5 The Margins Explanation for Boosting's Effectiveness

In chapter 4, we proved bounds on the generalization error of AdaBoost that all predicted classic overfitting. This prediction seemed reasonable and intuitive, given the apparently increasing complexity of AdaBoost's combined classifier with additional rounds of boosting. Although such overfitting is possible, we saw in section 1.3 that AdaBoost sometimes does *not* overfit. There, we gave an example in which a combined classifier of 1000 decision trees far outperforms on test data one consisting of only five trees, even though both perform perfectly on the training set. Indeed, extensive experiments indicate that AdaBoost generally tends to be quite resistant to overfitting. How can we account for this phenomenon? Why is the theory developed in chapter 4 inadequate? How can a combined classifier as huge and complex as the one above—consisting of 1000 trees and roughly two million decision nodes—perform so well on test data?

In this chapter, we find a way out of this seeming paradox. We develop an alternative theory for analyzing AdaBoost's generalization error that provides a qualitative explanation for its lack of overfitting, as well as specific predictions about the conditions under which boosting can fail. The concept at the core of the new approach is the notion of *confidence*, the idea that a classifier can be more sure of some predictions than of others, and that differences in confidence have consequences for generalization. Confidence was entirely ignored in chapter 4, where our analysis took into consideration only the number of incorrect classifications on the training set, rather than the sureness of the predictions. By explicitly taking confidence into account, our new analysis will give bounds that make very different predictions about how AdaBoost works and when to expect overfitting.

To quantify confidence formally, we introduce a measure called the *margin*. Our analysis then follows a two-part argument. First, we show that larger margins on the training set guarantee better generalization performance (or, more precisely, an improvement in a provable upper bound on the generalization error). And second, we show that AdaBoost provably tends to increase the margins on the training set, even after the training error is zero. Thus, we show that with continued training, AdaBoost tends to become more confident in its own predictions, and that the greater the confidence in a prediction, the more likely it is to be correct. Note that in this analysis, the number of rounds of boosting, which is proportional

to the overall size of the final classifier, has little or no impact on generalization, which is instead controlled by the margins; since these are likely to increase with further rounds, this theory predicts an absence of overfitting under identifiable circumstances.

The methods that we use to prove both parts of the analysis outlined above build directly on those developed in the preceding chapters. In addition, we also introduce another general and very powerful technique based on a different measure of hypothesis complexity called Rademacher complexity.

The view of AdaBoost as a margin-maximizing algorithm suggests that it may be possible to derive a better algorithm by modifying AdaBoost to maximize the margins more aggressively. Later in this chapter, we consider how this might be done, as well as some of the subtle difficulties that are involved. We also discuss AdaBoost's connection to other large-margin learning algorithms, particularly support-vector machines.

The margins explanation of boosting contrasts not only with the kind of analysis seen in chapter 4, but also with a competing explanation based on bias-variance theory and the notion that AdaBoost's strong performance is principally due to its "averaging" or "smoothing" effect on the predictions of an "unstable" base learning algorithm. We discuss further in this chapter why these explanations, though possibly relevant to related methods, are ultimately inadequate for AdaBoost.

Finally, we explore some practical applications of margins and their interpretation as a measure of confidence.

5.1 Margin as a Measure of Confidence

The basis of our analysis is the *margin*, a quantitative measure of the confidence of a prediction made by the combined classifier. Recall that the combined classifier has the form

$$H(x) = \text{sign}(F(x))$$

where

$$F(x) \doteq \sum_{t=1}^{T} \alpha_t h_t(x).$$

It will be convenient to normalize the nonnegative weights α_t on the base classifiers. Let

$$a_t \doteq \frac{\alpha_t}{\sum_{t'=1}^{T} \alpha_{t'}}, \tag{5.1}$$

and let

$$f(x) \doteq \sum_{t=1}^{T} a_t h_t(x) = \frac{F(x)}{\sum_{t=1}^{T} \alpha_t}. \tag{5.2}$$

Then $\sum_{t=1}^{T} a_t = 1$, and since multiplying by a positive constant does not change the sign of $F(x)$, we can write

$$H(x) = \text{sign}(f(x)). \tag{5.3}$$

For a given labeled example (x, y), we can now define the *margin* simply to be $yf(x)$. For clarity, this quantity is sometimes referred to as the *normalized margin* to distinguish it from the *unnormalized margin* $yF(x)$ obtained by omitting the normalization step above. Later, we will be interested in both quantities, although their properties are quite distinct. We will often use the shorter term *margin* when the context is sufficient to prevent confusion. In particular, throughout this chapter, this term will refer exclusively to the normalized margin.

Recall that the base classifiers h_t have range $\{-1, +1\}$, and that labels y also are in $\{-1, +1\}$. Because the weights a_t are normalized, this implies that f has range $[-1, +1]$, and so the margin also is in $[-1, +1]$. Furthermore, $y = H(x)$ if and only if y has the same sign as $f(x)$, that is, if and only if the margin of (x, y) is positive. Thus, the sign of the margin indicates whether or not the example is correctly classified by the combined classifier.

As has been noted before, the combined classifier H is simply a weighted majority vote of the predictions of the base classifiers in which the vote of h_t is given weight a_t. An equivalent way of thinking about the margin is as the difference between the weight of the base classifiers predicting the correct label y and the weight of those predicting the incorrect label $-y$. When this vote is very close, so that the predicted label $H(x)$ is based on a narrow majority, the margin will be small in magnitude and, intuitively, we will have little confidence in the prediction. On the other hand, when the prediction $H(x)$ is based on a clear and substantial majority of the base classifiers, the margin will be correspondingly large lending greater confidence in the predicted label. Thus, the magnitude of the margin (or, equivalently, of $f(x)$) is a reasonable measure of confidence. These interpretations of the range of values of the margin can be diagrammed as in figure 5.1.

We can visualize the effect AdaBoost has on the margins of the training examples by plotting their distribution. In particular, we can create a plot showing, for each $\theta \in [-1, +1]$, the fraction of training examples with margin at most θ. For such a cumulative distribution curve, the bulk of the distribution lies where the curve rises the most steeply. Figure 5.2 shows such a *margin distribution graph* for the same dataset used to create figure 1.7 (p. 16),

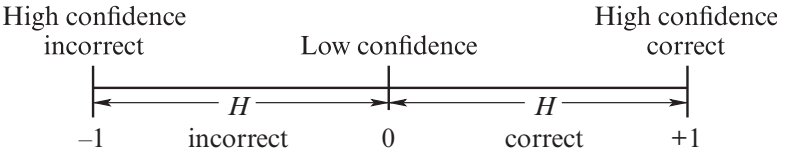

Figure 5.1
An example's margin is in the range $[-1, +1]$ with a positive sign if and only if the combined classifier H is correct. Its magnitude measures the confidence in the combined classifier's prediction.

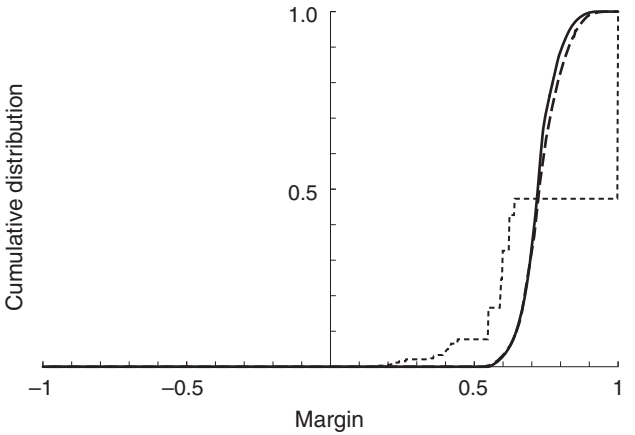

Figure 5.2
The margin distribution graph for boosting C4.5 on the letter dataset showing the cumulative distribution of margins of the training instances after 5, 100, and 1000 iterations, indicated by short-dashed, long-dashed (mostly hidden), and solid curves, respectively. (Reprinted with permission of the Institute of Mathematical Statistics.)

showing the margin distribution after 5, 100, and 1000 rounds of boosting. Whereas nothing at all is happening to the training error, these curves expose dramatic changes happening on the margin distribution. For instance, after five rounds, although the training error is zero (so that no examples have negative margin), a rather substantial fraction of the training examples (7.7%) have margin below 0.5. By round 100, all of these examples have been swept to the right so that not a single example has margin below 0.5, and nearly all have margin above 0.6. (On the other hand, many with margin 1.0 have slipped back to the 0.6–0.8 range.) In line with this trend, the minimum margin of any training example has increased from 0.14 at round 5 to 0.52 at round 100, and 0.55 at round 1000.

Thus, this example is indicative of the powerful effect AdaBoost has on the margins, aggressively pushing up those examples with small or negative margin. Moreover, in comparison with figure 1.7, we see that this overall increase in the margins appears to be correlated with better performance on the test set.

Indeed, as will be seen, AdaBoost can be analyzed theoretically along exactly these lines. We will first prove a bound on the generalization error of AdaBoost—or any other voting method—that depends only on the margins of the training examples, and *not* on the number of rounds of boosting. Thus, this bound predicts that AdaBoost will not overfit regardless of how long it is run, provided that large margins can be achieved (and provided, of course, that the base classifiers are not too complex relative to the size of the training set).

The second part of the analysis is to prove that, as observed empirically in figure 5.2, AdaBoost generally tends to increase the margins of all training examples. All of this will be made precise shortly.

5.2 A Margins-Based Analysis of the Generalization Error

We begin our analysis with a proof of a generalization-error bound in terms of the training-set margins.

5.2.1 Intuition

Let us first try to provide a bit of intuition behind the proof. AdaBoost's combined classifier is a (weighted) majority vote over a possibly very large "committee" of voting base classifiers. Similarly, real-world political elections also may be held with tens or hundreds of millions of voters. Even so, the outcome of such an election can often be predicted by taking a survey, that is, by randomly polling a relatively tiny subset of the electorate, usually around a thousand voters, regardless of the size of the entire electorate. This approach works provided that the election is not too close, that is, provided one candidate has a substantial lead over his or her opponent. This notion of closeness is exactly what is measured by the margin.

In the same manner, the overall prediction of even a very large combined classifier can be determined by sampling randomly among its base classifiers. The majority vote of these "polled" base classifiers will usually be the same as the entire committee represented by the combined classifier, provided that the margin of the overall vote is large. And the larger the margin, the fewer the base classifiers that need to be polled.

So if most examples have large margins, then the combined classifier can be approximated by a much smaller combination of base classifiers, allowing us to use techniques, like those in chapter 4, which are applicable to such classifiers composed of a relatively small number of base classifiers. Thus, the idea is to show that any combined classifier that attains large margins, even a very big one, must be close to a fairly small classifier, and then to use more direct techniques on this simpler approximating set of classifiers.

We now give a more formal treatment. As in chapter 4, we assume that all base classifiers belong to some space $\mathcal{H}$. For simplicity, we assume without loss of generality that $\mathcal{H}$ is closed under negation so that $-h \in \mathcal{H}$ whenever $h \in \mathcal{H}$. (This allows us to avoid considering negative weights on the base classifiers.) We define the *convex hull* $\mathrm{co}(\mathcal{H})$ of $\mathcal{H}$ as the set of all mappings that can be generated by taking a weighted average of classifiers from $\mathcal{H}$:

$$\mathrm{co}(\mathcal{H}) \doteq \left\{ f : x \mapsto \sum_{t=1}^{T} a_t h_t(x) \;\middle|\; a_1, \ldots, a_T \geq 0; \sum_{t=1}^{T} a_t = 1; h_1, \ldots, h_T \in \mathcal{H}; T \geq 1 \right\}. \tag{5.4}$$

Note that the function f generated by AdaBoost as in equation (5.2) is a member of this set.

As usual, $\mathcal{D}$ is the true distribution from which all examples are generated, and $S = \langle (x_1, y_1), \ldots, (x_m, y_m) \rangle$ is the training set. We will sometimes be interested in computing probabilities or expectations with respect to an example (x, y) chosen randomly according

to distribution $\mathcal{D}$, which we denote by $\mathbf{Pr}_{\mathcal{D}}[\cdot]$ or $\mathbf{E}_{\mathcal{D}}[\cdot]$. We also will sometimes consider the choice of (x, y) from the empirical distribution, that is, selected uniformly at random from the training set S. In this case, we use the notation $\mathbf{Pr}_S[\cdot]$ and $\mathbf{E}_S[\cdot]$. For instance, $\mathbf{Pr}_{\mathcal{D}}[H(x) \neq y]$ is the true generalization error of H, and

$$\mathbf{Pr}_S[H(x) \neq y] \doteq \frac{1}{m} \sum_{i=1}^{m} \mathbf{1}\{H(x_i) \neq y_i\}$$

is the training error. Recalling that H makes a mistake if and only if the margin $yf(x)$ is not positive, we can write H's generalization error equivalently as $\mathbf{Pr}_{\mathcal{D}}[yf(x) \leq 0]$, and similarly for the training error.

Theorems 5.1 and 5.5, the main results of this section, state that with high probability, the generalization error of any majority vote classifier can be bounded in terms of the number of training examples with margin below a threshold θ, plus an additional term which depends on the number of training examples, some "complexity" measure of $\mathcal{H}$, and the threshold θ (preventing us from choosing θ too close to zero). As in chapters 2 and 4, when $\mathcal{H}$ is finite, complexity is measured by $\log |\mathcal{H}|$; when $\mathcal{H}$ is infinite, its VC-dimension is used instead.

5.2.2 Finite Base Hypothesis Spaces

We begin with the simpler case that the space $\mathcal{H}$ of base classifiers is finite.

Theorem 5.1 Let $\mathcal{D}$ be a distribution over $\mathcal{X} \times \{-1, +1\}$, and let S be a sample of m examples chosen independently at random according to $\mathcal{D}$. Assume that the base classifier space $\mathcal{H}$ is finite, and let $\delta > 0$. Then with probability at least $1 - \delta$ over the random choice of the training set S, every weighted average function $f \in \mathrm{co}(\mathcal{H})$ satisfies the following bound:

$$\mathbf{Pr}_{\mathcal{D}}[yf(x) \leq 0] \leq \mathbf{Pr}_S[yf(x) \leq \theta] + O\left(\sqrt{\frac{\log |\mathcal{H}|}{m\theta^2} \cdot \log\left(\frac{m\theta^2}{\log |\mathcal{H}|}\right) + \frac{\log(1/\delta)}{m}}\right)$$

for all $\theta > \sqrt{(\ln |\mathcal{H}|)/(4m)}$.

The term on the left is the generalization error, as noted above. The first term on the right is the fraction of training examples with margin below some threshold θ. This term will be small if most training examples have large margin (i.e., larger than θ). The second term on the right is an additional term that becomes small as the size of the training set m gets larger, provided the complexity of the base classifiers is controlled for θ bounded away from zero. This bound is analogous to the sorts of bounds seen in chapter 2, such as theorems 2.2 and 2.5, which quantify how the generalization error depends upon fit to the training set and complexity of the hypotheses used. Here, however, fit to the data is measured by the number of examples with small margin (at most θ), rather than the training error and, importantly, only the complexity of the *base* classifiers enters the bound—the number of nonzero terms

comprising f, that is, the number of rounds T in the boosting context, does not appear anywhere in the bound.

Such an analysis is entirely consistent with the behavior observed in the example discussed in section 5.1, where no degradation in performance was observed with further rounds of boosting. Rather, performance improved with continued boosting in a manner apparently correlated with a general increase in the margins of the training examples. The overfitting behavior seen in section 1.2.3 is also qualitatively consistent with this analysis; in that case, it seems that a relatively small sample size and generally small margins together have doomed the performance beyond just a few rounds of boosting.

Proof For the sake of the proof, we define $\mathcal{A}_n$ to be the set of *unweighted* averages over n elements from $\mathcal{H}$:

$$\mathcal{A}_n \doteq \left\{ f : x \mapsto \frac{1}{n} \sum_{j=1}^{n} h_j(x) \,\middle|\, h_1, \ldots, h_n \in \mathcal{H} \right\}.$$

Note that the same $h \in \mathcal{H}$ may appear multiple times in such an average.

As outlined above, the main idea of the proof is to approximate any weighted average function $f \in \text{co}(\mathcal{H})$ by randomly polling its constituents. Any such function has the form given in equations (5.2) and (5.4). Note that the weights a_t on the base classifiers naturally define a probability distribution over $\mathcal{H}$ according to which individual base classifiers can be sampled randomly. Going a step further, we can imagine an experiment in which n base classifiers $\tilde{h}_1, \ldots, \tilde{h}_n$ from $\mathcal{H}$ are selected independently at random. Thus, each $\tilde{h}_j$ is selected independently at random from $\mathcal{H}$ where we choose $\tilde{h}_j$ to be equal to h_t with probability a_t. We can then form their unweighted average

$$\tilde{f}(x) \doteq \frac{1}{n} \sum_{j=1}^{n} \tilde{h}_j(x), \tag{5.5}$$

which is clearly a member of $\mathcal{A}_n$. It is this function $\tilde{f}$ that we use to approximate f.

We assume throughout this proof that $\tilde{f}$ is selected in this random manner, denoting probability and expectations with respect to its random selection by $\mathbf{Pr}_{\tilde{f}}[\cdot]$ and $\mathbf{E}_{\tilde{f}}[\cdot]$. The particular choice of n will come later.

Here is an informal outline of the proof, which has two main parts. First, we will show that $\tilde{f}$ is typically a good approximation of f in the sense that, for "most" examples (x, y),

$$\left| y f(x) - y \tilde{f}(x) \right| \leq \frac{\theta}{2}.$$

Thus, if $y f(x) \leq 0$, then it is likely that $y \tilde{f}(x) \leq \theta/2$, which means that

$$\mathbf{Pr}_{\mathcal{D}}[yf(x) \le 0] \lesssim \mathbf{Pr}_{\mathcal{D}}\left[y\tilde{f}(x) \le \frac{\theta}{2}\right], \tag{5.6}$$

where we use $\lesssim$ to indicate approximate inequality in a strictly informal sense. A similar argument will show that

$$\mathbf{Pr}_{S}\left[y\tilde{f}(x) \le \frac{\theta}{2}\right] \lesssim \mathbf{Pr}_{S}[yf(x) \le \theta]. \tag{5.7}$$

The second key ingredient of the proof is an argument that the margins of functions in $\mathcal{A}_n$ have statistics on the training set that are similar to those on the true distribution $\mathcal{D}$. In particular, we show that, with high probability, the empirical probability of a small margin is close to its true probability for all $\tilde{f} \in \mathcal{A}_n$. That is,

$$\mathbf{Pr}_{\mathcal{D}}\left[y\tilde{f}(x) \le \frac{\theta}{2}\right] \lesssim \mathbf{Pr}_{S}\left[y\tilde{f}(x) \le \frac{\theta}{2}\right]. \tag{5.8}$$

Combining equations (5.6), (5.7), and (5.8) will give

$$\mathbf{Pr}_{\mathcal{D}}[yf(x) \le 0] \lesssim \mathbf{Pr}_{\mathcal{D}}\left[y\tilde{f}(x) \le \frac{\theta}{2}\right] \lesssim \mathbf{Pr}_{S}\left[y\tilde{f}(x) \le \frac{\theta}{2}\right] \lesssim \mathbf{Pr}_{S}[yf(x) \le \theta],$$

proving the theorem.

We now proceed to the details. Our first observation is that, for fixed x, if n is sufficiently large, then $\tilde{f}(x)$ will be close to its expectation, which by construction turns out to be $f(x)$. Specifically, we have:

Lemma 5.2 For fixed x, $\theta > 0$, and $n \ge 1$,

$$\mathbf{Pr}_{\tilde{f}}\left[\left|\tilde{f}(x) - f(x)\right| \ge \frac{\theta}{2}\right] \le 2e^{-n\theta^2/8} \doteq \beta_{n,\theta}.$$

Proof With x fixed, $\tilde{h}_j(x)$ is a random variable with range $\{-1, +1\}$. Since $\tilde{h}_j = h_t$ with probability a_t, its expected value is

$$\mathbf{E}_{\tilde{f}}\left[\tilde{h}_j(x)\right] = \sum_{t=1}^{T} a_t h_t(x) = f(x),$$

and so, by equation (5.5), $\mathbf{E}_{\tilde{f}}\left[\tilde{f}(x)\right] = f(x)$ as well. Thus, with minor rescaling, we can apply Hoeffding's inequality (theorem 2.1) to this set of independent random variables $\tilde{h}_1(x), \ldots, \tilde{h}_n(x)$ to obtain

$$\mathbf{Pr}_{\tilde{f}}\left[\left|\tilde{f}(x) - f(x)\right| \ge \frac{\theta}{2}\right] \le \beta_{n,\theta}.$$ ∎

The next lemma shows further that the margin for f, $yf(x)$, will be close to the margin for $\tilde{f}$, $y\tilde{f}(x)$, "on average" if the pair (x, y) is chosen at random from an arbitrary distribution P. Below, $\mathbf{Pr}_P[\cdot]$ and $\mathbf{E}_P[\cdot]$ denote probability and expectation over the random choice of (x, y) from P, respectively.

The proof uses *marginalization*, the principle that if X and Y are random variables, then the probability of any event a can be computed as the expected probability of the event when one of the variables is held fixed:

$$\mathbf{Pr}_{X,Y}[a] = \mathbf{E}_X[\mathbf{Pr}_Y[a|X]]\,.$$

Lemma 5.3 Suppose P is any distribution over pairs (x, y). Then for $\theta > 0$ and $n \geq 1$,

$$\mathbf{Pr}_{P,\tilde{f}}\left[\left|yf(x) - y\tilde{f}(x)\right| \geq \frac{\theta}{2}\right] \leq \beta_{n,\theta}.$$

Proof Using marginalization and lemma 5.2, we have that

$$\begin{aligned}\mathbf{Pr}_{P,\tilde{f}}\left[\left|yf(x) - y\tilde{f}(x)\right| \geq \frac{\theta}{2}\right] &= \mathbf{Pr}_{P,\tilde{f}}\left[\left|f(x) - \tilde{f}(x)\right| \geq \frac{\theta}{2}\right] \\ &= \mathbf{E}_P\left[\mathbf{Pr}_{\tilde{f}}\left[\left|f(x) - \tilde{f}(x)\right| \geq \frac{\theta}{2}\right]\right] \\ &\leq \mathbf{E}_P\left[\beta_{n,\theta}\right] = \beta_{n,\theta}.\end{aligned}$$

■

Thus, $\tilde{f}$ is a good approximation of f. In particular, we can now prove equation (5.6) in more precise terms. Specifically, lemma 5.3, applied to distribution $\mathcal{D}$, gives that

$$\begin{aligned}\mathbf{Pr}_{\mathcal{D}}[yf(x) \leq 0] &= \mathbf{Pr}_{\mathcal{D},\tilde{f}}[yf(x) \leq 0] \\ &\leq \mathbf{Pr}_{\mathcal{D},\tilde{f}}\left[y\tilde{f}(x) \leq \frac{\theta}{2}\right] + \mathbf{Pr}_{\mathcal{D},\tilde{f}}\left[yf(x) \leq 0, y\tilde{f}(x) > \frac{\theta}{2}\right] \qquad (5.9) \\ &\leq \mathbf{Pr}_{\mathcal{D},\tilde{f}}\left[y\tilde{f}(x) \leq \frac{\theta}{2}\right] + \mathbf{Pr}_{\mathcal{D},\tilde{f}}\left[\left|yf(x) - y\tilde{f}(x)\right| > \frac{\theta}{2}\right] \\ &\leq \mathbf{Pr}_{\mathcal{D},\tilde{f}}\left[y\tilde{f}(x) \leq \frac{\theta}{2}\right] + \beta_{n,\theta}. \qquad (5.10)\end{aligned}$$

Here, equation (5.9) uses the simple fact that for any two events a and b,

$$\mathbf{Pr}[a] \;=\; \mathbf{Pr}[a, b] + \mathbf{Pr}[a, \neg b] \;\leq\; \mathbf{Pr}[b] + \mathbf{Pr}[a, \neg b]\,. \qquad (5.11)$$

Equation (5.7) follows from a similar derivation that again uses equation (5.11) and lemma 5.3 now applied instead to the empirical distribution:

$$\begin{aligned}\mathbf{Pr}_{S,\tilde{f}}\left[y\tilde{f}(x) \le \frac{\theta}{2}\right] &\le \mathbf{Pr}_{S,\tilde{f}}[yf(x) \le \theta] + \mathbf{Pr}_{S,\tilde{f}}\left[y\tilde{f}(x) \le \frac{\theta}{2}, yf(x) > \theta\right] \\ &\le \mathbf{Pr}_{S,\tilde{f}}[yf(x) \le \theta] + \mathbf{Pr}_{S,\tilde{f}}\left[\left|yf(x) - y\tilde{f}(x)\right| > \frac{\theta}{2}\right] \\ &\le \mathbf{Pr}_S[yf(x) \le \theta] + \beta_{n,\theta}. \end{aligned} \tag{5.12}$$

We move on now to the second part of the proof, in which we show that equation (5.8) holds for all $\tilde{f} \in \mathcal{A}_n$ with high probability.

Lemma 5.4 Let

$$\varepsilon_n \doteq \sqrt{\frac{\ln\left[n(n+1)^2|\mathcal{H}|^n/\delta\right]}{2m}}.$$

Then, with probability at least $1-\delta$ (where the probability is taken over the choice of the random training set S), for all $n \ge 1$, for all $\tilde{f} \in \mathcal{A}_n$, and for all $\theta \ge 0$,

$$\mathbf{Pr}_{\mathcal{D}}\left[y\tilde{f}(x) \le \frac{\theta}{2}\right] \le \mathbf{Pr}_S\left[y\tilde{f}(x) \le \frac{\theta}{2}\right] + \varepsilon_n. \tag{5.13}$$

Proof Let $p_{\tilde{f},\theta} = \mathbf{Pr}_{\mathcal{D}}\left[y\tilde{f}(x) \le \theta/2\right]$, and let $\hat{p}_{\tilde{f},\theta} = \mathbf{Pr}_S\left[y\tilde{f}(x) \le \theta/2\right]$. Consider first a particular *fixed* choice of n, $\tilde{f}$, and θ. Let B_i be a Bernoulli random variable that is 1 if $y_i\tilde{f}(x_i) \le \theta/2$, and 0 otherwise. Note that here the underlying random process is the choice of the random sample S. Then

$$\hat{p}_{\tilde{f},\theta} = \frac{1}{m}\sum_{i=1}^{m} B_i,$$

and

$$p_{\tilde{f},\theta} = \mathbf{E}[B_i] = \mathbf{E}\left[\hat{p}_{\tilde{f},\theta}\right].$$

Thus, by Hoeffding's inequality (theorem 2.1),

$$\mathbf{Pr}\left[p_{\tilde{f},\theta} \ge \hat{p}_{\tilde{f},\theta} + \varepsilon_n\right] = \mathbf{Pr}\left[\hat{p}_{\tilde{f},\theta} \le \mathbf{E}\left[\hat{p}_{\tilde{f},\theta}\right] - \varepsilon_n\right] \le e^{-2\varepsilon_n^2 m}, \tag{5.14}$$

which means that equation (5.13) holds for fixed $\tilde{f}$ and θ with high probability. We next use the union bound to show that it also holds for all $\tilde{f}$ and θ simultaneously with high probability.

Note that $y\tilde{f}(x) \le \theta/2$ if and only if

$$y\sum_{j=1}^{n}\tilde{h}_j(x) \le \frac{n\theta}{2}$$

(by definition of $\tilde{f}$), which in turn holds if and only if

$$y \sum_{j=1}^{n} \tilde{h}_j(x) \leq \left\lfloor \frac{n\theta}{2} \right\rfloor,$$

since the term on the left is an integer. Thus, $p_{\tilde{f},\theta} = p_{\tilde{f},\overline{\theta}}$ and $\hat{p}_{\tilde{f},\theta} = \hat{p}_{\tilde{f},\overline{\theta}}$, where $\overline{\theta}$ is chosen so that

$$\frac{n\overline{\theta}}{2} = \left\lfloor \frac{n\theta}{2} \right\rfloor,$$

that is, from the set

$$\Theta_n \doteq \left\{ \frac{2i}{n} : i = 0, 1, \ldots, n \right\}.$$

(There is never a need to consider $\theta > 2$ since $y\tilde{f}(x) \in [-1, +1]$.) Thus, for fixed n, the chance that $p_{\tilde{f},\theta} \geq \hat{p}_{\tilde{f},\theta} + \varepsilon_n$ for any $\tilde{f} \in \mathcal{A}_n$ and any $\theta \geq 0$ is

$$\begin{aligned}
\mathbf{Pr}\left[\exists \tilde{f} \in \mathcal{A}_n, \theta \geq 0 : p_{\tilde{f},\theta} \geq \hat{p}_{\tilde{f},\theta} + \varepsilon_n\right] &= \mathbf{Pr}\left[\exists \tilde{f} \in \mathcal{A}_n, \theta \in \Theta_n : p_{\tilde{f},\theta} \geq \hat{p}_{\tilde{f},\theta} + \varepsilon_n\right] \\
&\leq |\mathcal{A}_n| \cdot |\Theta_n| \cdot e^{-2\varepsilon_n^2 m} \qquad (5.15) \\
&\leq |\mathcal{H}|^n \cdot (n+1) \cdot e^{-2\varepsilon_n^2 m} \qquad (5.16) \\
&= \frac{\delta}{n(n+1)}. \qquad (5.17)
\end{aligned}$$

Equation (5.15) uses equation (5.14) and the union bound. Equation (5.16) is simple counting. And equation (5.17) follows from our choice of ε_n.

Applying the union bound one last time, we have that the probability of this happening for any $n \geq 1$ is at most

$$\sum_{n=1}^{\infty} \frac{\delta}{n(n+1)} = \delta. \qquad \blacksquare$$

We can now complete the proof of theorem 5.1. We assume that we are in the "good" case in which equation (5.13) holds for all $n \geq 1$, for all $\tilde{f} \in \mathcal{A}_n$, and for all $\theta \geq 0$ (as will happen with probability at least $1 - \delta$, by lemma 5.4). Using marginalization (twice), this implies that

$$\begin{aligned}
\mathbf{Pr}_{\mathcal{D},\tilde{f}}\left[y\tilde{f}(x) \leq \frac{\theta}{2}\right] &= \mathbf{E}_{\tilde{f}}\left[\mathbf{Pr}_{\mathcal{D}}\left[y\tilde{f}(x) \leq \frac{\theta}{2}\right]\right] \\
&\leq \mathbf{E}_{\tilde{f}}\left[\mathbf{Pr}_{S}\left[y\tilde{f}(x) \leq \frac{\theta}{2}\right] + \varepsilon_n\right] \\
&= \mathbf{Pr}_{S,\tilde{f}}\left[y\tilde{f}(x) \leq \frac{\theta}{2}\right] + \varepsilon_n. \qquad (5.18)
\end{aligned}$$

Thus, pulling everything together—specifically, equations (5.10), (5.18), and (5.12)—we have, with probability at least $1-\delta$, for every $f \in \text{co}(\mathcal{H})$, for every $n \geq 1$, and for every $\theta > 0$,

$$\begin{aligned}
\mathbf{Pr}_{\mathcal{D}}[yf(x) \leq 0] &\leq \mathbf{Pr}_{\mathcal{D},\tilde{f}}\left[y\tilde{f}(x) \leq \frac{\theta}{2}\right] + \beta_{n,\theta} \\
&\leq \mathbf{Pr}_{S,\tilde{f}}\left[y\tilde{f}(x) \leq \frac{\theta}{2}\right] + \varepsilon_n + \beta_{n,\theta} \\
&\leq \mathbf{Pr}_S[yf(x) \leq \theta] + \beta_{n,\theta} + \varepsilon_n + \beta_{n,\theta} \\
&= \mathbf{Pr}_S[yf(x) \leq \theta] + 4e^{-n\theta^2/8} + \sqrt{\frac{\ln\left[n(n+1)^2|\mathcal{H}|^n/\delta\right]}{2m}}.
\end{aligned}$$

The bound in the statement of the theorem can now be obtained by setting

$$n = \left\lceil \frac{4}{\theta^2} \ln\left(\frac{4m\theta^2}{\ln|\mathcal{H}|}\right)\right\rceil. \qquad \blacksquare$$

5.2.3 Infinite Base Hypothesis Spaces

Theorem 5.1 applies only to the case of a finite base classifier space $\mathcal{H}$. When this space is infinite, we instead use its VC-dimension as a measure of complexity, giving the following analogue of theorem 5.1:

Theorem 5.5 Let $\mathcal{D}$ be a distribution over $\mathcal{X} \times \{-1, +1\}$, and let S be a sample of m examples chosen independently at random according to $\mathcal{D}$. Suppose the base-classifier space $\mathcal{H}$ has VC-dimension d, and let $\delta > 0$. Assume that $m \geq d \geq 1$. Then, with probability at least $1-\delta$ over the random choice of the training set S, every weighted average function $f \in \text{co}(\mathcal{H})$ satisfies the following bound:

$$\mathbf{Pr}_{\mathcal{D}}[yf(x) \leq 0] \leq \mathbf{Pr}_S[yf(x) \leq \theta] + O\left(\sqrt{\frac{d\log(m/d)\log(m\theta^2/d)}{m\theta^2} + \frac{\log(1/\delta)}{m}}\right)$$

for all $\theta > \sqrt{8d\ln(em/d)/m}$.

Proof This theorem can be proved exactly like theorem 5.1, except that lemma 5.4 needs to be modified as follows:

Lemma 5.6 Let

$$\varepsilon_n \doteq \sqrt{\frac{32[\ln(n(n+1)^2) + dn\ln(em/d) + \ln(8/\delta)]}{m}}.$$

Then, with probability at least $1-\delta$ (over the choice of the random training set), for all $n \geq 1$, for all $\tilde{f} \in \mathcal{A}_n$ and for all $\theta \geq 0$,

$$\mathbf{Pr}_{\mathcal{D}}\left[y\tilde{f}(x) \leq \frac{\theta}{2}\right] \leq \mathbf{Pr}_{S}\left[y\tilde{f}(x) \leq \frac{\theta}{2}\right] + \varepsilon_n. \tag{5.19}$$

Proof To prove the lemma, we make use of theorem 2.6 rather than the union bound. To do so, we construct a family of subsets of the space $\mathcal{Z} = \mathcal{X} \times \{-1, +1\}$ of instance-label pairs. For any $\tilde{f} \in \mathcal{A}_n$ and $\theta \geq 0$, let

$$B_{\tilde{f},\theta} \doteq \left\{(x, y) \in \mathcal{Z} : y\tilde{f}(x) \leq \theta/2\right\}$$

be the set of pairs whose margin with respect to $\tilde{f}$ is at most $\theta/2$. Then let $\mathcal{B}_n$ be the collection of all such subsets:

$$\mathcal{B}_n \doteq \left\{B_{\tilde{f},\theta} : \tilde{f} \in \mathcal{A}_n, \theta \geq 0\right\}.$$

To apply theorem 2.6 to this collection, we first count the number of in-out behaviors realizable by sets in $\mathcal{B}_n$ on a finite set of m points, that is, $\Pi_{\mathcal{B}_n}(m)$. Let $x_1, \ldots, x_m \in \mathcal{X}$ and $y_1, \ldots, y_m \in \{-1, +1\}$. Since the VC-dimension of $\mathcal{H}$ is d, Sauer's lemma (lemma 2.4) and equation (2.12) give that the number of labelings of the x_i's by hypotheses in $\mathcal{H}$ is

$$|\{\langle h(x_1), \ldots, h(x_m)\rangle : h \in \mathcal{H}\}| \leq \sum_{i=0}^{d} \binom{m}{i} \leq \left(\frac{em}{d}\right)^d$$

for $m \geq d \geq 1$. This implies that the number of margin behaviors associated with functions $\tilde{f} \in \mathcal{A}_n$ is

$$\left|\left\{\langle y_1\tilde{f}(x_1), \ldots, y_m\tilde{f}(x_m)\rangle : \tilde{f} \in \mathcal{A}_n\right\}\right| \leq \left(\frac{em}{d}\right)^{dn},$$

since each $\tilde{f} \in \mathcal{A}_n$ is composed of n functions from $\mathcal{H}$. Since we need consider only $n+1$ distinct values of θ (that is, only $\theta \in \Theta_n$ as in the proof of lemma 5.4), it follows that

$$\Pi_{\mathcal{B}_n}(m) \leq (n+1)\left(\frac{em}{d}\right)^{dn}.$$

Applying theorem 2.6 now gives that, for $n \geq 1$, with probability at least $1-\delta/(n(n+1))$, for all $B_{\tilde{f},\theta} \in \mathcal{B}_n$,

$$\mathbf{Pr}_{z\sim\mathcal{D}}\left[z \in B_{\tilde{f},\theta}\right] \leq \mathbf{Pr}_{z\sim S}\left[z \in B_{\tilde{f},\theta}\right] + \varepsilon_n$$

for the choice of ε_n given in the lemma. This is equivalent to equation (5.19). Thus, by the union bound, this same statement holds for all $n \geq 1$ simultaneously with probability at least $1-\delta$, proving the lemma. ∎

The rest of the proof of theorem 5.5 is the same as before until it is time to plug in our new choice of ε_n giving, with probability at least $1-\delta$,

$$\mathbf{Pr}_{\mathcal{D}}[yf(x) \le 0] \le \mathbf{Pr}_S[yf(x) \le \theta] + 4e^{-n\theta^2/8} + \sqrt{\frac{32[\ln(n(n+1)^2) + dn\ln(em/d) + \ln(8/\delta)]}{m}}$$

for all $f \in \mathrm{co}(\mathcal{H})$, $n \ge 1$, and $\theta > 0$. Setting

$$n = \left\lceil \frac{4}{\theta^2} \ln\left(\frac{m\theta^2}{8d\ln(em/d)}\right) \right\rceil$$

gives the bound stated in the theorem. ∎

We have focused our attention on the general case in which some of the training examples may have small margins below some value θ of interest. This has led to an additional term in the bounds in theorems 5.1 and 5.5 of the form $\tilde{O}\left(1/\sqrt{m}\right)$ as a function of m. However, just as we saw in section 2.2.5 that better rates of convergence are possible with consistent hypotheses, for the same reasons given in that section, these theorems can be similarly modified to give much better bounds on the order of $\tilde{O}\left(1/m\right)$ when *all* training examples have margin above θ so that $\mathbf{Pr}_S[yf(x) \le \theta] = 0$.

5.3 Analysis Based on Rademacher Complexity

Before continuing forward, we pause to outline an alternative method of analysis that is perhaps more abstract mathematically but is very general and powerful. We sketch only the main ideas, and omit most of the proofs. (See the bibliographic notes for further reading.)

We have already explored a number of techniques for measuring the complexity of a space of classifiers. Here we introduce yet another measure, which is at the core of this approach. Intuitively, a space $\mathcal{H}$ is especially "rich" or "expressive" if we find it easy to fit any dataset using classifiers in $\mathcal{H}$. We have routinely measured how well a hypothesis h fits a dataset $(x_1, y_1), \ldots, (x_m, y_m)$ by its training error, a measure that is essentially equivalent to the correlation of the predictions $h(x_i)$ with the labels y_i, that is,

$$\frac{1}{m}\sum_{i=1}^{m} y_i h(x_i).$$

The hypothesis $h \in \mathcal{H}$ that has the best fit then has correlation

$$\max_{h\in\mathcal{H}} \frac{1}{m}\sum_{i=1}^{m} y_i h(x_i).$$

This gives a measure of how well the space $\mathcal{H}$ as a whole fits the data.

Suppose now that the labels y_i are chosen *at random* without regard to the x_i's. In other words, suppose we replace each y_i by a random variable σ_i that is -1 or $+1$ with equal probability, independent of everything else. Thus, the σ_i's represent labels that are pure noise. We can measure how well the space $\mathcal{H}$ can fit this noise in expectation by

$$\mathbf{E}_{\boldsymbol{\sigma}}\left[\max_{h\in\mathcal{H}} \frac{1}{m}\sum_{i=1}^{m}\sigma_i h(x_i)\right], \tag{5.20}$$

where we write $\mathbf{E}_{\boldsymbol{\sigma}}[\cdot]$ for expectation with respect to the choice of the σ_i's. Returning to our earlier intuition, if $\mathcal{H}$ is a rich class, it should have an easier time fitting even random noise, so that equation (5.20) will be large; conversely, for a more restricted class, we expect equation (5.20) to be small. This suggests that this expression may be a reasonable measure of $\mathcal{H}$'s complexity.

This notion generalizes immediately to families of real-valued functions, not just classifiers. In abstract terms, let $\mathcal{Z}$ be any space and $\mathcal{F}$ any family of functions $f : \mathcal{Z} \to \mathbb{R}$. Let $S = \langle z_1, \ldots, z_m\rangle$ be a sequence of points in $\mathcal{Z}$. Then the *Rademacher complexity* of $\mathcal{F}$ with respect to S, which is the focus of this section, is defined to be[1]

$$R_S(\mathcal{F}) \doteq \mathbf{E}_{\boldsymbol{\sigma}}\left[\sup_{f\in\mathcal{F}} \frac{1}{m}\sum_{i=1}^{m}\sigma_i f(z_i)\right]. \tag{5.21}$$

Note that equation (5.20) is the Rademacher complexity of $\mathcal{H}$ with respect to $\langle x_1, \ldots, x_m\rangle$ obtained by taking $\mathcal{F} = \mathcal{H}$ and $\mathcal{Z} = \mathcal{X}$.

Like the complexity measures introduced in section 2.2, the primary purpose of Rademacher complexity is in bounding the difference between empirical and true probabilities or expectations. In particular, the following very general result can be proved.

Theorem 5.7 Let $\mathcal{F}$ be any family of functions $f : \mathcal{Z} \to [-1, +1]$. Let S be a random sequence of m points chosen independently from $\mathcal{Z}$ according to some distribution $\mathcal{D}$. Then with probability at least $1 - \delta$,

$$\mathbf{E}_{z\sim\mathcal{D}}[f(z)] \;\le\; \mathbf{E}_{z\sim S}[f(z)] + 2R_S(\mathcal{F}) + \sqrt{\frac{2\ln(2/\delta)}{m}}$$

for all $f \in \mathcal{F}$.

1. Rademacher complexity is commonly defined instead to be $\mathbf{E}_{\boldsymbol{\sigma}}\left[\sup_{f\in\mathcal{F}} \frac{1}{m}\left|\sum_{i=1}^{m}\sigma_i f(z_i)\right|\right]$. We use the "one-sided" version given in equation (5.21) because it turns out to be simpler and more convenient for our purposes.

(As in section 2.2.4, $\mathbf{E}_{z\sim\mathcal{D}}[\cdot]$ and $\mathbf{E}_{z\sim S}[\cdot]$ denote expectation with respect to the true and empirical distributions, respectively.) Note that the Rademacher complexity that appears here is relative to the sample S. Alternative results can be obtained based on either expected or worst-case complexity.

Thus, proving uniform convergence results, according to this theorem, reduces to computing Rademacher complexity. We briefly outline three techniques that are useful for this purpose. When combined with theorem 5.7, these will be sufficient to give a complete analysis of margin-based voting classifiers.

First, in the special case given above in which $\mathcal{H}$ is a space of binary classifiers and $\mathcal{Z}=\mathcal{X}$, Rademacher complexity can be immediately related to the other complexity measures we have been using. In particular, if $\mathcal{H}$ is finite, then it can be shown (see exercise 6.4) that

$$R_S(\mathcal{H}) \leq \sqrt{\frac{2\ln|\mathcal{H}|}{m}} \tag{5.22}$$

(where m is the size of S, throughout). And in general, for any $\mathcal{H}$,

$$R_S(\mathcal{H}) \leq \sqrt{\frac{2\ln|\Pi_{\mathcal{H}}(S)|}{m}}$$

where $\Pi_{\mathcal{H}}(S)$ is the set of dichotomies realized by $\mathcal{H}$ on S, as in section 2.2.3. By Sauer's lemma (lemma 2.4) and equation (2.12), this implies that if $\mathcal{H}$ has VC-dimension d, then

$$R_S(\mathcal{H}) \leq \sqrt{\frac{2d\ln(em/d)}{m}} \tag{5.23}$$

for $m \geq d \geq 1$. Thus, in a sense, Rademacher complexity subsumes both $\lg|\mathcal{H}|$ and VC-dimension as a complexity measure, and yields results that are at least as general. For instance, theorems 2.2 and 2.5 can now be derived as corollaries of theorem 5.7 (possibly with some adjustment of constants).

In studying voting classifiers, we have been especially interested in the convex hull $\mathrm{co}(\mathcal{H})$ of a space of base classifiers $\mathcal{H}$, as defined in equation (5.4). Remarkably, the Rademacher complexity of the convex hull, despite being a much larger space, is always the same as that of the original space $\mathcal{H}$. That is,

$$R_S(\mathrm{co}(\mathcal{H})) = R_S(\mathcal{H}). \tag{5.24}$$

This follows immediately from the definition of Rademacher complexity given in equation (5.21) since, for any values of the σ_i's and x_i's, the maximum of

$$\sum_{i=1}^{m}\sigma_i f(x_i)$$

over functions f in $\mathrm{co}(\mathcal{H})$ will be realized "at a corner," that is, at an f that is actually equal to one of the classifiers h in the original space $\mathcal{H}$. This property makes Rademacher complexity particularly well suited to the study of voting classifiers, as we will soon see.

Finally, we consider what happens to the Rademacher complexity when all of the functions in a class $\mathcal{F}$ undergo the same transformation. Specifically, let $\phi : \mathbb{R} \to \mathbb{R}$ be any *Lipschitz function*, that is, a function such that, for some constant $L_\phi > 0$ called the *Lipschitz constant*, we have that

$$|\phi(u) - \phi(v)| \leq L_\phi \cdot |u - v|$$

for all $u, v \in \mathbb{R}$. Let $\phi \circ \mathcal{F}$ be the result of composing ϕ with all functions in $\mathcal{F}$:

$$\phi \circ \mathcal{F} \doteq \{z \mapsto \phi(f(z)) \mid f \in \mathcal{F}\}.$$

Then it can be shown (see exercise 5.5) that the Rademacher complexity of the transformed class scales that of the original class by at most L_ϕ. That is,

$$R_S(\phi \circ \mathcal{F}) \leq L_\phi \cdot R_S(\mathcal{F}). \tag{5.25}$$

With these general tools, we can now derive a margins-based analysis that is similar to (actually, slightly better than) the one given in section 5.2.

Let $\mathcal{H}$ be our space of base classifiers, and let $\mathcal{M}$ be the space of all "margin functions" of the form $yf(x)$ where f is any convex combination of base classifiers:

$$\mathcal{M} \doteq \{(x, y) \mapsto yf(x) \mid f \in \mathrm{co}(\mathcal{H})\}.$$

Note that

$$R_S(\mathcal{M}) = R_S(\mathrm{co}(\mathcal{H})) \tag{5.26}$$

since the labels y_i are "absorbed" by the σ_i's and so become irrelevant under the definition of Rademacher complexity given in equation (5.21).

For any $\theta > 0$, let ϕ be the piecewise-linear function

$$\phi(u) \doteq \begin{cases} 1 & \text{if } u \leq 0 \\ 1 - u/\theta & \text{if } 0 \leq u \leq \theta \\ 0 & \text{if } u \geq \theta. \end{cases} \tag{5.27}$$

See figure 5.3. This function is Lipschitz with $L_\phi = 1/\theta$.

We apply theorem 5.7 to the class $\phi \circ \mathcal{M}$. Working through definitions, for a sample of size m, this gives that with probability at least $1 - \delta$,

$$\mathbf{E}_{\mathcal{D}}[\phi(yf(x))] \leq \mathbf{E}_S[\phi(yf(x))] + 2R_S(\phi \circ \mathcal{M}) + \sqrt{\frac{2\ln(2/\delta)}{m}} \tag{5.28}$$

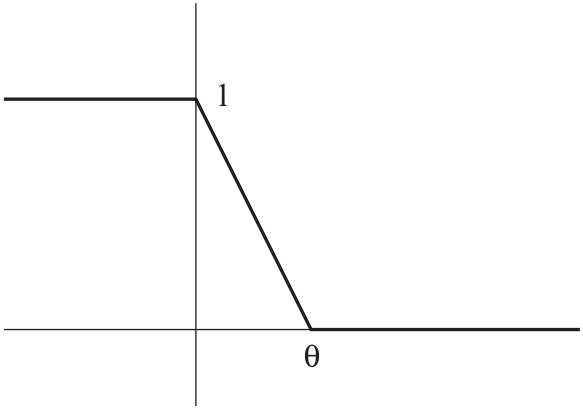

Figure 5.3
A plot of the piecewise-linear function ϕ given in equation (5.27).

for all $f \in \mathrm{co}(\mathcal{H})$. Using, in order, equations (5.25), (5.26), (5.24), and (5.23), we can compute the Rademacher complexity that appears in this expression to be

$$\begin{aligned} R_S(\phi \circ \mathcal{M}) &\leq L_\phi \cdot R_S(\mathcal{M}) \\ &= L_\phi \cdot R_S(\mathrm{co}(\mathcal{H})) \\ &= L_\phi \cdot R_S(\mathcal{H}) \\ &\leq \frac{1}{\theta} \cdot \sqrt{\frac{2d \ln(em/d)}{m}} \end{aligned} \tag{5.29}$$

where d is the VC-dimension of $\mathcal{H}$, and assuming $m \geq d \geq 1$. (Alternatively, a bound in terms of $\ln |\mathcal{H}|$ could be obtained using equation (5.22).)

Note that

$$\mathbf{1}\{u \leq 0\} \;\leq\; \phi(u) \;\leq\; \mathbf{1}\{u \leq \theta\},$$

as is evident from figure 5.3, so that

$$\mathbf{Pr}_{\mathcal{D}}[yf(x) \leq 0] = \mathbf{E}_{\mathcal{D}}[\mathbf{1}\{yf(x) \leq 0\}] \leq \mathbf{E}_{\mathcal{D}}[\phi(yf(x))]$$

and

$$\mathbf{E}_S[\phi(yf(x))] \leq \mathbf{E}_S[\mathbf{1}\{yf(x) \leq \theta\}] = \mathbf{Pr}_S[yf(x) \leq \theta].$$

Therefore, combining with equations (5.28) and (5.29) gives

$$\mathbf{Pr}_{\mathcal{D}}[yf(x) \leq 0] \leq \mathbf{Pr}_S[yf(x) \leq \theta] + \frac{2}{\theta} \cdot \sqrt{\frac{2d \ln(em/d)}{m}} + \sqrt{\frac{2 \ln(2/\delta)}{m}}$$

for all $f \in \text{co}(\mathcal{H})$, with probability at least $1 - \delta$. This is essentially the same as theorem 5.5 (actually, a bit better).

5.4 The Effect of Boosting on Margin Distributions

The analyses given in sections 5.2 and 5.3 apply to any voting classifier, not just those produced by boosting. In this section, we give theoretical evidence that AdaBoost is especially suited to the task of maximizing the number of training examples with large margin. Informally, this is because, at every round, AdaBoost puts the most weight on the examples with the smallest margins.

5.4.1 Bounding AdaBoost's Margins

In theorem 3.1, we proved that if the empirical γ-weak learning assumption holds or, more specifically, if the weighted training errors ϵ_t of the weak classifiers are all bounded below $\frac{1}{2} - \gamma$, then the training error of the combined classifier—that is, the fraction of training examples with margin below zero—decreases exponentially fast with the number of weak classifiers that are combined. Here, we extend this proof to give a more general bound on the fraction of training examples with margin below θ, for any $\theta \geq 0$. The resulting bound is in terms of the edges γ_t of the weak hypotheses, as well as θ, and shows that, under the same weak learning condition, if θ is not too large, then the fraction of training examples with margin below θ also decreases to zero exponentially fast with the number of rounds of boosting.

Note that theorem 3.1 is a special case of this theorem in which we set $\theta = 0$.

Theorem 5.8 Given the notation of algorithm 1.1 (p. 5), let $\gamma_t \doteq \frac{1}{2} - \epsilon_t$. Then the fraction of training examples with margin at most θ is at most

$$\prod_{t=1}^{T} \sqrt{(1+2\gamma_t)^{1+\theta}(1-2\gamma_t)^{1-\theta}}.$$

Proof Let f be as defined in equation (5.2). Note that $yf(x) \leq \theta$ if and only if

$$y \sum_{t=1}^{T} \alpha_t h_t(x) \leq \theta \sum_{t=1}^{T} \alpha_t,$$

which in turn holds if and only if

$$\exp\left(-y \sum_{t=1}^{T} \alpha_t h_t(x) + \theta \sum_{t=1}^{T} \alpha_t\right) \geq 1.$$

Thus,

$$\mathbf{1}\{yf(x) \le \theta\} \le \exp\left(-y\sum_{t=1}^{T}\alpha_t h_t(x) + \theta\sum_{t=1}^{T}\alpha_t\right).$$

Therefore, the fraction of training examples with margin at most θ is

$$\begin{aligned}
\mathbf{Pr}_S[yf(x) \le \theta] &= \frac{1}{m}\sum_{i=1}^{m}\mathbf{1}\{y_i f(x_i) \le \theta\} \\
&\le \frac{1}{m}\sum_{i=1}^{m}\exp\left(-y_i\sum_{t=1}^{T}\alpha_t h_t(x_i) + \theta\sum_{t=1}^{T}\alpha_t\right) \\
&= \frac{\exp\left(\theta\sum_{t=1}^{T}\alpha_t\right)}{m}\sum_{i=1}^{m}\exp\left(-y_i\sum_{t=1}^{T}\alpha_t h_t(x_i)\right) \\
&= \exp\left(\theta\sum_{t=1}^{T}\alpha_t\right)\left(\prod_{t=1}^{T}Z_t\right) \qquad (5.30)
\end{aligned}$$

where the last equality follows from the identical derivation used in the proof of theorem 3.1. Plugging in the values of α_t and Z_t from equation (3.9) gives the theorem. ∎

To get a feeling for this bound, consider what happens when, for all t, $\epsilon_t \le \frac{1}{2} - \gamma$ for some $\gamma > 0$. Given this assumption, we can simplify the upper bound in theorem 5.8 to

$$\left(\sqrt{(1-2\gamma)^{1-\theta}(1+2\gamma)^{1+\theta}}\right)^T.$$

When the expression inside the parentheses is strictly smaller than 1, that is, when

$$\sqrt{(1-2\gamma)^{1-\theta}(1+2\gamma)^{1+\theta}} < 1, \qquad (5.31)$$

this bound implies that the fraction of training examples with $yf(x) \le \theta$ decreases to zero exponentially fast with T, and must actually be equal to zero at some point since this fraction must always be a multiple of $1/m$. Moreover, by solving for θ, we see that equation (5.31) holds if and only if

$$\theta < \Upsilon(\gamma)$$

where

$$\Upsilon(\gamma) \doteq \frac{-\ln(1-4\gamma^2)}{\ln\left(\frac{1+2\gamma}{1-2\gamma}\right)}. \qquad (5.32)$$

This function is plotted in figure 5.4, where it can be seen that $\gamma \le \Upsilon(\gamma) \le 2\gamma$ for $0 \le \gamma \le \frac{1}{2}$, and that $\Upsilon(\gamma)$ is close to γ when γ is small. So, to rephrase, we have shown that

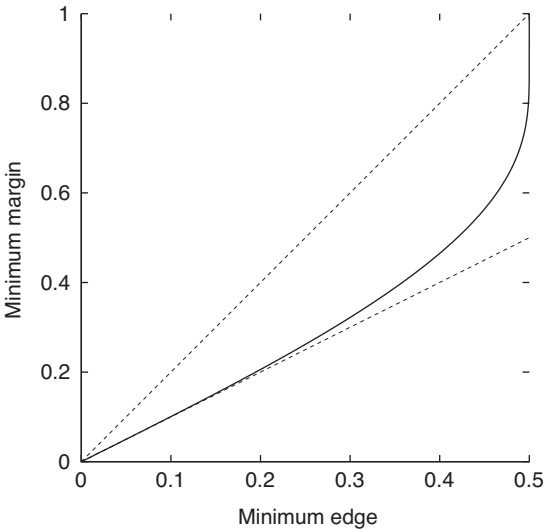

Figure 5.4
A plot of the minimum margin $\Upsilon(\gamma)$ guaranteed for AdaBoost as a function of the minimum edge γ. Also plotted are the linear lower and upper bounds, γ and 2γ.

if every weak hypothesis has edge at least γ (as will happen when the empirical γ-weak learning assumption holds), then in the limit of a large number of rounds T, all examples will eventually have margin at least $\Upsilon(\gamma) \geq \gamma$. In this sense, $\Upsilon(\gamma)$ bounds the minimum margin as a function of the minimum edge.

Thus, when the weak classifiers are consistently better than random guessing, the margins of the training examples are guaranteed to be large after a sufficient number of boosting iterations. Moreover, we see that there is a direct relationship at work here: The higher the edges γ_t of the weak classifiers, the higher the margins that will be attained by AdaBoost's combined classifier. This tight connection between edges and margins, which arose in section 3.2, turns out to be rooted in the game-theoretic view of boosting which will be explored in chapter 6.

This also suggests that stronger base classifiers, such as decision trees, which produce higher accuracy prediction rules, and therefore larger edges, will also yield larger margins and less overfitting, exactly as observed in the example in section 5.1. Conversely, weaker base classifiers, such as decision stumps, tend to produce smaller edges, and therefore also smaller margins, as can be seen, for instance, in the margin distributions shown in figure 5.5 on a benchmark dataset using stumps (see further discussion of this figure below). On the other hand, stronger base classifiers generally have higher complexity than weaker ones, and this complexity, according both to intuition and to the bounds in theorems 5.1 and 5.5, is likely to be a detriment to performance. Thus, we are again faced with the fundamental

trade-off between complexity (of the base classifiers) and fit to the data (as measured by their edges).

5.4.2 More Aggressive Margin Maximization

Theorem 5.8 shows that, under the empirical γ-weak learning assumption, all training examples will eventually have margin at least $\Upsilon(\gamma) \geq \gamma$. This is encouraging since the analysis in section 5.2 suggests that larger margins are conducive to better generalization. However, this turns out not to be the best that can be done. Although in practice AdaBoost often seems to achieve the largest possible minimum margin (that is, the smallest of the margins of the training examples), theoretically it can be shown that $\Upsilon(\gamma)$ is the best general bound that can be proved on the minimum margin attained by AdaBoost under the γ-weak learning assumption (see exercise 5.1). In contrast, it turns out that other methods can achieve a margin of 2γ, which is roughly twice as large as $\Upsilon(\gamma)$ when γ is small.

In fact, the proof of theorem 5.8 can be used to derive variations of AdaBoost for more directly maximizing the number of training examples with margin above some prespecified level θ. AdaBoost, as was seen in the proof of theorem 3.1, was derived for the purpose of minimizing the usual training error $\mathbf{Pr}_S[yf(x) \leq 0]$. Suppose instead that our goal is to minimize $\mathbf{Pr}_S[yf(x) \leq \theta]$ for a chosen value of θ. Then equation (5.30) combined with equation (3.8) tells us generally that

$$\mathbf{Pr}_S[yf(x) \leq \theta] \leq \prod_{t=1}^{T} \left[e^{(\theta-1)\alpha_t} \left(\tfrac{1}{2} + \gamma_t\right) + e^{(\theta+1)\alpha_t} \left(\tfrac{1}{2} - \gamma_t\right) \right]. \tag{5.33}$$

Rather than choosing α_t as in AdaBoost, we can instead select α_t to minimize equation (5.33) directly. Doing so gives

$$\alpha_t = \frac{1}{2} \ln\left(\frac{1+2\gamma_t}{1-2\gamma_t}\right) - \frac{1}{2} \ln\left(\frac{1+\theta}{1-\theta}\right), \tag{5.34}$$

which is smaller than the AdaBoost choice by the additive constant appearing as the rightmost term of this expression. Assuming each $\alpha_t \geq 0$ (which is equivalent to assuming $\gamma_t \geq \theta/2$), we can plug this choice into equation (5.33), which gives a bound that can be written succinctly as

$$\mathbf{Pr}_S[yf(x) \leq \theta] \leq \exp\left(-\sum_{t=1}^{T} \mathrm{RE}_b\left(\frac{1}{2} + \frac{\theta}{2} \;\|\; \frac{1}{2} + \gamma_t\right)\right). \tag{5.35}$$

Here, $\mathrm{RE}_b(p \;\|\; q)$, for $p, q \in [0, 1]$, is the *(binary) relative entropy*:

$$\mathrm{RE}_b(p \;\|\; q) = p \ln\left(\frac{p}{q}\right) + (1-p) \ln\left(\frac{1-p}{1-q}\right), \tag{5.36}$$

which is really just a special case of the more general relative entropy encountered in section 6.2.3 applied to Bernoulli distributions $(p, 1-p)$ and $(q, 1-q)$. As in the general case, binary relative entropy is always nonnegative, and is equal to zero if and only if $p = q$. Furthermore, it is increasing in q for $q \geq p$. See section 8.1.2 for further background.

So if θ is chosen ahead of time, and if the γ-weak learning assumption holds for some $\gamma > \theta/2$, then the fraction of training examples with margin at most θ will be no more than

$$\exp\left(-T \cdot \mathrm{RE}_b\left(\frac{1}{2}+\frac{\theta}{2} \;\middle\|\; \frac{1}{2}+\gamma\right)\right),$$

which tends to zero exponentially fast in the number of rounds T. Thus, when T is sufficiently large, all of the training examples will have margin at least θ. If γ is known ahead of time, then θ can be chosen to be slightly smaller than 2γ. This shows that, with additional information regarding the edges, AdaBoost can be modified so that all training examples will have margins arbitrarily close to 2γ, roughly twice the bound that we derived from theorem 5.8 for (unmodified) AdaBoost, and also the best bound attainable by any algorithm, as will be discussed in section 5.4.3.

When γ is not known ahead of time, methods have been developed, such as arc-gv and AdaBoost$^*_\nu$, for adjusting θ dynamically so as to achieve the same bound on the margins without such prior information (see exercise 5.3). In this fashion, AdaBoost can be modified so that the minimum margin provably converges to the largest value possible. Theorems 5.1 and 5.5, which say roughly that larger margins are better, suggest that this should benefit the algorithm's performance. However, in practice such methods often fail to give improvement, apparently for two reasons. First, by attempting to more aggressively maximize the minimum margin, the base learning algorithm is often forced to return base classifiers of higher complexity so that the complexity terms ($\lg|\mathcal{H}|$ or d) appearing in these theorems will effectively be larger, counteracting improvements in the margin. This can especially be a problem with very flexible base classifiers, such as decision trees, which can vary considerably in complexity based on overall size and depth.

For instance, this can be seen in table 5.1, which shows the results of running AdaBoost and arc-gv on five benchmark datasets using the decision-tree algorithm CART as base learner. Arc-gv consistently gives larger minimum margins than AdaBoost, but also gives consistently higher test error. Although an attempt was made in these experiments to control complexity by forcing CART always to return trees with a fixed number of nodes, a more careful examination of the results shows that when run with arc-gv, CART is likely to produce deeper, skinnier trees which, it can be argued, tend to be more specialized in their predictions and thus more prone to overfitting.

Even when the base-classifier complexity can be controlled (for instance, by using decision stumps), there may be a second reason for a lack of improvement. Although such methods may succeed at increasing the *minimum* margin among all training examples, this increase may come at the expense of the vast majority of the other training examples,

Table 5.1
Test errors (in percent), minimum margins, and average tree depths, averaged over ten trials, for AdaBoost and arc-gv, run for 500 rounds using CART decision trees pruned to 16-leaf nodes as weak classifiers

	Test Error		Minimum Margin		Tree Depth	
	arc-gv	AdaBoost	arc-gv	AdaBoost	arc-gv	AdaBoost
breast cancer	3.04	2.46	0.64	0.61	9.71	7.86
ionosphere	7.69	3.46	0.97	0.77	8.89	7.23
ocr 17	1.76	0.96	0.95	0.88	7.47	7.41
ocr 49	2.38	2.04	0.53	0.49	7.39	6.70
splice	3.45	3.18	0.46	0.42	7.12	6.67

so that although the minimum margin increases, the bulk of the margin distribution actually decreases. Note that the bounds in theorems 5.1 and 5.5 depend on the *entire* margin distribution, not just the minimum margin.

For instance, figure 5.5 shows the margin distributions produced when running AdaBoost and arc-gv using decision stumps as the weak hypotheses on one of the benchmark datasets. Arc-gv does indeed achieve higher *minimum* margin (-0.01 for arc-gv versus -0.06 for AdaBoost), but, as the figure shows, the bulk of the training examples have substantially higher margin for AdaBoost.

5.4.3 A Necessary and Sufficient Condition for Weak Learnability

In section 3.2, we gave a sufficient condition for the empirical γ-weak learning assumption to hold, namely, that the training data be linearly separable with margin 2γ, meaning that there exists some linear threshold function (that is, some combined classifier) under which every training example has margin at least 2γ. Now we have the tools to prove the exact converse, and to show that this condition is both sufficient and necessary. Suppose the empirical γ-weak learning assumption holds. Then the argument above shows that modified AdaBoost, for any $\theta < 2\gamma$, will find a combined classifier under which all training examples have margin at least θ, in other words, witnessing that the data is linearly separable with margin θ. Since θ can be made arbitrarily close to 2γ, this essentially proves the converse. Thus, there exists a combined classifier for which every training example has margin at least 2γ if and only if for every distribution over the training set there exists a weak hypothesis with edge at least γ.

Furthermore, we can define a natural notion of *optimal margin*, meaning the largest value θ^* such that for some combined classifier, every training example has margin at least θ^*. And we can define a corresponding notion of *optimal edge*, meaning the largest value γ^* such that for every distribution, there is some weak hypothesis with edge at least γ^*. (Like the other concepts in this section, both of these are defined with respect to a particular dataset and hypothesis space.) Then the equivalence outlined above implies further that the optimal edge is equal to some value γ^* if and only if the optimal margin is $2\gamma^*$.

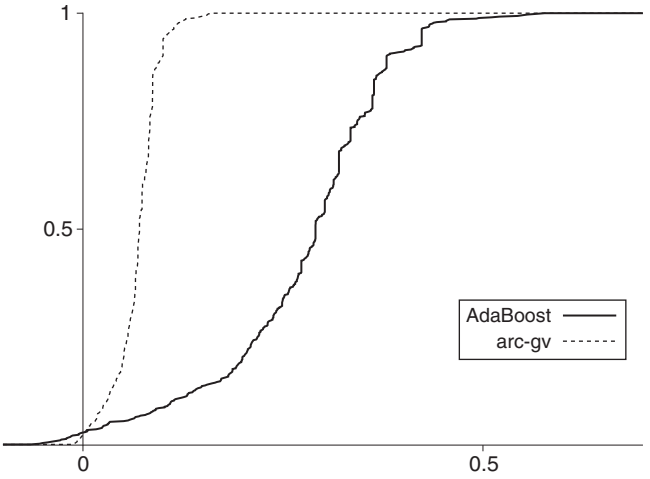

Figure 5.5
Cumulative margins for AdaBoost and arc-gv for the breast cancer dataset after 100 rounds of boosting on decision stumps.

Here, we again encounter the inseparable relationship between edges and margins. Understood more deeply, this equivalence between margins and edges, and between linear separability and the empirical weak learning assumption, turns out to be a direct consequence of fundamental results of game theory, as will be seen in chapter 6.

5.5 Bias, Variance, and Stability

In this chapter, we have presented an explanation of AdaBoost's successes and failures in terms of the margins theory. One of the main alternative explanations for the improvements achieved by voting classifiers is based instead on separating the expected generalization error of a classifier into a *bias* term and a *variance* term. While the details of these definitions differ from author to author, they are all attempts to capture the following quantities: The bias term measures the *persistent* error of the learning algorithm, the error that would remain even if we had an infinite number of independently trained classifiers. The variance term measures the error that is due to *fluctuations* that are a part of generating a single classifier. The idea is that by averaging over many classifiers, one can reduce the variance term and in that way reduce the expected error. In this section, we discuss a few of the strengths and weaknesses of bias-variance theory as an explanation for the performance of voting methods, especially boosting.

The origins of bias-variance analysis are in quadratic regression where performance is measured using the squared error (see chapter 7). Averaging several independently trained

regression functions will never increase the expected error. This encouraging fact is nicely reflected in the bias-variance separation of the expected quadratic error. Both bias and variance are always nonnegative, and averaging decreases the variance term without changing the bias term.

One would naturally hope that this beautiful analysis would carry over from quadratic regression to classification. Unfortunately, taking the majority vote over several classification rules can sometimes result in an *increase* in the expected classification error (and we will shortly see an example of how voting can make things worse). This simple observation suggests that it may be inherently more difficult or even impossible to find a bias-variance decomposition for classification as natural and satisfying as in quadratic regression. This difficulty is reflected in the myriad definitions that have been proposed for bias and variance.

The principle of variance reduction is the basis of other voting methods, notably *bagging*. This is a procedure quite similar to boosting, but one in which the distributions D_t are fixed for all iterations to be uniform over the training set, and resampling, as in section 3.4.1, is always employed so that the base classifiers are each trained on so-called *bootstrap samples* of the data. That is, on each round t, the base learner is trained on a dataset consisting of m examples, each selected uniformly at random from the original dataset (with replacement, of course). Thus, some examples will be included more than once in a given dataset, while more than a third, on average, will be omitted entirely.

The notion of variance certainly seems to be helpful in understanding bagging; empirically, bagging appears to be most effective for learning algorithms with large variance which are unstable in the sense that small changes in the data can cause large changes in the learned classifier. In fact, variance has sometimes been *defined* to be the amount of decrease in error effected by bagging a large number of base classifiers under idealized conditions. This ideal situation is one in which the bootstrap samples used in bagging faithfully approximate truly independent samples. However, this assumption can fail to hold in practice, in which case bagging may not perform as well as expected, even when variance dominates the error of the base learning algorithm.

It has been argued that boosting is also primarily a variance-reducing procedure. Some of the evidence for this comes from the observed effectiveness of boosting when used with decision-tree learning algorithms like C4.5 or CART, algorithms known empirically to have high variance. As the error of these algorithms is mostly due to variance, it is not surprising that the reduction in the error is primarily due to a reduction in the variance. However, boosting can also be highly effective when used with learning algorithms whose error tends to be dominated by bias rather than variance. Indeed, boosting is intended for use with quite weak base learning algorithms, such as decision stumps, which often have high bias and low variance.

To illustrate this point, table 5.2 shows the results of running boosting and bagging on three artificial datasets on training sets of size 300. For the base learning algorithm, both the decision-tree algorithm C4.5 (section 1.3) and decision stumps (section 3.4.2) were used.

Table 5.2
Results of bias-variance experiments using boosting and bagging on three synthetic datasets

		Kong & Dietterich Definitions						Breiman Definitions					
		Stumps			C4.5			Stumps			C4.5		
Name		–	Boost	Bag	–	Boost	Bag	–	Boost	Bag	–	Boost	Bag
twonorm	bias	2.5	0.6	2.0	0.5	0.2	0.5	1.3	0.3	1.1	0.3	0.1	0.3
	variance	28.5	2.3	17.3	18.7	1.8	5.4	29.6	2.6	18.2	19.0	1.9	5.6
	error	33.3	5.3	21.7	21.6	4.4	8.3	33.3	5.3	21.7	21.6	4.4	8.3
threenorm	bias	24.5	6.3	21.6	4.7	2.9	5.0	14.2	4.1	13.8	2.6	1.9	3.1
	variance	6.9	5.1	4.8	16.7	5.2	6.8	17.2	7.3	12.6	18.8	6.3	8.6
	error	41.9	22.0	36.9	31.9	18.6	22.3	41.9	22.0	36.9	31.9	18.6	22.3
ringnorm	bias	46.9	4.1	46.9	2.0	0.7	1.7	32.3	2.7	37.6	1.1	0.4	1.1
	variance	–7.9	6.6	–7.1	15.5	2.3	6.3	6.7	8.0	2.2	16.4	2.6	6.9
	error	40.6	12.2	41.4	19.0	4.5	9.5	40.6	12.2	41.4	19.0	4.5	9.5

For each dataset and each learning method, bias, variance, and generalization error rate were estimated, then reported in percent, using two sets of definitions for bias and variance. Both C4.5 and decision stumps were used as base learning algorithms. Columns headed with a dash indicate that the base learning algorithm was run by itself.

Bias, variance, and average generalization error were estimated by rerunning each algorithm many times. Two different definitions of bias and variance were used, one due to Kong and Dietterich, and the other due to Breiman. (See the bibliographic notes for references with details.)

Clearly, these experiments show that boosting is doing more than reducing variance. For instance, on the "ringnorm" dataset, boosting decreases the overall error of the stump algorithm from 40.6% to 12.2%, but *increases* the variance from −7.9% to 6.6% using Kong and Dietterich's definitions, or from 6.7% to 8.0% using Breiman's definitions. The decrease in error is instead due to a very substantial drop in the bias.

The view of boosting as mainly a variance-reducing procedure predicts that boosting will fail when combined with a "stable" learning algorithm with low variance. This is clearly false, as the experiments above demonstrate. The theory presented in this chapter suggests a different characterization of the cases in which boosting might fail. Theorems 5.1 and 5.5, together with theorem 5.8, predict that boosting can perform poorly only when either (1) there is insufficient training data relative to the complexity of the base classifiers, or (2) the training errors of the base classifiers (the ϵ_t's in theorem 5.8) become too large too quickly.

Moreover, although bagging was originally introduced as a method based on variance reduction, it too can be analyzed using the part of the margins theory developed in section 5.2 since this theory is generally applicable to any voting method, including bagging. Such an analysis would, as usual, be in terms of the margin distribution, as well as base-classifier complexity and training set size, and would not depend on the number of rounds of bagging. In the case of bagging, the margin of a training example is simply a measure of the fraction of selected base classifiers that correctly classify it, a quantity that must converge after a large number of rounds to the probability of a base classifier, randomly generated according to the bootstrap process, correctly classifying it. Thus, this analysis predicts little or no overfitting, while providing nonasymptotic bounds on performance in terms of intuitive quantities. As an example, figure 5.6 shows the learning curves and margin distribution when bagging is used instead of boosting with the same base learner and dataset as in section 5.1, for comparison with figures 1.7 and 5.2. As is typical, bagging's margin distribution has a qualitatively different form than boosting's, but nevertheless shows that a fairly small fraction of the examples have low margin (though not as few as with boosting, in this case).

The bias-variance interpretation of boosting and other voting methods is closely related to an intuition that averaging (or really voting) many classifiers is sure to lead to better predictions than the individual base classifiers, just as one expects that the average of many estimates (say, of the bias of a coin) will be better than the individual estimates. This view is supported by a supposition that a combined classifier formed by voting does not have higher complexity than the base classifiers. Unfortunately, these intuitions do not hold true in general for classification problems. A majority-vote classifier may be substantially more complex and prone to overfitting than its constituent classifiers, which might be very simple.

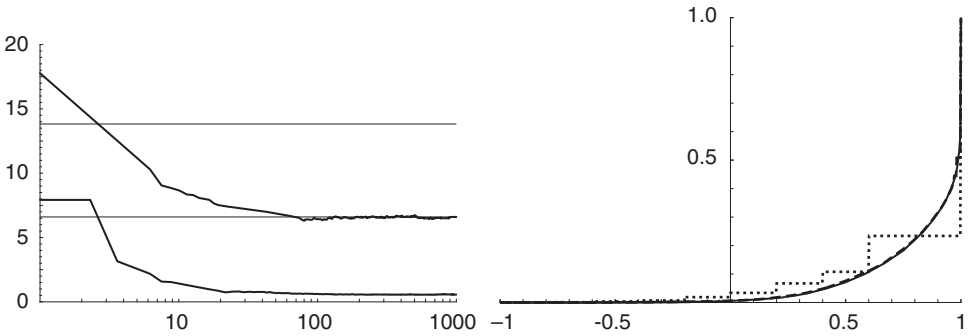

Figure 5.6
Results of running bagging on C4.5 on the letter dataset. The figure on the left shows the test (top) and training (bottom) percent error rates for bagging as a function of the number of rounds. The figure on the right shows the margin distribution graph. See the analogous figures 1.7 and 5.2 for further description. (Reprinted with permission of the Institute of Mathematical Statistics.)

As an example, suppose we use base classifiers that are delta-functions which predict $+1$ on a single point in the input space and -1 everywhere else, or vice versa (-1 on one point and $+1$ elsewhere), or that are constant functions predicting -1 everywhere or $+1$ everywhere. For any training set of size m, assuming the same instance never appears twice with different labels, and for any distribution D over this set, there must always exist a delta-function with error (with respect to D) at most

$$\frac{1}{2} - \frac{1}{2m}.$$

This is because one training example (x_i, y_i) must have probability at least $1/m$ under D, so an appropriately constructed delta-function will classify x_i correctly, as well as at least half of the probability mass of the remaining examples. Thus, the empirical γ-weak learning assumption holds for $\gamma = 1/(2m)$, which implies that, by theorem 3.1, AdaBoost will eventually construct a combined classifier that correctly classifies all m training examples.

As discussed in chapter 2, the very fact that we can easily fit such a rule to *any* training set implies that we do not expect the rule to be very good on new test examples outside of the training set. In other words, the complexity of these voting rules is too large, relative to the size of the sample, to make them useful. In fact, exactly this argument shows that their VC-dimension is infinite. Note that this complexity is entirely the result of voting. Each one of the delta-functions is very simple (the VC-dimension of this class is exactly 3), and would likely underfit most datasets. By voting many such simple rules, we end up with a combined classifier that is instead overly complex, one that would certainly overfit nearly any dataset.

Our analysis shows that AdaBoost controls the complexity of the combined classifier by striving for one with large margins. Indeed, when large margins can be attained, theorems 5.1 and 5.5 show that AdaBoost will perform as if the complexity of the combined classifier is on the same order as that of the *base* classifiers, so that the penalty for forming a majority vote of a large number of these is minimized.

AdaBoost's predicted poor performance in the example above is entirely consistent with our margin-based analysis; if AdaBoost is run for a long time, as noted earlier, all of the training examples will be correctly classified, but only with tiny margins of size $O(1/m)$, far too small to predict good generalization performance. (To be meaningful, theorems 5.1 and 5.5 require margins of size at least $\Omega(1/\sqrt{m})$.)

5.6 Relation to Support-Vector Machines

Boosting is not the only classification method that seems to operate on the principle of (approximate) margin maximization. In particular, *support-vector machines (SVMs)*, which are based explicitly on this principle, are currently very popular due to their effectiveness for general machine-learning tasks. Although boosting and SVMs are both learning methods based on maximization of quantities referred to loosely as "margins," we will see in this section how they differ significantly in important respects.

5.6.1 Brief Overview of SVMs

Since a full treatment of SVMs is well beyond the scope of this book, we give only an overview of the main ingredients of this approach.

Let us for now suppose that the instances $\mathbf{x}$ being classified are actually points in Euclidean space $\mathbb{R}^n$. Thus, the learner is given $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)$ where $\mathbf{x}_i \in \mathbb{R}^n$ and $y_i \in \{-1, +1\}$. For instance, we might be given the examples in figure 5.7, where $n = 2$. Already we see an important difference from boosting: SVMs are based on a strongly geometrical view of the data.

Given such data, the first idea of SVMs is to find a linear classifier, or linear threshold function, that correctly labels the data. In general, if there is even one, then there are likely to be many such linear classifiers. Rather than choosing one arbitrarily, in SVMs, we choose the hyperplane which separates the positive examples from the negative examples, and is maximally far from the closest data point. For instance, in figure 5.7 we might find a hyperplane (in this case, a line) like the one shown so as to maximize the indicated separation distance. Thus, not only do we want to correctly classify the training points, we also want those training points to be as far from the dividing boundary as possible.

More formally, a separating hyperplane is given by the equation[2] $\mathbf{w} \cdot \mathbf{x} = 0$, where $\mathbf{w}$, without loss of generality, has unit length ($\|\mathbf{w}\|_2 = 1$). An instance $\mathbf{x}$ is classified by such a

2. We have simplified our discussion by assuming that the hyperplane passes through the origin.

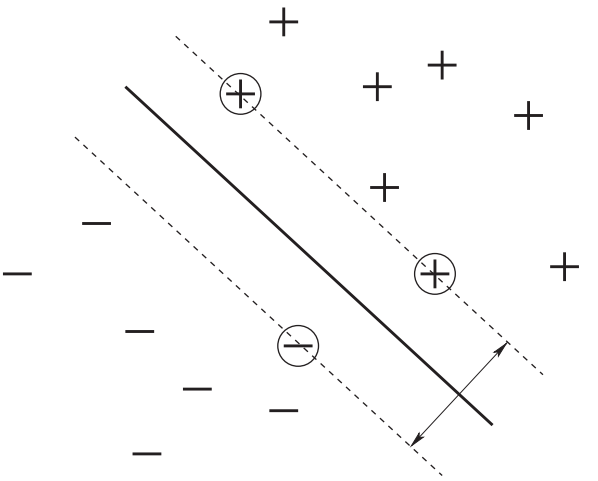

Figure 5.7
Sample data in two dimensions, and the separating hyperplane (a line in this case) that might be found by SVMs in this case. The *support vectors*, the examples closest to the hyperplane, have been circled.

hyperplane according to which side it falls on, that is, using the prediction rule

$$\text{sign}(\mathbf{w} \cdot \mathbf{x}).$$

With respect to the hyperplane defined by $\mathbf{w}$, the (signed) distance of an example from the separating hyperplane is called the *margin*. As we will see, it is related to, but distinct from, the margin used in boosting. The margin of example $(\mathbf{x}, y)$ can be computed to be $y(\mathbf{w} \cdot \mathbf{x})$. The margin of an entire training set is the minimum of the margins of the individual training examples, that is, $\min_i y_i(\mathbf{w} \cdot \mathbf{x}_i)$. The idea then is to find the hyperplane $\mathbf{w}$ that maximizes this minimum margin.

Of course, it is well known that linear threshold functions are limited in their expressiveness, especially in low dimensions. Nevertheless, data that starts out being linearly inseparable in its original low-dimensional space may become separable if mapped into a higher-dimensional space.

For instance, the data in figure 5.8 is clearly linearly inseparable. However, suppose we map these two-dimensional points $\mathbf{x} = \langle x_1, x_2 \rangle$ into $\mathbb{R}^6$ by the map

$$\mathbf{h}(\mathbf{x}) = \mathbf{h}(x_1, x_2) \doteq \langle 1, x_1, x_2, x_1 x_2, x_1^2, x_2^2 \rangle.$$

Then a linear hyperplane defined on these mapped points has the form

$$\mathbf{w} \cdot \mathbf{h}(\mathbf{x}) = w_1 + w_2 x_1 + w_3 x_2 + w_4 x_1 x_2 + w_5 x_1^2 + w_6 x_2^2 = 0$$

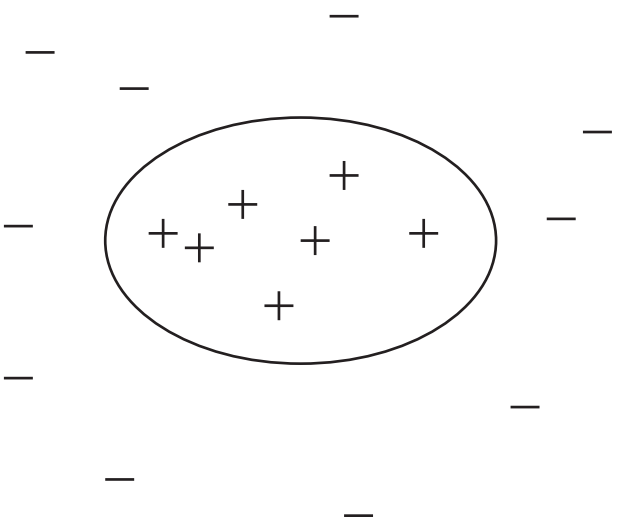

Figure 5.8
Data in two dimensions that cannot be linearly separated, but can be separated using an ellipse or, equivalently, a hyperplane following projection into six dimensions.

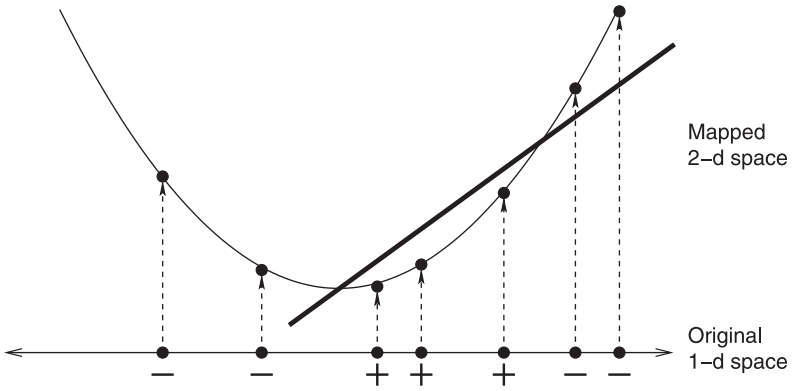

Figure 5.9
In their original, one-dimensional space, the seven points comprising this dataset are evidently linearly inseparable. However, when each point x is mapped to the two-dimensional vector $\langle x, x^2 \rangle$, that is, onto the parabola shown in the figure, the data now becomes linearly separable.

for scalars $w_1, \ldots, w_6$. In other words, a linear hyperplane in the mapped space can be used to represent any conic section in the original space, including, for instance, the ellipse in figure 5.8, which clearly does separate the positive and negative examples. An even simpler example is shown in figure 5.9.

Thus, in general, the instances $\mathbf{x} \in \mathbb{R}^n$ may be mapped to a higher-dimensional space $\mathbb{R}^N$ using a map $\mathbf{h}$, simply by replacing all appearances of $\mathbf{x}$ in the algorithm with $\mathbf{h}(\mathbf{x})$. In this example, $n = 2$ dimensions were mapped to $N = 6$. In practice, however, points starting out in a reasonable number of dimensions (say, 100) can easily end up being mapped into an extremely large number of dimensions (perhaps in the billions, or worse), so this would seem to be a very expensive computational operation.

Fortunately, in many cases a remarkable technique based on *kernels* can be applied to make for computational feasibility. It turns out that the only operation that is needed to implement SVMs is inner product between pairs of (mapped) points, that is, $\mathbf{h}(\mathbf{x}) \cdot \mathbf{h}(\mathbf{z})$. This sometimes can be done very efficiently. For instance, we can modify the example above slightly so that

$$\mathbf{h}(\mathbf{x}) = \mathbf{h}(x_1, x_2) \doteq \langle 1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2, x_1^2, x_2^2 \rangle.$$

The insertion of a few constants does not change the expressiveness of the linear threshold functions that can be computed using this mapping, but now it can be verified that

$$\begin{aligned} \mathbf{h}(\mathbf{x}) \cdot \mathbf{h}(\mathbf{z}) &= 1 + 2x_1z_1 + 2x_2z_2 + 2x_1x_2z_1z_2 + x_1^2z_1^2 + x_2^2z_2^2 \\ &= (1 + \mathbf{x} \cdot \mathbf{z})^2. \end{aligned} \tag{5.37}$$

Thus, the inner product of mapped points can be computed *without ever expanding explicitly* into the higher-dimensional space, but rather simply by taking inner product in the *original* low-dimensional space, adding 1, and squaring.

The function on the right-hand side of equation (5.37) is called a *kernel function*, and there are many other such functions which make it possible to implement SVMs even when mapping into very high-dimensional spaces. The computational savings effected by this trick can be tremendous. For instance, generalizing the example above, if we wanted to add all terms up to degree k (rather than degree 2, as above), so that we are mapping from n dimensions to $O(n^k)$ dimensions, we can compute inner products in this very high-dimensional space using a kernel identical to the one in equation (5.37), but with the exponent 2 replaced by k; this kernel can be computed in $O(n + \ln k)$ time. Using kernels to quickly compute inner products in very high-dimensional spaces is the second key ingredient of SVMs.

Many types of kernels have been developed; the polynomial kernels above are just one example. In fact, kernels can even be defined on objects other than vectors, such as strings and trees. Because the original objects need not be vectors, we therefore revert in the following to writing instances as the more generic x rather than $\mathbf{x}$.

Although the computational difficulty of mapping to a very high-dimensional space can sometimes be made tractable, there remains the statistical "curse of dimensionality," which suggests that generalization based on high-dimensional data (relative to the number of training examples) is likely to be poor. Indeed, the VC-dimension of general linear threshold functions in $\mathbb{R}^N$ is equal to N (see lemma 4.1), suggesting that the number of training examples must be on the same order as the number of dimensions. However, the VC-dimension of linear threshold functions with large margin may be much lower. In particular, suppose without loss of generality that all examples are mapped inside a unit ball so that $\|\mathbf{h}(x)\|_2 \leq 1$. Then it can be shown that the VC-dimension of linear threshold functions with margin $\gamma > 0$ is at most $1/\gamma^2$, regardless of the number of dimensions. This suggests that generalization may be possible even in extremely high-dimensional spaces, provided it is possible to achieve large margins.

5.6.2 Comparison to Boosting

When using a mapping function $\mathbf{h}$ as above, the linear classifier produced by SVMs has the form

$$\text{sign}(\mathbf{w} \cdot \mathbf{h}(x)).$$

AdaBoost, on the other hand, computes a final classifier of the form given in equation (5.3):

$$\text{sign}\left(\sum_{t=1}^{T} a_t h_t(x)\right),$$

where the a_t's, as in equation (5.1), are nonnegative and sum to 1. In fact, with a little bit more notation, these can be seen to be of exactly the same form as in SVMs. For simplicity, let us assume that the base-classifier space $\mathcal{H}$ is finite, and consists of the functions $\hbar_1, \ldots, \hbar_N$. Then we can define a vector

$$\mathbf{h}(x) \doteq \langle \hbar_1(x), \ldots, \hbar_N(x) \rangle.$$

Although $\mathcal{H}$ is finite, it will typically be huge, so $\mathbf{h}(x)$ is an extremely high-dimensional vector. On each round t of boosting, one coordinate j_t of this vector is selected corresponding to the chosen base classifier $h_t = \hbar_{j_t}$. By setting

$$w_j = \sum_{t : j_t = j} a_t$$

for $j = 1, \ldots, N$, we can also define a weight vector $\mathbf{w} \in \mathbb{R}^N$ in terms of the a_t's so that

$$\mathbf{w} \cdot \mathbf{h}(x) = \sum_{t=1}^{T} a_t h_t(x).$$

Thus, AdaBoost's final classifier now has the identical form as SVMs, both being linear threshold functions, though over rather different spaces. This representation also emphasizes the fact that AdaBoost, like SVMs, employs a mapping $\mathbf{h}$ into a very high-dimensional space; indeed, as already noted, the number of dimensions of the mapped space is equal to the cardinality of the *entire* space of base classifiers—typically, an extremely large space.

As noted earlier, SVMs and boosting can both be understood and analyzed as methods for maximizing some notion of margin. However, the precise forms of margin used for the two methods are different in subtle but important ways. The margin used in SVMs for an example (x, y) is defined to be $y(\mathbf{w} \cdot \mathbf{h}(x))$. The margin used in boosting would appear to be identical:

$$yf(x) = y \sum_{t=1}^{T} a_t h_t(x) = y(\mathbf{w} \cdot \mathbf{h}(x)).$$

However, there is a major difference not revealed by the notation. In analyzing SVMs, we assumed that $\mathbf{w}$ has unit Euclidean length (so that $\|\mathbf{w}\|_2 = 1$) and, moreover, that $\mathbf{h}$ maps into the unit ball so that $\|\mathbf{h}(x)\|_2 \leq 1$ for all x. In contrast, for boosting we found it natural to normalize the weights a_t so that $\sum_{t=1}^{T} |a_t| = 1$, that is, so that $\|\mathbf{w}\|_1 = 1$. Further, the coordinates of the mapping $\mathbf{h}$ correspond to base classifiers, each with range $\{-1, +1\}$. Thus,

$$\max_j |h_j(x)| = 1$$

or, more succinctly, $\|\mathbf{h}(x)\|_\infty = 1$ for all x. (See appendix A.2 for more about ℓ_p-norms.)

Thus, both definitions of margin assume that the weight vector $\mathbf{w}$ and the map $\mathbf{h}$ are bounded, but using different norms. The SVM approach, being intrinsically geometrical, uses Euclidean norms, while boosting utilizes the ℓ_1- and ℓ_∞-norms.

This choice of norms can make a big difference. For instance, suppose all the components of $\mathbf{h}(x)$ have range $\{-1, +1\}$, and that the weight vector $\mathbf{w}$ assigns unit weights to k of the N coordinates (where k is odd), and zero weight to all others. In other words, $\text{sign}(\mathbf{w} \cdot \mathbf{h}(x))$ is computing a simple majority vote of k of the dimensions or base classifiers. Although overly simplistic, this is suggestive of learning problems in which only a subset of a very large number of features/dimensions/base classifiers are actually relevant to what is being learned. Normalizing appropriately, we see that the boosting (ℓ_1/ℓ_∞) margin of this classifier is $1/k$, which is reasonable if k is not too large. Also, this margin is independent of the number of dimensions N. On the other hand, the SVM (ℓ_2/ℓ_2) margin would be $1/\sqrt{kN}$, which could be far worse if N is very large. Other examples in which the SVM margin is far superior can also be constructed.

There is another important difference between boosting and SVMs. Both aim to find a linear classifier in a very high-dimensional space. However, computationally they are quite

different in how they manage to do this: SVMs use the method of kernels to perform computations in the high-dimensional space, while boosting relies on a base learning algorithm that explores the high-dimensional space one coordinate at a time.

Finally, we point out that the SVM approach is predicated on explicitly maximizing the *minimum* margin (although some variants relax this objective somewhat). AdaBoost, as discussed in section 5.4, does not provably maximize the minimum margin, but only tends to increase the overall distribution of margins, a property that empirically seems sometimes to be advantageous.

5.7 Practical Applications of Margins

Although we have focused largely on their theoretical utility, in practical terms margins can be quite useful as a reasonable measure of confidence. In this section, we describe two applications of this principle.

5.7.1 Rejecting Low-Confidence Predictions for Higher Accuracy

As previously discussed, the larger the magnitude of the margin, the greater our confidence in the predictions of the combined classifier. Intuitively, and also in line with our earlier theoretical development, we expect such high-confidence examples to have a correspondingly greater chance of being correctly classified. Moreover, note that the absolute margin—that is, $|yf(x)| = |f(x)|$—can be computed without knowledge of the label y. As we will see, these properties can be very useful in applications that demand high accuracy predictions, even if they are limited to only a part of the domain, since such settings require the use of a classifier that "knows what it knows" (or does not know).

For example, consider a classifier that is used as part of a spoken-dialogue system to categorize verbal utterances according to their meaning (see section 10.3). Knowing that a particular classification was made with high confidence means that it can be relied upon by the rest of the system. On the other hand, a low-confidence classification can be handled accordingly, for instance, by asking the user to repeat a response or to provide further information. Likewise, a classifier designed for the automatic categorization of news articles by major topic can be used and trusted when producing high-confidence predictions, while articles classified with low confidence can be handed off to a human for manual annotation. In other tasks, like spam filtering, we may instead want to treat all low-confidence predictions as ham so that only email messages that are predicted spam with high confidence are filtered out, thus minimizing the number of legitimate emails mistaken for spam.

In general, we might select a threshold so that all predictions with absolute margin above this value are trusted for their "high" confidence, while those with "low" confidence, below the chosen threshold, are rejected, for instance, in one of the ways described above. The particular threshold value can be chosen based on performance on held-out data not used

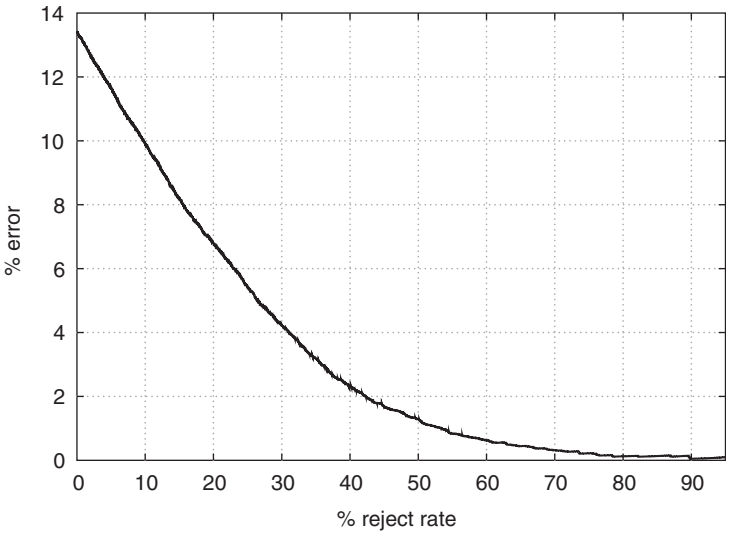

Figure 5.10
The trade-off between error and reject rate on the census dataset. One point is plotted for every possible margin threshold with the x-coordinate indicating the fraction of test examples rejected (that is, with absolute margin below threshold), and the y-coordinate giving the error as measured over the part of the test set not rejected (with margin above threshold).

for training, taking into account the requirements of the application. Naturally, the more examples rejected, the higher the accuracy on the examples that remain.

Figure 5.10 shows this trade-off on actual data. In this case, the instances are persons from a 1994 US Census database, each described by age, education, marital status, and so on. The problem is to predict whether or not a given individual's income exceeds \$50,000. In this experiment, AdaBoost was run on 10,000 training examples using decision stumps for 1000 rounds. (Also, real-valued weak hypotheses were employed, as described in chapter 9.)

On the entire test set of 20,000 examples, the overall test error was 13.4%. However, as the figure shows, a much lower error can be attained by rejecting a fraction of the test data. For instance, when the 20% of the test set with smallest absolute margins are rejected, the error on the remaining 80% drops by about half to 6.8%. A test error below 2% can be achieved at the cost of rejecting about 43% of the test examples. Thus, quite high accuracy can be attained on an identifiable and nonnegligible fraction of the dataset.

5.7.2 Active Learning

We turn next to a second application of margins. Throughout this book, we have taken for granted an adequate supply of labeled examples. In many applications, however, although there may well be an abundance of *un*labeled examples, we may find that reliable labels

Algorithm 5.1
An active learning method based on AdaBoost, using the absolute margin as a measure of confidence

Given: large set of unlabeled examples
limited annotation resources.
Initialize: choose an initial set of random examples for labeling.
Repeat:

- Train AdaBoost on all examples labeled so far.
- Obtain (normalized) final hypothesis $f(x)$ as in equation (5.2).
- Choose the k unlabeled examples x with minimum $|f(x)|$ for labeling.

are rather scarce due to the difficulty, expense, or time involved in obtaining human annotations. For example, in a vision task like face detection (section 3.4.3), it is not hard to gather thousands or millions of images, for instance, off the Internet. However, manually identifying all of the faces (and non-faces) in a large collection of images can be exceedingly slow and tedious. Likewise, in the kind of spoken-dialogue task mentioned earlier, obtaining recordings of utterances is relatively cheap; the expense is in annotating those recordings according to their proper categorization.

In such a setting where we have a large number of unlabeled examples but limited resources for obtaining labels, it makes sense to carefully and selectively choose which examples will be labeled, an approach known as *active learning*. Ideally, we would like to choose examples whose labels will be most "informative" and most helpful in driving the learning process forward. These are generally difficult notions to quantify and measure, especially without knowledge of the true labels. Nevertheless, intuitively, examples with low-confidence predictions are likely to have these properties: If we are very unsure of the correct label for a given example, then whatever that label turns out to be will be new information that can move learning forward. So the idea is to iteratively train a classifier using a growing pool of labeled examples where, on each iteration, the unlabeled examples we are least confident about are selected for labeling.

In boosting, as we have discussed at length, the absolute margin $|f(x)|$ can be used as a measure of confidence. Putting these ideas together leads to a procedure like algorithm 5.1. This simple approach to active learning can be surprisingly effective.

For instance, this approach has been applied to the spoken-dialogue task mentioned above and described in detail in section 10.3. Figure 5.11 shows how actively selecting examples for labeling in the manner described above compares with choosing the examples at random on each iteration. In these experiments, an initial set of 1000 examples was chosen for labeling, and $k = 500$ examples were added on each iteration. Decision stumps were used as described in section 10.3, and boosting was run for 500 rounds. These experiments

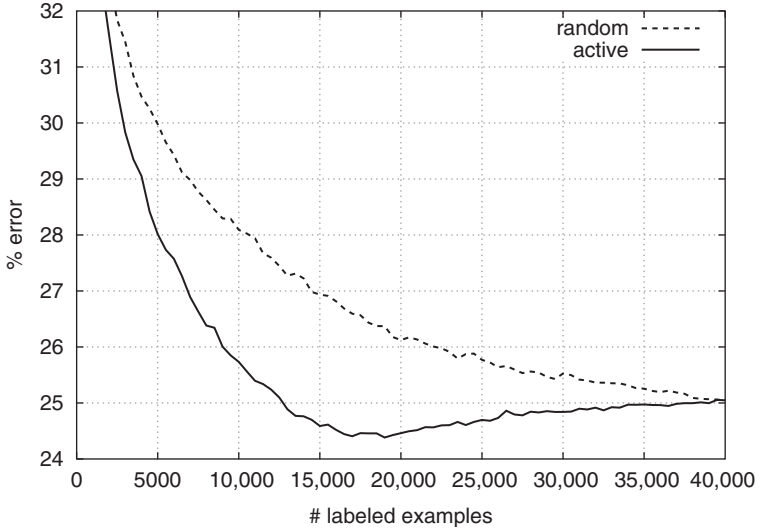

Figure 5.11
A comparison of the percent test error achieved on a spoken-dialogue task for an increasing number of labeled examples selected from a fixed pool using either active learning or random selection.

were repeated ten times and the results were averaged. In each case, examples were selected from a training pool of 40,000 examples, initially all unlabeled.[3]

In terms of labeling effort, figure 5.11 shows that the savings can be tremendous. For example, as shown in table 5.3, to obtain a test error rate of 25%, some 40,000 randomly selected examples must be labeled (that is, the entire training set), while only 13,000 actively selected examples suffice—more than a threefold savings.

In these controlled experiments, a fixed set of 40,000 examples was used for training so that the performance of both active and random selection must finally converge to the same point. This means that the latter part of the active-learning curve reflects the addition of less informative examples added later in the process, once all the other examples have been labeled, and suggests that even greater improvements in performance may be possible with larger pools of unlabeled examples. (In some applications, where a steady stream of unlabeled examples arrives every day, the supply is virtually infinite.) Furthermore, it is interesting that on this dataset, using only about half the data, if chosen selectively, gives better results than using the entire dataset: With 19,000 labeled examples, active learning gives a test error of 24.4% compared to 25.0% using the entire set of 40,000 examples.

3. Since this dataset is multiclass and multi-label, a modified definition of margin was used, namely, the difference of the "scores" predicted by the final classifier for the top two labels. Also, "one-error" was used instead of classification error. See chapter 10.

Table 5.3
The number of rounds needed to achieve various test error rates for the experimental results in figure 5.11, as well as the percentage labeling effort reduced by active learning

	First Reached		
% error	Random	Active	% Label Savings
27.0	14,500	7,000	51.7
26.0	22,000	9,000	59.1
25.0	40,000	13,000	67.5
24.5	–	16,000	–

Apparently, the examples labeled toward the end of the process not only are uninformative, but actually seem to have a "disinformative" effect, perhaps due to mislabeling.

Summary

We have explored in this chapter a theory for understanding AdaBoost's generalization capabilities in terms of its propensity to maximize the margins of the training examples. Specifically, the theory provides that AdaBoost's generalization performance is largely a function of the number of training examples, the complexity of the base classifiers, and the margins of the training examples. These margins in turn are intimately related to the edges of the base classifiers. The theory gives an explanation of why AdaBoost often does not overfit, as well as qualitative predictions of the conditions under which the algorithm can fail.

The margins theory seems to provide a more complete explanation for AdaBoost's behavior than bias-variance analysis does, and links AdaBoost with SVMs, another margins-based learning method. Unfortunately, attempts to directly apply insights from the theory as a means of improving AdaBoost have met with mixed success for a number of reasons. Even so, margins are practically useful as a natural measure of confidence.

In the following chapters, we turn to some alternative interpretations of the AdaBoost algorithm itself.

Bibliographic Notes

The margins explanation for the effectiveness of AdaBoost and other voting methods, as presented in sections 5.1 and 5.2 (including figure 5.2), is due to Schapire et al. [202]. Their analysis, in turn, was based significantly on a related result of Bartlett [11] for neural networks. An improved bound for the case in which all examples have margin above θ (as in the last paragraph of section 5.2) was proved by Breiman [36]. See also the refined analysis of Wang et al. [230] based on the notion of an "equilibrium margin."

The use of Rademacher complexity as a tool in the analysis of voting methods, as in section 5.3, was introduced by Koltchinskii and Panchenko [139]. An excellent review of these methods, including proofs and references, is given by Boucheron, Bousquet, and Lugosi [30].

Theorem 5.8 was proved by Schapire et al. [202]. The bound $\Upsilon(\gamma)$ derived in section 5.4.1 on the asymptotic minimum margin is due to Rätsch and Warmuth [187]. This bound was shown to be tight by Rudin, Schapire, and Daubechies [196]. That AdaBoost need not always achieve the largest possible minimum margin, even when using an exhaustive weak learner, was first proved by Rudin, Daubechies, and Schapire [194].

The modified version of AdaBoost given by the choice of α_t in equation (5.34) was initially studied by Rätsch et al. [186] and Breiman [36], and later by Rätsch and Warmuth [187], who called the resulting algorithm AdaBoost$_\rho$, and who gave an analysis similar to the one in section 5.4.2. The algorithms arc-gv and AdaBoost$^*_\nu$, which provably maximize the minimum margin, are due to Breiman [36] and Rätsch and Warmuth [187], respectively. Other algorithms with this property have also been given by Grove and Schuurmans [111], Rudin, Schapire, and Daubechies [196], and Shalev-Shwartz and Singer [211]. A different approach for directly optimizing the margins (though not necessarily the minimum margin) is given by Mason, Bartlett, and Baxter [167].

Breiman [36] conducted experiments with arc-gv which showed that it tends to achieve higher margins than AdaBoost, but also slightly higher test errors. Results of a similar flavor were also obtained by Grove and Schuurmans [111]. The experiments reported in table 5.1 and figure 5.5, as well as the explanation given of Breiman's findings, are due to Reyzin and Schapire [188].

As noted in chapter 3, the connection between optimal margins and optimal edges, as in section 5.4.3, was first made explicit by Rätsch and Warmuth [187].

Bagging, as discussed in section 5.5, is due to Breiman [34], who also proposed a bias-variance explanation for both bagging's and boosting's effectiveness [35]. The definitions of bias and variance used here are due to Breiman [35] and Kong and Dietterich [140], although others have been proposed [138, 217]. The main arguments and results of this section, including table 5.2 (adapted) and figure 5.6, are taken from Schapire et al. [202]. The synthetic datasets used in table 5.2 are from Breiman [35]. Bagging is closely related to Breiman's random forests [37], another highly effective method for combining decision trees.

Support-vector machines were pioneered by Boser, Guyon, and Vapnik [29] and Cortes and Vapnik [56]. See also, for instance, the books by Cristianini and Shawe-Taylor [58], and Schölkopf and Smola [208]. The comparison with boosting given in section 5.6 is taken from Schapire et al. [202].

The census dataset used in section 5.7.1 originated with the U.S. Census Bureau and was prepared by Terran Lane and Ronny Kohavi.

Research on active learning dates to the work of Cohn, Atlas, and Ladner [51], and Lewis and Catlett [151]. The use of boosting for active learning, essentially along the lines of the

method used in section 5.7.2, is due to Abe and Mamitsuka [1]. The experiments and results appearing in this section (including figure 5.11, adapted) are taken from Tur, Schapire, and Hakkani-Tür [220]; see also Tur, Hakkani-Tür, and Schapire [219]. The observation that less data can be more effective when using active learning was previously noted in another context by Schohn and Cohn [207].

Some of the exercises in this chapter are based on material from [10, 13, 26, 150, 187, 196].

Exercises

5.1 Suppose AdaBoost is run for an unterminating number of rounds. In addition to our usual notation, let

$$F_T(x) \doteq \sum_{t=1}^{T} \alpha_t h_t(x) \text{ and } s_T \doteq \sum_{t=1}^{T} \alpha_t.$$

We assume without loss of generality that each $\alpha_t \geq 0$. Let the minimum (normalized) margin on round t be denoted

$$\theta_t \doteq \min_i \frac{y_i F_t(x_i)}{s_t}.$$

Finally, we define the *smooth margin* on round t to be

$$g_t \doteq \frac{-\ln\left(\frac{1}{m}\sum_{i=1}^{m} e^{-y_i F_t(x_i)}\right)}{s_t}.$$

a. Prove that

$$\theta_t \leq g_t \leq \theta_t + \frac{\ln m}{s_t}.$$

Thus, if s_t gets large, then g_t gets very close to θ_t.

b. Prove that g_T is a weighted average of the values $\Upsilon(\gamma_t)$, specifically,

$$g_T = \frac{\sum_{t=1}^{T} \alpha_t \Upsilon(\gamma_t)}{s_T}.$$

c. Let $0 < \gamma_{\min} < \gamma_{\max} < \frac{1}{2}$. Show that if the edges γ_t eventually all lie in the narrow range $[\gamma_{\min}, \gamma_{\max}]$, then the smooth margins g_t—and therefore also the minimum margins θ_t—must similarly converge to the narrow range $[\Upsilon(\gamma_{\min}), \Upsilon(\gamma_{\max})]$. More precisely, suppose for some $t_0 > 0$ that $\gamma_{\min} \leq \gamma_t \leq \gamma_{\max}$ for all $t \geq t_0$. Prove that

$$\liminf_{t\to\infty} \theta_t = \liminf_{t\to\infty} g_t \geq \Upsilon(\gamma_{\min}),$$

and that

$$\limsup_{t\to\infty} \theta_t = \limsup_{t\to\infty} g_t \leq \Upsilon(\gamma_{\max}).$$

(See appendix A.4 for definitions.)

d. Prove that if the edges γ_t converge (as $t \to \infty$) to some value $\gamma \in (0, \frac{1}{2})$, then the minimum margins θ_t converge to $\Upsilon(\gamma)$.

5.2 Prove the following properties of binary relative entropy:

a. $\mathrm{RE}_b(p \parallel q)$ is convex in q (for any fixed p), and convex in p (for any fixed q). (Refer to appendix A.7 for definitions.)

b. $\mathrm{RE}_b(p \parallel q) \geq 2(p-q)^2$ for all $p, q \in [0, 1]$. [*Hint:* Use Taylor's theorem (theorem A.1).]

5.3 Suppose the γ^*-weak learning assumption holds for some $\gamma^* > 0$ that is *not* known ahead of time. In this case, the algorithm *AdaBoost*$^*_\nu$ can be used to efficiently find a combined classifier with minimum margin arbitrarily close to $2\gamma^*$, that is, with margin at least $\theta \doteq 2\gamma^* - \nu$ on all training examples, where $\nu > 0$ is a given accuracy parameter. This algorithm proceeds exactly like AdaBoost (algorithm 1.1 (p. 5)), except that α_t is computed on round t as follows:

- $\gamma_t = \frac{1}{2} - \epsilon_t$. (Note that $\gamma_t \geq \gamma^*$ by assumption.)
- $\hat{\gamma}_t = \min\{\gamma_1, \ldots, \gamma_t\}$.
- $\hat{\theta}_t = 2\hat{\gamma}_t - \nu$.
- $\alpha_t = \frac{1}{2}\ln\left(\frac{1+2\gamma_t}{1-2\gamma_t}\right) - \frac{1}{2}\ln\left(\frac{1+\hat{\theta}_t}{1-\hat{\theta}_t}\right)$.

a. Prove that after T rounds, the fraction of training examples with margin below θ is at most

$$\exp\left(-\sum_{t=1}^{T} \mathrm{RE}_b\left(\frac{1}{2} + \frac{\hat{\theta}_t}{2} \;\middle\|\; \frac{1}{2} + \gamma_t\right)\right).$$

b. Show that if $T > 2(\ln m)/\nu^2$, then the margin on *all* training examples is at least θ.

5.4 Let $X_1, \ldots, X_n$ be independent Bernoulli random variables with

$$X_i = \begin{cases} 1 & \text{with probability } p \\ 0 & \text{with probability } 1-p. \end{cases}$$

Let $A_n \doteq \frac{1}{n}\sum_{i=1}^{n} X_i$.

a. Generalize the technique of section 3.3 to prove that if $q \leq p$, then

$$\begin{aligned}\mathbf{Pr}[A_n \leq q] &\leq \exp\left(-n \cdot \mathrm{RE}_b\left(q \parallel p\right)\right) \\ &\leq e^{-2n(q-p)^2}.\end{aligned}$$

b. By reducing to the previous case in part (a), state and prove bounds analogous to those in (a) on $\mathbf{Pr}[A_n \geq q]$.

5.5 This exercise develops a proof of equation (5.25). As in section 5.3, let $\mathcal{F}$ be a family of real-valued functions on $\mathcal{Z}$, and let $S = \langle z_1, \ldots, z_m \rangle$ be a sequence of points in $\mathcal{Z}$.

a. Suppose $\phi(u) \doteq au + b$ for all u, where $a \geq 0$ and $b \in \mathbb{R}$. Find $R_S(\phi \circ \mathcal{F})$ exactly in terms of a, b, and $R_S(\mathcal{F})$.

b. Now let $\phi : \mathbb{R} \to \mathbb{R}$ be any *contraction*, that is, a Lipschitz function with Lipschitz constant $L_\phi = 1$. Let $U \subseteq \mathbb{R}^2$ be any set of pairs of real numbers. Prove that

$$\mathbf{E}_\sigma\left[\sup_{(u,v)\in U}(u + \sigma\phi(v))\right] \leq \mathbf{E}_\sigma\left[\sup_{(u,v)\in U}(u + \sigma v)\right]$$

where expectation is with respect to a uniformly random choice of $\sigma \in \{-1, +1\}$. [*Hint:* First show that for all $u_1, v_1, u_2, v_2 \in \mathbb{R}$, $(u_1 + \phi(v_1)) + (u_2 - \phi(v_2)) \leq \max\{(u_1 + v_1) + (u_2 - v_2), (u_1 - v_1) + (u_2 + v_2)\}$.]

c. Use part (b) to prove that if ϕ is a contraction, then $R_S(\phi \circ \mathcal{F}) \leq R_S(\mathcal{F})$.

d. Conclude that if ϕ is a Lipschitz function with Lipschitz constant $L_\phi > 0$, then $R_S(\phi \circ \mathcal{F}) \leq L_\phi \cdot R_S(\mathcal{F})$.

5.6 This exercise derives a generalization error bound for margin-based classifiers which use ℓ_2/ℓ_2-norms, such as SVMs. Let $\mathcal{X}$ be the unit ball in $\mathbb{R}^n$:

$$\mathcal{X} \doteq \left\{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\|_2 \leq 1\right\}.$$

Thus, each of the m random training examples in S is a pair $(\mathbf{x}_i, y_i)$ in $\mathcal{X} \times \{-1, +1\}$. Let $\mathcal{F}$ be the set of all possible margin functions defined by unit-length weight vectors $\mathbf{w}$:

$$\mathcal{F} \doteq \left\{(\mathbf{x}, y) \mapsto y(\mathbf{w} \cdot \mathbf{x}) \mid \mathbf{w} \in \mathbb{R}^n, \|\mathbf{w}\|_2 = 1\right\}.$$

a. Prove that $\mathcal{F}$'s Rademacher complexity is

$$R_S(\mathcal{F}) \leq \frac{1}{\sqrt{m}}.$$

[*Hint:* First show that $R_S(\mathcal{F}) = \frac{1}{m}\mathbf{E}_{\boldsymbol{\sigma}}\left[\left\|\sum_{i=1}^m \sigma_i \mathbf{x}_i\right\|_2\right]$, and then apply Jensen's inequality (equation (A.4)).]

b. For any $\theta > 0$, show that with probability at least $1 - \delta$, for all weight vectors $\mathbf{w} \in \mathbb{R}^n$ with $\|\mathbf{w}\|_2 = 1$,

$$\mathbf{Pr}_{\mathcal{D}}[y(\mathbf{w} \cdot \mathbf{x}) \leq 0] \leq \mathbf{Pr}_S[y(\mathbf{w} \cdot \mathbf{x}) \leq \theta] + O\left(\frac{1}{\theta\sqrt{m}} + \sqrt{\frac{\ln(1/\delta)}{m}}\right).$$

Give explicit constants.

5.7 Suppose, as in the example given in section 5.5, that we are using delta-functions and constant functions for base classifiers. Give an example of a random data source such that for any training set of any (finite) size $m \geq 1$, there always exists a (weighted) majority-vote classifier (defined over these base classifiers) whose training error is zero but whose generalization error is 100%.

5.8 Suppose we are using simplified decision stumps, as in exercise 2.10, as base classifiers. Assume that the same instance x can never appear in the training set with opposite labels.

a. When the number of dimensions $n = 1$, prove or disprove that for every training set, there must always exist a weighted majority-vote classifier defined over decision stumps that is consistent (that is, whose training error is zero).

b. Prove or disprove the same statement for $n \geq 2$.

5.9 Let the domain $\mathcal{X}$ be the unit sphere $\mathcal{S}$ in $\mathbb{R}^n$, that is,

$$\mathcal{S} \doteq \left\{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\|_2 = 1\right\}.$$

Given training data $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)$ in $\mathcal{S} \times \{-1, +1\}$, suppose there exists an unknown weight vector $\mathbf{w}^* \in \mathcal{S}$ such that $y_i(\mathbf{w}^* \cdot \mathbf{x}_i) \geq \gamma$ for all i, where $\gamma \geq 0$ is known. Thus, the data is linearly separable with positive margin, but using ℓ_2/ℓ_2-norms rather than ℓ_1/ℓ_∞.

Consider the following weak learning algorithm for a given distribution D over the data:

- Choose $\mathbf{w}$ uniformly at random from $\mathcal{S}$, and let $h_{\mathbf{w}}(\mathbf{x}) \doteq \text{sign}(\mathbf{w} \cdot \mathbf{x})$.
- If $\text{err}_D(h_{\mathbf{w}}) \doteq \mathbf{Pr}_{i \sim D}[h_{\mathbf{w}}(\mathbf{x}_i) \neq y_i]$ is at most $1/2 - \gamma/4$, then halt and output $h_{\mathbf{w}}$.
- Otherwise, repeat.

If this procedure halts, then clearly it has succeeded in finding a weak classifier $h_{\mathbf{w}}$ with edge $\gamma/4$. But in principle, it could take a very long time (or forever) for it to halt. We will see that this is unlikely to happen when γ is not too small.

For parts (a) and (b), fix a particular example $(\mathbf{x}_i, y_i)$.

a. Show that the angle between $\mathbf{w}^*$ and $y_i\mathbf{x}_i$ is at most $\pi/2 - \gamma$. (You can use the inequality $\sin\theta \leq \theta$ for $\theta \geq 0$.)

b. Conditional on $\mathbf{w}$ being chosen so that $\mathbf{w} \cdot \mathbf{w}^* \geq 0$, show that the probability that $h_{\mathbf{w}}(\mathbf{x}_i) \neq y_i$ is at most $1/2 - \gamma/\pi$. That is, show that

$$\mathbf{Pr}_{\mathbf{w}}\left[h_{\mathbf{w}}(\mathbf{x}_i) \neq y_i \mid \mathbf{w} \cdot \mathbf{w}^* \geq 0\right] \leq \frac{1}{2} - \frac{\gamma}{\pi}$$

where $\mathbf{Pr}_{\mathbf{w}}[\cdot]$ denotes probability with respect to the random choice of $\mathbf{w}$. [*Hint:* Consider the projection $\overline{\mathbf{w}}$ of $\mathbf{w}$ into the two-dimensional plane defined by $\mathbf{w}^*$ and $y_i\mathbf{x}_i$. Start by arguing that its direction, $\overline{\mathbf{w}}/\|\overline{\mathbf{w}}\|_2$, is uniformly distributed on the unit circle in this plane.]

c. For some absolute constant $c > 0$, show that

$$\mathbf{Pr}_{\mathbf{w}}\left[\mathrm{err}_D(h_{\mathbf{w}}) \leq \frac{1}{2} - \frac{\gamma}{4}\right] \geq c\gamma,$$

and therefore that the above procedure, in expectation, will halt in $O(1/\gamma)$ iterations for any distribution D.