13 Optimally Efficient Boosting

Much of this book has been concerned with the efficiency of boosting, especially of AdaBoost. In section 3.1, we proved a bound on how quickly AdaBoost drives down the training error in terms of the edges of the weak classifiers. In chapters 4 and 5, we proved an assortment of bounds on the generalization error that in one way or another made use of the training-error analysis. These bounds on AdaBoost's performance are quite good in many respects, indicating, for instance, that AdaBoost's training error drops exponentially fast when the weak learning assumption holds. However, they also leave us wondering if it might be possible to do even better, perhaps with a different algorithm. In other words, these results raise basic questions about the nature of "optimal" boosting: Is AdaBoost the best possible algorithm? If not, what algorithm is, and how close does AdaBoost come? These questions concern the fundamental resource requirements that are necessary for boosting to be possible.

To find answers, we begin by studying how to optimally minimize the training error when allowed up to T calls to a weak learning algorithm for which the empirical γ-weak learning assumption is guaranteed to hold. Here, as in chapter 6, the interaction between the booster and the weak learner is viewed as a game. However, whereas in chapter 6 we regarded each *round* of boosting as a complete game that is repeated T times, now we regard the *entire sequence* of T rounds of boosting as a single game that is played just once.

Using this formulation, we derive an algorithm called *boost-by-majority* (BBM) which is very nearly optimal for this game. In terms of γ and T, its training error turns out to be exactly the tail of a certain binomial distribution, whereas the bound for AdaBoost given in theorem 3.1 is precisely the upper bound on this same tail that would be obtained by applying Hoeffding's inequality (theorem 2.1). Thus, in terms of the training error, the gap between AdaBoost and optimality is the same as the difference between Hoeffding's inequality and the true probability that it is used to approximate—a gap that, in a certain sense, vanishes asymptotically.

We next consider the generalization error, whose minimization is of course the true purpose of boosting. Not surprisingly, the results of chapter 4 can be immediately applied to derive an upper bound on the generalization error of BBM. More interestingly, the bounds

so obtained turn out to be exactly the best possible for *any* boosting algorithm. In other words, in terms of T and γ, there exist learning problems for which any boosting algorithm will have generalization error at least as large as that given by the upper bound for BBM (up to an additive difference which vanishes as the number of training examples gets large). Equivalently, this lower bound provides a floor on the minimum number of rounds of boosting needed to achieve a desired accuracy. Thus, in these terms, BBM is essentially optimal, and AdaBoost is close behind.

Besides being optimal in the senses discussed above, BBM may have another potential advantage over AdaBoost, namely, in its handling of outliers. As seen in sections 10.3 and 12.3.3, when some of the data are mislabeled or ambiguous, AdaBoost piles more and more weight on such difficult examples, sometimes substantially degrading performance. BBM also concentrates on harder examples but, in contrast to AdaBoost, actually puts *less* weight on the *very* hardest examples, effectively "giving up" on the outliers. This may be an important benefit on noisy datasets.

Unfortunately, BBM also has an important disadvantage. Unlike AdaBoost, it is non-adaptive, meaning that the minimum edge γ must be provided before boosting begins. This property seriously hinders its use in practical applications. In chapter 14, we describe a technique for making BBM adaptive.

13.1 The Boost-by-Majority Algorithm

We begin by considering how to optimally minimize the training error. This will lead to a derivation of the boost-by-majority algorithm.

13.1.1 The Voting Game

As usual, we assume we have been given m training examples $(x_1, y_1), \ldots, (x_m, y_m)$. We also assume access to a weak learning algorithm satisfying the empirical γ-weak learning assumption, meaning that, for any distribution D over the sample, the weak learner is guaranteed to return a weak hypothesis h whose weighted error with respect to D is at most $\frac{1}{2} - \gamma$. Finally, we assume that the booster is allowed to access the weak learner T times. Under these conditions, our goal is to determine the minimum training error that can be guaranteed by any boosting algorithm. Note that this is essentially equivalent to asking for the minimum number of rounds of boosting necessary to achieve some desired accuracy.

For now, we further restrict our attention to boosting algorithms whose combined classifier takes the form of a simple (unweighted) majority vote over the weak hypotheses. This restriction may slightly limit what is possible for the booster. Even so, the results of section 13.2 will show that no boosting algorithm can do significantly better, even without this limitation.

We can regard the boosting process as a game between the two interacting players, the booster and the weak learner. The game is played as follows: On each of a sequence of rounds $t = 1, \ldots, T$:

1. the booster chooses a distribution D_t over the training set;
2. the weak learner chooses a hypothesis h_t such that

$$\mathbf{Pr}_{i \sim D_t}[h_t(x_i) \neq y_i] \leq \tfrac{1}{2} - \gamma. \tag{13.1}$$

At the end of T rounds, the final hypothesis is formed as a simple majority vote of the weak hypotheses:

$$H(x) = \operatorname{sign}\left(\sum_{t=1}^{T} h_t(x)\right). \tag{13.2}$$

The loss of the booster in this game is the training error

$$\frac{1}{m}\sum_{i=1}^{m} \mathbf{1}\{H(x_i) \neq y_i\} = \frac{1}{m}\sum_{i=1}^{m} \mathbf{1}\left\{y_i \sum_{t=1}^{T} h_t(x_i) \leq 0\right\}. \tag{13.3}$$

The booster's goal is to minimize this loss, while the weak learner's goal is to maximize it.

When compared with our earlier game-theoretic formulation of boosting given in section 6.4, elements of the game described above may seem odd, or even incorrect. Indeed, there do exist very significant differences between the two formulations, each capturing different aspects of the boosting problem and thereby yielding different insights.

To be specific, in the formulation of section 6.4, the game of interest is played *repeatedly*, once on every round, with loss suffered at the end of *each* round. Also, this game was defined by a matrix over training examples and a *fixed space* of weak hypotheses. In the current setup, we instead regard the *entire* sequence of T rounds as a *single* game. Although in principle it might be possible to describe this game by a matrix, it is more natural to define it in terms of sequential play that alternates between the two players with loss incurred only at the end of the sequence. Further, we make no restrictions on the weak hypotheses other than that they satisfy the γ-weak learning assumption.

A final, more subtle difference between the two games is in the apparent goals of the two players, especially the weak learner: In the setup of section 6.4, the weak learner wishes to *minimize* the weighted error of its weak hypotheses, while the booster tries to make this difficult by choosing a hard distribution. Now, instead, the weak learner has a diametrically opposite interest in choosing weak hypotheses with the *largest* weighted error possible (though not exceeding $\frac{1}{2} - \gamma$), since these are intuitively more likely to make it difficult for the booster to achieve its goal of producing a combined hypothesis with low error. Thus, the weak learner's goal is exactly reversed in the two game-theoretic models of boosting.

Nevertheless, despite this seeming contradiction, both formulations lead to sensible insights and algorithms.

To get some intuition for the game, let us consider some simple cases. First, consider a "lazy booster" that chooses the uniform distribution over the training set on every round. The response of the weak learner to this strategy is likely to be simple: Choose some weak hypothesis h that is correct on $\frac{1}{2}+\gamma$ of the training examples, and output this same weak hypothesis on every round. The final majority-vote classifier will then be equivalent to h, so that its training error is exactly as large as h's. Clearly, and unsurprisingly, the booster has to change the distribution in order to prevent the weak learner from always outputting the same weak hypothesis.

As a second example, and one which will play a key role in the following, consider an *oblivious weak learner* that entirely ignores the distributions D_t. Instead, on every round, a random weak hypothesis is chosen by this weak learner whose prediction on every training example x_i is selected independently to match the correct label y_i with probability $\frac{1}{2}+\gamma$, and otherwise is equal to its opposite $-y_i$ (with probability $\frac{1}{2}-\gamma$). In other words, conditional on the correct label, the predictions of the weak hypotheses on the examples are independent of one another, and each is correct with probability $\frac{1}{2}+\gamma$. Regardless of the distributions provided by the booster, the expected weighted error of such a weak hypothesis will be exactly $\frac{1}{2}-\gamma$. Strictly speaking, this leaves open the possibility of the randomly chosen weak hypothesis having an *actual* weighted error that exceeds $\frac{1}{2}-\gamma$. Nevertheless, for the moment, for the purposes of this informal discussion, we ignore this complication and allow such a weak learner, even though it does not technically satisfy the requirements of the game.

If the weak learner uses this oblivious strategy, then the booster's final training error is likely to be very small. This is because each example is correctly classified independently by each weak hypothesis with probability $\frac{1}{2}+\gamma$, and it is correctly classified by the final hypothesis if and only if more than half of the weak hypotheses are correct. Thus, the chance that it is misclassified is the same as the probability of at most $T/2$ heads in a sequence of T coin flips when the probability of heads on each flip is $\frac{1}{2}+\gamma$. This probability is exactly

$$\sum_{j=0}^{\lfloor T/2 \rfloor} \binom{T}{j} \left(\tfrac{1}{2}+\gamma\right)^{j} \left(\tfrac{1}{2}-\gamma\right)^{T-j}. \tag{13.4}$$

Since this holds for each training example, the expected training error will also be exactly equal to this quantity, which, by Hoeffding's inequality (theorem 2.1), is at most $e^{-2\gamma^2 T}$. As informally suggested here, when made rigorous, this argument shows that for any booster, the oblivious weak learner (with some technical modifications) can force the training error to be very close to the quantity in equation (13.4) when the number of training examples is large.

One option for playing this game, of course, is to apply AdaBoost or, rather, the α-Boost version given in section 6.4.3, in which, on every round, the hypothesis weight α_t in algorithm 1.1 is fixed to the constant

$$\alpha = \frac{1}{2} \ln \left(\frac{1+2\gamma}{1-2\gamma} \right) \tag{13.5}$$

so that the final hypothesis will be a simple majority vote as required in this game. We refer to this algorithm, with this setting of α, as *NonAdaBoost* since it is a nonadaptive boosting algorithm. A straightforward modification of theorem 3.1 then shows that the loss of this boosting algorithm will be at most

$$(1-4\gamma^2)^{T/2} \leq e^{-2\gamma^2 T},$$

which exactly matches the upper bound on equation (13.4) provided by Hoeffding's inequality. Thus, already we can see that the gap between AdaBoost and optimality for this game is not large, though perhaps not the best possible.

In fact, as we will see, the boost-by-majority algorithm achieves an upper bound on the training error that is *exactly* equal to equation (13.4) for any weak learner. This will show that both BBM and the oblivious weak learner are essentially optimal for their respective roles in the game.

13.1.2 A Chip Game

To simplify the presentation, let us define, for example i and round t, the variable

$$z_{t,i} \doteq y_i h_t(x_i),$$

which is $+1$ if h_t correctly classifies (x_i, y_i), and -1 otherwise. We also define the variable

$$s_{t,i} \doteq y_i \sum_{t'=1}^{t} h_{t'}(x_i) = \sum_{t'=1}^{t} z_{t',i},$$

which is the unnormalized margin of the classifier constructed through round t. We write $\mathbf{s}_t$ and $\mathbf{z}_t$ for the corresponding vectors with components as above.

In terms of these variables, the voting game we are studying can be redescribed more visually as a "chip game." Here, each training example is identified with a *chip*, and each of the m chips has an integer *position*; specifically, the position of chip i at the end of round t is $s_{t,i}$. Initially, all chips are at position 0 so that $\mathbf{s}_0 = \mathbf{0}$. On every round t, the booster chooses a distribution D_t over the chips. In general, chips at the same position need not be assigned the same weight under D_t, although this will usually happen naturally. Given D_t, the weak learner next chooses to increment (move up by 1) the positions of some of the

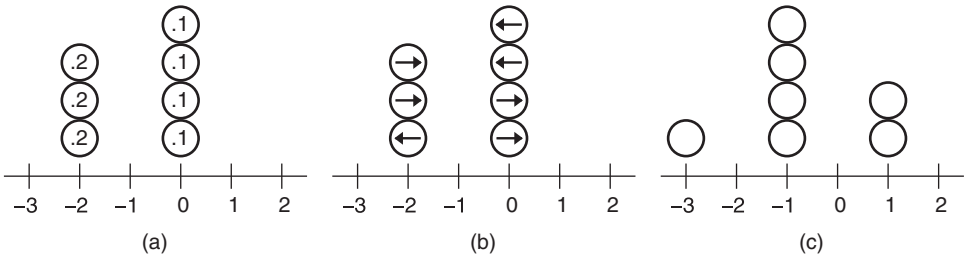

Figure 13.1
One round of the chip game: (a) The booster selects a distribution over the chips, indicated by the numbers appearing in each chip; (b) the weak learner chooses some of the chips to be incremented (moved right one position), and the rest to be decremented (moved left one position), as indicated by the arrows; (c) the chips are moved to their new positions, as specified by the weak learner.

chips, and to decrement (move down by 1) the positions of all the rest. In other words, the weak learner chooses a vector $\mathbf{z}_t \in \{-1, +1\}^m$ and updates the chip positions:

$$\mathbf{s}_t = \mathbf{s}_{t-1} + \mathbf{z}_t. \tag{13.6}$$

See figure 13.1 for an example of a single round of the game.

Importantly, the weak learner is required to increment the positions of at least $\frac{1}{2} + \gamma$ of the chips, as weighted by D_t; that is, the weak learner must choose $\mathbf{z}_t$ so that

$$\mathbf{Pr}_{i \sim D_t}\big[z_{t,i} = +1\big] \geq \tfrac{1}{2} + \gamma \tag{13.7}$$

or, equivalently,

$$\mathbf{E}_{i \sim D_t}\big[z_{t,i}\big] \geq 2\gamma. \tag{13.8}$$

The choice of $\mathbf{z}_t$ of course corresponds to the choice of h_t, and the condition in equation (13.7) is then simply the γ-weak learning assumption.

After T rounds, the loss suffered by the booster is the fraction of chips at nonpositive positions:

$$L(\mathbf{s}_T) \doteq \frac{1}{m}\sum_{i=1}^{m} \mathbf{1}\big\{s_{T,i} \leq 0\big\}, \tag{13.9}$$

a direct translation of equation (13.3).

13.1.3 Deriving Optimal Play

How, then, should this game be played optimally? In section 6.1.2, we saw the benefit of analyzing a sequentially played game from the end of the game backward to its beginning. We can apply the same idea here.

Suppose at the beginning of the final round T that the chip positions are given by the vector $\mathbf{s}_{T-1}$, and that the booster chooses distribution D_T over the chips. How would an optimal weak learner respond? The weak learner's goal is to choose $\mathbf{z}_T \in \{-1, +1\}^m$ so as to maximize the resulting loss for the final chip positions, namely,

$$L(\mathbf{s}_T) = L(\mathbf{s}_{T-1} + \mathbf{z}_T).$$

However, $\mathbf{z}_T$ must satisfy the constraint in equation (13.8), that is, it must belong to the set $\mathcal{Z}(D_T)$ where

$$\mathcal{Z}(D) \doteq \left\{\mathbf{z} \in \{-1, +1\}^m : \mathbf{E}_{i\sim D}[z_i] \geq 2\gamma\right\}.$$

So if the booster chooses D_T, then its final loss will be

$$\max_{\mathbf{z}_T \in \mathcal{Z}(D_T)} L(\mathbf{s}_{T-1} + \mathbf{z}_T). \tag{13.10}$$

Therefore, when the chips are in positions $\mathbf{s}_{T-1}$ on round T, an optimal booster will select D_T which minimizes equation (13.10), giving a loss of

$$\min_{D_T} \max_{\mathbf{z}_T \in \mathcal{Z}(D_T)} L(\mathbf{s}_{T-1} + \mathbf{z}_T)$$

(where such a minimum will always be understood to be taken over all distributions on $\{1, \ldots, m\}$). This expression is a function of the chip positions $\mathbf{s}_{T-1}$. Given these positions, it computes the loss that will result if both players play optimally for the rest of the game. It also specifically prescribes that the optimal booster use the distribution D_T that realizes the minimum.

To continue this argument, let us define for each round t a function $\Lambda_t(\mathbf{s}_t)$ that is equal to the loss that would result if the chips are in positions given by vector $\mathbf{s}_t$ at the end of round t, and given that both players play optimally for the part of the game that remains after this round. Note that after round T, the game is over and the loss suffered is already determined. Thus,

$$\Lambda_T(\mathbf{s}_T) = L(\mathbf{s}_T). \tag{13.11}$$

For earlier rounds $t \leq T$, we can use the same reasoning as above. The chips begin the round in positions $\mathbf{s}_{t-1}$. If the booster chooses D_t, an optimal weak learner will respond with $\mathbf{z}_t \in \mathcal{Z}(D_t)$ to maximize the loss for the remainder of the game $\Lambda_t(\mathbf{s}_{t-1} + \mathbf{z}_t)$. Thus, the booster should select D_t to minimize

$$\max_{\mathbf{z}_t \in \mathcal{Z}(D_t)} \Lambda_t(\mathbf{s}_{t-1} + \mathbf{z}_t),$$

so that the loss suffered under optimal play beginning after round $t-1$ is

$$\Lambda_{t-1}(\mathbf{s}_{t-1}) = \min_{D_t} \max_{\mathbf{z}_t \in \mathcal{Z}(D_t)} \Lambda_t(\mathbf{s}_{t-1} + \mathbf{z}_t). \tag{13.12}$$

This recurrence, in principle, allows us to compute the optimal loss under optimal play and, furthermore, provides the optimal strategy for both players—the booster should play the distribution D_t that realizes the minimum and, given D_t, the weak learner should play the vector $\mathbf{z}_t$ that realizes the maximum. On the other hand, these strategies do not lend themselves easily to analysis or implementation.

At the beginning of the game, under optimal play, the loss suffered by the booster with all chips starting at position 0 at time 0 is $\Lambda_0(\mathbf{0})$. Thus, unraveling the recurrence in equation (13.12) gives an explicit expression for the value of the game, that is, the loss for the entire game under optimal play:

$$\Lambda_0(\mathbf{0}) = \min_{D_1} \max_{\mathbf{z}_1 \in \mathcal{Z}(D_1)} \cdots \min_{D_T} \max_{\mathbf{z}_T \in \mathcal{Z}(D_T)} L\left(\sum_{t=1}^{T} \mathbf{z}_t\right).$$

Needless to say, this is a rather unwieldy formula.

13.1.4 A Tractable Approximation

The function Λ_t characterizes optimality exactly, but is difficult to compute and work with mathematically, as are the optimal strategies that it implicitly defines. Fortunately, as we show next, Λ_t can be usefully approximated in a way that admits both a closed-form analysis of the game and the derivation of the BBM algorithm, a strategy for the booster that is close to optimal and straightforward to implement. This approximation will eventually be stated in terms of a "potential function," a concept at the heart of BBM and its analysis. Although this algorithm can perhaps be stated and analyzed without giving a full derivation of Λ_t's approximation, we provide one anyway with the purpose of revealing where the potential function and the algorithm itself are coming from, while also illustrating a more general approach.

The basic recurrence in equation (13.12) is especially unpleasant for two reasons: first, because the maximum is constrained to the set $\mathcal{Z}(D_t)$, making it more complicated to handle than if there were no constraint on $\mathbf{z}_t$; and second, because the optimization requires consideration of all m chips at once. The next important lemma eliminates both of these difficulties. By making a slight approximation, the lemma will allow us to rewrite equation (13.12) in such a way that the maximum is unconstrained and, moreover, the optimization will decompose so that each chip can be considered separately and independently from all the rest of the chips. Indeed, these simplifications will make it possible to solve the (approximated) recurrence exactly.

Lemma 13.1 Let $G : \{-1, +1\}^m \to \mathbb{R}$, and assume

$$G(\mathbf{z}) \leq \sum_{i=1}^{m} g_i(z_i) \tag{13.13}$$

for all $\mathbf{z} \in \{-1, +1\}^m$, and some sequence of functions $g_i : \{-1, +1\} \to \mathbb{R}$. Then

$$\min_D \max_{\mathbf{z}\in\mathcal{Z}(D)} G(\mathbf{z}) \le \sum_{i=1}^{m} \inf_{w_i\ge 0} \max_{z_i\in\{-1,+1\}} [g_i(z_i) + w_i \cdot (z_i - 2\gamma)]. \tag{13.14}$$

Note that the right-hand side of equation (13.12) has exactly the form given on the left of equation (13.14) for any fixed $\mathbf{s}_{t-1}$ since we can take

$$G(\mathbf{z}) = \Lambda_t(\mathbf{s}_{t-1} + \mathbf{z}).$$

The lemma says that if such a function G can be (approximately) decomposed chip by chip, then the entire min-max expression can be as well. And, moreover, the resulting optimization problems involve only individual chips.

At the heart of the proof is a reversal in the order of taking a minimum or a maximum as seen in section 6.1.3.

Proof We first eliminate the restriction on the choice of $\mathbf{z}$ by introducing a new variable λ and modifying the quantity being maximized. Specifically, for any D,

$$\max_{\mathbf{z}\in\mathcal{Z}(D)} G(\mathbf{z}) = \max_{\mathbf{z}\in\{-1,+1\}^m} \inf_{\lambda\ge 0} \left[G(\mathbf{z}) + \lambda \left(\sum_{i=1}^{m} D(i) z_i - 2\gamma \right) \right]. \tag{13.15}$$

This is because if $\mathbf{z} \in \mathcal{Z}(D)$, so that

$$\sum_{i=1}^{m} D(i) z_i \ge 2\gamma, \tag{13.16}$$

then the infimum appearing in equation (13.15), taken over all $\lambda \ge 0$, will be realized when $\lambda = 0$, and so will be equal to $G(\mathbf{z})$. On the other hand, if $\mathbf{z} \notin \mathcal{Z}(D)$, so that equation (13.16) does not hold, then the infimum will be $-\infty$, as can be seen by setting λ to be arbitrarily large.

We now introduce an approximation based on the fact, pointed out in section 6.1.3, that the "max min" of any function taken over any set is always upper bounded by its "min max" (and likewise when using infima or suprema). In other words, for any function $f : U \times V \to \mathbb{R}$ defined over sets U and V,

$$\sup_{u\in U} \inf_{v\in V} f(u, v) \le \inf_{v\in V} \sup_{u\in U} f(u, v).$$

Applied here, this shows that the right-hand side of equation (13.15) is at most

$$\inf_{\lambda\ge 0} \max_{\mathbf{z}\in\{-1,+1\}^m} \left[G(\mathbf{z}) + \lambda \left(\sum_{i=1}^{m} D(i) z_i - 2\gamma \right) \right].$$

Thus,

$$\min_D \max_{z \in \mathcal{Z}(D)} G(\mathbf{z}) \leq \min_D \inf_{\lambda \geq 0} \max_{\mathbf{z} \in \{-1,+1\}^m} \left[G(\mathbf{z}) + \lambda \left(\sum_{i=1}^m D(i) z_i - 2\gamma \right) \right]$$

$$= \min_D \inf_{\lambda \geq 0} \max_{\mathbf{z} \in \{-1,+1\}^m} \left[G(\mathbf{z}) + \sum_{i=1}^m \lambda D(i)(z_i - 2\gamma) \right] \tag{13.17}$$

since D is a distribution. Note that by setting

$$w_i = \lambda D(i), \tag{13.18}$$

the minimum over D and the infimum over λ can be collapsed into a single infimum over a vector $\mathbf{w}$ with all nonnegative components (which do not necessarily sum to 1). In this way, equation (13.17) can be rewritten as

$$\inf_{\mathbf{w} \in \mathbb{R}_+^m} \max_{\mathbf{z} \in \{-1,+1\}^m} \left[G(\mathbf{z}) + \sum_{i=1}^m w_i \cdot (z_i - 2\gamma) \right]. \tag{13.19}$$

As a final simplification, equation (13.13) implies that equation (13.19) is at most

$$\inf_{\mathbf{w} \in \mathbb{R}_+^m} \max_{\mathbf{z} \in \{-1,+1\}^m} \sum_{i=1}^m [g_i(z_i) + w_i \cdot (z_i - 2\gamma)]$$

$$= \inf_{\mathbf{w} \in \mathbb{R}_+^m} \sum_{i=1}^m \max_{z_i \in \{-1,+1\}} [g_i(z_i) + w_i \cdot (z_i - 2\gamma)]$$

$$= \sum_{i=1}^m \inf_{w_i \geq 0} \max_{z_i \in \{-1,+1\}} [g_i(z_i) + w_i \cdot (z_i - 2\gamma)] \tag{13.20}$$

since each maximum and each infimum can be evaluated independently for each component. This completes the proof. ∎

In fact, the simplified optimization problem appearing on the right-hand side of equation (13.14) can easily be solved separately for each chip using the following:

Lemma 13.2 If $g(+1) \leq g(-1)$, then

$$\inf_{w \geq 0} \max_{z \in \{-1,+1\}} [g(z) + w \cdot (z - 2\gamma)] = \left(\tfrac{1}{2} + \gamma\right) g(+1) \; + \; \left(\tfrac{1}{2} - \gamma\right) g(-1). \tag{13.21}$$

Moreover, the infimum on the left is realized when

$$w = \frac{g(-1) - g(+1)}{2}. \tag{13.22}$$

Proof Writing out the maximum gives

$$\max_{z\in\{-1,+1\}} [g(z) + w \cdot (z - 2\gamma)] = \max\{\, g(-1) + w \cdot (-1 - 2\gamma),\ g(+1) + w \cdot (1 - 2\gamma)\,\}. \tag{13.23}$$

As a function of w, this is the maximum of two lines, one with negative slope and the other with positive slope; moreover, the y-intercept of the latter line is below that of the former. Thus, the function is as plotted in figure 13.2. Evidently, the minimum occurs where the two lines intersect, that is, at the value given in equation (13.22). Plugging in this value for w gives equation (13.21). ■

Armed with these lemmas, we can recursively derive a good, decomposable upper bound on Λ_t. Specifically, we will find a bound of the form

$$\Lambda_t(\mathbf{s}) \le \frac{1}{m}\sum_{i=1}^{m}\Phi_t(s_i) \tag{13.24}$$

for all rounds t and all position vectors $\mathbf{s}$. To do so, we first let

$$\Phi_T(s) \doteq \mathbf{1}\{s \le 0\} \tag{13.25}$$

so that equation (13.24) holds with equality when $t = T$ (by equations (13.9) and (13.11)). Next, for $t = 1, \ldots, T$, we define

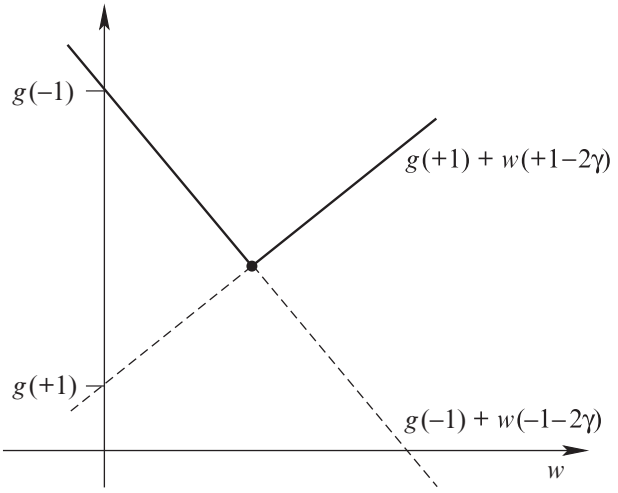

Figure 13.2
A plot of equation (13.23) as a function of w.

$$\Phi_{t-1}(s) \doteq \inf_{w \geq 0} \max_{z \in \{-1,+1\}} [\Phi_t(s+z) + w \cdot (z - 2\gamma)]. \tag{13.26}$$

Then equation (13.24) will hold by backwards induction since, by the recursive expression for Λ_t given in equation (13.12) (multiplied on both sides by m), we have

$$\begin{aligned} m\Lambda_{t-1}(\mathbf{s}) &= \min_{D} \max_{\mathbf{z} \in \mathcal{Z}(D)} m\Lambda_t(\mathbf{s}+\mathbf{z}) \\ &\leq \sum_{i=1}^{m} \inf_{w \geq 0} \max_{z \in \{-1,+1\}} [\Phi_t(s_i+z) + w \cdot (z - 2\gamma)] \\ &= \sum_{i=1}^{m} \Phi_{t-1}(s_i). \end{aligned}$$

Here, we used equation (13.24) inductively and applied lemma 13.1 with

$$G(\mathbf{z}) = m\Lambda_t(\mathbf{s}+\mathbf{z}) \tag{13.27}$$

and

$$g_i(z) = \Phi_t(s_i + z). \tag{13.28}$$

Moreover, lemma 13.2 gives

$$\Phi_{t-1}(s) = \left(\tfrac{1}{2} + \gamma\right) \Phi_t(s+1) \; + \; \left(\tfrac{1}{2} - \gamma\right) \Phi_t(s-1), \tag{13.29}$$

which, with equation (13.25), can be solved in closed form to give

$$\Phi_t(s) = \text{Binom}\left(T - t, \; \frac{T - t - s}{2}, \; \frac{1}{2} + \gamma\right) \tag{13.30}$$

where $\text{Binom}\,(n, k, p)$ denotes the probability of at most k heads (k not necessarily an integer) in n flips of a coin whose probability of heads is p:

$$\text{Binom}\,(n, k, p) \doteq \sum_{j=0}^{\lfloor k \rfloor} \binom{n}{j} p^j (1-p)^{n-j}.$$

Equation (13.30) can be verified by backwards induction using equation (13.29), while simultaneously verifying that the conditions of lemma 13.2 are satisfied (see exercise 13.3).

The function $\Phi_t(s)$ is called the *potential function*. As we have seen, it can be intuitively interpreted as the potential loss associated with a single chip at position s at the end of round t. This is discussed further in section 13.1.7. Note that $\Phi_t(s)$ depends implicitly on both the total number of rounds T and the edge γ.

13.1.5 Algorithm

Based on this development, we can now state a bound on the loss suffered by an optimal boosting algorithm, which we saw earlier is $\Lambda_0(\mathbf{0})$. In particular, setting $t = 0$, and noting that all chips begin at position 0, we have shown that this optimal loss is bounded as

$$\Lambda_0(\mathbf{0}) \leq \frac{1}{m}\sum_{i=1}^{m}\Phi_0(0) = \Phi_0(0) = \text{Binom}\left(T,\ \frac{T}{2},\ \frac{1}{2}+\gamma\right) \tag{13.31}$$

by equations (13.24) and (13.30). This is exactly equal to equation (13.4), our earlier lower bound for the oblivious weak learner.

This optimal algorithm, according to the argument above, selects on round t that distribution D_t which realizes the minimum in equation (13.12). It is unclear how to compute this distribution tractably. However, we can instead use the distribution given by our approximation. In particular, the proof of lemma 13.1, specifically equation (13.18), suggests that this distribution should be set proportionally to the values w_i, where, tracing through the proof, we see that w_i realizes the infimum on the right-hand side of equation (13.14).

In our case, on round t with the chips at position $\mathbf{s}$, lemma 13.1 is applied with G and g_i as in equations (13.27) and (13.28). The foregoing discussion then prescribes that we first choose a weight w_i for each chip i in the manner described above. Specifically, by our choice of g_i, we should choose $w_i = w_t(s_i)$ where w_t is the *weighting function*

$$w_t(s) \doteq \arg\min_{w\geq 0}\max_{z\in\{-1,+1\}}\left[\Phi_t(s+z) + w\cdot(z-2\gamma)\right]. \tag{13.32}$$

Like the potential function, the weighting function is central to our development. As for $\Phi_t(s)$, the notation $w_t(s)$ hides implicit dependence on T and γ.

Using lemma 13.2 and equation (13.30), the expression appearing in equation (13.32) can be put in closed form as

$$w_t(s) = \frac{\Phi_t(s-1) - \Phi_t(s+1)}{2} \tag{13.33}$$

$$= \frac{1}{2}\binom{T-t}{\lfloor\frac{T-t-s+1}{2}\rfloor}\left(\frac{1}{2}+\gamma\right)^{\lfloor(T-t-s+1)/2\rfloor}\left(\frac{1}{2}-\gamma\right)^{\lceil(T-t+s-1)/2\rceil} \tag{13.34}$$

(see exercise 13.3). Finally, having computed the weights w_i, we can choose a distribution for round t that is proportional to these weights:

$$D_t(i) \propto w_t(s_i).$$

Pulling all of these ideas together, we can finally see the boost-by-majority algorithm emerging, as in algorithm 13.1, where we have reverted to a description in terms of the original boosting problem rather than the chip-game abstraction. The algorithm proceeds like

Algorithm 13.1
The boost-by-majority algorithm

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in \mathcal{X}$, $y_i \in \{-1, +1\}$
edge $\gamma > 0$ and number of rounds T.
Initialize: $s_{0,i} = 0$ for $i = 1, \ldots, m$.
For $t = 1, \ldots, T$:

- $D_t(i) = \dfrac{w_t(s_{t-1,i})}{\mathcal{Z}_t}$ for $i = 1, \ldots, m$
  where $\mathcal{Z}_t$ is a normalization factor and

$$w_t(s) \doteq \frac{1}{2}\binom{T-t}{\lfloor \frac{T-t-s+1}{2} \rfloor}\left(\frac{1}{2}+\gamma\right)^{\lfloor (T-t-s+1)/2 \rfloor}\left(\frac{1}{2}-\gamma\right)^{\lceil (T-t+s-1)/2 \rceil}.$$

- Train weak learner using distribution D_t.
- Get weak hypothesis $h_t : \mathcal{X} \to \{-1, +1\}$ with sufficiently small error:

$$\mathbf{Pr}_{i \sim D_t}[h_t(x_i) \neq y_i] \leq \tfrac{1}{2} - \gamma.$$

- Update, for $i = 1, \ldots, m$:

$$s_{t,i} = s_{t-1,i} + y_i h_t(x_i).$$

Output the final hypothesis:

$$H(x) = \operatorname{sign}\left(\sum_{t=1}^{T} h_t(x)\right).$$

AdaBoost, on each round constructing a distribution and training a given weak learning algorithm. Here, of course, we require that each weak hypothesis have weighted error at most $\frac{1}{2} - \gamma$. The principal difference from AdaBoost is in the choice of distribution. BBM chooses $D_t(i)$, as described above, proportional to the weighting function w_t given in equation (13.34) evaluated at the current unnormalized margin (chip position) $s_{t-1,i}$ of example i. The final hypothesis is an unweighted majority vote of all the weak hypotheses.

NonAdaBoost, the nonadaptive version of AdaBoost described in section 13.1.1, is identical except that we instead would use $w_t(s) = e^{-\alpha s}$ where α is as set in equation (13.5).

13.1.6 Analysis

We also have everything in place to analyze this algorithm. The key to this analysis is a proof that the total potential of all the chips (training examples) together can never increase

from one round to the next. This will immediately yield a bound on BBM's training error, which is exactly equal to the average potential at the end of the game.

Theorem 13.3 Using the notation above and in algorithm 13.1, the total potential of all training examples in BBM can never increase. That is, for $t = 1, \ldots, T$:

$$\sum_{i=1}^{m} \Phi_{t-1}(s_{t-1,i}) \geq \sum_{i=1}^{m} \Phi_t(s_{t,i}).$$

Proof From equations (13.32) and (13.26), we have that for any s,

$$\Phi_{t-1}(s) = \max_{z \in \{-1,+1\}} [\Phi_t(s+z) + w_t(s) \cdot (z - 2\gamma)].$$

Therefore, letting $z_{t,i} \doteq y_i h_t(x_i)$ and plugging in $s = s_{t-1,i}$, we see that

$$\Phi_{t-1}(s_{t-1,i}) \geq \Phi_t(s_{t-1,i} + z_{t,i}) + w_t(s_{t-1,i}) \cdot (z_{t,i} - 2\gamma) \tag{13.35}$$

$$= \Phi_t(s_{t,i}) + w_t(s_{t-1,i}) \cdot (z_{t,i} - 2\gamma). \tag{13.36}$$

(Actually, it can be shown that equation (13.35) must always hold with equality, but this stronger fact is not needed for the proof.) Since we assumed empirical γ-weak learnability, h_t must have edge γ as in equation (13.1) or, equivalently, equation (13.8). These conditions can be rewritten as

$$\frac{\sum_{i=1}^{m} w_t(s_{t-1,i}) z_{t,i}}{\sum_{i=1}^{m} w_t(s_{t-1,i})} \geq 2\gamma$$

by definition of D_t. That is,

$$\sum_{i=1}^{m} w_t(s_{t-1,i}) \cdot (z_{t,i} - 2\gamma) \geq 0.$$

Combining with equation (13.36) gives

$$\sum_{i=1}^{m} \Phi_{t-1}(s_{t-1,i}) \geq \sum_{i=1}^{m} \left[\Phi_t(s_{t,i}) + w_t(s_{t-1,i}) \cdot (z_{t,i} - 2\gamma)\right]$$

$$= \sum_{i=1}^{m} \Phi_t(s_{t,i}) + \sum_{i=1}^{m} w_t(s_{t-1,i}) \cdot (z_{t,i} - 2\gamma)$$

$$\geq \sum_{i=1}^{m} \Phi_t(s_{t,i}),$$

as claimed. ∎

A bound on the training error now follows immediately. The optimality that the bound implies for BBM is discussed below.

Corollary 13.4 Using the notation of algorithm 13.1, the training error of BBM's final classifier H is at most

$$\text{Binom}\left(T,\ \frac{T}{2},\ \frac{1}{2}+\gamma\right) \doteq \sum_{j=0}^{\lfloor T/2 \rfloor} \binom{T}{j} \left(\tfrac{1}{2}+\gamma\right)^j \left(\tfrac{1}{2}-\gamma\right)^{T-j}. \tag{13.37}$$

Proof Repeatedly applying theorem 13.3 gives that

$$\Phi_0(0) = \frac{1}{m}\sum_{i=1}^{m} \Phi_0(s_{0,i}) \geq \frac{1}{m}\sum_{i=1}^{m} \Phi_T(s_{T,i}).$$

The expression on the right is, by definition, exactly the training error of the final hypothesis H. And by the general formula for $\Phi_t(s)$ in equation (13.30), the expression on the left, $\Phi_0(0)$, is equal to equation (13.37). ∎

13.1.7 Game-Theoretic Optimality

In section 13.1.1, we saw that when the number of chips is large, a version of the oblivious weak learner forces any booster to have training error approaching equation (13.37). Thus, corollary 13.4 shows that BBM and the oblivious weak learner are essentially optimal for their respective roles in the game.

Recall that the oblivious weak learner treats each chip independently of the other chips, as well as the history of the game. In the boosting setting, this corresponds to weak hypotheses whose predictions are independent of one another. Such a case would intuitively seem to be especially favorable to learning. However, we now see that this case is actually (close to) the worst possible in our current adversarial setting. Furthermore, we see that when the weak hypotheses are not independent, BBM is able to effectively force them to behave as if they were, achieving exactly the same training error as if in the case of full independence.

The near optimality of the oblivious weak learner is also helpful in interpreting the potential function $\Phi_t(s)$ and its relationship to BBM. Indeed, as can be seen from the expression for $\Phi_t(s)$ in equation (13.30) or, alternatively, from its recursive formulation in equations (13.29) and (13.25), $\Phi_t(s)$ is exactly the probability that a chip at position s will end up at a nonpositive position at the end of the game when the remaining $T-t$ rounds of the game are played against the oblivious weak learner. That is,

$$\Phi_t(s) = \mathbf{Pr}[s + z_{t+1} + \cdots + z_T \leq 0], \tag{13.38}$$

where $z_{t+1}, \ldots, z_T$ are independent random variables, each equal to $+1$ with probability $\frac{1}{2} + \gamma$, and -1 otherwise. Thus, the average potential of all the training examples, which is the key quantity used in the analysis of section 13.1.6, is exactly the expected training error under the same assumption. In other words, theorem 13.3 can be understood as a proof that the expected training error never increases from one round to the next, where expectation is over imagined random play of all future rounds by an oblivious weak learner. And corollary 13.4 then follows by observing that the training error at the end of the game, when there are no more rounds to be played, is therefore at most the expected training error before the game begins.

Regarding the weights $w_t(s)$, note that a chip that begins round t at position s will either end the round at position $s + 1$ with potential $\Phi_t(s + 1)$, or at position $s - 1$ with potential $\Phi_t(s - 1)$. Thus, the relative impact of decrementing rather than incrementing the position of that chip is proportional to $\Phi_t(s - 1) - \Phi_t(s + 1)$, which intuitively helps explain the choice of weights $w_t(s)$ that BBM places on the chip as in equation (13.33).

As we have formulated the chip game, the optimality of BBM and the oblivious weak learner is not entirely satisfying since the latter is technically not even a valid weak learner for this game, and the former only approximates the optimal player of section 13.1.3. We can, however, modify the game in such a way that the two players are both valid and optimal. We briefly sketch two such modifications.

In the first of these, we allow the weak learner to make *randomized* choices in its predictions for the individual examples, and thus in the movement of the chips. In other words, the weak learner must now choose on every round a random weak hypothesis h_t whose *expected* error cannot exceed $\frac{1}{2} - \gamma$. In terms of the chip game, this is equivalent to the weak learner selecting a distribution over vectors $\mathbf{z}_t \in \{-1, +1\}^m$ with the requirement that equations (13.7) and (13.8) hold *in expectation* over the random choice of $\mathbf{z}_t$ when chosen according to the selected distribution. The loss of the game can then be computed in expectation with respect to all of the randomized choices of the weak learner. Note that the oblivious weak learner, as described in section 13.1.1, is now a valid weak learner under this relaxed reformulation of the game. Moreover, our analysis of BBM can be shown to hold for this relaxed game as well, thus yielding matching upper and lower bounds on the (expected) loss, and so implying that both BBM and the oblivious weak learner are exactly optimal for their respective roles. (See exercise 13.9.)

In an alternative but closely related relaxation of the game, we do not allow the weak learner to make randomized choices, but instead endow the weak learner with the additional power to split or divide chips. In other words, rather than having to increment or decrement each chip as a single, indivisible unit, the weak learner may choose to split the chip into two parts—not necessarily of equal size—one which is incremented and one which is decremented. Thus, the chips behave more like globs of Jell-O which can be cut arbitrarily into

smaller globs. These split globs can in turn be split again on future rounds. As before, the weak learner must on each round increment at least a fraction $\frac{1}{2} + \gamma$ of the Jell-O, as weighted by the distribution chosen by the booster; and the final loss is the fraction of the initial quantity of Jell-O at a final position of zero or below.

For this modified game, the counterpart of the oblivious weak learner can be implemented exactly as the strategy that divides every glob into unequal halves, incrementing a fraction $\frac{1}{2} + \gamma$ of the glob, and decrementing the remaining fraction. For this strategy, the fraction of all the Jell-O at nonpositive positions after T rounds will be exactly as given in equation (13.4) by a similar argument.

Furthermore, BBM can be shown to be exactly optimal for this relaxed game as well. The algorithm can be derived just as before, where now it can be shown that the central approximation proved in lemma 13.1 holds with equality. Thus, the optimal game-theoretic algorithm, which turns out to be BBM (suitably modified for globs rather than chips), can be obtained exactly in this case. The argument leading to corollary 13.4 can also be modified and shown to hold for this game. Since, as before, the upper and lower bounds match exactly, we see that BBM and the oblivious weak learner are the game-theoretic optimal players for this game as well.

All this suggests a different, though closely related, approach to the derivation of tractable, nearly optimal players. To approximate the optimal player for the original game, we first relax the game itself in a way that increases the power of the adversary (weak learner), and then compute the optimal player for the modified game, which in this case yields BBM.

13.2 Optimal Generalization Error

Having analyzed BBM's training error and its near optimality for the voting game, we turn next to a study of the generalization error, whose minimization is the object of learning. We continue to focus in this section on the case that the boosting algorithm is permitted to make T calls to a weak learning algorithm satisfying the empirical γ-weak learning condition. Under this assumption, we will see that BBM's generalization error is *exactly* the best possible for *any* boosting algorithm when the number of training examples becomes large (with T and γ held fixed).

13.2.1 An Upper Bound for BBM

Not surprisingly, the results of chapter 4 can be immediately and directly applied to derive bounds on the generalization error of BBM. (In fact, in most cases an even simpler analysis could have been used since BBM always outputs a combined classifier that is an *unweighted* majority vote of the base hypotheses.) Specifically, theorems 4.3 and 4.6, applied to BBM, show that the generalization error $\mathrm{err}(H)$ of the combined hypothesis H can be bounded in terms of the training error $\widehat{\mathrm{err}}(H)$ as

$$\text{err}(H) \leq \widehat{\text{err}}(H) + \tilde{O}\left(\sqrt{\frac{TC}{m}}\right) \tag{13.39}$$

where m is the number of training examples, T is the number of rounds, and C is a measure of the complexity of the base hypothesis space $\mathcal{H}$, either $\ln |\mathcal{H}|$ or its VC-dimension d.

Likewise, similar to the discussion in section 4.2, when boosting by resampling is used, we can represent H by a sequence of Tm_0 training examples, where m_0 is the number of examples required by the weak learner. In other words, BBM can be viewed as a compression scheme of size Tm_0. Applying theorem 2.8 with $\kappa = Tm_0$ then gives again a bound of the form in equation (13.39), where the complexity C is now replaced by the weak learning sample size m_0.

Thus, assuming either a bound on the number of examples needed for weak learning or a bound on the complexity of the base hypothesis space $\mathcal{H}$, we see that the generalization error of BBM can be upper bounded as in equation (13.39). Moreover, applying corollary 13.4 immediately gives us a bound of the form

$$\text{err}(H) \leq \text{Binom}\left(T,\ \frac{T}{2},\ \frac{1}{2}+\gamma\right) + \tilde{O}\left(\sqrt{\frac{TC}{m}}\right).$$

This means that as m becomes large with T and γ fixed, the rightmost term becomes negligible, and we obtain an upper bound on the generalization error that approaches the bound given in corollary 13.4. As we show next, this latter bound is exactly the best achievable by any boosting algorithm, meaning that BBM is, in this sense, optimal.

13.2.2 A General Lower Bound

To prove such a lower bound on the generalization error, we need to begin by defining what we mean by a boosting algorithm. Here, we will return to the formal definitions given in section 2.3. Recall that a weak learning algorithm A in the PAC model for target class $\mathcal{C}$ has the property that for some $\gamma > 0$, for all $c \in \mathcal{C}$, and for all distributions $\mathcal{D}$ over the domain $\mathcal{X}$, if given $\delta > 0$ and $m_0 = m_0(\delta)$ examples $(x_1, c(x_1)), \ldots, (x_{m_0}, c(x_{m_0}))$ where each x_i is independently distributed according to $\mathcal{D}$, the algorithm will, with probability at least $1 - \delta$, output a hypothesis h with error at most $\frac{1}{2} - \gamma$ with respect to $\mathcal{D}$. In this context, we refer to γ as A's edge.

A boosting algorithm B is one which, when provided with access to a weak PAC learning algorithm A for $\mathcal{C}$, as well as $\epsilon > 0$, $\delta > 0$, and $m = m(\epsilon, \delta)$ examples labeled according to any $c \in \mathcal{C}$, and drawn according to any distribution $\mathcal{D}$, will with probability at least $1 - \delta$, output a hypothesis H with error at most ϵ with respect to $\mathcal{D}$. Note, importantly, that a boosting algorithm should be general in the sense of *not* requiring knowledge of the target class $\mathcal{C}$ (although we do allow it to have other information about the weak learner, such as the required sample size m_0, the associated edge γ, etc.). We also require that B's sample size m be polynomial in the appropriate parameters. Aside from this requirement, B is entirely

unrestricted and is allowed, for instance, to have a superpolynomial running time, or to output a combined hypothesis of any form whatsoever.

We will prove a lower bound on the generalization error that can be achieved by a boosting algorithm for a fixed number of calls T to the weak learner, and with fixed edge $\gamma > 0$. Alternatively, such a bound can be inverted to give a sharp lower bound on the number of rounds that are necessary for any boosting algorithm to achieve a given target generalization error ϵ (still with fixed edge γ). Thus, these bounds characterize the optimal efficiency of any boosting algorithm in terms of how the number of rounds must depend on the desired accuracy.

The intuitive idea for the proof of the lower bound is the same as in section 13.1.1, namely, to use a variant of the oblivious weak learner which produces a random hypothesis that is correct on each example with probability $\frac{1}{2} + \gamma$. Such a hypothesis will, in expectation, have error $\frac{1}{2} - \gamma$ for any distribution. Moreover, no matter how such weak hypotheses are combined for making predictions on new data, there will always linger some chance of a mistake; this probability, which happens to match the upper bound for BBM given in corollary 13.4, will provide the lower bound we seek.

Formally, we will prove the following:

Theorem 13.5 Let B be any boosting algorithm as defined above, let $0 < \gamma < \frac{1}{2}$, and let T be a positive odd integer. Then for any $\nu > 0$, there exist a target class $\mathcal{C}$, a distribution $\mathcal{D}$, a target function $c \in \mathcal{C}$, and a weak learning algorithm A for $\mathcal{C}$ with edge γ such that if B makes T calls to A, then the generalization error of its combined classifier will be at least

$$\text{Binom}\left(T,\ \frac{T}{2},\ \frac{1}{2}+\gamma\right) - \nu \tag{13.40}$$

with probability at least $1 - \nu$ (where the probability is taken with respect to the random sample provided to B and any internal randomization used by A and B).

The theorem says that it is nearly certain that B's generalization error will be at least equation (13.40). In other words, if B's confidence parameter δ is chosen to be smaller than $1 - \nu$, then its error parameter ϵ cannot be made smaller than equation (13.40) without increasing T.

The proof will occupy the remainder of this section. Although the intuitive idea outlined above is simple, there are many subtle but technical details that will need to be worked out to ensure that all of the formal requirements of the learning model are satisfied, especially with respect to the definition of a weak learning algorithm. This proof is not crucial to understanding the other material in this chapter.

13.2.3 The Construction

We begin the proof with the construction of the target class $\mathcal{C}$ and the weak learning algorithm A, and later prove that the generalization error will be as given in the theorem when A is used as a weak learner for B.

Let the domain $\mathcal{X} = \{0, 1\}^n$, the set of all n-bit strings, and let the target distribution $\mathcal{D}$ be uniform over $\mathcal{X}$. The positive integer n will act as a "complexity parameter"—instances are all of length n; the hypotheses that A constructs will be representable using strings of length polynomial in n; and A will have time and sample complexity polynomial in n. We will also allow B to use any number of examples m that can be bounded by a polynomial in n. Since ϵ, γ, δ, and T are all effectively fixed, this will be the case for any boosting algorithm whose sample complexity is polynomial either in the sample complexity or in the hypothesis complexity of the weak learner.

We will use the *probabilistic method* to construct both $\mathcal{C}$ and the base hypothesis space $\mathcal{H}$ used by A. This means that we will imagine that $\mathcal{C}$ and $\mathcal{H}$ are chosen *randomly* according to some appropriately constructed probability distribution. We will then show that, in expectation over the choice of the classes $\mathcal{C}$ and $\mathcal{H}$, the conclusion of the theorem is satisfied, which clearly implies the *existence* of such classes.

To construct the target class $\mathcal{C}$, we first select a random function $c : \mathcal{X} \to \{-1, +1\}$ by independently, for each $x \in \mathcal{X}$, choosing $c(x)$ to be -1 or $+1$ with equal probability. Then the class $\mathcal{C}$ is chosen simply to consist of this single function: $\mathcal{C} = \{c\}$. This class is obviously very small and trivial. In fact, with knowledge of $\mathcal{C}$, the target c can be "learned" with no data at all since c is the only function in the class. However, as pointed out earlier, the boosting algorithm does *not* know $\mathcal{C}$, even though the weak learner does.

We next construct the weak learning algorithm A. Of course, since A knows c, it might simply output the hypothesis $h = c$, but this would make the learning process rather trivial for the booster, whereas our purpose in this construction is just the opposite—that is, for A to release as little information about c as possible while still satisfying the γ-weak learning condition.

As suggested informally above, we would ideally like to use the notion of an oblivious weak learner that randomly chooses a hypothesis h such that, for each x, $h(x)$ is chosen randomly to be $c(x)$ with probability $\frac{1}{2} + \gamma$ and $-c(x)$ otherwise. However, there are a number of technicalities that need to be addressed. First, although the *expected* error of such a hypothesis is exactly $\frac{1}{2} - \gamma$, weak learning demands that the error *actually* be at most $\frac{1}{2} - \gamma$ with high probability. This difficulty can first be addressed by choosing $h(x)$ to be correct with slightly higher probability than $\frac{1}{2} + \gamma$, say $\frac{1}{2} + \gamma'$ for some $\gamma' > \gamma$. It turns out that this will be sufficient to ensure an error of $\frac{1}{2} - \gamma$ (with high probability), provided that the target distribution D generating examples for the weak learner is "smooth" in the sense of no examples having "large" weight under the distribution. (This distribution should not be confused with the distribution $\mathcal{D}$ that generates examples for the booster. The target distribution D, from the weak learner's perspective, is one that will be constructed by the booster.)

However, when substantial probability mass is concentrated on one or more examples, we face a further difficulty. At an extreme, when D is concentrated entirely on a single example x_0, choosing h randomly as above will result, with respect to D, in an error below $\frac{1}{2} - \gamma$

exactly when $h(x_0) = c(x_0)$, which happens with probability $\frac{1}{2} + \gamma'$. Thus, in this case, there is no way to guarantee that the weak learning condition will hold *with high probability*, that is, with probability close to 1.

We can address this problem by designing a weak learner that identifies the examples with large mass under the target distribution D, and then augments the random oblivious hypothesis with an *exception list* that includes the correct classifications of all of the large-mass examples. In other words, such a hypothesis predicts $h(x) = c(x)$ if x has large mass, and otherwise chooses $h(x)$ randomly as before. Since the number of large-mass examples in any distribution is necessarily small, the exception lists will also always be short.

Finally, we note that the hypotheses we have so far discussed have very long descriptions: every function mapping $\mathcal{X}$ to $\{-1, +1\}$ has a nonzero probability of being generated, which means each hypothesis requires 2^n bits to specify, and that the size of the entire hypothesis space is 2^{2^n}, far too large to admit learning. To alleviate this difficulty, before learning begins, we will construct a much smaller hypothesis space consisting of a relatively small number of "ground" hypotheses, each produced using the random, oblivious process above, together with all possible hypotheses that can be obtained by adding exception lists. The weak learner can then choose from this preselected space of hypotheses.

Having sketched the main ideas of the construction, we turn now to the details. The hypothesis space $\mathcal{H}$ used by the weak learner is constructed as follows. First, a set of *ground hypotheses* is selected:

$$\mathcal{G} \doteq \left\{\overline{g}_r : r \in \{0,1\}^n\right\}.$$

Each of the 2^n ground hypotheses is indexed by an n-bit string r called the *seed*. The classifier $\overline{g}_r$ is constructed randomly by letting

$$\overline{g}_r(x) = \begin{cases} c(x) & \text{with probability } \frac{1}{2} + \gamma' \\ -c(x) & \text{with probability } \frac{1}{2} - \gamma' \end{cases} \tag{13.41}$$

independently for each x where we define

$$\gamma' \doteq \gamma + 2\Delta,$$

and

$$\Delta \doteq \frac{1}{\sqrt{n}}.$$

The weak hypotheses in $\mathcal{H}$ include all ground hypotheses augmented with all possible exception lists of length at most n^2. That is, each hypothesis in $\mathcal{H}$ has the form $\hbar_{r,E}$ where r is a seed, and E is the exception list, a set of at most n^2 examples:

$$\mathcal{H} \doteq \left\{\hbar_{r,E} : r \in \{0,1\}^n, E \subseteq \mathcal{X}, |E| \leq n^2\right\}.$$

Algorithm 13.2
The weak learning algorithm A', designed for boosting by reweighting

Given: distribution D over $\mathcal{X}$ (and built-in knowledge of $\mathcal{C}$, $\mathcal{H}$, γ, and Δ).

- Choose an n-bit seed r uniformly at random.
- Let E be the set of all examples with probability mass at least $1/n^2$:

$$E = \left\{ x \in \mathcal{X} : D(x) \geq \frac{1}{n^2} \right\}.$$

- If the error of $\hbar_{r,E}$ with respect to D is at most $\frac{1}{2} - \gamma - \Delta$, then output $\hbar_{r,E}$; otherwise, abort by outputting c.

Such a hypothesis correctly classifies all examples in E, and classifies all other examples using the ground hypothesis $\overline{g}_r$:

$$\hbar_{r,E}(x) = \begin{cases} c(x) & \text{if } x \in E \\ \overline{g}_r(x) & \text{else.} \end{cases}$$

Finally, we are ready to describe the weak learning algorithm. For simplicity, we first describe a weak learner A' designed for boosting by reweighting. As discussed in section 3.4.1, such a weak learner receives as input an actual distribution D over a set of training examples (though here treated formally as a distribution over the entire domain $\mathcal{X}$), and must produce a weak hypothesis with error at most $\frac{1}{2} - \gamma$ with respect to the given distribution D. Such a weak learning algorithm does not satisfy the formal definition reviewed in section 13.2.2, although boosting is commonly combined with such weak learners, as discussed in section 3.4.1. Later, we describe a boosting-by-resampling version as formally required. Thus, the proof will actually apply to either form of boosting.

The algorithm A', shown as algorithm 13.2, pulls together the informal ideas presented earlier, identifying high-mass examples (with probability at least $1/n^2$) which are placed on an exception list, and classifying all other examples using a randomly chosen ground hypothesis. Note that $|E| \leq n^2$ since D is a distribution. If the resulting hypothesis $\hbar_{r,E}$ still has unacceptably high error, we say that an *abort* occurs, and in this case the target c is used as the output hypothesis (its error always being zero). This guarantees that the generated weak hypothesis will always have error below $\frac{1}{2} - \gamma$ (actually, even slightly better than this). The next lemma shows, moreover, that aborts occur only very rarely:

Lemma 13.6 Let c be fixed, and suppose A' is run on distribution D. Let r be the seed chosen on the first step, and assume that $\overline{g}_r$, as a random variable, is independent of D. Then

the probability of an abort, that is, the probability that $\hbar_{r,E}$ has error exceeding $\frac{1}{2} - \gamma - \Delta$ with respect to D, is at most e^{-2n}.

Proof The error of $\hbar_{r,E}$ can be written as

$$\mathrm{err}(\hbar_{r,E}) = \sum_{x \in \mathcal{X}-E} D(x)\mathbf{1}\{\overline{g}_r(x) \neq c(x)\} = \sum_{x \in \mathcal{X}-E} D(x) I_x$$

where each I_x is an independent random variable that is equal to 1 with probability $\frac{1}{2} - \gamma'$ and 0 otherwise. To bound this weighted sum, we use a generalized form of Hoeffding's inequality (theorem 2.1) which states the following:

Theorem 13.7 Let $X_1, \ldots, X_m$ be independent random variables such that $X_i \in [0, 1]$. Let $w_1, \ldots, w_m$ be a set of nonnegative weights. Denote the weighted sum of the random variables by $S_m = \sum_{i=1}^m w_i X_i$. Then for any $\varepsilon > 0$ we have

$$\mathbf{Pr}[S_m \geq \mathbf{E}[S_m] + \varepsilon] \leq \exp\left(-\frac{2\varepsilon^2}{\sum_{i=1}^m w_i^2}\right)$$

and

$$\mathbf{Pr}[S_m \leq \mathbf{E}[S_m] - \varepsilon] \leq \exp\left(-\frac{2\varepsilon^2}{\sum_{i=1}^m w_i^2}\right).$$

The expected error of $\hbar_{r,E}$ is

$$\mathbf{E}\big[\mathrm{err}(\hbar_{r,E})\big] = \left(\tfrac{1}{2} - \gamma'\right) \sum_{x \in \mathcal{X}-E} D(x) \leq \tfrac{1}{2} - \gamma'.$$

So applying theorem 13.7 gives

$$\begin{aligned}\mathbf{Pr}\big[\mathrm{err}(\hbar_{r,E}) > \tfrac{1}{2} - \gamma' + \Delta\big] &\leq \mathbf{Pr}\big[\mathrm{err}(\hbar_{r,E}) > \mathbf{E}\big[\mathrm{err}(\hbar_{r,E})\big] + \Delta\big] \\ &\leq \exp\left(-\frac{2\Delta^2}{\sum_{x \in \mathcal{X}-E} D(x)^2}\right). \qquad (13.42)\end{aligned}$$

Since $D(x) < 1/n^2$ for every x not in E,

$$\sum_{x \in \mathcal{X}-E} D(x)^2 \leq \frac{1}{n^2} \sum_{x \in \mathcal{X}-E} D(x) \leq \frac{1}{n^2}.$$

Thus, equation (13.42) is at most e^{-2n} by our choice of Δ. ∎

We can now build a boost-by-resampling weak learner A which uses A' as a subroutine. Such a weak learner, when provided with $m_0(\delta)$ labeled examples chosen independently

Algorithm 13.3
The weak learning algorithm A, designed for boosting by resampling

Given: $(x_1, c(x_1)), \ldots, (x_{m_0}, c(x_{m_0}))$.

- Let $\hat{D}$ be the empirical distribution on the sample:

$$\hat{D}(x) \doteq \frac{1}{m_0} \sum_{i=1}^{m_0} \mathbf{1}\{x_i = x\}\,.$$

- Run A' on distribution $\hat{D}$, and output the returned hypothesis.

from an unknown distribution D, must, with probability at least $1-\delta$, output a hypothesis with error at most $\frac{1}{2}-\gamma$ with respect to D. Our algorithm A, shown as algorithm 13.3, simply forms the empirical distribution $\hat{D}$ in which each of the given m_0 examples is assigned probability $1/m_0$, and then runs A' on this distribution, outputting the hypothesis so obtained.

By our construction of the algorithm A', this returned hypothesis h will always have training error (which is the same as the error measured with respect to $\hat{D}$) at most $\frac{1}{2}-\gamma-\Delta$. Thus, in order that this hypothesis have error at most $\frac{1}{2}-\gamma$ with respect to the distribution D that generated the training set, it suffices to show that this true error $\mathrm{err}(h)$ exceeds its training error by at most Δ with high probability. If we are in the case $h = c$, this is trivially true since both errors are exactly zero. In all other cases, we can apply the results of section 2.2.2. In particular, each hypothesis $\hbar_{r,E}$ can be represented using n bits for the seed r, and n bits for each of up to n^2 examples on the exception list. Thus,

$$\lg|\mathcal{H}| = O(n^3).$$

Plugging into theorem 2.2, we can then calculate that a sample of size

$$m_0 = \left\lceil \frac{\ln|\mathcal{H}| + \ln(1/\delta)}{2\Delta^2} \right\rceil = O(n^4 + n\ln(1/\delta)) \tag{13.43}$$

is sufficient to guarantee that

$$\mathrm{err}(\hbar_{r,E}) \le \widehat{\mathrm{err}}(\hbar_{r,E}) + \Delta$$

for all $\hbar_{r,E} \in \mathcal{H}$ with probability at least $1-\delta$. Thus, in particular, for the h output by A, we will have

$$\mathrm{err}(h) \le \widehat{\mathrm{err}}(h) + \Delta \le \left(\tfrac{1}{2}-\gamma-\Delta\right) + \Delta = \tfrac{1}{2}-\gamma.$$

We conclude that A satisfies the definition of a weak learning algorithm for $\mathcal{C}$ with edge γ, and that its sample complexity is as given in equation (13.43).

13.2.4 Overview of the Analysis

Having finally completed our construction of the weak learner, we are ready to analyze the generalization error of the boosting algorithm B when it is used with this weak learner. Note that A' gets called on every round, whether directly by B (if using boosting by reweighting) or as a subroutine of A.

There are many sources of randomness that are part of either the learning process or of our construction, namely:

- the randomly constructed target function c;
- the randomly constructed weak hypothesis space $\mathcal{H}$;
- the training set S consisting of a sequence of m random training instances (but not their labels, which are determined by c);
- the random seeds $\mathbf{r} = \langle r_1, \ldots, r_T \rangle$ selected on the T calls to A';
- the boosting algorithm's own internal randomization, denoted by the random variable $\mathcal{R}$. (This randomization might be used for various purposes, such as random resampling of the training set when calling the weak learner. Concretely, $\mathcal{R}$ might take the form, for instance, of an infinite sequence of random bits, although such details are of no concern to us.)

To prove theorem 13.5, we will show that with respect to *all* of the sources of randomness, B's error is likely to be at least $\beta^* - \nu$, for n sufficiently large, where

$$\beta^* \doteq \text{Binom}\left(T,\ \frac{T}{2},\ \frac{1}{2} + \gamma\right). \tag{13.44}$$

That is, we will show that

$$\mathbf{Pr}_{c,\mathcal{H},S,\mathbf{r},\mathcal{R}}\big[\text{err}(H, c) \geq \beta^* - \nu\big] \geq 1 - \nu, \tag{13.45}$$

where $\text{err}(H, c)$ denotes the true error of the final hypothesis H output by B relative to the target c. Here and throughout this proof, we often add subscripts to probabilities in order to emphasize which random quantities the probability is taken over. Equation (13.45) is sufficient for the proof since it is equivalent, by marginalization, to

$$\mathbf{E}_{c,\mathcal{H}}\big[\mathbf{Pr}_{S,\mathbf{r},\mathcal{R}}\big[\text{err}(H, c) \geq \beta^* - \nu \mid c, \mathcal{H}\big]\big] \geq 1 - \nu,$$

which, in turn, implies that there exist a *particular* target c and hypothesis space $\mathcal{H}$ for which

$$\mathbf{Pr}_{S,\mathbf{r},\mathcal{R}}\big[\text{err}(H, c) \geq \beta^* - \nu \mid c, \mathcal{H}\big] \geq 1 - \nu,$$

exactly the statement of the theorem.

Here is a rough outline of how we will prove equation (13.45). The notation used here is somewhat informal and will be made more precise shortly.

First, since they have no influence on the computation of H, we can regard the choice of labels c on instances not in S as if they were still random, even after H has been computed. We can then compare the error of H with respect to the actual choice of c to its expectation. By Hoeffding's inequality, these will be close, allowing us to show that

$$\mathbf{E}_c[\mathrm{err}(H, c)] \lesssim \mathrm{err}(H, c) \tag{13.46}$$

with high probability. (As used elsewhere in this book, we write $\lesssim$ to denote informal, approximate inequality.)

Next, we will argue that with full knowledge of the random process generating c and $\mathcal{H}$, there is an optimal rule *opt* for predicting the label of a test instance, given the predictions of the weak hypotheses. Since it is optimal, we will have

$$\mathbf{E}_c[\mathrm{err}(opt, c)] \leq \mathbf{E}_c[\mathrm{err}(H, c)]\,. \tag{13.47}$$

The quantity on the left-hand side of equation (13.47) depends on the particular predictions of the weak hypotheses, which are fixed in this expression. By Hoeffding's inequality, applied a second time, this expression will be close to its expectation under the random choice of $\mathcal{H}$, so that

$$\mathbf{E}_{c,\mathcal{H}}[\mathrm{err}(opt, c)] \lesssim \mathbf{E}_c[\mathrm{err}(opt, c)] \tag{13.48}$$

with high probability.

Finally, this quantity on the left, in which both c and $\mathcal{H}$ are random according to the process used in our construction, turns out to converge to exactly β^* as in equation (13.44) as n gets large. Combined with equations (13.46), (13.47), and (13.48), we will thus obtain

$$\beta^* \approx \mathbf{E}_{c,\mathcal{H}}[\mathrm{err}(opt, c)] \lesssim \mathbf{E}_c[\mathrm{err}(opt, c)] \leq \mathbf{E}_c[\mathrm{err}(H, c)] \lesssim \mathrm{err}(H, c)$$

with high probability, a rough equivalent of equation (13.45), whose proof is our goal.

13.2.5 Viewing the Booster as a Fixed Function

In more detail, we begin by formulating a mathematical view of the boosting algorithm as a *fixed* function, a perspective that is crucial to the proof. The boosting algorithm B computes its final hypothesis H on the basis of the training sample and the weak hypotheses it receives from the weak learner. And although B may be randomized, we can regard its randomization $\mathcal{R}$ as itself an input to the algorithm. In this way, B's computation of a final hypothesis can be viewed as a *fixed and deterministic* function of:

- the training sample S;
- the labels (values of c) on the training instances in S, denoted $c_{|S}$;
- the weak hypotheses $h_1, \ldots, h_T$ returned on the T calls to A', including their values on *all* instances in $\mathcal{X}$;
- B's internal randomization $\mathcal{R}$.

As a function, B maps these inputs to a final hypothesis H which, for simplicity, we assume does not use randomization in formulating its predictions (although our argument can be generalized to handle this case as well).

We can take this understanding of B's computation a step deeper so that we are working directly with the ground hypotheses $\overline{g}_r$, rather than the actual weak hypotheses h_t returned by A'. This will simplify the analysis since the weak hypotheses may be muddled by exception lists or aborts. Let r_t and E_t denote the random seed and exception list selected by A' on the t-th call. Let $g_t \doteq \overline{g}_{r_t}$ be the corresponding ground hypothesis, and let us further define the function

$$g'_t \doteq \begin{cases} c & \text{if abort occurs on } t\text{-th call to } A' \\ g_t & \text{else.} \end{cases}$$

This definition allows us to write h_t in the unified form

$$h_t(x) = \begin{cases} c(x) & \text{if } x \in E_t \\ g'_t(x) & \text{else,} \end{cases} \tag{13.49}$$

which holds whether or not an abort occurs.

We claim that the boosting algorithm can now be viewed instead as a fixed and deterministic function of:

- the modified ground hypotheses $g'_1, \ldots, g'_T$ (rather than $h_1, \ldots, h_T$);
- the training sample S, the training labels $c_{|S}$, and B's randomization $\mathcal{R}$, just as before.

In other words, in addition to the latter items, we claim that we can view B's computation as a function of $\mathbf{g}'$ rather than $\mathbf{h}$ (where we use vector notation $\mathbf{h}$ to stand for all of the weak hypotheses $\langle h_1, \ldots, h_T \rangle$ together, and similarly for $\mathbf{g}$ and $\mathbf{g}'$). This is because equation (13.49) shows that each h_t is itself a function only of g'_t, E_t, and the labels $c(x)$ on the instances in E_t. But because each exception list is a subset of the sample S, the labels of instances appearing on such lists are actually included in $c_{|S}$. Furthermore, the exception list E_t is determined by the distribution received by A', or the sample received by A, that is, by the history up to the point at which the weak learner was invoked. Therefore, E_t is itself a fixed function of the other elements which determine B's computation. Thus, B's computation of its final hypothesis H can be viewed as a deterministic function only of S, $c_{|S}$, $\mathcal{R}$, and $\mathbf{g}'$.

Let us now *fix* several of the sources of randomness, namely, the sample S, its labeling $c_{|S}$, and the randomization $\mathcal{R}$ and $\mathbf{r}$ used by B and A'. Later, we will take expectation over these to obtain equation (13.45). For now, these can all be arbitrary, except that we assume that all of the seeds $r_1, \ldots, r_T$ are distinct.

With all of these variables held fixed and treated as constants, by the above argument the boosting algorithm can be viewed as a deterministic function only of $\mathbf{g}'$ so that its final hypothesis H is computed as

$$H = \mathcal{B}(\mathbf{g}')$$

for some fixed, deterministic function $\mathcal{B}$. We also write

$$\mathcal{B}(\mathbf{g}', x) \doteq \mathcal{B}(\mathbf{g}')(x)$$

to denote its (fixed and deterministic) prediction on a test instance x.

We assume without loss of generality that $\mathcal{B}$ is a total function in the sense of being defined for *all* inputs of the correct syntactic form (that is, functions g'_t that map $\mathcal{X}$ to $\{-1, +1\}$, and test instances $x \in \mathcal{X}$). Although $\mathcal{B}$ should properly be applied only to $\mathbf{g}'$, this assumption allows us to consider its application to $\mathbf{g}$ instead, as in $\mathcal{B}(\mathbf{g})$ or $\mathcal{B}(\mathbf{g}, x)$. Essentially, this means always using the ground hypotheses g_t on each round, ignoring the possibility of an abort condition. Mathematically, this substitution will be very convenient since although aborts are rare (by lemma 13.6), they are still a nuisance. Later, of course, we will have to account for aborts as well.

13.2.6 Analyzing the Error

Given the fixed function $\mathcal{B}$, our goal now is to analyze the error of the resulting final hypothesis $\mathcal{B}(\mathbf{g})$ relative to the target c, where c and the g_t's are generated according to the random process described in section 13.2.3 (but with S and $c_{|S}$ fixed). In particular, since all of the seeds r_t are distinct, the g_t's are generated independently of one another (conditional on c) as in equation (13.41).

As before, we denote the error, for any given H and c, by

$$\mathrm{err}(H, c) \doteq \mathbf{Pr}_{x\sim\mathcal{D}}[H(x) \neq c(x)] = 2^{-n} \cdot \sum_{x\in\mathcal{X}} \mathbf{1}\{H(x) \neq c(x)\}.$$

Also, since we are interested primarily in what happens off the training set, let us define $\overline{\mathcal{X}}$ to be the set of all instances in $\mathcal{X}$ *not* included in the sample S; let $\overline{c}$ be the restriction of c to $\overline{\mathcal{X}}$ (that is, the labels on all the points in $\overline{\mathcal{X}}$); and let

$$\overline{\mathrm{err}}(H, \overline{c}) \doteq \mathbf{Pr}_{x\sim\mathcal{D}}\big[H(x) \neq c(x) \mid x \in \overline{\mathcal{X}}\big] = \frac{1}{|\overline{\mathcal{X}}|} \cdot \sum_{x\in\overline{\mathcal{X}}} \mathbf{1}\{H(x) \neq c(x)\}$$

denote the error just on $\overline{\mathcal{X}}$.

Following the outline above, we first show that for any $\mathbf{g}$, the error of $\mathcal{B}(\mathbf{g})$ is likely to be close to its expectation under the random choice of $\overline{c}$.

Lemma 13.8 Let $\mathbf{g}$ be fixed, and let $\overline{c}$ be chosen at random, conditional on $\mathbf{g}$. Then with probability at least $1 - e^{-2n}$,

$$\overline{\mathrm{err}}(\mathcal{B}(\mathbf{g}), \overline{c}) \geq \mathbf{E}_{\overline{c}}[\overline{\mathrm{err}}(\mathcal{B}(\mathbf{g}), \overline{c}) \mid \mathbf{g}] - \sqrt{\frac{n}{|\overline{\mathcal{X}}|}}.$$

Proof Given $\mathbf{g}$, the $c(x)$'s remain independent of one another. Therefore, the random variables

$$M_x \doteq \mathbf{1}\{\mathcal{B}(\mathbf{g}, x) \neq c(x)\},$$

for $x \in \overline{\mathcal{X}}$, are independent of one another. Applying Hoeffding's inequality (theorem 2.1) to their average

$$\overline{\mathrm{err}}(\mathcal{B}(\mathbf{g}), \overline{c}) = \frac{1}{|\overline{\mathcal{X}}|} \cdot \sum_{x \in \overline{\mathcal{X}}} M_x$$

now gives the result. ∎

Let us consider a single example x in $\overline{\mathcal{X}}$. Given $\mathbf{g}$, the probability of misclassifying x depends only on $c(x)$, and can be computed to be

$$\mathbf{Pr}_{c(x)}[c(x) \neq \mathcal{B}(\mathbf{g}, x) \mid \mathbf{g}].$$

Clearly, this is at least

$$\min_{y \in \{-1,+1\}} \mathbf{Pr}_{c(x)}[c(x) \neq y \mid \mathbf{g}]\,.$$

And since $c(x)$ is conditionally independent, given $\mathbf{g}(x)$, of all the values of $\mathbf{g}$ on instances other than x, this is simply equal to

$$\min_{y \in \{-1,+1\}} \mathbf{Pr}_{c(x)}[c(x) \neq y \mid \mathbf{g}(x)]\,. \tag{13.50}$$

Let $opt(\mathbf{g}, x)$ denote the value of y that minimizes this expression, and let $opt(\mathbf{g})$ denote the prediction function $opt(\mathbf{g}, \cdot)$. This is the *Bayes optimal* classifier encountered in section 12.1.

By taking into account the manner in which $c(x)$ and $\mathbf{g}(x)$ are generated, we can determine $opt(\mathbf{g}, x)$ explicitly as follows. For $y \in \{-1, +1\}$, we have

$$\begin{aligned}
\mathbf{Pr}[c(x) = y \mid \mathbf{g}(x)] &= \frac{\mathbf{Pr}[\mathbf{g}(x) \mid c(x) = y] \cdot \mathbf{Pr}[c(x) = y]}{\mathbf{Pr}[\mathbf{g}(x)]} && (13.51)\\
&\propto \mathbf{Pr}[\mathbf{g}(x) \mid c(x) = y] && (13.52)\\
&= \prod_{t=1}^{T} \left[\left(\tfrac{1}{2} + \gamma'\right)^{\mathbf{1}\{g_t(x)=y\}} \left(\tfrac{1}{2} - \gamma'\right)^{\mathbf{1}\{g_t(x)\neq y\}} \right] && (13.53)\\
&= \prod_{t=1}^{T} \left[\left(\tfrac{1}{2} + \gamma'\right)^{(1+y g_t(x))/2} \left(\tfrac{1}{2} - \gamma'\right)^{(1-y g_t(x))/2} \right]\\
&\propto \prod_{t=1}^{T} \left(\frac{1+2\gamma'}{1-2\gamma'} \right)^{y g_t(x)/2}.
\end{aligned}$$

(In this context, we write $f \propto g$ to mean f is equal to g times a positive value that does not depend on y.) Here, equation (13.51) is exactly Bayes rule. Equation (13.52) uses the fact that $c(x)$ is equally likely to be each label. Equation (13.53) follows from the random process of generating ground hypotheses as in equation (13.41). And the last two lines are straightforward manipulations.

Thus, taking the logarithm of the ratio of this final expression when $y = +1$ or $y = -1$ gives

$$\ln\left(\frac{\mathbf{Pr}[c(x) = +1 \mid \mathbf{g}(x)]}{\mathbf{Pr}[c(x) = -1 \mid \mathbf{g}(x)]}\right) = \ln\left(\frac{1+2\gamma'}{1-2\gamma'}\right) \cdot \sum_{t=1}^{T} g_t(x).$$

The sign of the quantity on the left tells us which value of $c(x)$ is more likely, and thus which should be chosen by $opt(\mathbf{g}, x)$ to realize the minimum of equation (13.50). It follows, therefore, that

$$opt(\mathbf{g}, x) = \text{sign}\left(\sum_{t=1}^{T} g_t(x)\right). \tag{13.54}$$

(Recall that we are assuming that T is odd, so that a tie, in which the sign function receives an argument of zero, can never occur.) In other words, taking a majority vote of the T ground hypotheses is the best possible prediction in this setting.

Since it is optimal for every $\mathbf{g}$ and every x, the expected error of $opt(\mathbf{g})$ is a lower bound on that of $\mathcal{B}(\mathbf{g})$:

$$\begin{aligned}
\mathbf{E}_{\overline{c}}[\overline{\text{err}}(\mathcal{B}(\mathbf{g}), \overline{c})] &= \frac{1}{|\overline{\mathcal{X}}|} \cdot \sum_{x \in \overline{\mathcal{X}}} \mathbf{Pr}_{c(x)}[\mathcal{B}(\mathbf{g}, x) \neq c(x) \mid \mathbf{g}] \\
&\geq \frac{1}{|\overline{\mathcal{X}}|} \cdot \sum_{x \in \overline{\mathcal{X}}} \mathbf{Pr}_{c(x)}[opt(\mathbf{g}, x) \neq c(x) \mid \mathbf{g}] \\
&= \mathbf{E}_{\overline{c}}[\overline{\text{err}}(opt(\mathbf{g}), \overline{c})].
\end{aligned} \tag{13.55}$$

Note that for random $\mathbf{g}$, the expected optimal error appearing in this expression can be computed directly. This is because, for any $x \in \overline{\mathcal{X}}$,

$$\begin{aligned}
\mathbf{E}_{\mathbf{g}(x)}\left[\mathbf{Pr}_{c(x)}[opt(\mathbf{g}, x) \neq c(x) \mid \mathbf{g}]\right] &= \mathbf{Pr}_{c(x),\mathbf{g}(x)}[opt(\mathbf{g}, x) \neq c(x)] \\
&= \mathbf{Pr}_{c(x),\mathbf{g}(x)}\left[c(x) \neq \text{sign}\left(\sum_{t=1}^{T} g_t(x)\right)\right] \\
&= \text{Binom}\left(T,\ \frac{T}{2},\ \frac{1}{2}+\gamma'\right) \doteq \text{err}^*,
\end{aligned} \tag{13.56}$$

that is, the chance of fewer than half the random g_t's correctly classifying x. Call this probability err^*.

Using Hoeffding's inequality again, we can further show that equation (13.55) is very likely to be close to its expectation err^*.

Lemma 13.9 With probability at least $1 - e^{-2n}$ over the random choice of $\mathbf{g}$,

$$\mathbf{E}_{\bar{c}}[\overline{\text{err}}(opt(\mathbf{g}), \bar{c})] \geq \text{err}^* - \sqrt{\frac{n}{|\overline{\mathcal{X}}|}}.$$

Proof Let us define the random variables

$$O_x \doteq \mathbf{Pr}_{c(x)}[opt(\mathbf{g}, x) \neq c(x) \mid \mathbf{g}(x)]$$

for $x \in \overline{\mathcal{X}}$. Note that $\mathbf{E}_{\mathbf{g}}[O_x] = \text{err}^*$ by equation (13.56). Thus, applying Hoeffding's inequality (theorem 2.1) to their average

$$\mathbf{E}_{\bar{c}}[\overline{\text{err}}(opt(\mathbf{g}), \bar{c})] = \frac{1}{|\overline{\mathcal{X}}|} \cdot \sum_{x \in \overline{\mathcal{X}}} O_x$$

gives the result. ∎

Combining lemma 13.8, equation (13.55), and lemma 13.9, along with the union bound, we thus have shown that with probability at least $1 - 2e^{-2n}$,

$$\begin{aligned}
\overline{\text{err}}(\mathcal{B}(\mathbf{g}), \bar{c}) &\geq \mathbf{E}_{\bar{c}}[\overline{\text{err}}(\mathcal{B}(\mathbf{g}), \bar{c}) \mid \mathbf{g}] - \sqrt{\frac{n}{|\overline{\mathcal{X}}|}} \\
&\geq \mathbf{E}_{\bar{c}}[\overline{\text{err}}(opt(\mathbf{g}), \bar{c}) \mid \mathbf{g}] - \sqrt{\frac{n}{|\overline{\mathcal{X}}|}} \\
&\geq \text{err}^* - 2\sqrt{\frac{n}{|\overline{\mathcal{X}}|}}.
\end{aligned}$$

This implies that

$$\begin{aligned}
\text{err}(\mathcal{B}(\mathbf{g}), c) &\geq 2^{-n} \sum_{x \in \overline{\mathcal{X}}} \mathbf{1}\{\mathcal{B}(\mathbf{g}, x) \neq c(x)\} \\
&= \frac{|\overline{\mathcal{X}}|}{2^n} \cdot \overline{\text{err}}(\mathcal{B}(\mathbf{g}), \bar{c}) \\
&\geq \frac{|\overline{\mathcal{X}}|}{2^n} \cdot \left[\text{err}^* - 2\sqrt{\frac{n}{|\overline{\mathcal{X}}|}}\right] \\
&\geq \left(1 - \frac{m}{2^n}\right) \text{err}^* - 2\sqrt{\frac{n}{2^n}} \doteq \beta_n
\end{aligned} \tag{13.57}$$

with probability at least $1 - 2e^{-2n}$, where, in the last line, we used $2^n - m \leq |\overline{\mathcal{X}}| \leq 2^n$ since S is a sample of m (not necessarily distinct) instances. We denote the quantity in equation (13.57) by β_n.

13.2.7 Bringing Everything Together

We can now use lemma 13.6 to factor in the possibility of an abort when $\mathcal{B}$ is applied, more properly, to $\mathbf{g}'$ rather than $\mathbf{g}$. In particular, we have that

$$\begin{aligned}\mathbf{Pr}_{\bar{c},\mathcal{H}}\left[\mathrm{err}(\mathcal{B}(\mathbf{g}'), c) < \beta_n\right] &\leq \mathbf{Pr}_{\bar{c},\mathcal{H}}\left[\mathrm{err}(\mathcal{B}(\mathbf{g}), c) < \beta_n \vee \mathbf{g} \neq \mathbf{g}'\right] \\ &\leq \mathbf{Pr}_{\bar{c},\mathcal{H}}[\mathrm{err}(\mathcal{B}(\mathbf{g}), c) < \beta_n] + \mathbf{Pr}_{\bar{c},\mathcal{H}}\left[\exists t : g_t \neq g'_t\right] \\ &\leq 2e^{-2n} + Te^{-2n}\end{aligned} \tag{13.58}$$

where the last two lines use the union bound (repeatedly) together with equation (13.57) and lemma 13.6 (which implies that $g_t \neq g'_t$ with probability at most e^{-2n}).

We can now take expectation with respect to S, $c_{|S}$, $\mathbf{r}$, and $\mathcal{R}$, which, until this point, had been fixed. Let H denote the final hypothesis. To handle the possibility of $\mathbf{r}$ including two identical seeds, we use the fact that for any two events a and b,

$$\begin{aligned}\mathbf{Pr}[a] &= \mathbf{Pr}[a, b] + \mathbf{Pr}[a, \neg b] \\ &\leq \mathbf{Pr}[a|b] + \mathbf{Pr}[\neg b].\end{aligned}$$

Thus,

$$\begin{aligned}\mathbf{Pr}_{S,c,\mathcal{H},\mathbf{r},\mathcal{R}}[\mathrm{err}(H, c) < \beta_n] \leq\ & \mathbf{Pr}_{S,c,\mathcal{H},\mathbf{r},\mathcal{R}}[\mathrm{err}(H, c) < \beta_n \mid r_1, \ldots, r_T \text{ distinct}] \\ &+ \mathbf{Pr}_{S,c,\mathcal{H},\mathbf{r},\mathcal{R}}[r_1, \ldots, r_T \text{ not all distinct}]\,.\end{aligned} \tag{13.59}$$

The first term on the right is at most $(T + 2)e^{-2n}$ since it is the (conditional) expectation of the probability appearing on the left-hand side of equation (13.58). As for the second term on the right of equation (13.59), the chance that two particular random seeds are identical is exactly 2^{-n}. Therefore, by the union bound, the chance that T seeds are not all distinct is at most $\binom{T}{2} \cdot 2^{-n}$. Therefore,

$$\mathbf{Pr}_{S,c,\mathcal{H},\mathbf{r},\mathcal{R}}[\mathrm{err}(H, c) < \beta_n] \leq (T + 2)e^{-2n} + \binom{T}{2} \cdot 2^{-n}. \tag{13.60}$$

Clearly, the right-hand side of equation (13.60) can be made smaller than $\nu > 0$ for n sufficiently large. Furthermore, keeping in mind that m, γ' and err* all depend implicitly on n, we see that for n large, $\beta_n \to \beta^*$ (where β^* is as in equation (13.44)) since $\gamma' \to \gamma$ and since we assume that m, the sample size, is bounded by a polynomial in n. Therefore, $\beta_n \geq \beta^* - \nu$ for n sufficiently large. Thus, we have proved eqution (13.45), completing the proof of theorem 13.5. ∎

13.3 Relation to AdaBoost

We next consider the relationship between BBM and the more familiar AdaBoost algorithm. Not only are their error bounds related, but we will further see that a nonadaptive version of AdaBoost can be derived as a special case of BBM. On the other hand, there also exist some fundamental differences between the algorithms.

13.3.1 Comparison of Error Bounds

We have already seen that BBM's error, as a function of T and γ, takes the form of the tail of a binomial distribution as given in equation (13.37). This bound applies to the training error, as proved in corollary 13.4, as well as to the generalization error, when the number of training examples is large, as seen in section 13.2.1. Furthermore, in section 13.2.2, we showed that this bound is the best possible for any boosting algorithm. So BBM is essentially optimal when T and γ are known and fixed.

As already noted, when AdaBoost or NonAdaBoost is used, theorem 3.1 implies that the training error will be at most $e^{-2\gamma^2 T}$, a bound that also holds for the generalization error in the limit of large training sets. As earlier noted, this is exactly the bound Hoeffding's inequality (theorem 2.1) gives for the tail of the binomial in equation (13.37).

In fact, the better bound implied by theorem 3.1 for AdaBoost's training error of

$$\left(\sqrt{1-4\gamma^2}\right)^T \tag{13.61}$$

for this case is also a Chernoff bound. In particular, it can be proved (see exercise 5.4) that the chance of at most qn heads in n flips of a coin with bias p, where $q < p$, is bounded as

$$\text{Binom}\,(n, qn, p) \le \exp\left(-n \cdot \text{RE}_b\,(q \parallel p)\right), \tag{13.62}$$

where $\text{RE}_b\,(\cdot \parallel \cdot)$ is the binary relative entropy given in equation (5.36). Thus, in the case of the bound for BBM, we get

$$\text{Binom}\left(T, \frac{T}{2}, \frac{1}{2}+\gamma\right) \le \exp\left(-T \cdot \text{RE}_b\left(\frac{1}{2} \parallel \frac{1}{2}+\gamma\right)\right), \tag{13.63}$$

which is exactly equal to equation (13.61). Furthermore, it can be shown that

$$\text{Binom}\left(T, \frac{T}{2}, \frac{1}{2}+\gamma\right) \ge \exp\left(-T \cdot \left[\text{RE}_b\left(\frac{1}{2} \parallel \frac{1}{2}+\gamma\right) + O\left(\frac{\ln T}{T}\right)\right]\right). \tag{13.64}$$

Since $(\ln T)/T$ becomes negligible as T becomes large, this means that the approximation in equation (13.63) is tight for a large number of rounds. Thus, AdaBoost's error bounds are quite close to the optimal bounds enjoyed by BBM.

Figure 13.3 shows a plot of the bounds for BBM and AdaBoost as a function of T. As T grows, we know that the bounds must exhibit the same exponential behavior, as can be

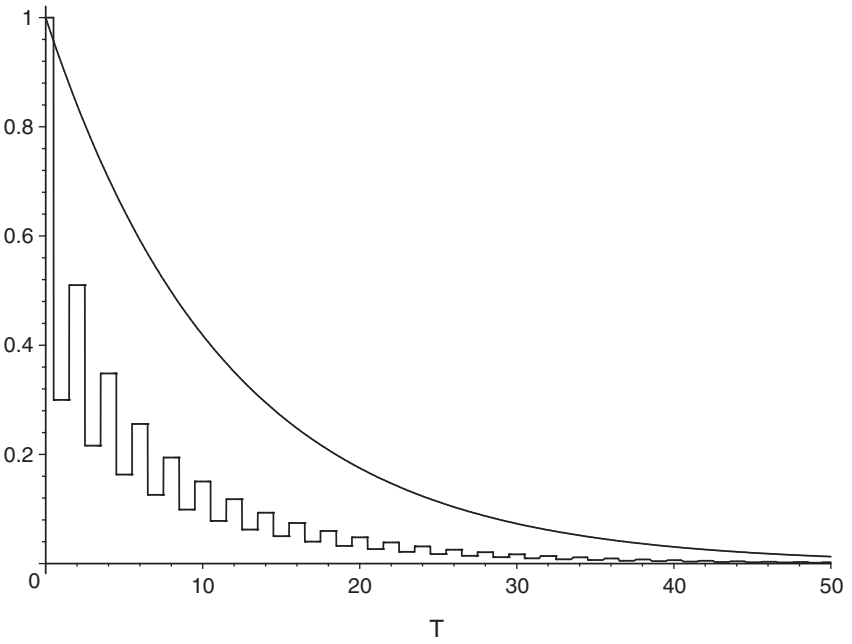

Figure 13.3
A comparison of the error bounds for nonadaptive AdaBoost (top curve, from equation (13.61)) and BBM (bottom curve, from equation (13.37)) as a function of T with $\gamma = 0.2$. The BBM bound is defined only at integral values of T, and appears jagged because the bound is generally worse when T is even (see exercise 13.8).

seen in the figure. On the other hand, for small T, there is clearly a big numerical difference between them.

13.3.2 Deriving AdaBoost from BBM

In addition to the close relationship between their error bounds, AdaBoost and BBM turn out also to be strongly connected as algorithms, with NonAdaBoost, the nonadaptive version of AdaBoost discussed in section 13.1.1, a kind of limiting special case of BBM.

In particular, when the number of rounds T is chosen to be very large, the first T_0 rounds of BBM, for any fixed T_0, will behave nearly exactly like NonAdaBoost. In this sense, nonadaptive AdaBoost can be regarded as the limit of BBM that is obtained by letting $T \to \infty$ with γ fixed. (In chapter 14, we will look at a different limit that will yield a different algorithm.)

To see this, we show that the weights used by BBM on round t converge, up to an irrelevant constant factor, to those for NonAdaBoost when t is smaller than any fixed T_0, and T is increased without bound. For this, we also need to assume that the same weak hypotheses were computed on the preceding rounds as for NonAdaBoost. Since the weights

on all the preceding rounds of BBM are converging as $T \to \infty$ to the same weights as for NonAdaBoost, this is likely to be the case. However, this cannot be proven to always be true in general since some weak learning algorithms are numerically unstable, meaning that even tiny changes in the distribution selected on a given round can cause significant changes in the computed weak hypothesis.

In this section, we write $w_t^T(s)$, rather than $w_t(s)$, to make explicit the dependence of the weighting function on the number of rounds T. Referring to algorithm 13.1, each training example i begins round t with unnormalized margin $s_{t-1,i}$ and is assigned weight $w_t^T(s_{t-1,i})$. Thus, to prove the claim above, it suffices to show that as $T \to \infty$, the weights $w_t^T(s)$, suitably scaled, converge to the weights used by NonAdaBoost (with α as in equation (13.5)), namely

$$e^{-\alpha s} = \left(\frac{1-2\gamma}{1+2\gamma}\right)^{s/2} \tag{13.65}$$

for $|s| \le t \le T_0$. (Multiplicative constants can be ignored because the weights are normalized in forming the distribution D_t.)

For simplicity of presentation, we consider only the case that T is odd and s is even. The other cases can be handled similarly. Note that because all the training examples begin at position 0, and all are incremented or decremented on every round, their parities will always remain in agreement and, furthermore, will always be opposite to the parity of t. Let $\overline{T} \doteq T - t$, which, by the preceding discussion and assumptions, must be even.

From equation (13.34), the weight $w_t^T(s)$ used by BBM can be rewritten as

$$\frac{1}{2}\left(\frac{1-2\gamma}{1+2\gamma}\right)^{s/2} \cdot \binom{\overline{T}}{\frac{\overline{T}}{2} - \frac{s}{2}} \left(\frac{1}{2}+\gamma\right)^{\overline{T}/2} \left(\frac{1}{2}-\gamma\right)^{\overline{T}/2}. \tag{13.66}$$

To deal with the binomial coefficient, we note that for any integers $n \ge k \ge 1$,

$$\begin{aligned}\frac{\binom{2n}{n}}{\binom{2n}{n+k}} = \frac{\binom{2n}{n}}{\binom{2n}{n-k}} &= \frac{(n+k)(n+k-1)\cdots(n+1)}{n(n-1)\cdots(n-k+1)} \\ &= \prod_{j=0}^{k-1}\left(1+\frac{k}{n-j}\right).\end{aligned}$$

In the limit $n \to \infty$ with k fixed, each of the k terms in the product converges to 1; therefore, the entire product converges to 1 as well.

Thus, taking $n = \overline{T}/2$ and $k = |s|/2$ in equation (13.66), and defining the constant

$$C_{\overline{T}} = \frac{1}{2}\binom{\overline{T}}{\overline{T}/2}\left(\frac{1}{2}+\gamma\right)^{\overline{T}/2}\left(\frac{1}{2}-\gamma\right)^{\overline{T}/2},$$

we see that for fixed s and t, $w_t^T(s)/C_{T-t}$ converges to equation (13.65) as T (odd) grows to infinity, proving the claim (for this case).

It follows by an inductive argument that the behavior of BBM with large T will be essentially indistinguishable from NonAdaBoost on the first T_0 rounds, for any fixed T_0 (modulo the technical numerical caveats mentioned above concerning the weak learning algorithm).

13.3.3 Comparison of Weights

Although their behavior is very similar on the first several rounds when T is large, there exist important differences between AdaBoost and BBM at later stages of the algorithm, particularly with respect to the weighting functions that differentiate the two algorithms.

We have seen that NonAdaBoost weights examples with unnormalized margin s proportionally to equation (13.65). Such a weighting function is shown at the top of figure 13.4. Note that this weighting function is fixed, and does not change with additional rounds of boosting.

In stark contrast, at the bottom, figure 13.4 shows the weighting function $w_t(s)$ used by BBM for various values of t. Note first that this function changes significantly over time: As t, the round number, increases, its peak value shifts steadily rightward. Moreover, the weights become much more concentrated as the end of boosting approaches. Indeed, on the very last round, all of the weight will be concentrated solely on the examples exactly on the boundary between a correct or incorrect prediction since, at this point, the fate of all other examples has already been decided.

Perhaps the most striking difference from AdaBoost is the non-monotonicity of BBM's weighting function. This means that whereas AdaBoost piles ever more weight on examples which are continually misclassified by the weak hypotheses, BBM will do the same, but only up to a point. Eventually, examples that are misclassified too many times will see their weight actually *decrease* with further misclassifications. In other words, BBM is actually giving up on these very hard examples.

This could potentially be an important advantage. AdaBoost is known to sometimes "spin its wheels" on outliers—examples that may be hard due to labeling errors or inherent ambiguity (see sections 10.3 and 12.3.3). The shape of BBM's weighting function suggests that this algorithm may instead abandon such examples for the greater good of the overall learning process.

On the other hand, BBM is not a practical algorithm because, unlike AdaBoost, it is not adaptive. To use it, we need to choose T and to anticipate a lower bound γ on all of the edges of the forthcoming weak hypotheses *before* boosting begins. In practice, guessing the right values for these parameters can be very difficult. To address this issue, in chapter 14 we describe a technique for making BBM adaptive.

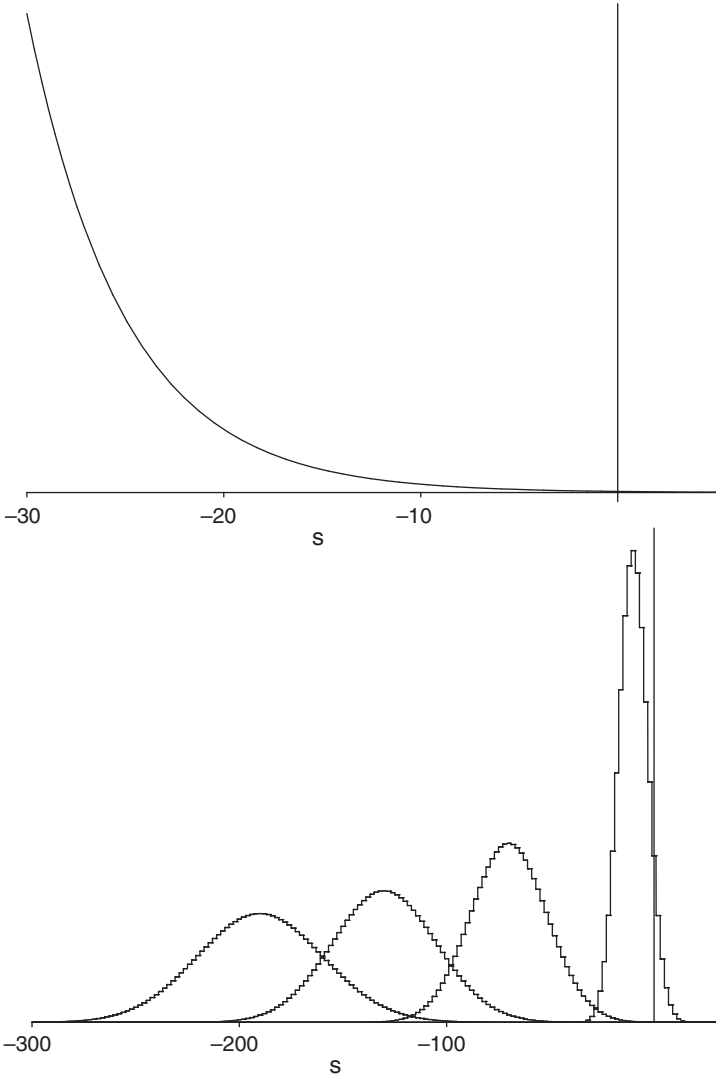

Figure 13.4
A comparison of the weighting function used by nonadaptive AdaBoost (top) and BBM (bottom) when $T = 1000$ and $\gamma = 0.1$. The BBM curves are plotted, from left to right, for rounds $t = 50$, 350, 650, and 950. (These curves appear jagged because they are defined only at integer values.)

Summary

In this chapter, we took a close look at the nature of optimality in boosting, leading to the boost-by-majority algorithm, which we derived in an abstract framework based on a chip game. BBM is nearly optimal in terms of minimizing both the training error and the generalization error, as was seen, in both cases, by matching its performance to lower bounds based on an oblivious weak learner.

We also studied the close relationship between AdaBoost and BBM, and noted that the previously proved bounds for AdaBoost indicate that it is not far behind optimal BBM in performance. On the other hand, the behavior of the two algorithms may be very different on outliers. In any case, AdaBoost is much more practical than BBM because of its adaptiveness. To overcome this limitation, in chapter 14 we consider a technique for making BBM adaptive.

Bibliographic Notes

The boost-by-majority algorithm and analysis of section 13.1, including the voting-game formulation, are due to Freund [88]. The derivation given in section 13.1.4 is based on ideas from Schapire's [201] work on "drifting games," a framework that generalizes BBM and its analysis. The proof of lemma 13.1 incorporates some key (unpublished) insights that are due to Indraneel Mukherjee.

The lower bound given in section 13.2.2 is a substantial elaboration of one originally given by Freund [88].

Theorem 13.7 and equation (13.62) are due to Hoeffding [123]. Equation (13.64) is based on known lower bounds on the tail of a binomial distribution which can be found, for instance, in section 12.1 of Cover and Thomas [57], and the references therein.

Some of the exercises in this chapter are based on material from [46, 201].

Exercises

13.1 All of the boosting algorithms we have considered in this book compute the distribution D_t in a strongly sequential fashion, with each D_t depending on the preceding weak hypotheses $h_1, \ldots, h_{t-1}$. Is this kind of adaptiveness truly necessary? Can there exist "universal" distributions which are effective for boosting against any weak learner without adjustment based on the weak hypotheses actually received?

To formalize this question, consider a variant of the voting game given in section 13.1.1 that proceeds as follows:

1. the booster chooses T distributions $D_1, \ldots, D_T$ over a given and fixed training set of m examples;

2. the weak learner chooses T hypotheses $h_1, \ldots, h_T$ such that equation (13.1) holds for $t = 1, \ldots, T$.

The final hypothesis is then formed as a simple majority vote as in equation (13.2). Let us say that the booster wins the game if H has training error zero; otherwise, the weak learner wins.

Show *either* that there exists a strategy for the booster that wins the game always against any weak learner, for some appropriate choice of T; *or* prove that no such winning strategy can exist.

13.2 Give an example showing that the inequality given in equation (13.31) can be strict, that is, showing that it is possible that $\Lambda_0(\mathbf{0}) < \Phi_0(0)$.

13.3 Verify that equation (13.30) satisfies both equations (13.25) and (13.29). Also verify equation (13.34).

13.4 Suppose in the construction given in section 13.1 that abstaining weak hypotheses are used with range $\{-1, 0, +1\}$. Now, in place of equation (13.1), we require $\mathbf{E}_{i\sim D_t}[y_i h_t(x_i)] \geq 2\gamma$. And in terms of the chip game, this means that chip i's position is increased on round t by $z_{t,i} \in \{-1, 0, +1\}$ with the requirement that equation (13.8) hold. The derivation leading to equations (13.24), (13.25), and (13.26) (*except* lemma 13.2) can be straightforwardly modified for this case simply by replacing $\{-1, +1\}$ with $\{-1, 0, +1\}$. Thus, the potential $\Phi_T(s)$ is as in equation (13.25), but for $t = 1, \ldots, T$, is now redefined to be

$$\Phi_{t-1}(s) \doteq \inf_{w\geq 0} \max_{z\in\{-1,0,+1\}} [\Phi_t(s+z) + w \cdot (z - 2\gamma)]. \tag{13.67}$$

Likewise, BBM can be modified for this case simply by plugging in a different definition of $w_t(s)$. The rest of this exercise refers to these modified definitions.

a. Show that $\Phi_t(s)$ is nonincreasing for all t, that is, $\Phi_t(s) \geq \Phi_t(s')$ if $s < s'$.

b. For $t = 1, \ldots, T$, show that

$$\Phi_{t-1}(s) = \max\left\{(\tfrac{1}{2}+\gamma)\Phi_t(s+1) + (\tfrac{1}{2}-\gamma)\Phi_t(s-1),\right.$$
$$\left.(1-2\gamma)\Phi_t(s) + 2\gamma\ \Phi_t(s+1)\right\}.$$

Also, find $w_t(s)$, the value of $w \geq 0$ which realizes the infimum in equation (13.67), in terms of $\Phi_t(s-1)$, $\Phi_t(s)$, and $\Phi_t(s+1)$. (Your answer should give $w_t(s)$ explicitly, not using an "arg min".)

13.5 Let $\theta > 0$ be a desired minimum normalized margin that is given and known in advance. As in BBM, assume the γ-weak learning condition holds, where $\gamma > 0$ is also known.

a. Show how to modify BBM and its analysis so that the fraction of training examples with normalized margin at most θ is guaranteed not to exceed

$$\text{Binom}\left(T,\ \left(\frac{1+\theta}{2}\right)T,\ \frac{1}{2}+\gamma\right).$$

b. For what (fixed) values of θ and γ does this bound approach zero as $T \to \infty$?

13.6 Suppose in equation (13.25) that we instead defined $\Phi_T(s) \doteq e^{-\alpha s}$ for some $\alpha > 0$, thus also redefining $\Phi_t(s)$, for $t < T$, via equation (13.26), as well as $w_t(s)$ via equation (13.32).

a. Explain why equation (13.24) holds for this redefined version of Φ_t.

b. Compute the new versions of $\Phi_t(s)$ and $w_t(s)$ in closed form.

c. Show how to choose α to optimize $\Phi_0(0)$, which bounds the training error.

d. Verify that the resulting bound on the training error is the same as can be obtained for NonAdaBoost from theorem 3.1. Also verify that if the new version of $w_t(s)$ is substituted in BBM (with the optimized choice of α), then the resulting algorithm is exactly equivalent to NonAdaBoost.

13.7 Consider the following online prediction problem, similar to the one studied in section 6.3. There are m experts. On each round t, expert i predicts $x_{t,i} \in \{-1, +1\}$, and the learner makes its own prediction $\hat{y}_t$ as a weighted majority vote of the expert predictions. The true label y_t is then revealed. Thus, formally, on each round $t = 1, \ldots, T$:

- the learner chooses a weight vector $\mathbf{v}_t \in \mathbb{R}^m_+$;
- the expert predictions $\mathbf{x}_t \in \{-1, +1\}^m$ are revealed;
- the learner predicts $\hat{y}_t = \text{sign}(\mathbf{v}_t \cdot \mathbf{x}_t)$;
- nature reveals $y_t \in \{-1, +1\}$.

The learner makes a mistake if $\hat{y}_t \neq y_t$; likewise, expert i makes a mistake if $x_{t,i} \neq y_t$.

In this problem, we assume that one of the experts makes at most k mistakes, where k is known ahead of time. We also assume that the learner is *conservative*, meaning that rounds on which $\hat{y}_t = y_t$ are entirely ignored in the sense that the state of the algorithm does not change. For such algorithms, we can assume without loss of generality that a mistake occurs on *every* round (since other rounds are ignored). Thus, t is actually counting the number of mistakes of the learner, rather than the number of rounds.

This can be formulated as a chip game in which the chips are now identified with experts. In particular, we make the following redefinitions of the variables and quantities appearing in section 13.1.2:

- $z_{t,i} \doteq -y_t x_{t,i}$;
- $D_t(i) = v_{t,i}/\mathcal{Z}_t$, where $\mathcal{Z}_t$ is a normalization factor;
- $\gamma = 0$;
- $L(\mathbf{s}_T) \doteq \frac{1}{m}\sum_{i=1}^m \mathbf{1}\{s_{T,i} \leq 2k - T\}$.

All of the following are with respect to these new definitions, which of course also impact $\mathbf{s}_t$, Φ_t, etc.

a. Show that equation (13.8) holds for all t.

b. If the game is played for a full T rounds, show that $L(\mathbf{s}_T) \geq 1/m$.

c. Calculate $\Phi_t(s)$ and $w_t(s)$ in closed form.

d. Suppose the learner chooses the weight vector $\mathbf{v}_t$ by setting $v_{t,i} = w_t(s_{t-1,i})$. Furthermore, suppose we let $T = 1 + T_0$ where T_0 is the largest positive integer for which

$$2^{T_0} \leq m \cdot \sum_{j=0}^{k} \binom{T_0}{j}.$$

Prove that the number of mistakes made by such a learner cannot exceed T_0. (It can be shown that $T_0 \leq 2k + 2\sqrt{k \ln m} + \lg m$.)

13.8 As seen in figure 13.3, the bound on BBM's training error is significantly worse for an even number of rounds than for an odd number of rounds. This is largely the result of our convention of counting a tied vote among the weak hypotheses as a full mistake. This exercise considers an alternative in which such a tie counts as only half a mistake, as is natural if ties result in random-guess predictions.

More specifically, suppose $T > 0$ is even, and that the loss or training error of the booster in equation (13.3), as well as the corresponding loss $L(\mathbf{s}_T)$ of the chips in equation (13.9), are replaced by

$$\frac{1}{m}\sum_{i=1}^{m} \ell\left(y_i \sum_{t=1}^{T} h_t(x_i)\right) = \frac{1}{m}\sum_{i=1}^{m} \ell(s_{T,i})$$

where

$$\ell(s) \doteq \begin{cases} 1 & \text{if } s < 0 \\ \frac{1}{2} & \text{if } s = 0 \\ 0 & \text{if } s > 0. \end{cases}$$

Note that our treatment for an odd number of rounds is unchanged since, in this case, $s_{T,i}$ will never be 0.

In this problem, we write $\Phi_t^T(s)$ and $w_t^T(s)$ with superscripts to make explicit the dependence of the potential and weighting functions on the total number of rounds T. These functions are still defined by equations (13.26) and (13.32), although equation (13.25) naturally requires appropriate modification. By the same analysis, if this modified weighting function is used in BBM, then the training error will be at most the (modified) initial potential $\Phi_0^T(0)$.

a. Show, under this revised definition, that

$$\Phi_t^T(s) = \tfrac{1}{2}\left(\Phi_{t+1}^{T+1}(s-1) + \Phi_{t+1}^{T+1}(s+1)\right).$$

(Keep in mind our assumption that T is even.)

b. Find an analogous expression for $w_t^T(s)$.

c. Prove that

$$\Phi_0^{T+1}(0) \le \Phi_0^T(0) = \Phi_0^{T-1}(0).$$

This shows that the bound on the (modified) training error is nonincreasing as a function of T, and also that there is no advantage to using T rounds, rather than $T-1$, if T is even.

d. Sketch how to modify the proof of theorem 13.5 so that it applies when T is even where, in this case, equation (13.40) is replaced by $\Phi_0^T(0) - \nu$, that is,

$$\frac{1}{2}\left[\text{Binom}\left(T,\ \frac{T-1}{2},\ \frac{1}{2}+\gamma\right) + \text{Binom}\left(T,\ \frac{T+1}{2},\ \frac{1}{2}+\gamma\right)\right] - \nu.$$

13.9 Section 13.1.7 briefly discusses a relaxed game in which weak hypotheses—or, equivalently, chip movements $\mathbf{z}_t$—are selected in a randomized fashion. More formally, the game is played as follows. On each round t, the booster chooses a distribution D_t over chips, and the weak learner responds with a distribution Q_t over $\{-1,+1\}^m$, the set of possible chip movements. The vector $\mathbf{z}_t$ is then selected at random according to Q_t, and the chips are then moved as usual as in equation (13.6). Rather than equations (13.7) and (13.8), we instead require that these hold in expectation, that is,

$$\mathbf{E}_{\mathbf{z}_t \sim Q_t, i \sim D_t}\left[z_{t,i}\right] \ge 2\gamma.$$

The goal is to minimize the expected loss

$$\mathbf{E}_{\mathbf{z}_1 \sim Q_1, \ldots, \mathbf{z}_T \sim Q_T}\left[L\left(\sum_{t=1}^{T} \mathbf{z}_t\right)\right].$$

(Technically, we assume that the booster and weak learner map the preceding history of events to a decision (D_t or Q_t) in a deterministic fashion so that the only source of randomness is in the choice of $\mathbf{z}_t$'s.)

As before, let $\Lambda_t(\mathbf{s}_t)$ be the expected loss that will be incurred if the chips are in position $\mathbf{s}_t$ on round t, and if the game is henceforth played optimally by both players.

a. Analogous to equations (13.11) and (13.12), for this relaxed version of the game, give an expression for Λ_T, and also a recursive expression for Λ_{t-1} in terms of Λ_t. Justify your answers.

b. Prove that

$$\Lambda_t(\mathbf{s}) = \frac{1}{m}\sum_{i=1}^{m} \Phi_t(s_i)$$

for all **s** and t (where the definition of Φ_t is unchanged). In particular, conclude that the value of the game is exactly as given in equation (13.37).

c. Suppose now that the weak learner (deterministically) chooses on each round t a distribution Q_t over weak hypotheses, and that h_t is then selected at random from Q_t. Rather than equation (13.1), we now assume

$$\mathbf{E}_{h_t \sim Q_t}\left[\mathbf{Pr}_{i \sim D_t}[h_t(x_i) \neq y_i]\right] \leq \tfrac{1}{2} - \gamma.$$

Under this modified assumption, show that the expected training error of the final hypothesis H generated by BBM is at most that given in equation (13.37).

13.10 Consider boosting in a multiclass setting when the number of classes is $K > 2$. For the purposes of this exercise, we modify the definition of a weak learning algorithm, as reviewed in section 13.2.2, replacing the requirement that h's error be at most $\frac{1}{2} - \gamma$ with the weaker condition that h's error be at most $1 - 1/K - \gamma$.

Prove formally that for any boosting algorithm B, and for any $\nu > 0$, there exist a target class $\mathcal{C}$, a distribution $\mathcal{D}$, a target function $c \in \mathcal{C}$, and a weak learning algorithm for $\mathcal{C}$ such that, regardless of the number of times that B calls A, the generalization error of its combined classifier will be at least

$$1 - \frac{1}{K-1} - \nu$$

with probability at least $1 - \nu$ (where the probability is taken with respect to the same quantities as in theorem 13.5). In other words, show that boosting, as formally defined, is not possible when $\epsilon < 1 - 1/(K-1)$.