump corresponds to a single axis-aligned partition of the space, as illustrated in figure 7.2. If the data is in $\mathbb{R}^N$, we can associate a stump to each of the N components. Thus, to determine the stump with the minimal weighted error at each round of boosting, the best component and the best threshold for each component must be computed.

To do so, we can first presort each component in $O(m \log m)$ time with a total computational cost of $O(mN \log m)$. For a given component, there are only $m+1$ possible distinct thresholds, since two thresholds between the same consecutive component values are equivalent. To find the best threshold at each round of boosting, all of these possible $m+1$ values can be compared, which can be done in $O(m)$ time. Thus, the total computational complexity of the algorithm for T rounds of boosting is $O(mN \log m + mNT)$.

Observe, however, that while boosting stumps are widely used in combination with AdaBoost and can perform well in practice, the algorithm that returns the stump with the minimal (weighted) empirical error is *not a weak learner* (see definition 7.1)! Consider, for example, the simple XOR example with four data points lying in $\mathbb{R}^2$ (see figure 6.3a), where points in the second and fourth quadrants are labeled positively and those in the first and third quadrants negatively. Then, no decision stump can achieve an accuracy better than $\frac{1}{2}$.

7.3 Theoretical results

In this section we present a theoretical analysis of the generalization properties of AdaBoost.

7.3.1 VC-dimension-based analysis

We start with an analysis of AdaBoost based on the VC-dimension of its hypothesis set. The family of functions $\mathcal{F}_T$ out of which AdaBoost selects its output after T rounds of boosting is

$$\mathcal{F}_T = \left\{ \operatorname{sgn}\left(\sum_{t=1}^{T} \alpha_t h_t \right) : \alpha_t \geq 0, h_t \in \mathcal{H}, t \in [T] \right\}. \tag{7.8}$$

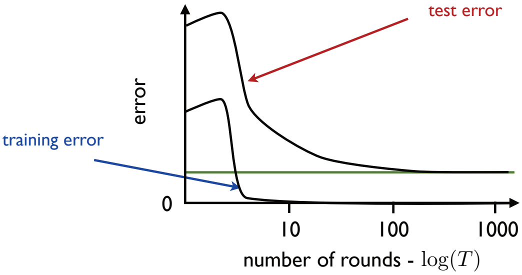


Figure 7.5
An empirical result using AdaBoost with C4.5 decision trees as base learners. In this example, the training error goes to zero after about 5 rounds of boosting ($T \approx 5$), yet the test error continues to decrease for larger values of T.

The VC-dimension of $\mathcal{F}_T$ can be bounded as follows in terms of the VC-dimension d of the family of base hypothesis $\mathcal{H}$ (exercise 7.1):

$$\mathrm{VCdim}(\mathcal{F}_T) \leq 2(d+1)(T+1)\log_2((T+1)e)\,. \tag{7.9}$$

The upper bound grows as $O(dT\log T)$, thus, the bound suggests that AdaBoost could overfit for large values of T, and indeed this can occur. However, in many cases, it has been observed empirically that the generalization error of AdaBoost decreases as a function of the number of rounds of boosting T, as illustrated in figure 7.5! How can these empirical results be explained? The following sections present a margin-based analysis in support of AdaBoost that can serve as a theoretical explanation for these empirical observations.

7.3.2 L_1-geometric margin

In chapter 5, we introduced the definition of confidence margin and presented a series of general learning bounds based on that notion which, in particular, provided strong learning guarantees for SVMs. Here, we will similarly derive general learning bounds based on that same notion of confidence margin for ensemble methods, which we will use, in particular, to derive learning guarantees for AdaBoost.

Recall that the confidence margin of a real-valued function f at a point x labeled with y is the quantity $yf(x)$. For SVMs, we also defined the notion of geometric margin which, in the separable case, is a lower bound on the confidence margin of a linear hypothesis with a normalized weighted vector $\mathbf{w}$, $\|\mathbf{w}\|_2 = 1$. Here, we will also define a notion of geometric margin for linear hypotheses with a norm-1 constraint, such as the ensemble hypotheses returned by AdaBoost, and similarly relate that notion to that of confidence margin. This will also serve as an opportunity for us to point out the connection between several concepts and terminology used in the context of SVMs and those used in the context of boosting.

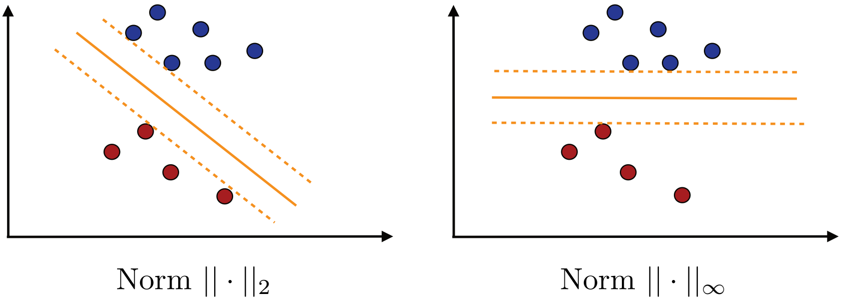


Figure 7.6
Maximum margin hyperplanes for norm-2 and norm-∞.

First note that a function $f = \sum_{t=1}^T \alpha_t h_t$ that is a linear combination of base hypotheses $h_1, \ldots, h_T$ can be equivalently expressed as an inner product $f = \boldsymbol{\alpha} \cdot \mathbf{h}$, where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_T)^\top$ and $\mathbf{h} = [h_1, \ldots, h_T]^\top$. This makes the similarity between the linear hypotheses considered in this chapter and those of chapter 5 and chapter 6 evident: the vector of base hypothesis values $\mathbf{h}(x)$ can be viewed as a feature vector associated to x, which was denoted by $\mathbf{\Phi}(x)$ in previous chapters, and $\boldsymbol{\alpha}$ is the weight vector that was denoted by $\mathbf{w}$. For ensemble linear combinations such as those returned by AdaBoost, additionally, the weight vector is non-negative: $\boldsymbol{\alpha} \geq 0$. Next, we introduce a notion of geometric margin for such ensemble functions which differs from the one introduced for SVMs only by the norm-1 used instead of norm-2, using the notation just introduced.

Definition 7.3 (L_1-geometric margin) *The* L_1-geometric margin $\rho_f(x)$ *of a linear function* $f = \sum_{t=1}^T \alpha_t h_t$ *with* $\boldsymbol{\alpha} \neq 0$ *at a point* $x \in \mathcal{X}$ *is defined by*

$$\rho_f(x) = \frac{|f(x)|}{\|\boldsymbol{\alpha}\|_1} = \frac{|\sum_{t=1}^T \alpha_t h_t(x)|}{\|\boldsymbol{\alpha}\|_1} = \frac{\big|\boldsymbol{\alpha} \cdot \mathbf{h}(x)\big|}{\|\boldsymbol{\alpha}\|_1}. \tag{7.10}$$

The L_1-margin *of* f *over a sample* $S = (x_1, \ldots, x_m)$ *is its minimum margin at the points in that sample:*

$$\rho_f = \min_{i \in [m]} \rho_f(x_i) = \min_{i \in [m]} \frac{\big|\boldsymbol{\alpha} \cdot \mathbf{h}(x_i)\big|}{\|\boldsymbol{\alpha}\|_1}. \tag{7.11}$$

This definition of geometric margin differs from definition 5.1 given in the context of the SVM algorithm only by the norm used for the weight vector: L_1-norm here, L_2-norm in definition 5.1. To distinguish them in the discussion that follows, let $\rho_1(x)$ denote the L_1-margin and $\rho_2(x)$ the L_2-margin at point x (definition 5.1):

$$\rho_1(x) = \frac{|\boldsymbol{\alpha} \cdot \mathbf{h}(x)|}{\|\boldsymbol{\alpha}\|_1} \quad \text{and} \quad \rho_2(x) = \frac{|\boldsymbol{\alpha} \cdot \mathbf{h}(x)|}{\|\boldsymbol{\alpha}\|_2}.$$

$\rho_2(x)$ is then the norm-2 distance of the vector $\mathbf{h}(x)$ to the hyperplane of equation $\boldsymbol{\alpha} \cdot \mathbf{x} = 0$ in $\mathbb{R}^T$. Similarly, $\rho_1(x)$ is the norm-∞ distance of $\mathbf{h}(x)$ to that hyperplane. This geometric difference is illustrated by figure 7.6.[8]

We will denote by

$$\bar{f} = \frac{f}{\sum_{t=1}^{T} \alpha_t} = \frac{f}{\|\boldsymbol{\alpha}\|_1}$$

the normalized version of the function f returned by AdaBoost. Note that if a point x with label y is correctly classified by f (or $\bar{f}$), then the confidence margin of $\bar{f}$ at x coincides with the L_1-geometric margin of f: $y\bar{f}(x) = \frac{yf(x)}{\|\boldsymbol{\alpha}\|_1} = \rho_f(x)$. Observe that, since the coefficients α_t are non-negative, $\rho_f(x)$ is then a convex combination of the base hypothesis values $h_t(x)$. In particular, if the base hypotheses h_t take values in $[-1, +1]$, then $\rho_f(x)$ is in $[-1, +1]$.

7.3.3 Margin-based analysis

To analyze the generalization properties of AdaBoost, we start by examining the Rademacher complexity of convex linear ensembles. For any hypothesis set $\mathcal{H}$ of real-valued functions, we denote by $\operatorname{conv}(\mathcal{H})$ its convex hull defined by

$$\operatorname{conv}(\mathcal{H}) = \left\{ \sum_{k=1}^{p} \mu_k h_k \colon p \geq 1, \forall k \in [p], \mu_k \geq 0, h_k \in \mathcal{H}, \sum_{k=1}^{p} \mu_k \leq 1 \right\}. \tag{7.12}$$

The following lemma shows that, remarkably, the empirical Rademacher complexity of $\operatorname{conv}(\mathcal{H})$, which in general is a strictly larger set including $\mathcal{H}$, coincides with that of $\mathcal{H}$.

Lemma 7.4 *Let $\mathcal{H}$ be a set of functions mapping from $\mathcal{X}$ to $\mathbb{R}$. Then, for any sample S, we have*

$$\widehat{\mathfrak{R}}_S\big(\operatorname{conv}(\mathcal{H})\big) = \widehat{\mathfrak{R}}_S(\mathcal{H}).$$

[8] More generally, for $p, q \geq 1$, p and q *conjugate*, that is $\frac{1}{p} + \frac{1}{q} = 1$, $\frac{|\boldsymbol{\alpha} \cdot \mathbf{h}(x)|}{\|\boldsymbol{\alpha}\|_p}$ is the norm-q distance of $\mathbf{h}(x)$ to the hyperplane of equation $\boldsymbol{\alpha} \cdot \mathbf{h}(x) = 0$.

Proof: The proof follows from a straightforward series of equalities:

$$\begin{aligned}\widehat{\mathfrak{R}}_S\big(\operatorname{conv}(\mathcal{H})\big) &= \frac{1}{m}\mathop{\mathbb{E}}_{\sigma}\left[\sup_{h_1,\ldots,h_p\in\mathcal{H},\boldsymbol{\mu}\geq 0,\|\boldsymbol{\mu}\|_1\leq 1}\sum_{i=1}^{m}\sigma_i\sum_{k=1}^{p}\mu_k h_k(x_i)\right]\\ &= \frac{1}{m}\mathop{\mathbb{E}}_{\sigma}\left[\sup_{h_1,\ldots,h_p\in\mathcal{H}}\sup_{\boldsymbol{\mu}\geq 0,\|\boldsymbol{\mu}\|_1\leq 1}\sum_{k=1}^{p}\mu_k\sum_{i=1}^{m}\sigma_i h_k(x_i)\right]\\ &= \frac{1}{m}\mathop{\mathbb{E}}_{\sigma}\left[\sup_{h_1,\ldots,h_p\in\mathcal{H}}\max_{k\in[p]}\sum_{i=1}^{m}\sigma_i h_k(x_i)\right]\\ &= \frac{1}{m}\mathop{\mathbb{E}}_{\sigma}\left[\sup_{h\in\mathcal{H}}\sum_{i=1}^{m}\sigma_i h(x_i)\right] = \widehat{\mathfrak{R}}_S(\mathcal{H}),\end{aligned}$$

where the third equality follows the definition of the dual norm (see section A.1.2) or the observation that the maximizing vector $\boldsymbol{\mu}$ for a convex combination of p terms is the one placing all the weight on the largest term. □

This theorem can be used directly in combination with theorem 5.8 to derive the following Rademacher complexity generalization bound for convex combination ensembles of hypotheses. Recall that $\widehat{R}_{S,\rho}(h)$ denotes the empirical margin loss with margin ρ.

Corollary 7.5 (Ensemble Rademacher margin bound) *Let $\mathcal{H}$ denote a set of real-valued functions. Fix $\rho > 0$. Then, for any $\delta > 0$, with probability at least $1-\delta$, each of the following holds for all $h \in \operatorname{conv}(\mathcal{H})$:*

$$R(h) \leq \widehat{R}_{S,\rho}(h) + \frac{2}{\rho}\mathfrak{R}_m\big(\mathcal{H}\big) + \sqrt{\frac{\log\frac{1}{\delta}}{2m}} \tag{7.13}$$

$$R(h) \leq \widehat{R}_{S,\rho}(h) + \frac{2}{\rho}\widehat{\mathfrak{R}}_S\big(\mathcal{H}\big) + 3\sqrt{\frac{\log\frac{2}{\delta}}{2m}}\,. \tag{7.14}$$

Using corollary 3.8 and corollary 3.18 to bound the Rademacher complexity in terms of the VC-dimension yields immediately the following VC-dimension-based generalization bounds for convex combination ensembles of hypotheses.

Corollary 7.6 (Ensemble VC-Dimension margin bound) *Let $\mathcal{H}$ be a family of functions taking values in $\{+1,-1\}$ with VC-dimension d. Fix $\rho > 0$. Then, for any $\delta > 0$, with probability at least $1-\delta$, the following holds for all $h \in \operatorname{conv}(\mathcal{H})$:*

$$R(h) \leq \widehat{R}_{S,\rho}(h) + \frac{2}{\rho}\sqrt{\frac{2d\log\frac{em}{d}}{m}} + \sqrt{\frac{\log\frac{1}{\delta}}{2m}}\,. \tag{7.15}$$

These bounds can be generalized to hold uniformly for all $\rho \in (0,1]$, at the price of an additional term of the form $\sqrt{\frac{\log\log_2\frac{2}{\delta}}{m}}$ as in theorem 5.9. They cannot be directly applied to the function f returned by AdaBoost, since it is not a convex combination of base hypotheses, but they can be applied to its normalized version, $\bar{f} = \frac{\sum_{t=1}^T \alpha_t h_t}{\|\boldsymbol{\alpha}\|_1} \in \operatorname{conv}(\mathcal{H})$. Notice that from the point of view of binary classification, f and $\bar{f}$ are equivalent since $\operatorname{sgn}(f) = \operatorname{sgn}\big(\frac{f}{\|\boldsymbol{\alpha}\|_1}\big)$, thus $R(f) = R\big(\frac{f}{\|\boldsymbol{\alpha}\|_1}\big)$, but their empirical margin losses are distinct.

Let $f = \sum_{t=1}^T \alpha_t h_t$ denote the function defining the classifier returned by AdaBoost after T rounds of boosting when trained on sample S. Then, in view of (7.13), for any $\delta > 0$, the following holds with probability at least $1 - \delta$:

$$R(f) \leq \widehat{R}_{S,\rho}\left(\bar{f}\right) + \frac{2}{\rho}\mathfrak{R}_m(\mathcal{H}) + \sqrt{\frac{\log \frac{1}{\delta}}{2m}}. \tag{7.16}$$

Similar bounds can be derived from (7.14) and (7.15). Remarkably, the number of rounds of boosting T does not appear in the generalization bound (7.16). The bound depends only on the confidence margin ρ, the sample size m, and the Rademacher complexity of the family of base classifiers $\mathcal{H}$. Thus, the bound guarantees an effective generalization if the margin loss $R_\rho\big(\bar{f}\big)$ is small for a relatively large ρ. Recall that the margin loss can be upper bounded by the fraction of the points x labeled with y in the training sample with confidence margin at most ρ, that is $\frac{yf(x)}{\|\boldsymbol{\alpha}\|_1} \leq \rho$ (see (5.38)). With our definition of L_1-margin, this can also be written as follows:

$$\widehat{R}_{S,\rho}\left(\bar{f}\right) \leq \frac{|\{i \in [m] \colon y_i\, \rho_f(x_i) \leq \rho\}|}{m}. \tag{7.17}$$

Additionally, the following theorem provides a bound on the empirical margin loss, which decreases with T under conditions discussed later.

Theorem 7.7 *Let $f = \sum_{t=1}^T \alpha_t h_t$ denote the function returned by AdaBoost after T rounds of boosting and assume for all $t \in [T]$ that $\epsilon_t < \frac{1}{2}$, which implies $\alpha_t > 0$. Then, for any $\rho > 0$, the following holds:*

$$\widehat{R}_{S,\rho}(\bar{f}) \leq 2^T \prod_{t=1}^T \sqrt{\epsilon_t^{1-\rho}(1-\epsilon_t)^{1+\rho}}\,.$$

Proof: Using the general inequality $1_{u\leq 0} \leq \exp(-u)$ valid for all $u \in \mathbb{R}$, identity 7.2, that is $\mathcal{D}_{t+1}(i) = \frac{e^{-y_i f(x_i)}}{m\prod_{t=1}^T Z_t}$, the equality $Z_t = 2\sqrt{\epsilon_t(1-\epsilon_t)}$ from the proof of theorem 7.2, and the definition of $\alpha_t = \frac{1}{2}\log(\frac{1-\epsilon_t}{\epsilon_t})$ in AdaBoost, we can write:

$$\begin{aligned}
\frac{1}{m}\sum_{i=1}^{m} 1_{y_i f(x_i)-\rho\|\boldsymbol{\alpha}\|_1 \leq 0} &\leq \frac{1}{m}\sum_{i=1}^{m} \exp(-y_i f(x_i) + \rho\|\boldsymbol{\alpha}\|_1) \\
&= \frac{1}{m}\sum_{i=1}^{m} e^{\rho\|\boldsymbol{\alpha}\|_1}\left[m\prod_{t=1}^{T} Z_t\right]\mathcal{D}_{T+1}(i) \\
&= e^{\rho\|\boldsymbol{\alpha}\|_1}\prod_{t=1}^{T} Z_t = e^{\rho\sum_{t'}\alpha_{t'}}\prod_{t=1}^{T} Z_t \\
&= 2^T\prod_{t=1}^{T}\left[\sqrt{\tfrac{1-\epsilon_t}{\epsilon_t}}\right]^{\rho}\sqrt{\epsilon_t(1-\epsilon_t)}\,,
\end{aligned}$$

which concludes the proof. □

Moreover, if for all $t \in [T]$ we have $\gamma \leq (\frac{1}{2} - \epsilon_t)$ and $\rho \leq 2\gamma$, then the expression $4\epsilon_t^{1-\rho}(1-\epsilon_t)^{1+\rho}$ is maximized at $\epsilon_t = \frac{1}{2} - \gamma$.[9] Thus, the upper bound on the empirical margin loss can then be bounded by

$$\widehat{R}_{S,\rho}(\bar{f}) \leq \left[(1-2\gamma)^{1-\rho}(1+2\gamma)^{1+\rho}\right]^{\frac{T}{2}}. \tag{7.18}$$

Observe that $(1-2\gamma)^{1-\rho}(1+2\gamma)^{1+\rho} = (1-4\gamma^2)\big(\frac{1+2\gamma}{1-2\gamma}\big)^{\rho}$. This is an increasing function of ρ since we have $\big(\frac{1+2\gamma}{1-2\gamma}\big) > 1$ as a consequence of $\gamma > 0$. Thus, if $\rho < \gamma$, it can be strictly upper bounded as follows

$$(1-2\gamma)^{1-\rho}(1+2\gamma)^{1+\rho} < (1-2\gamma)^{1-\gamma}(1+2\gamma)^{1+\gamma}.$$

The function $\gamma \mapsto (1-2\gamma)^{1-\gamma}(1+2\gamma)^{1+\gamma}$ is strictly upper bounded by 1 over the interval $(0, 1/2)$, thus, if $\rho < \gamma$, then $(1-2\gamma)^{1-\rho}(1+2\gamma)^{1+\rho} < 1$ and the right-hand side of (7.18) decreases exponentially with T. Since the condition $\rho \gg O(1/\sqrt{m})$ is necessary in order for the given margin bounds to converge, this places a condition of $\gamma \gg O(1/\sqrt{m})$ on the edge value. In practice, the error ϵ_t of the base classifier at round t may increase as a function of t. Informally, this is because boosting presses the weak learner to concentrate on instances that are harder and harder to classify, for which even the best base classifier could not achieve an error significantly better than random. If ϵ_t becomes close to $\frac{1}{2}$ relatively fast as a function of t, then the bound of theorem 7.7 becomes uninformative.

[9] The differential of $f\colon \epsilon \mapsto \log[\epsilon^{1-\rho}(1-\epsilon)^{1+\rho}] = (1-\rho)\log\epsilon + (1+\rho)\log(1-\epsilon)$ over the interval $(0,1)$ is given by $f'(\epsilon) = \frac{1-\rho}{\epsilon} - \frac{1+\rho}{1-\epsilon} = 2\frac{(\frac{1}{2}-\frac{\rho}{2})-\epsilon}{\epsilon(1-e)}$. Thus, f is an increasing function over $(0, \frac{1}{2}-\frac{\rho}{2})$, which implies that it is increasing over $(0, \frac{1}{2}-\gamma)$ when $\gamma \geq \frac{\rho}{2}$.

The analysis and discussion that precede show that if AdaBoost admits a positive edge ($\gamma > 0$), then, for $\rho < \gamma$, the empirical margin loss $\widehat{R}_{S,\rho}(\bar{f})$ becomes zero for T sufficiently large (it decreases exponentially fast). Thus, AdaBoost achieves an L_1-geometric margin of γ over the training sample. In section 7.3.5, we will see that the edge γ is positive if and only if the training sample is separable. In that case, the edge can be chosen to be as large as half the maximum L_1-geometric margin $\rho_{\max}$ that can be achieved on the sample: $\gamma = \frac{\rho_{\max}}{2}$. Thus, for a separable data set, AdaBoost can asymptotically achieve a geometric margin that is at least half the maximum geometric margin, $\frac{\rho_{\max}}{2}$.

This analysis can serve as a theoretical explanation of the empirical observation that, in some tasks, the generalization error decreases as a function of T even after the error on the training sample is zero: the geometric margin continues to increase when the training sample is separable. In (7.16), for the ensemble function f determined by AdaBoost after T rounds, as T increases, ρ can be chosen as a larger quantity for which the first term on the right-hand side vanishes ($\widehat{R}_{S,\rho}(\bar{f}) = 0$) while the second term becomes more favorable since it decreases as $\frac{1}{\rho}$.

But, does AdaBoost achieve the maximum L_1-geometric margin $\rho_{\max}$? No. It has been shown that AdaBoost may converge, for a linearly separable sample, to a geometric margin that is significantly smaller than the maximum margin (e.g., $\frac{1}{3}$ instead of $\frac{3}{8}$).

7.3.4 Margin maximization

In view of these results, several algorithms have been devised with the explicit goal of maximizing the L_1-geometric margin. These algorithms correspond to different methods for solving a linear program (LP).

By definition of the L_1-margin, the maximum margin for a linearly separable sample $S = ((x_1, y_1), \ldots, (x_m, y_m))$ is given by

$$\rho = \max_{\boldsymbol{\alpha}} \min_{i \in [m]} \frac{y_i\big(\boldsymbol{\alpha} \cdot \mathbf{h}(x_i)\big)}{\|\boldsymbol{\alpha}\|_1}\,. \tag{7.19}$$

By definition of the maximization, the optimization problem can be written as:

$$\max_{\boldsymbol{\alpha}} \rho$$
$$\text{subject to: } \frac{y_i\big(\boldsymbol{\alpha} \cdot \mathbf{h}(x_i)\big)}{\|\boldsymbol{\alpha}\|_1} \geq \rho, \ \forall i \in [m].$$

Since $\frac{\boldsymbol{\alpha} \cdot \mathbf{h}(x_i)}{\|\boldsymbol{\alpha}\|_1}$ is invariant to the scaling of $\boldsymbol{\alpha}$, we can restrict ourselves to $\|\boldsymbol{\alpha}\|_1 = 1$. Further seeking a non-negative $\boldsymbol{\alpha}$ as in the case of AdaBoost leads to the following optimization:

$$\begin{aligned} &\max_{\boldsymbol{\alpha}} \rho \\ \text{subject to: } & y_i\big(\boldsymbol{\alpha} \cdot \mathbf{h}(x_i)\big) \geq \rho, \ \forall i \in [m]; \\ & \left(\sum_{t=1}^{T} \alpha_t = 1\right) \wedge \Big(\alpha_t \geq 0, \forall t \in [T]\Big). \end{aligned}$$

This is a linear program (LP), that is, a convex optimization problem with a linear objective function and linear constraints. There are several different methods for solving relative large LPs in practice, using the simplex method, interior-point methods, or a variety of special-purpose solutions.

Note that the solution of this algorithm differs from the margin-maximization defining SVMs in the separable case only by the definition of the geometric margin used (L_1 versus L_2) and the non-negativity constraint on the weight vector. Figure 7.6 illustrates the margin-maximizing hyperplanes found using these two distinct margin definitions in a simple case. The left figure shows the SVM solution, where the distance to the closest points to the hyperplane is measured with respect to the norm $\|\cdot\|_2$. The right figure shows the solution for the L_1-margin, where the distance to the closest points to the hyperplane is measured with respect to the norm $\|\cdot\|_\infty$.

By definition, the solution of the LP just described admits an L_1-margin that is larger or equal to that of the AdaBoost solution. However, empirical results do not show a systematic benefit for the solution of the LP. In fact, it appears that in many cases, AdaBoost outperforms that algorithm. The margin theory described does not seem sufficient to explain that performance.

7.3.5 Game-theoretic interpretation

In this section, we show that AdaBoost admits a natural game-theoretic interpretation. The application of von Neumann's theorem then helps us relate the maximum margin and the optimal edge and clarify the connection of AdaBoost's weak-learning assumption with the notion of L_1-margin. We first introduce the definition of the edge of a base classifier for a particular distribution.

Definition 7.8 *The* edge of a base classifier h_t for a distribution $\mathcal{D}$ *over the training sample* $S = ((x_1, y_1), \ldots, (x_m, y_m))$ *is defined by*

$$\gamma_t(\mathcal{D}) = \frac{1}{2} - \epsilon_t = \frac{1}{2} \sum_{i=1}^{m} y_i h_t(x_i) \mathcal{D}(i). \tag{7.20}$$

AdaBoost's weak learning condition can now be formulated as follows: there exists $\gamma > 0$ such that for any distribution $\mathcal{D}$ over the training sample and any base

Table 7.1
The loss matrix for the standard rock-paper-scissors game.

	rock	paper	scissors
rock	0	+1	-1
paper	-1	0	+1
scissors	+1	-1	0

classifier h_t, the following holds:

$$\gamma_t(\mathcal{D}) \geq \gamma. \tag{7.21}$$

This condition is required for the analysis of theorem 7.2 and the non-negativity of the coefficients α_t. We will frame boosting as a two-person zero-sum game.

Definition 7.9 (Zero-sum game) *A* finite two-person zero-sum game *consists of a* loss matrix $\mathbf{M} \in \mathbb{R}^{m \times n}$*, where m is the number of possible* actions *(or* pure strategies*) for the row player and n the number of possible actions for the column player. The entry M_{ij} is the loss for the row player (or equivalently the payoff for the column payer) when the row player takes action i and the column player takes action j.*[10]

An example of a loss matrix for the familiar "rock-paper-scissors" game is shown in table 7.1.

Definition 7.10 (Mixed strategy) *A* mixed strategy *for the row player is a distribution p over the m possible row actions; a* mixed strategy *for the column player is a distribution q over the n possible column actions. The* expected loss *for the row player (expected payoff for the column player) with respect to the mixed strategies p and q is*

$$\mathop{\mathbb{E}}_{\substack{i \sim p \\ j \sim q}}[M_{ij}] = \sum_{i=1}^{m} \sum_{j=1}^{n} p_i M_{ij} q_j = \mathbf{p}^\top \mathbf{M} \mathbf{q}.$$

The following is a fundamental result in game theory proven in chapter 8.

Theorem 7.11 (Von Neumann's minimax theorem) *For any finite two-person zero-sum game defined by the matrix $\mathbf{M}$, the following equality holds:*

$$\min_{\mathbf{p}} \max_{\mathbf{q}} \mathbf{p}^\top \mathbf{M} \mathbf{q} = \max_{\mathbf{q}} \min_{\mathbf{p}} \mathbf{p}^\top \mathbf{M} \mathbf{q}\,. \tag{7.22}$$

The common value in (7.22) is called the *value of the game.* The theorem states that for any two-person zero-sum game, there exists a mixed strategy for each player

[10] To be consistent with the results discussed in other chapters, we consider the loss matrix as opposed to the payoff matrix (its opposite).

such that the expected loss for one is the same as the expected payoff for the other, both of which are equal to the value of the game.

Note that, given the row player's strategy, the column player can choose a pure strategy optimizes their payoff. That is, the column player can choose the single strategy corresponding the largest coordinate of the vector $\mathbf{p}^\top \mathbf{M}$. A similar comment applies to the reverse. Thus, an alternative and equivalent form of the minimax theorem is

$$\min_{\mathbf{p}} \max_{j \in [n]} \mathbf{p}^\top \mathbf{M} \mathbf{e}_j = \max_{\mathbf{q}} \min_{i \in [m]} \mathbf{e}_i^\top \mathbf{M} \mathbf{q}, \tag{7.23}$$

where $\mathbf{e}_i$ denotes the ith unit vector.

We can now view AdaBoost as a zero-sum game, where an action of the row player is the selection of a training instance x_i, $i \in [m]$, and an action of the column player the selection of a base learner h_t, $t \in [T]$. A mixed strategy for the row player is thus a distribution $\mathcal{D}$ over the training points' indices $[m]$. A mixed strategy for the column player is a distribution over the based classifiers' indices $[T]$. This can be defined from a non-negative vector $\boldsymbol{\alpha} \geq 0$: the weight assigned to $t \in [T]$ is $\alpha_t / \|\boldsymbol{\alpha}\|_1$. The loss matrix $\mathbf{M} \in \{-1, +1\}^{m \times T}$ for AdaBoost is defined by $M_{it} = y_i h_t(x_i)$ for all $(i, t) \in [m] \times [T]$. By von Neumann's theorem (7.23), the following holds:

$$\min_{\mathcal{D} \in \mathscr{D}} \max_{t \in [T]} \sum_{i=1}^{m} \mathcal{D}(i) y_i h_t(x_i) = \max_{\boldsymbol{\alpha} \geq 0} \min_{i \in [m]} \sum_{t=1}^{T} \frac{\alpha_t}{\|\boldsymbol{\alpha}\|_1} y_i h_t(x_i), \tag{7.24}$$

where $\mathscr{D}$ denotes the set of all distributions over the training sample. Let $\rho_{\boldsymbol{\alpha}}(x)$ denote the margin of point x for the classifier defined by $f = \sum_{t=1}^{T} \alpha_t h_t$. The result can be rewritten as follows in terms of the margins and edges:

$$2\gamma^* = 2 \min_{\mathcal{D}} \max_{t \in [T]} \gamma_t(\mathcal{D}) = \max_{\boldsymbol{\alpha}} \min_{i \in [m]} \rho_{\boldsymbol{\alpha}}(x_i) = \rho^*, \tag{7.25}$$

where ρ^* is the maximum margin of a classifier and γ^* the best possible edge. This result has several implications. First, it shows that the weak learning condition ($\gamma^* > 0$) implies $\rho^* > 0$ and thus the existence of a classifier with positive margin, which motivates the search for a non-zero margin. AdaBoost can be viewed as an algorithm seeking to achieve such a non-zero margin, though, as discussed earlier, AdaBoost does not always achieve an optimal margin and is thus suboptimal in that respect. Furthermore, we see that the "weak learning" assumption, which originally appeared to be the weakest condition one could require for an algorithm (that of performing better than random), is in fact a strong condition: it implies that the training sample is linearly separable with margin $2\gamma^* > 0$. Linear separability often does not hold for the data sets found in practice.

7.4 L_1-regularization

In practice, the training sample may not be linearly separable and AdaBoost may not admit a positive edge, in which case the weak learning condition does not hold. It may also be that AdaBoost does admit a positive edge but with γ very small. In such cases, running AdaBoost may result in large total mixture weights for some base classifiers h_j. This can be because the algorithm increasingly concentrates on a few examples that are hard to classify and whose weights keep growing. Only a few base classifiers might achieve the best performance for those examples and the algorithm keeps selecting them, thereby increasing their total mixture weights. These base classifiers with relatively large total mixture weight end up dominating in an ensemble f and therefore solely dictating the classification decision. The performance of the resulting ensemble is typically poor since it almost entirely hinges on that of a few base classifiers.

There are several methods for avoiding such situations. One consists of limiting the number of rounds of boosting T, which is also known as *early-stopping*. Another one consists of controlling the magnitude of the mixture weights. This can be done by augmenting the objective function of AdaBoost with a regularization term based on a norm of the vector of mixture weights. Using a norm-1 regularization leads to an algorithm that we will refer to as *L_1-regularized AdaBoost*. Given a labeled sample $S = ((x_1, y_1), \ldots, (x_m, y_m))$, the objective function G minimized by L_1-regularized AdaBoost is defined for all $\bar{\boldsymbol{\alpha}} = (\bar{\alpha}_1, \ldots, \bar{\alpha}_N) \in \mathbb{R}^N$ by

$$G(\bar{\boldsymbol{\alpha}}) = \frac{1}{m}\sum_{i=1}^{m} e^{-y_i f(x_i)} + \lambda \|\bar{\boldsymbol{\alpha}}\|_1 = \frac{1}{m}\sum_{i=1}^{m} e^{-y_i \sum_{j=1}^{N} \bar{\alpha}_j h_j(x_i)} + \lambda \|\bar{\boldsymbol{\alpha}}\|_1, \qquad (7.26)$$

where, as for AdaBoost, f is an ensemble function defined by $f = \sum_{j=1}^{N} \bar{\alpha}_j h_j$, with $\bar{\alpha}_j \geq 0$. The objective function G is a convex function of $\bar{\boldsymbol{\alpha}}$ as the sum of the convex objective of AdaBoost and the norm-1 of $\bar{\boldsymbol{\alpha}}$. L_1-regularized AdaBoost consists of applying coordinate-descent to the objective function G.

We now show that the algorithm can be directed derived from the margin-based guarantee for ensemble methods of Corollary 7.5 or Corollary 7.6. Thus, in that way, L_1-regularized AdaBoost benefits from a more favorable and natural theoretical guarantee than AdaBoost.

By the generalization of Corollary 7.5 to a uniform convergence bound over ρ, for any $\delta > 0$, with probability at least $1 - \delta$, the following holds for all ensemble functions $f = \sum_{j=1}^{N} \bar{\alpha}_j h_j$ with $\|\bar{\boldsymbol{\alpha}}\|_1 \leq 1$ and all $\rho \in (0, 1]$:

$$R(f) \leq \frac{1}{m}\sum_{i=1}^{m} 1_{f(x_i) \leq \rho} + \frac{2}{\rho}\mathfrak{R}_m(\mathcal{H}) + \sqrt{\frac{\log \log_2 \frac{2}{\rho}}{m}} + \sqrt{\frac{\log \frac{2}{\delta}}{2m}}. \qquad (7.27)$$

The inequality also trivially holds for $\rho > 1$ since, in that case, the first term on the right-hand side of the bound is equal to one. Indeed, in that case, by Hölder's inequality, for any $x \in \mathcal{X}$, we have $f(x) = \sum_{j=1}^{N} \bar{\alpha}_i h_j(x) \leq \|\bar{\boldsymbol{\alpha}}\|_1 \max_{j \in [N]} |h_j(x)| \leq \|\bar{\boldsymbol{\alpha}}\|_1 \leq 1 < \rho$.

Now, in view of the general upper bound $1_{u \leq 0} \leq e^{-u}$ valid for all $u \in \mathbb{R}$, with probability at least $1 - \delta$, the following holds for all $f = \sum_{j=1}^{N} \bar{\alpha}_j h_j$ with $\|\bar{\boldsymbol{\alpha}}\|_1 \leq 1$ and all $\rho > 0$:

$$R(f) \leq \frac{1}{m} \sum_{i=1}^{m} e^{1 - \frac{f(x_i)}{\rho}} + \frac{2}{\rho} \mathfrak{R}_m(\mathcal{H}) + \sqrt{\frac{\log \log_2 \frac{2}{\rho}}{m}} + \sqrt{\frac{\log \frac{2}{\delta}}{2m}}. \tag{7.28}$$

Since for any $\rho > 0$, f/ρ admits the same generalization error as f, with probability at least $1 - \rho$, the following inequality holds for all $f = \sum_{j=1}^{N} \bar{\alpha}_j h_j$ with $\|\bar{\boldsymbol{\alpha}}\|_1 \leq 1/\rho$ and all $\rho > 0$:

$$R(f) \leq \frac{1}{m} \sum_{i=1}^{m} e^{1 - f(x_i)} + \frac{2}{\rho} \mathfrak{R}_m(\mathcal{H}) + \sqrt{\frac{\log \log_2 \frac{2}{\rho}}{m}} + \sqrt{\frac{\log \frac{2}{\delta}}{2m}}. \tag{7.29}$$

This inequality can be used to derive an algorithm that selects $\bar{\boldsymbol{\alpha}}$ and $\rho > 0$ to minimize the right-hand side. The minimization with respect to ρ does not lead to a convex optimization and depends on theoretical constant factors affecting the second and third terms, which may not be optimal. Thus, instead, ρ is left as a free parameter of the algorithm, typically determined via cross-validation.

Now, since only the first term of the right-hand side depends on $\bar{\boldsymbol{\alpha}}$, the bound suggests selecting $\bar{\boldsymbol{\alpha}}$ as the solution of the following optimization problem:

$$\min_{\|\bar{\boldsymbol{\alpha}}\|_1 \leq \frac{1}{\rho}} \frac{1}{m} \sum_{i=1}^{m} e^{-f(x_i)} = \frac{1}{m} \sum_{i=1}^{m} e^{-\sum_{j=1}^{N} \bar{\alpha}_j h_j(x_i)}. \tag{7.30}$$

Introducing a Lagrange variable $\lambda \geq 0$, the optimization problem can be written equivalently as

$$\min_{\|\bar{\boldsymbol{\alpha}}\|_1 \leq \frac{1}{\rho}} \frac{1}{m} \sum_{i=1}^{m} e^{-\sum_{j=1}^{N} \bar{\alpha}_j h_j(x_i)} + \lambda \|\bar{\boldsymbol{\alpha}}\|_1. \tag{7.31}$$

Since for any choice of ρ in the constraint of (7.30) there exists an equivalent dual variable λ in the formulation (7.31) that achieves the same optimal $\bar{\boldsymbol{\alpha}}$, $\lambda \geq 0$ can be freely selected via cross-validation. The resulting objective function therefore precisely coincides with that of L_1-regularized AdaBoost.

7.5 Discussion

AdaBoost offers several advantages: it is simple, its implementation is straightforward, and the time complexity of each round of boosting as a function of the sample size is rather favorable. As already discussed, when using decision stumps, the time complexity of each round of boosting is in $O(mN)$. Of course, if the dimension of the feature space N is very large, then the algorithm could become in fact quite slow.

AdaBoost additionally benefits from a rich theoretical analysis. Nevertheless, there are still many theoretical questions related to the algorithm. For example, as we saw, the algorithm in fact does not maximize the margin, and yet algorithms that do maximize the margin do not always outperform it. This suggests that perhaps a finer analysis based on a notion different from that of minimal margin could shed more light on the properties of the algorithm.

A minor drawback of the algorithm is the need to select the parameter T and the base classifier set. The choice of the number of rounds of boosting T (stopping criterion) is crucial to the performance of the algorithm. As suggested by the VC-dimension analysis, larger values of T can lead to overfitting. In practice, T is typically determined via cross-validation. The choice of the base classifiers is also crucial. The complexity of the family of base classifiers $\mathcal{H}$ appeared in all the bounds presented and it is important to control it in order to guarantee generalization. On the other hand, insufficiently complex hypothesis sets could lead to low margins.

Probably the most serious disadvantage of AdaBoost is its performance in the presence of noise, at least in some tasks. The distribution weight assigned to examples that are harder to classify substantially increases with the number of rounds of boosting, by the nature of the algorithm. These examples may end up dominating the selection of the base classifiers, which, with a large enough number of rounds, will play a detrimental role in the definition of the linear combination defined by AdaBoost. Several solutions have been proposed to address these issues. One consists of using a "less aggressive" objective function than the exponential function of AdaBoost, such as the logistic loss, to penalize less incorrectly classified points. Another solution is based on a regularization, e.g., the L_1-regularized AdaBoost described in the previous section.

An empirical study of AdaBoost has shown that uniform noise severely damages its accuracy. This has also been corroborated by recent theoretical results showing that boosting algorithms based on convex potentials do not tolerate even low levels of random noise. Moreover, these issues have been shown to persist even when using L_1-regularization or early stopping. However, the uniform noise model used in those experiments or analysis is rather unrealistic and seems unlikely to appear in practice. The model assumes that a label corruption with some fixed probability affects all instances uniformly. Clearly, the performance of any algorithm should degrade in the presence of such noise. Empirical results suggest, however, that the performance of AdaBoost tends to degrade more than that of other algorithms for this uniform noise model.

Finally, notice that the behavior of AdaBoost in the presence of noise can be used, in fact, as a useful feature for detecting *outliers*, that is, examples that are incorrectly labeled or that are hard to classify. Examples with large weights after a certain number of rounds of boosting can be identified as outliers.

7.6 Chapter notes

The question of whether a weak learning algorithm could be *boosted* to derive a strong learning algorithm was first posed by Kearns and Valiant [1988, 1994], who also gave a negative proof of this result for a distribution-dependent setting. The first positive proof of this result in a distribution-independent setting was given by Schapire [1990], and later by Freund [1990].

These early boosting algorithms, boosting by filtering [Schapire, 1990] or boosting by majority [Freund, 1990, 1995] were not practical. The AdaBoost algorithm introduced by Freund and Schapire [1997] solved several of these practical issues. Freund and Schapire [1997] further gave a detailed presentation and analysis of the algorithm including the bound on its empirical error, a VC-dimension analysis, and its applications to multi-class classification and regression.

Early experiments with AdaBoost were carried out by Drucker, Schapire, and Simard [1993], who gave the first implementation in OCR with weak learners based on neural networks and Drucker and Cortes [1995], who reported the empirical performance of AdaBoost combined with decision trees, in particular decision stumps.

The fact that AdaBoost coincides with coordinate descent applied to an exponential objective function was later shown by Duffy and Helmbold [1999], Mason et al. [1999], and Friedman [2000]. Friedman, Hastie, and Tibshirani [2000] also gave an interpretation of boosting in terms of additive models. They also pointed out the close connections between AdaBoost and logistic regression, in particular the fact that their objective functions have a similar behavior near zero or the fact that their expectation admit the same minimizer, and derived an alternative boosting algorithm, LogitBoost, based on the logistic loss. Lafferty [1999] showed how an incremental family of algorithms, including LogitBoost, can be derived from Bregman divergences and designed to closely approximate AdaBoost when varying a parameter. Kivinen and Warmuth [1999] gave an equivalent view of AdaBoost as an entropy projection. They showed that the distribution over the sample found by Adaboost at each round is approximately the solution to the problem of finding the closest distribution to the one at the previous round, subject to the constraint that it be orthogonal to the vector of errors of the current base hypotheses. Here, closeness is measured by a Bregman divergence, which, for AdaBoost is the unnormalized relative entropy. Collins, Schapire, and Singer [2002] later showed that boosting and logistic regression were special instances of a common framework based on Bregman divergences and used that to give the first convergence proof of AdaBoost. Another direct relationship between AdaBoost and logistic regression is given by Lebanon and Lafferty [2001] who showed that the two algorithms minimize the same extended relative entropy objective function subject to the same feature constraints, except from an additional normalization constraint for logistic regression.

A margin-based analysis of AdaBoost was first presented by Schapire, Freund, Bartlett, and Lee [1997], including theorem 7.7 which gives a bound on the empirical margin loss. Our presentation is based on the elegant derivation of margin bounds by Koltchinskii and Panchenko [2002] using the notion of Rademacher complexity. Rudin et al. [2004] gave an example showing that, in general, AdaBoost does not maximize the L_1-margin. Rätsch and Warmuth [2002] provided asymptotic lower bounds for the margin achieved by AdaBoost under some conditions. The L_1-margin maximization based on an LP is due to Grove and Schuurmans [1998]. Rätsch, Onoda, and Müller [2001] suggested a modification of that algorithm using a soft-margin instead and pointed out its connections with SVMs. The game-theoretic interpretation of boosting and the application of von Neumann's minimax theorem [von Neumann, 1928] in that context were pointed out by Freund and Schapire [1996, 1999b]; see also Grove and Schuurmans [1998] and Breiman [1999].

The L_1-regularized AdaBoost algorithm described in Section 7.4 is presented and analyzed by Rätsch, Mika, and Warmuth [2001]. Cortes, Mohri, and Syed [2014] introduced a new boosting algorithm, *DeepBoost*, which they proved to benefit from finer learning guarantees, including favorable ones even when using as base classifier set relatively rich families, for example a family of very deep decision trees, or other similarly complex families. In DeepBoost, the decisions in each iteration of which classifier to add to the ensemble and which weight to assign to that classifier, depend on the (data-dependent) complexity of the sub-family to which the classifier belongs. Cortes, Mohri, and Syed [2014] further showed that empirically DeepBoost achieves a better performance than AdaBoost, Logistic Regression, and their L_1-regularized variants. Both AdaBoost and L_1-regularized AdaBoost can be viewed as special instances of DeepBoost.

Dietterich [2000] provided extensive empirical evidence for the fact that uniform noise can severely damage the accuracy of AdaBoost. This has been reported by a number of other authors since then. Long and Servedio [2010] further recently showed the failure of boosting algorithms based on convex potentials to tolerate random noise, even with L_1-regularization or early stopping.

There are several excellent surveys and tutorials related to boosting [Schapire, 2003, Meir and Rätsch, 2002, Meir and Rätsch, 2003], including the recent book of Schapire and Freund [2012] fully dedicated to this topic, with an extensive list of references and a detailed presentation.

7.7 Exercises

7.1 VC-dimension of the hypothesis set of AdaBoost. Prove the upper bound on the VC-dimension of the hypothesis set $\mathcal{F}_T$ of AdaBoost after T rounds of boosting, as stated in equation (7.9).

7.2 Alternative objective functions. This problem studies boosting-type algorithms defined with objective functions different from that of AdaBoost. We assume that the training data are given as m labeled examples $(x_1, y_1), \ldots, (x_m, y_m) \in \mathcal{X} \times \{-1, +1\}$. We further assume that Φ is a strictly increasing convex and differentiable function over $\mathbb{R}$ such that: $\forall x \geq 0, \Phi(x) \geq 1$ and $\forall x < 0, \Phi(x) > 0$.

(a) Consider the loss function $L(\alpha) = \sum_{i=1}^m \Phi(-y_i f(x_i))$ where f is a linear combination of base classifiers, i.e., $f = \sum_{t=1}^T \alpha_t h_t$ (as in AdaBoost). Derive a new boosting algorithm using the objective function L. In particular, characterize the best base classifier h_u to select at each round of boosting if we use coordinate descent.

(b) Consider the following functions: (1) zero-one loss $\Phi_1(-u) = 1_{u \leq 0}$; (2) least squared loss $\Phi_2(-u) = (1-u)^2$; (3) SVM loss $\Phi_3(-u) = \max\{0, 1-u\}$; and (4) logistic loss $\Phi_4(-u) = \log(1 + e^{-u})$. Which functions satisfy the assumptions on Φ stated earlier in this problem?

(c) For each loss function satisfying these assumptions, derive the corresponding boosting algorithm. How do the algorithm(s) differ from AdaBoost?

7.3 Update guarantee. Assume that the main weak learner assumption of AdaBoost holds. Let h_t be the base learner selected at round t. Show that the base learner h_{t+1} selected at round $t+1$ must be different from h_t.

7.4 Weighted instances. Let the training sample be $S = ((x_1, y_1), \ldots, (x_m, y_m))$. Suppose we wish to penalize differently errors made on x_i versus x_j. To do that, we associate some non-negative importance weight w_i to each point x_i and define the objective function $F(\boldsymbol{\alpha}) = \sum_{i=1}^{m} w_i e^{-y_i f(x_i)}$, where $f = \sum_{t=1}^{T} \alpha_t h_t$. Show that this function is convex and differentiable and use it to derive a boosting-type algorithm.

7.5 Define the unnormalized correlation of two vectors $\mathbf{x}$ and $\mathbf{x}'$ as the inner product between these vectors. Prove that the distribution vector $(\mathcal{D}_{t+1}(1), \ldots, \mathcal{D}_{t+1}(m))$ defined by AdaBoost and the vector of components $y_i h_t(x_i)$ are uncorrelated.

7.6 Fix $\epsilon \in (0, 1/2)$. Let the training sample be defined by m points in the plane with $\frac{m}{4}$ negative points all at coordinate $(1, 1)$, another $\frac{m}{4}$ negative points all at coordinate $(-1, -1)$, $\frac{m(1-\epsilon)}{4}$ positive points all at coordinate $(1, -1)$, and $\frac{m(1+\epsilon)}{4}$ positive points all at coordinate $(-1, +1)$. Describe the behavior of AdaBoost when run on this sample using boosting stumps. What solution does the algorithm return after T rounds?

7.7 Noise-tolerant AdaBoost. AdaBoost may be significantly overfitting in the presence of noise, in part due to the high penalization of misclassified examples. To reduce this effect, one could use instead the following objective function:

$$F = \sum_{i=1}^{m} G(-y_i f(x_i)), \tag{7.32}$$

where G is the function defined on $\mathbb{R}$ by

$$G(x) = \begin{cases} e^x & \text{if } x \leq 0 \\ x + 1 & \text{otherwise.} \end{cases} \tag{7.33}$$

(a) Show that the function G is convex and differentiable.

(b) Use F and greedy coordinate descent to derive an algorithm similar to AdaBoost.

(c) Compare the reduction of the empirical error rate of this algorithm with that of AdaBoost.

7.8 Simplified AdaBoost. Suppose we simplify AdaBoost by setting the parameter α_t to a fixed value $\alpha_t = \alpha > 0$, independent of the boosting round t.

(a) Let γ be such that $(\frac{1}{2} - \epsilon_t) \geq \gamma > 0$. Find the best value of α as a function of γ by analyzing the empirical error.

(b) For this value of α, does the algorithm assign the same probability mass to correctly classified and misclassified examples at each round? If not, which set is assigned a higher probability mass?

```
AdaBoost($\mathbf{M}, t_{\max}$)
$\lambda_{1,j} \leftarrow 0$ for $i = 1, \ldots, m$
for $t \leftarrow 1$ to $t_{\max}$ do
    $d_{t,i} \leftarrow \frac{\exp(-(\mathbf{M}\boldsymbol{\lambda}_t)_i)}{\sum_{k=1}^{m} \exp(-(\mathbf{M}\boldsymbol{\lambda}_t)_k)}$ for $i = 1, \ldots, m$
    $j_t \leftarrow \operatorname{argmax}_j (\mathbf{d}_t^\top \mathbf{M})_j$
    $r_t \leftarrow (\mathbf{d}_t^\top \mathbf{M})_{j_t}$
    $\alpha_t \leftarrow \frac{1}{2} \log \left(\frac{1+r_t}{1-r_t}\right)$
    $\boldsymbol{\lambda}_{t+1} \leftarrow \boldsymbol{\lambda}_t + \alpha_t \mathbf{e}_{j_t}$, where $\mathbf{e}_{j_t}$ is 1 in position $j_t$ and 0 elsewhere.
return $\frac{\boldsymbol{\lambda}_{t_{\max}}}{\|\boldsymbol{\lambda}_{t_{\max}}\|_1}$
```

Figure 7.7
AdaBoost defined with respect to a matrix $\mathbf{M}$, which encodes the accuracy of each weak classifier on each training point.

(c) Using the previous value of α, give a bound on the empirical error of the algorithm that depends only on γ and the number of rounds of boosting T.

(d) Using the previous bound, show that for $T > \frac{\log m}{2\gamma^2}$, the resulting hypothesis is consistent with the sample of size m.

(e) Let s be the VC-dimension of the base learners used. Give a bound on the generalization error of the consistent hypothesis obtained after $T = \left\lfloor \frac{\log m}{2\gamma^2} \right\rfloor + 1$ rounds of boosting. (*Hint*: Use the fact that the VC-dimension of the family of functions $\{\operatorname{sgn}(\sum_{t=1}^{T} \alpha_t h_t) : \alpha_t \in \mathbb{R}\}$ is bounded by $2(s+1)T \log_2(eT)$). Suppose now that γ varies with m. Based on the bound derived, what can be said if $\gamma(m) = O(\sqrt{\frac{\log m}{m}})$?)

7.9 AdaBoost example.

In this exercise we consider a concrete example that consists of eight training points and eight weak classifiers.

(a) Define an $m \times n$ matrix $\mathbf{M}$ where $\mathbf{M}_{ij} = y_i h_j(\mathbf{x}_i)$, i.e., $\mathbf{M}_{ij} = +1$ if training example i is classified correctly by weak classifier h_j, and -1 otherwise. Let $\mathbf{d}_t, \boldsymbol{\lambda}_t \in \mathbb{R}^n$, $\|\mathbf{d}_t\|_1 = 1$ and $d_{t,i}$ (respectively $\lambda_{t,i}$) equal the i^{th} component of $\mathbf{d}_t$ (respectively $\boldsymbol{\lambda}_t$). Now, consider AdaBoost as described in figure 7.7

and define $\mathbf{M}$ as below with eight training points and eight weak classifiers.

$$\mathbf{M} = \begin{pmatrix} -1 & 1 & 1 & 1 & 1 & -1 & -1 & 1 \\ -1 & 1 & 1 & -1 & -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 & 1 & -1 & 1 & 1 \\ 1 & -1 & 1 & 1 & -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & 1 & 1 & -1 \\ 1 & 1 & -1 & 1 & 1 & 1 & 1 & -1 \\ 1 & 1 & -1 & 1 & 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & 1 & -1 \end{pmatrix}$$

Assume that we start with the following initial distribution over the datapoints:

$$\mathbf{d}_1 = \left(\frac{3-\sqrt{5}}{8}, \frac{3-\sqrt{5}}{8}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{\sqrt{5}-1}{8}, \frac{\sqrt{5}-1}{8}, 0\right)^\top$$

Compute the first few steps of the AdaBoost algorithm using $\mathbf{M}$, $\mathbf{d}_1$, and $t_{max} = 7$. What weak classifier is picked at each round of boosting? Do you notice any pattern?

(b) What is the L_1 norm margin produced by AdaBoost for this example?

(c) Instead of using AdaBoost, imagine we combined our classifiers using the following coefficients: $[2, 3, 4, 1, 2, 2, 1, 1] \times \frac{1}{16}$. What is the margin in this case? Does AdaBoost maximize the margin?

7.10 Boosting in the presence of unknown labels. Consider the following variant of the classification problem where, in addition to the positive and negative labels $+1$ and -1, points may be labeled with 0. This can correspond to cases where the true label of a point is unknown, a situation that often arises in practice, or more generally to the fact that the learning algorithm incurs no loss for predicting -1 or $+1$ for such a point. Let $\mathcal{X}$ be the input space and let $\mathcal{Y} = \{-1, 0, +1\}$. As in standard binary classification, the loss of $f\colon \mathcal{X} \to \mathbb{R}$ on a pair $(x, y) \in \mathcal{X} \times \mathcal{Y}$ is defined by $1_{yf(x)<0}$.

Consider a sample $S = ((x_1, y_1), \ldots, (x_m, y_m)) \in (\mathcal{X} \times \mathcal{Y})^m$ and a hypothesis set H of base functions taking values in $\{-1, 0, +1\}$. For a base hypothesis $h_t \in \mathcal{H}$ and a distribution $\mathcal{D}_t$ over indices $i \in [m]$, define ϵ_t^s for $s \in \{-1, 0, +1\}$ by $\epsilon_t^s = \mathbb{E}_{i \sim \mathcal{D}_t}[1_{y_i h_t(x_i)=s}]$.

(a) Derive a boosting-style algorithm for this setting in terms of ϵ_t^ss, using the same objective function as that of AdaBoost. You should carefully justify the definition of the algorithm.

(b) What is the weak-learning assumption in this setting?

(c) Write the full pseudocode of the algorithm.

(d) Give an upper bound on the training error of the algorithm as a function of the number of rounds of boosting and ϵ_t^ss.

7.11 HingeBoost. As discussed in the chapter, AdaBoost can be viewed as coordinate descent applied to an exponential objective function. Here, we consider an alternative ensemble method algorithm, HingeBoost, that consists of applying coordinate descent to an objective function based on the hinge loss. Consider the function F defined for all $\boldsymbol{\alpha} \in \mathbb{R}^N$ by

$$F(\boldsymbol{\alpha}) = \sum_{i=1}^{m} \max\left(0, 1 - y_i \sum_{j=1}^{N} \alpha_j h_j(x_i)\right), \tag{7.34}$$

where the h_js are base classifiers belonging to a hypothesis set H of functions taking values -1 or $+1$.

(a) Show that F is convex and admits a right- and left-derivative along any direction.

(b) For any $j \in [N]$, let $\boldsymbol{e}_j$ denote the direction corresponding to the base hypothesis h_j. Let $\boldsymbol{\alpha}_t$ denote the vector of coefficients $\alpha_{t,j}$, $j \in [N]$ obtained after $t \geq 0$ iterations of coordinate descent and $f_t = \sum_{j=1}^{N} \alpha_{t,j} h_j$ the predictor obtained after t iterations.

Give the expression of the right-derivative $F'_+(\boldsymbol{\alpha}_{t-1}, \boldsymbol{e}_j)$ and the left-derivative $F'_-(\boldsymbol{\alpha}_{t-1}, \boldsymbol{e}_j)$ after $t-1$ iterations in terms of f_{t-1}.

(c) For any $j \in [N]$, define the maximum directional derivative $\delta F(\boldsymbol{\alpha}_{t-1}, \boldsymbol{e}_j)$ at $\boldsymbol{\alpha}_{t-1}$ as follows:

$$\delta F(\boldsymbol{\alpha}_{t-1}, \boldsymbol{e}_j) = \begin{cases} 0 & \text{if } F'_-(\boldsymbol{\alpha}_{t-1}, \boldsymbol{e}_j) \leq 0 \leq F'_+(\boldsymbol{\alpha}_{t-1}, \boldsymbol{e}_j) \\ F'_+(\boldsymbol{\alpha}_{t-1}, \boldsymbol{e}_j) & \text{if } F'_-(\boldsymbol{\alpha}_{t-1}, \boldsymbol{e}_j) \leq F'_+(\boldsymbol{\alpha}_{t-1}, \boldsymbol{e}_j) \leq 0 \\ F'_-(\boldsymbol{\alpha}_{t-1}, \boldsymbol{e}_j) & \text{if } 0 \leq F'_-(\boldsymbol{\alpha}_{t-1}, \boldsymbol{e}_j) \leq F'_+(\boldsymbol{\alpha}_{t-1}, \boldsymbol{e}_j). \end{cases}$$

The direction $\boldsymbol{e}_j$ considered by the coordinate descent considered here is the one maximizing $|\delta F(\boldsymbol{\alpha}_{t-1}, \boldsymbol{e}_j)|$. Once the best direction j is selected, the step η can be determined by minimizing $F(\boldsymbol{\alpha}_{t-1} + \eta \boldsymbol{e}_j)$ using a grid search. Give the pseudocode of HingeBoost.

7.12 Empirical margin loss boosting. As discussed in the chapter, AdaBoost can be viewed as coordinate descent applied to a convex upper bound on the empirical error. Here, we consider an algorithm seeking to minimize the empirical margin loss. For any $0 \leq \rho < 1$ let $\widehat{R}_{S,\rho}(f) = \frac{1}{m}\sum_{i=1}^{m} 1_{y_i f(x_i) \leq \rho}$ denote the empirical margin loss of a function f of the form $f = \frac{\sum_{t=1}^{T} \alpha_t h_t}{\sum_{t=1}^{T} \alpha_t}$ for a labeled sample $S = ((x_1, y_1), \ldots, (x_m, y_m))$.

(a) Show that $\widehat{R}_{S,\rho}(f)$ can be upper bounded as follows:

$$\widehat{R}_{S,\rho}(f) \leq \frac{1}{m}\sum_{i=1}^{m} \exp\left(-y_i \sum_{t=1}^{T} \alpha_t h_t(x_i) + \rho \sum_{t=1}^{T} \alpha_t\right).$$

(b) For any $\rho > 0$, let G_ρ be the objective function defined for all $\boldsymbol{\alpha} \geq 0$ by

$$G_\rho(\boldsymbol{\alpha}) = \frac{1}{m}\sum_{i=1}^{m} \exp\left(-y_i \sum_{j=1}^{N} \alpha_j h_j(x_i) + \rho \sum_{j=1}^{N} \alpha_j\right),$$

with $h_j \in H$ for all $j \in [N]$, with the notation used in class in the boosting lecture. Show that G_ρ is convex and differentiable.

(c) Derive a boosting-style algorithm $\mathcal{A}_\rho$ by applying (maximum) coordinate descent to G_ρ. You should justify in detail the derivation of the algorithm, in particular the choice of the base classifier selected at each round and that of the step. Compare both to their counterparts in AdaBoost.

(d) What is the equivalent of the weak learning assumption for $\mathcal{A}_\rho$ (*Hint*: use non-negativity of the step value)?

(e) Give the full pseudocode of the algorithm $\mathcal{A}_\rho$. What can you say about the $\mathcal{A}_0$ algorithm?

(f) Provide a bound on $\widehat{R}_{S,\rho}(f)$.

i. Prove the upper bound $\widehat{R}_{S,\rho}(f) \leq \exp\left(\sum_{t=1}^{T} \alpha_t \rho\right) \prod_{t=1}^{T} Z_t$, where the normalization factors Z_t are defined as in the case of AdaBoost (with α_t the step chosen by $\mathcal{A}_\rho$ at round t).

ii. Give the expression of Z_t as a function of ρ and ϵ_t, where ϵ_t is the weighted error of the hypothesis found by $\mathcal{A}_\rho$ at round t (defined in the same way as for AdaBoost in class). Use that to prove the following upper bound

$$\widehat{R}_{S,\rho}(f) \leq \left(u^{\frac{1+\rho}{2}} + u^{-\frac{1-\rho}{2}}\right)^T \prod_{t=1}^{T} \sqrt{\epsilon_t^{1-\rho}(1-\epsilon_t)^{1+\rho}},$$

where $u = \frac{1-\rho}{1+\rho}$.

iii. Assume that for all $t \in [T]$, $\frac{1-\rho}{2} - \epsilon_t > \gamma > 0$. Use the result of the previous question to show that

$$\widehat{R}_{S,\rho}(f) \leq \exp\left(-\frac{2\gamma^2 T}{1-\rho^2}\right).$$

(*Hint*: you can use without proof the following identity:

$$\left(u^{\frac{1+\rho}{2}} + u^{-\frac{1-\rho}{2}}\right) \sqrt{\epsilon_t^{1-\rho}(1-\epsilon_t)^{1+\rho}} \leq 1 - 2\frac{\left(\frac{1-\rho}{2} - \epsilon_t\right)^2}{1-\rho^2},$$

valid for $\frac{1-\rho}{2} - \epsilon_t > 0$.) Show that for $T \geq \frac{(\log m)(1-\rho^2)}{2\gamma^2}$, all points of the training data have margin at least ρ.