4 Direct Bounds on the Generalization Error

In chapter 3, we proved a bound on the training error of AdaBoost. However, as has already been pointed out, what we really care about in learning is how well we can generalize to data not seen during training. Indeed, an algorithm that drives down the training error does not necessarily qualify as a boosting algorithm. Rather, as discussed in section 2.3, a boosting algorithm is one that can drive the *generalization error* arbitrarily close to zero; in other words, it is a learning algorithm that makes nearly perfect predictions on data *not* seen during training, provided the algorithm is supplied with a reasonable number of training examples and access to a weak learning algorithm that can consistently find weak hypotheses that are slightly better than random.

In fact, there are numerous ways of analyzing AdaBoost's generalization error, several of which will be explored in this book. In this chapter, we present the first of these methods, focusing on the direct application of the general techniques outlined in chapter 2, and basing our analyses on the structural form of the final classifier as a combination of base hypotheses. This will be enough to prove that AdaBoost is indeed a boosting algorithm. However, we will also see that the bound we derive on the generalization error predicts that AdaBoost will overfit, a prediction that often turns out to be false in actual experiments. This deficiency in the analysis will be addressed in chapter 5, where a margins-based analysis is presented.

4.1 Using VC Theory to Bound the Generalization Error

We begin with an analysis based directly on the form of the hypotheses output by AdaBoost.

4.1.1 Basic Assumptions

In proving a bound on the training error in chapter 3, we did not need to make any assumptions about the data. The (x_i, y_i) pairs were entirely arbitrary, as were the weak hypotheses h_t. Theorem 3.1 held regardless, without any assumptions. In turning now to the study of generalization error, we can no longer afford this luxury, and must accept additional assumptions. This is because, as discussed in chapter 2, if there is no relationship between the data

observed during training and the data encountered during testing, then we cannot possibly hope to do well in the test phase. Therefore, as in chapter 2, we assume that all examples, during both training and testing, are generated at random according to the same (unknown) distribution $\mathcal{D}$ over $\mathcal{X} \times \{-1, +1\}$. As before, our goal is to find a classifier h with low generalization error

$$\mathrm{err}(h) \doteq \mathbf{Pr}_{(x,y)\sim\mathcal{D}}[h(x) \neq y],$$

that is, low probability of misclassifying a new example (x, y) chosen at random from the same distribution $\mathcal{D}$ that generated each of the training examples $(x_1, y_1), \ldots, (x_m, y_m)$. We assume this probabilistic framework throughout all of our analyses of AdaBoost's generalization error.

As discussed in chapters 1 and 2, learning is all about fitting the data well, but not overfitting it. As in science, we want to "explain" our observations (the data) using the "simplest" explanation (classifier) possible. A boosting algorithm has no direct control over the base classifiers h_t that are selected on each round. If these base classifiers are already of an extremely complex form that overfits the data, then the boosting algorithm is immediately doomed to suffer overfitting as well. Therefore, in order to derive a meaningful bound on the generalization error, we also must assume something about the complexity or expressiveness of the base classifiers. Said differently, our generalization error bounds for AdaBoost will inevitably depend on some measure of the complexity of the base classifiers.

To be more precise, we assume that all base classifiers are selected from some space of classifiers $\mathcal{H}$. For instance, this might be the space of all decision stumps, or the space of all decision trees (perhaps of bounded size). As in section 2.2.2, when $\mathcal{H}$ is finite in cardinality, we can measure its complexity by $\lg |\mathcal{H}|$, which can be interpreted as the number of bits needed to specify one of its members. When $\mathcal{H}$ is infinite, we instead use the VC-dimension of $\mathcal{H}$, a combinatorial measure which, as seen in section 2.2.3, is appropriate for measuring the difficulty of learning a class of functions. Thus, we expect our bounds to depend on one of these two complexity measures.

Having proved in section 3.1 a bound on the training error, in this chapter we derive bounds on the generalization error by proving a bound on the magnitude of the difference between the generalization error and the training error. This is essentially the mode of analysis presented in chapter 2. There, we saw that for a particular classifier h, the training error $\widehat{\mathrm{err}}(h)$ can be regarded as an empirical estimate of the generalization error, an estimate that gets better and better with more data. However, in learning, we generally select the classifier h based on the training data, and usually from among those classifiers with the lowest training error. Such a process for selecting h typically leads to a significant gap between the training and generalization errors. Moreover, because we do not know which classifier h will be chosen prior to the choice of training examples, we must bound the difference $\mathrm{err}(h) - \widehat{\mathrm{err}}(h)$ for *all* h which might potentially be generated by the learning

algorithm. Here, we apply to boosting the powerful tools developed in section 2.2 for proving such general results.

4.1.2 The Form and Complexity of AdaBoost's Classifiers

As above, $\mathcal{H}$ is the base classifier space from which all of the h_t's are selected. Let $\mathcal{C}_T$ be the space of *combined* classifiers that might potentially be generated by AdaBoost if run for T rounds. Such a combined classifier H computes a weighted majority vote of T base classifiers so that

$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right) \tag{4.1}$$

for some real numbers $\alpha_1, \ldots, \alpha_T$, and some base classifiers $h_1, \ldots, h_T$ in $\mathcal{H}$. Expressed differently, we can write H in the form

$$H(x) = \sigma(h_1(x), \ldots, h_T(x))$$

where $\sigma : \mathbb{R}^T \to \{-1, 0, +1\}$ is some linear threshold function of the form

$$\sigma(\mathbf{x}) = \text{sign}(\mathbf{w} \cdot \mathbf{x}) \tag{4.2}$$

for some $\mathbf{w} \in \mathbb{R}^T$. Let Σ_T be the space of all such linear threshold functions. Then $\mathcal{C}_T$ is simply the space of linear threshold functions defined over T hypotheses from $\mathcal{H}$. That is,

$$\mathcal{C}_T = \{x \mapsto \sigma(h_1(x), \ldots, h_T(x)) : \sigma \in \Sigma_T; h_1, \ldots, h_T \in \mathcal{H}\}.$$

Our goal, then, is to show that the training error $\widehat{\text{err}}(h)$ is a good estimate of $\text{err}(h)$ for all $h \in \mathcal{C}_T$. We saw in section 2.2 that this can be proved by counting the number of functions in $\mathcal{C}_T$. Unfortunately, since Σ_T is infinite (each linear threshold function being defined by a vector $\mathbf{w} \in \mathbb{R}^T$), $\mathcal{C}_T$ is as well. However, we also saw that it suffices to count the number of possible behaviors or dichotomies that can be realized by functions in $\mathcal{C}_T$ on a finite set of points. We make this computation in the lemmas below.

Technically, to apply the formalism from section 2.2, the combined classifier must output predictions in $\{-1, +1\}$, not $\{-1, 0, +1\}$. Therefore, in this chapter only, we redefine the sign function in equations (4.1) and (4.2) to have range $\{-1, +1\}$ simply by redefining $\text{sign}(0)$ to be -1, rather than 0 as it is defined in the rest of the book. There are other ways of handling this technical annoyance which avoid this bit of inelegance, but this is perhaps the most straightforward (see exercise 4.1).

In section 2.2.3, we noted that Σ_n, the space of linear threshold functions over $\mathbb{R}^n$, has VC-dimension n, a property that we exploit below. Here we give a proof.

Lemma 4.1 The space Σ_n of linear threshold functions over $\mathbb{R}^n$ has VC-dimension n.

Proof Let $\mathbf{e}_i \in \mathbb{R}^n$ be a basis vector with a 1 in dimension i and 0 in all other dimensions. Then $\mathbf{e}_1, \ldots, \mathbf{e}_n$ is shattered by Σ_n. For if $y_1, \ldots, y_n$ is any set of labels in $\{-1, +1\}$, then $\mathbf{w} = \langle y_1, \ldots, y_n \rangle$ realizes the corresponding dichotomy since

$$\text{sign}(\mathbf{w} \cdot \mathbf{e}_i) = y_i.$$

Thus, the VC-dimension of Σ_n is at least n.

By way of reaching a contradiction, suppose now that there exists a set of $n+1$ points $\mathbf{x}_1, \ldots, \mathbf{x}_{n+1} \in \mathbb{R}^n$ which are shattered by Σ_n. Then, being $n+1$ points in n-dimensional space, there must exist real numbers $\beta_1, \ldots, \beta_{n+1}$, not all zero, such that

$$\sum_{i=1}^{n+1} \beta_i \mathbf{x}_i = \mathbf{0}.$$

Assume without loss of generality that $\beta_{n+1} > 0$. Since these points are shattered by Σ_n, there exists $\mathbf{w} \in \mathbb{R}^n$ such that

$$\text{sign}(\mathbf{w} \cdot \mathbf{x}_{n+1}) = +1, \tag{4.3}$$

and such that

$$\text{sign}(\mathbf{w} \cdot \mathbf{x}_i) = \begin{cases} +1 & \text{if } \beta_i > 0 \\ -1 & \text{if } \beta_i \leq 0 \end{cases} \tag{4.4}$$

for $i = 1, \ldots, n$. Then equation (4.3) says that $\mathbf{w} \cdot \mathbf{x}_{n+1} > 0$, while equation (4.4) implies that $\beta_i(\mathbf{w} \cdot \mathbf{x}_i) \geq 0$ for $i = 1, \ldots, n$. This gives the following contradiction:

$$\begin{aligned} 0 &= \mathbf{w} \cdot \mathbf{0} \\ &= \mathbf{w} \cdot \sum_{i=1}^{n+1} \beta_i \mathbf{x}_i \\ &= \sum_{i=1}^{n} \beta_i (\mathbf{w} \cdot \mathbf{x}_i) + \beta_{n+1}(\mathbf{w} \cdot \mathbf{x}_{n+1}) \\ &> 0. \end{aligned}$$

Thus, the VC-dimension of Σ_n is at most n. ∎

4.1.3 Finite Base Hypothesis Spaces

We are now ready to count the number of dichotomies induced by classifiers in $\mathcal{C}_T$ on any sample S. For simplicity, we focus mainly on the case where the base hypothesis space $\mathcal{H}$ is finite.

Lemma 4.2 Assume $\mathcal{H}$ is finite. Let $m \geq T \geq 1$. For any set S of m points, the number of dichotomies realizable by $\mathcal{C}_T$ is bounded as follows:

$$\left|\Pi_{\mathcal{C}_T}(S)\right| \leq \Pi_{\mathcal{C}_T}(m) \leq \left(\frac{em}{T}\right)^T |\mathcal{H}|^T.$$

Proof Let $S = \langle x_1, \ldots, x_m \rangle$. Consider a specific *fixed* sequence of base hypotheses $h_1, \ldots, h_T \in \mathcal{H}$. With respect to these, we create a modified sample $S' \doteq \langle \mathbf{x}'_1, \ldots, \mathbf{x}'_m \rangle$ where we define

$$\mathbf{x}'_i \doteq \langle h_1(x_i), \ldots, h_T(x_i) \rangle$$

to be the vector $\mathbb{R}^T$ obtained by applying $h_1, \ldots, h_T$ to x_i.

Since Σ_T has VC-dimension equal to T by lemma 4.1, we have by Sauer's lemma (lemma 2.4) and equation (2.12) applied to S' that

$$\left|\Pi_{\Sigma_T}(S')\right| \leq \left(\frac{em}{T}\right)^T. \tag{4.5}$$

That is, for *fixed* $h_1, \ldots, h_T$, the number of dichotomies defined by functions of the form

$$\sigma(h_1(x), \ldots, h_T(x))$$

for $\sigma \in \Sigma_T$ is bounded as in equation (4.5). Since the number of choices for $h_1, \ldots, h_T$ is equal to $|\mathcal{H}|^T$, and since for each one of these, the number of dichotomies is as in equation (4.5), we thus obtain the bound stated in the lemma. ∎

We can now directly apply theorems 2.3 and 2.7 to obtain the following theorem, which provides general bounds on the generalization error of AdaBoost or, for that matter, of any combined classifier H formed by taking a weighted majority vote of base classifiers. In line with the results and intuitions of chapter 2, the bound is in terms of the training error $\widehat{\text{err}}(H)$, the sample size m, and two terms which effectively stand in for the complexity of H, namely, the number of rounds T, and $\lg|\mathcal{H}|$, a measure of the complexity of the base classifiers. These are intuitive measures of H's complexity since they roughly correspond to the overall size of H, which consists of T base classifiers, each of size (in bits) $\lg|\mathcal{H}|$.

Theorem 4.3 Suppose AdaBoost is run for T rounds on $m \geq T$ random examples, using base classifiers from a finite space $\mathcal{H}$. Then, with probability at least $1 - \delta$ (over the choice of the random sample), the combined classifier H satisfies

$$\text{err}(H) \leq \widehat{\text{err}}(H) + \sqrt{\frac{32[T \ln(em|\mathcal{H}|/T) + \ln(8/\delta)]}{m}}.$$

Furthermore, with probability at least $1 - \delta$, if H is consistent with the training set (so that $\widehat{\text{err}}(H) = 0$), then

$$\text{err}(H) \leq \frac{2T \lg(2em|\mathcal{H}|/T) + 2\lg(2/\delta)}{m}.$$

Proof This is simply a matter of plugging the bound from lemma 4.2 into theorems 2.3 and 2.7. ∎

It is now possible to prove that the empirical weak learning assumption is enough to guarantee that AdaBoost will achieve arbitrarily low generalization error, given sufficient data. This is almost but not quite equivalent to saying that AdaBoost is a boosting algorithm in the technical sense given in section 2.3, an issue we discuss in section 4.3. Nevertheless, corollary 4.4 provides practical conditions under which AdaBoost is guaranteed to give nearly perfect generalization.

Corollary 4.4 Assume, in addition to the assumptions of theorem 4.3, that each base classifier has weighted error $\epsilon_t \leq \frac{1}{2} - \gamma$ for some $\gamma > 0$. Let the number of rounds T be equal to the smallest integer exceeding $(\ln m)/(2\gamma^2)$. Then, with probability at least $1 - \delta$, the generalization error of the combined classifier H will be at most

$$O\left(\frac{1}{m}\left[\frac{(\ln m)(\ln m + \ln|\mathcal{H}|)}{\gamma^2} + \ln\left(\frac{1}{\delta}\right)\right]\right).$$

Proof By theorem 3.1, the training error of the combined classifier is at most $e^{-2\gamma^2 T} < 1/m$. Since there are m examples, this means that the training error must actually be zero. Applying the second part of theorem 4.3 gives the result. ∎

Note that, in terms of the sample size m, this bound converges to zero at the rate $O((\ln m)^2/m)$, and therefore can be made smaller than ϵ, for any $\epsilon > 0$, for a setting of m that is polynomial in the relevant parameters: $1/\gamma$, $1/\epsilon$, $1/\delta$, and $\ln|\mathcal{H}|$.

Corollary 4.4 gives us a bound on the generalization error when AdaBoost is stopped just when the theoretical bound on the training error reaches zero. What happens on other rounds? Combining theorems 3.1 and 4.3, we get a bound on the generalization error of the form

$$e^{-2\gamma^2 T} + O\left(\sqrt{\frac{T\ln(m|\mathcal{H}|/T) + \ln(1/\delta)}{m}}\right). \tag{4.6}$$

This function is plotted in figure 4.1. When T is small, the first term dominates and we see an exponential drop in the bound. However, as T becomes large, the second term dominates, leading to a substantial increase in the bound. In other words, the bound predicts classic overfitting behavior. This would seem to make sense since, as T grows, the number of base classifiers comprising the combined classifier steadily increases, suggesting an increase in the size and complexity of the combined classifier. Switching to the second bound of theorem 4.3 once H is consistent does not help much since this bound also increases without limit as a function of T, suggesting that it is best to stop running AdaBoost the moment the training error reaches zero (if not sooner). Indeed, such overfitting behavior is sometimes observed with AdaBoost, as was seen in section 1.2.3. However, we also saw in section 1.3

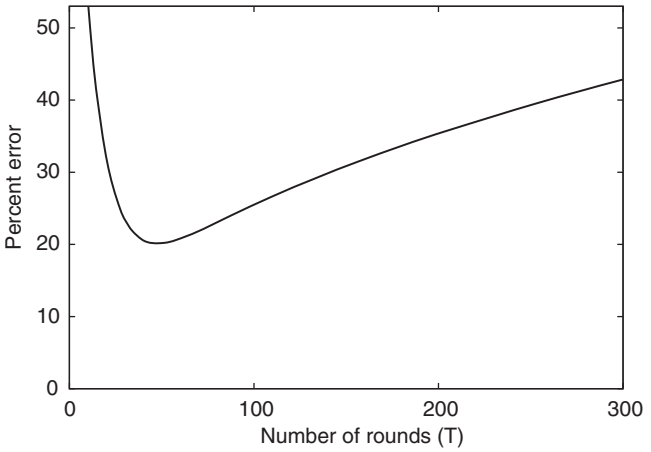

Figure 4.1
A plot of the bound on the generalization error given in equation (4.6) as a function of the number of rounds T, using the constants from theorem 4.3 with $\gamma = 0.2$, $m = 10^6$, $\ln|\mathcal{H}| = 10$, and $\delta = 0.05$.

that AdaBoost is often resistant to overfitting, and that there can be significant benefit in running AdaBoost long after consistency on the training set is reached. These phenomena cannot be explained by the analysis given above. We take up an alternative explanation of AdaBoost's behavior in chapter 5.

4.1.4 Infinite Base Classifier Spaces

Theorem 4.3 was proved for the case that the space of base classifiers $\mathcal{H}$ is finite. If $\mathcal{H}$ is infinite, a similar argument can be made using its VC-dimension d. Essentially, this is just a matter of adjusting our calculation of $\Pi_{\mathcal{C}_T}(m)$ in lemma 4.2.

Lemma 4.5 Assume $\mathcal{H}$ has finite VC-dimension $d \geq 1$. Let $m \geq \max\{d, T\}$. For any set S of m points, the number of dichotomies realizable by $\mathcal{C}_T$ is bounded as follows:

$$\left|\Pi_{\mathcal{C}_T}(S)\right| \leq \Pi_{\mathcal{C}_T}(m) \leq \left(\frac{em}{T}\right)^T \left(\frac{em}{d}\right)^{dT}.$$

Proof Let $S = \langle x_1, \ldots, x_m\rangle$. We know that $\mathcal{H}$ can realize only a finite set of dichotomies on S. Let $\mathcal{H}'$ be a subset of $\mathcal{H}$ containing exactly one "representative" for each such dichotomy. In other words, for every $h \in \mathcal{H}$ there exists exactly one $h' \in \mathcal{H}'$ such that $h(x_i) = h'(x_i)$ for every example x_i appearing in S. By definition and Sauer's lemma (lemma 2.4), together with equation (2.12),

$$|\mathcal{H}'| = |\Pi_{\mathcal{H}}(S)| \leq \left(\frac{em}{d}\right)^d.$$

Since every function in $\mathcal{H}$, with regard to its behavior on S, is represented in $\mathcal{H}'$, choosing a set of functions $h_1, \ldots, h_T \in \mathcal{H}$ as in the proof of lemma 4.2 is equivalent to choosing functions from $\mathcal{H}'$. Thus, the number of such choices is $|\mathcal{H}'|^T$. Therefore, by the argument used to prove lemma 4.2,

$$\begin{aligned}\left|\Pi_{\mathcal{C}_T}(S)\right| &\leq \left(\frac{em}{T}\right)^T |\mathcal{H}'|^T \\ &\leq \left(\frac{em}{T}\right)^T \left(\frac{em}{d}\right)^{dT}.\end{aligned}$$

■

The modification of theorem 4.3 and corollary 4.4 using lemma 4.5 is straightforward. Essentially, we end up with bounds in which $\ln|\mathcal{H}|$ is replaced by d, plus some additional log factors.

Theorem 4.6 Suppose AdaBoost is run for T rounds on m random examples, using base classifiers from a space $\mathcal{H}$ of finite VC-dimension $d \geq 1$. Assume $m \geq \max\{d, T\}$. Then with probability at least $1-\delta$ (over the choice of the random sample), the combined classifier H satisfies

$$\operatorname{err}(H) \leq \widehat{\operatorname{err}}(H) + \sqrt{\frac{32[T(\ln(em/T) + d\ln(em/d)) + \ln(8/\delta)]}{m}}.$$

Furthermore, with probability at least $1-\delta$, if H is consistent with the training set (so that $\widehat{\operatorname{err}}(H) = 0$), then

$$\operatorname{err}(H) \leq \frac{2T(\lg(2em/T) + d\lg(2em/d)) + 2\lg(2/\delta)}{m}.$$

Corollary 4.7 Assume, in addition to the assumptions of theorem 4.6, that each base classifier has weighted error $\epsilon_t \leq \frac{1}{2} - \gamma$ for some $\gamma > 0$. Let the number of rounds T be equal to the smallest integer exceeding $(\ln m)/(2\gamma^2)$. Then, with probability at least $1-\delta$, the generalization error of the combined classifier H will be at most

$$O\left(\frac{1}{m}\left[\frac{\ln m}{\gamma^2}\left(\ln m + d\ln\left(\frac{m}{d}\right)\right) + \ln\left(\frac{1}{\delta}\right)\right]\right).$$

Thus, summarizing, theorems 4.3 and 4.6 show that, ignoring log factors,

$$\operatorname{err}(H) \leq \widehat{\operatorname{err}}(H) + \tilde{O}\left(\sqrt{\frac{T \cdot C_{\mathcal{H}}}{m}}\right)$$

where $C_{\mathcal{H}}$ is some measure of the complexity of the base hypothesis space $\mathcal{H}$. Likewise, if H is consistent, the bounds state that

$$\mathrm{err}(H) \leq \tilde{O}\left(\frac{T \cdot C_{\mathcal{H}}}{m}\right).$$

As in the generalization bounds of section 2.2, these combine fit to data, complexity, and training set size, where now we measure the overall complexity of the combination of T base hypotheses by $T \cdot C_{\mathcal{H}}$.

Corollaries 4.4 and 4.7 show that when the complexity $C_{\mathcal{H}}$ is finite and fixed, the generalization error rapidly approaches zero, given the empirical weak learning assumption.

4.2 Compression-Based Bounds

So far, we have seen how AdaBoost can be analyzed in terms of the complexity of the base hypotheses. Thus, we have focused on the hypotheses *output* by the base learning algorithm. In this section, we will explore the opposing idea of instead analyzing AdaBoost based on the *input* to the base learning algorithm, in particular, the number of examples used by the base learner. This curious mode of analysis turns out to be very natural for boosting. In addition to providing generalization bounds, this approach will allow us to prove the general equivalence of strong and weak learnability by showing that AdaBoost, when appropriately configured, is a true boosting algorithm. Moreover, in studying AdaBoost from this perspective, we will highlight the remarkable property that in boosting, only a tiny fraction of the training set is ever used by the weak learning algorithm—the vast majority of the examples are never even seen by the weak learner. Indeed, it is exactly this property that forms the basis for this analysis.

4.2.1 The Idea

We assume throughout this section that boosting by resampling is employed. In other words, as described in section 3.4.1, we assume on each round t that the weak learner is trained on an *unweighted* sample selected at random by resampling with replacement from the entire training set according to the current distribution D_t. (Thus, these results are not directly applicable when boosting by reweighting is used instead.) We further assume that the unweighted sample generated on each round consists of a fixed size $m' = m_0$, not dependent (or at least not heavily dependent) on the overall sample size m. This is not unreasonable since the weak learner is aiming for a fixed accuracy $\frac{1}{2} - \gamma$, and therefore should require only a fixed sample size.

We also assume explicitly that the weak learning algorithm does not employ randomization, so that it can be regarded as a fixed, deterministic mapping from a sequence of m_0 unweighted examples to a hypothesis h. Under this assumption, any weak hypothesis produced by the weak learner can be represented rather trivially by the very sequence of m_0 examples on which it was trained.

Moreover, if we were to suppose momentarily that AdaBoost has been modified to output a combined classifier that is a simple (unweighted) majority vote, then this combined classifier can similarly be represented by the Tm_0 examples on which its T weak hypotheses were trained. In other words, under this scheme a sequence of Tm_0 examples represents the combined classifier which is a majority vote of weak hypotheses that can be computed simply by breaking the sequence into T blocks of m_0 examples, each of which is then converted into a weak hypothesis by running the weak learning algorithm on it.

Thus AdaBoost, under the assumptions above, is in fact a compression scheme of size $\kappa = Tm_0$ as described in section 2.2.6. In other words, because the combined classifier can be represented by Tm_0 of the training examples, and because m_0 is fixed, and consistency with the training set can be achieved for $T \ll m$, we can immediately apply theorem 2.8 to obtain a bound on the generalization error. Such an analysis is based solely on these properties, without any consideration of the form of the base hypotheses used.

But how can we apply this idea to AdaBoost, which in fact outputs a *weighted* majority vote? We can use the same idea above to represent the weak hypotheses by a sequence of examples, but how can we represent the real-valued weights $\alpha_1, \ldots, \alpha_T$? To answer this, we provide a general *hybrid* approach that combines the compression-based analysis of section 2.2.6 with the VC theory presented in section 2.2.3.

4.2.2 Hybrid Compression Schemes

In a standard compression scheme, as described in section 2.2.6, the learning algorithm outputs a hypothesis h that can itself be represented by a sequence of training examples. In a *hybrid compression scheme* of size κ, the hypothesis is instead selected from a *class* of hypotheses $\mathcal{F}$ where the *class* (rather than the hypothesis itself) can be represented by a sequence of κ training examples. Thus, a hybrid compression scheme is defined by a size κ and a mapping $\mathcal{K}$ from κ-tuples of labeled examples to *sets* of hypotheses. Given a training set $(x_1, y_1), \ldots, (x_m, y_m)$, the associated learning algorithm first chooses indices $i_1, \ldots, i_\kappa \in \{1, \ldots, m\}$, thus specifying a class

$$\mathcal{F} = \mathcal{K}((x_{i_1}, y_{i_1}), \ldots, (x_{i_\kappa}, y_{i_\kappa})). \tag{4.7}$$

The algorithm then chooses and outputs one hypothesis $h \in \mathcal{F}$ from this class.

Note that a standard compression scheme is simply a special case in which $\mathcal{F}$ is always a singleton.

AdaBoost is an example of a hybrid compression scheme for the setting above. We have already seen that the T weak hypotheses $h_1, \ldots, h_T$ can be represented by a sequence of $\kappa = Tm_0$ training examples. Then the resulting class $\mathcal{F}$ from which the final hypothesis H is selected consists of all linear threshold functions (that is, weighted majority vote classifiers) over the selected, fixed set of weak hypotheses $h_1, \ldots, h_T$:

$$\mathcal{F} = \left\{ H : x \mapsto \operatorname{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x) \right) \;\middle|\; \alpha_1, \ldots, \alpha_T \in \mathbb{R} \right\}. \tag{4.8}$$

By combining theorem 2.7 with theorem 2.8, we obtain a general result for hybrid compression schemes that depends both on the size κ and on the complexity of the class $\mathcal{F}$ selected by the scheme. For simplicity, we focus only on the consistent case, although the same technique can certainly be generalized.

Theorem 4.8 Suppose a learning algorithm based on a hybrid compression scheme of size κ with an associated function $\mathcal{K}$ as in equation (4.7) is provided with a random training set S of size m. Suppose further that for every κ-tuple, the resulting class $\mathcal{F}$ has VC-dimension at most $d \geq 1$. Assume $m \geq d + \kappa$. Then, with probability at least $1 - \delta$, any hypothesis h produced by this algorithm that is consistent with S must satisfy

$$\operatorname{err}(h) \leq \frac{2d \lg(2e(m-\kappa)/d) + 2\kappa \lg m + 2\lg(2/\delta)}{m - \kappa}. \tag{4.9}$$

Proof Let ε be equal to the quantity on the right-hand side of equation (4.9).

First, let us fix the indices $i_1, \ldots, i_\kappa$, and let $I = \{i_1, \ldots, i_\kappa\}$. Once the examples $(x_{i_1}, y_{i_1}), \ldots, (x_{i_\kappa}, y_{i_\kappa})$ with indices in this set have been selected, this also fixes the class $\mathcal{F}$ as in equation (4.7), Moreover, because the training examples are assumed to be independent, the training points not in I, that is, $S' = \langle (x_i, y_i) \rangle_{i \notin I}$, are also independent of the class $\mathcal{F}$. Thus, we can apply theorem 2.7, specifically equation (2.19), where we regard $\mathcal{F}$ as the hypothesis space and S' as a training set of size $m - |I| \geq m - \kappa$, and where we replace δ by δ / m^κ. Then, since $\mathcal{F}$ has VC-dimension at most d, with probability at least $1 - \delta/m^\kappa$, this result implies that $\operatorname{err}(h) \leq \varepsilon$ for every $h \in \mathcal{F}$ that is consistent with S', and therefore also for every $h \in \mathcal{F}$ that is consistent with the entire sample S. This holds true for any particular selection of examples $(x_{i_1}, y_{i_1}), \ldots, (x_{i_\kappa}, y_{i_\kappa})$, which means that it also holds true if these examples are selected at random.

Thus we have argued that for any fixed choice of indices $i_1, \ldots, i_\kappa$, with probability at least $1 - \delta/m^\kappa$, $\operatorname{err}(h) \leq \varepsilon$ for any consistent $h \in \mathcal{F}$. Therefore, by the union bound, since there are m^κ choices for these indices, this result holds for *all* sequences of indices with probability at least $1 - \delta$, implying the result. ∎

4.2.3 Application to AdaBoost

We can apply this result immediately to AdaBoost, where we already have discussed the appropriate hybrid compression scheme. Here, the class $\mathcal{F}$ consists of all linear threshold functions over a fixed set of T weak hypotheses as in equation (4.8). This class cannot have VC-dimension greater than that of linear threshold functions over points in $\mathbb{R}^T$, a class that

we showed in lemma 4.1 has VC-dimension exactly T. Thus, in constructing this scheme for AdaBoost, we have proved the following:

Theorem 4.9 Suppose AdaBoost is run for T rounds on m random examples. Assume each weak hypothesis is trained using a deterministic weak learning algorithm on m_0 unweighted examples selected using resampling, and assume $m \geq (m_0 + 1)T$. Then, with probability at least $1 - \delta$ (over the choice of the random sample), if the combined classifier H is consistent with the entire training set, then

$$\mathrm{err}(H) \leq \frac{2T \lg(2e(m - Tm_0)/T) + 2Tm_0 \lg m + 2\lg(2/\delta)}{m - Tm_0}.$$

Proof Just plug $\kappa = Tm_0$ and $d = T$ into theorem 4.8. ∎

When we add the weak learning assumption, we get the following corollary analogous to corollary 4.4.

Corollary 4.10 Assume, in addition to the assumptions of theorem 4.9, that each base classifier has weighted error $\epsilon_t \leq \frac{1}{2} - \gamma$ for some $\gamma > 0$. Let the number of rounds T be equal to the smallest integer exceeding $(\ln m)/(2\gamma^2)$. Then, with probability at least $1 - \delta$, the generalization error of the combined classifier H will be at most

$$O\left(\frac{1}{m}\left[\frac{m_0(\ln m)^2}{\gamma^2} + \ln\left(\frac{1}{\delta}\right)\right]\right) \tag{4.10}$$

(where, for purposes of $O(\cdot)$ notation, we assume $Tm_0 \leq cm$ for some constant $c < 1$).

In both these bounds m_0, the number of examples used to train the weak learner, is acting as a complexity measure rather than some measure based on the weak hypotheses that it outputs. Otherwise, the bounds have essentially the same form as in section 4.1.

4.3 The Equivalence of Strong and Weak Learnability

Finally, we are ready to prove that AdaBoost is a boosting algorithm in the technical sense, and that strong and weak learnability are equivalent in the PAC model described in section 2.3. Note that corollaries 4.4 and 4.7 do not quite prove this since their bounds depend on a measure of the complexity of the weak hypotheses. Thus, they implicitly require that the sample size m be sufficiently large relative to this complexity measure. This goes beyond a bare assumption of weak learnability. A compression-based analysis, however, allows us to sidestep this difficulty.

Theorem 4.11 A target class $\mathcal{C}$ is (efficiently) weakly PAC learnable if and only if it is (efficiently) strongly PAC learnable.

Proof That strong learning implies weak learning is trivial. To prove the converse, we apply AdaBoost to a given weak learning algorithm. Here are the details.

Suppose $\mathcal{C}$ is weakly PAC learnable. Then there exists a constant $\gamma > 0$, and an algorithm A such that for any distribution $\mathcal{D}$ over the instance space $\mathcal{X}$, and for any $c \in \mathcal{C}$, A takes as input m_0 random examples $(x_1, c(x_1)), \ldots, (x_{m_0}, c(x_{m_0}))$ and, with probability at least $\frac{1}{2}$, outputs a hypothesis with $\mathrm{err}(h) \leq \frac{1}{2} - \gamma$. Note that here we have weakened the requirement for A even further than the definition given in section 2.3 by requiring only that A succeed with probability at least $\frac{1}{2}$, effectively fixing δ in the earlier definition to this constant.

We assume A is deterministic. If it is not, there are general constructions that can be used here for converting a randomized PAC algorithm into a deterministic one; however, these go beyond the scope of this book.

To construct a strong PAC learning algorithm, we apply AdaBoost with A as the weak learning algorithm. Given m examples from some unknown target $c \in \mathcal{C}$, and given $\delta > 0$, we run AdaBoost for T rounds where T is the smallest integer exceeding $(\ln m)/(2\gamma^2)$. On each round t, we use boosting by resampling to select a sample of size m_0 according to distribution D_t. This sample is fed to the weak learning algorithm A, which produces a weak hypothesis h_t. If the error of h_t on D_t is bigger than $\frac{1}{2} - \gamma$, that is, if it is *not* the case that

$$\mathbf{Pr}_{i \sim D_t}[h_t(x_i) \neq c(x_i)] \leq \tfrac{1}{2} - \gamma, \tag{4.11}$$

then h_t is discarded and the process is repeated until h_t satisfying equation (4.11) is found. If no such h_t is found after $L \doteq \lceil \lg(2T/\delta) \rceil$ attempts, then boosting fails.

What is the probability of such failure? The weak learning algorithm's training set on round t consists of m_0 examples selected from distribution D_t, so from A's perspective, D_t is the "true" distribution. Therefore, according to our assumptions regarding this algorithm, the probability that its hypothesis h_t will have weighted error greater than $\frac{1}{2} - \gamma$ on this "true" distribution is at most $\frac{1}{2}$. Thus, the chance of failure on all L (independent) attempts is at most

$$2^{-L} \leq \frac{\delta}{2T}.$$

Therefore, the chance of failure on any of the T rounds is at most $\delta/2$ by the union bound.

When no such failures occur, each hypothesis h_t will have weighted error $\epsilon_t \leq \frac{1}{2} - \gamma$, so that corollary 4.10 can be applied where we replace δ with $\delta/2$ so that the overall probability of failure either in the search for weak hypotheses or in the choice of the training set is at most δ (again, by the union bound).

Thus, with probability at least $1 - \delta$, AdaBoost produces a combined classifier H with error at most as given in equation (4.10). This bound can be made smaller than any $\epsilon > 0$ by choosing m to be a suitable polynomial in m_0, $1/\gamma$, $1/\epsilon$, and $1/\delta$. Thus, the class $\mathcal{C}$ is

strongly learnable in the PAC model. Furthermore, if A is efficient (that is, polynomial-time), then AdaBoost, as we have described it, will be as well since the overall running time is polynomial in m, $1/\delta$, $1/\gamma$, and the running time of the weak learner A itself. ■

Note that in the construction used in this proof, only a total of

$$Tm_0 = O\left(\frac{m_0 \ln m}{\gamma^2}\right)$$

examples are used by the weak learning algorithm in the computation of the weak hypotheses comprising the combined classifier. (Even if we count runs in which the weak learner fails to provide an adequate weak hypothesis, this number goes up by only a small factor.) Since we regard m_0 and γ as fixed, this means that only a vanishingly small fraction of the training set—just $O(\ln m)$ of the m examples—are ever even presented as input to the weak learner. All the work of boosting apparently goes into the careful selection of this tiny sliver of the dataset.

Furthermore, our analysis provides bounds not only for boosting but also for learning in a much more general sense. For instance, the proof of theorem 4.11 gave a construction in which the generalization error of AdaBoost was shown to drop at the rate

$$O\left(\frac{(\ln m)^2}{m}\right) \tag{4.12}$$

as a function of m (for T chosen as above). But this same construction also shows that *any* PAC learning algorithm A can be converted into one with such behavior. To make such a conversion, we simply hardwire A's parameters, say, to $\epsilon = \frac{1}{4}$ and $\delta = \frac{1}{2}$. Then the resulting algorithm will be a weak learning algorithm which, when combined with AdaBoost, will have the same rate of generalization as in equation (4.12). Thus, if a class is (efficiently) learnable at all, then it is (efficiently) learnable at the learning rate given in equation (4.12). This kind of argument is applicable to other measures of performance as well.

Summary

In summary, we have described several modes of analysis applicable to AdaBoost. Each of these has measured the complexity of the combined classifier in terms of its gross size, that is, the number of base hypotheses being combined, and some varying measure of the complexity of the base hypotheses themselves. We have seen that AdaBoost's generalization error can be made very small if the weak hypotheses are a bit better than random, and thus, that strong and weak PAC learnability, which seem superficially to be so different, actually turn out to be equivalent. However, all of our analyses have predicted overfitting, which is only sometimes a problem for AdaBoost. In chapter 5, we present a rather different analysis that appears to better match AdaBoost's behavior in many practical cases.

Bibliographic Notes

The style of analysis presented in section 4.1 was applied to AdaBoost by Freund and Schapire [95], and is based directly on the work of Baum and Haussler [16]. Lemma 4.1 was proved (in a more general setting) by Dudley [77]. See Anthony and Bartlett's book [8] for further background.

The hybrid compression schemes of section 4.2 are based on the standard compression schemes of Littlestone and Warmuth [154] and Floyd and Warmuth [85]. The propensity of boosting algorithms to compress a dataset was noted by Schapire [199], and was first used as a basis for analyzing their generalization error by Freund [88].

The equivalence of strong and weak learnability shown in section 4.3 was first proved by Schapire [199], and later by Freund [88], though using boosting algorithms which preceded AdaBoost. Both of these works also proved general resource requirements for PAC learning.

The fact noted in the proof of theorem 4.11 that a randomized PAC learning algorithm can be converted into one that is deterministic was proved by Haussler et al. [121].

Some of the exercises in this chapter are based on material from [16, 77, 85, 88, 199].

Exercises

4.1 In the development given in section 4.1, we found it necessary to redefine sign(0) to be -1, rather than 0, so that the combined classifier H would have range $\{-1, +1\}$ rather than $\{-1, 0, +1\}$ (with predictions of 0 always counting as a mistake). Show how to modify the proofs leading to theorems 4.3 and 4.6 when sign(0) is instead defined to be 0. [*Hint:* Apply the results of section 2.2.4 to an appropriate family of subsets of $\mathcal{X} \times \{-1, +1\}$.]

4.2 Let Σ_n' be the space of all functions mapping $\mathbb{R}^n$ to $\{-1, +1\}$ of the form

$$\mathbf{x} \mapsto \text{sign}(\mathbf{w} \cdot \mathbf{x} + b)$$

for some $\mathbf{w} \in \mathbb{R}^n$ and $b \in \mathbb{R}$ (where we continue to define sign(0) to be -1). These are sometimes called *affine threshold functions*, and differ from linear threshold functions only in the "bias term" b. Find the VC-dimension of Σ_n' exactly.

4.3 A *feedforward network*, as in the example in figure 4.2, is defined by a directed acyclic graph on a set of *input nodes* $x_1, \ldots, x_n$, and *computation nodes* $u_1, \ldots, u_N$. The input nodes have no incoming edges. One of the computation nodes is called the *output node*, and has no outgoing edges. Each computation node u_k is associated with a function $f_k : \mathbb{R}^{n_k} \to \{-1, +1\}$, where n_k is u_k's indegree (number of ingoing edges). On input $\mathbf{x} \in \mathbb{R}^n$, the network computes its output $g(\mathbf{x})$ in a natural, feedforward fashion. For instance, given input $\mathbf{x} = \langle x_1, x_2, x_3 \rangle$, the network in figure 4.2 computes $g(\mathbf{x})$ as follows:

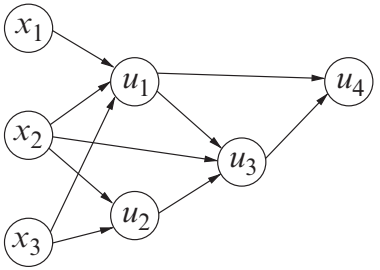

Figure 4.2
An example feedforward network with $n = 3$ input nodes, x_1, x_2, x_3; $N = 4$ computation nodes, u_1, u_2, u_3, u_4; and $W = 10$ edges. The output node is u_4.

$$u_1 = f_1(x_1, x_2, x_3)$$

$$u_2 = f_2(x_2, x_3)$$

$$u_3 = f_3(u_1, x_2, u_2)$$

$$u_4 = f_4(u_1, u_3)$$

$$g(\mathbf{x}) = u_4.$$

(Here, we slightly abuse notation, writing x_j and u_k both for nodes of the network and for the input/computed values associated with these nodes.) The number of edges in the graph is denoted W.

In what follows, we regard the underlying graph as fixed, but allow the functions f_k to vary, or to be learned from data. In particular, let $\mathcal{F}_1, \ldots, \mathcal{F}_N$ be spaces of functions. As explained above, every choice of functions $f_1, \ldots, f_N$ induces an overall function $g : \mathbb{R}^n \to \{-1, +1\}$ for the network. We let $\mathcal{G}$ denote the space of all such functions when f_k is chosen from $\mathcal{F}_k$ for $k = 1, \ldots, N$.

a. Prove that

$$\Pi_{\mathcal{G}}(m) \leq \prod_{k=1}^{N} \Pi_{\mathcal{F}_k}(m).$$

b. Let d_k be the VC-dimension of $\mathcal{F}_k$, and let $d \doteq \sum_{k=1}^{N} d_k$. Assume $m \geq d_k \geq 1$ for all k. Prove that

$$\Pi_{\mathcal{G}}(m) \leq \left(\frac{emN}{d}\right)^d.$$

[*Hint:* Use Jensen's inequality (equation (A.4)).]

c. In a typical *neural network* or *multilayer perceptron*, the functions f_k are affine threshold functions as in exercise 4.2, so that $\mathcal{F}_k = \Sigma'_{n_k}$. For this case, give an exact expression for d in terms of N, n, and W. Conclude by deriving a bound, analogous to theorems 4.3 and 4.6, on the generalization error of every $g \in \mathcal{G}$ which holds with probability at least $1 - \delta$ for $m \geq d$, and which is expressed in terms of $\widehat{\text{err}}(g)$, N, n, W, m, and δ. Also give a bound for when g is consistent.

4.4 This exercise relates the size of compression schemes to VC dimension. Assume throughout that all training examples are labeled according to some unknown target c in a known class $\mathcal{C}$ of VC-dimension $d \geq 1$. Assume also the existence of a (deterministic) algorithm A which, given any dataset S, outputs some $h \in \mathcal{C}$ consistent with S. You need not consider issues of efficiency.

a. Suppose B is a (standard) compression scheme of size κ which, when given $m \geq d$ training examples labeled as above, always produces a hypothesis h that is consistent with the given data. Being a compression scheme, each such hypothesis h can be represented by κ of the training examples. Prove that $\kappa \geq d/(1 + \lg d)$. [*Hint:* By counting the number of possible inputs and outputs for B on a shattered training set, show that, when κ is too small, there must exist two distinct labelings of the shattered set that are mapped to the same hypothesis, leading to a contradiction.]

b. Show that there exists a compression scheme B with the same properties as in part (a) whose size κ (when $m \geq d$) is at most $Cd \ln m$, for some absolute constant C. (We here allow the size κ to depend moderately on the sample size m.) [*Hint:* First show how A can be used as a weak learning algorithm requiring a sample of size $O(d)$. Then apply boosting.]

4.5 Support-vector machines, which will be discussed in section 5.6, produce classifiers of the form

$$h(x) = \text{sign}\left(\sum_{i=1}^{m} b_i\, g(x_i, x)\right)$$

for some $b_1, \ldots, b_m \in \mathbb{R}$, where $x_1, \ldots, x_m$ are the training examples, and where $g : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a *fixed* function. We say such a classifier is κ*-sparse* if at most κ of the b_i's are nonzero. Show that, with probability at least $1 - \delta$, every classifier of this form which is κ-sparse and which is consistent with a random training set of size m has generalization error at most

$$O\left(\frac{\kappa \ln m + \ln(1/\delta)}{m - \kappa}\right).$$

Give explicit constants.