

Information-theoretic Bounds on the Training and Testing Error of Boosting

Sébastien M. Lahaie
Senior Honors Thesis
Advised by Javed A. Aslam

Technical Report TR 2002-428
Department of Computer Science
Dartmouth College

Abstract

Boosting is a means of using weak learners as subroutines to produce a strong learner with markedly better accuracy. Recent results showing the connection between logistic regression and boosting provide the foundation for an information-theoretic analysis of boosting. We describe the analogy between boosting and gambling, which allows us to derive a new upper bound on training error. This upper bound explicitly describes the effect of noisy data on training error. We also use information-theoretic techniques to derive an alternative upper-bound on testing error which is independent of the size of the weak-learner space.

1 Introduction

In this paper, we will use information-theory to analyze a computational learning technique called *boosting* from a new perspective, and ultimately prove new information-theoretic upper bounds on boosting's error rate. A *learning algorithm* is faced with the following problem. Given a sample of instances which positively or negatively exemplify (i.e are or are not examples of) a given concept, the algorithm must output a *hypothesis*, or classification rule, which can correctly classify instances to which the algorithm was never exposed. The algorithm will therefore have “learned” the concept given some examples.

We model the process of learning using the ‘Probably Approximately Correct’ (PAC) learning model [4]. A PAC learning algorithm satisfies the following criterion: with high probability, or *confidence*, the algorithm will output a hypothesis that has high *accuracy* (low error). The confidence and accuracy are specified as inputs to the algorithm, so it must be able to adapt to requests for arbitrarily high confidence and accuracy. A learning algorithm that can provide any desired level of confidence and accuracy is called a *strong*

learner. A *weak learner*, on the other hand, only promises to meet the PAC criterion for a fixed accuracy.

A boosting algorithm uses a weak learner as a subroutine to generate a strong classification rule. The algorithm therefore ‘boosts’ the weak learner’s fixed accuracy to an arbitrarily high accuracy. The most popular and well-studied boosting algorithm is AdaBoost [7]. AdaBoost proceeds in rounds; at each round, the weak learner is trained using a local distribution on the training examples, and its hypothesis is retained. AdaBoost then determines which examples the hypothesis classifies correctly and incorrectly. The distribution is then modified so that incorrectly classified examples are weighted more heavily, and vice-versa. In the next round, the weak learner is retrained using this new distribution on the examples. Intuitively, the next hypothesis will then specialize in classifying examples the current hypothesis performed poorly on, and thus make up for the latter’s deficiencies. AdaBoost then takes a majority vote of its underlying weak learner’s predictions to determine its own final prediction.

There are two related measures of error for a hypothesis. The *training error* is the hypothesis’ error over the training examples (i.e. the proportion of training examples it classifies incorrectly). The *testing error* is its error over the whole instance space, including instances the hypothesis has never encountered. Intuitively, low training error should imply low testing error. However, there is a risk that the learning algorithm might concentrate on features of the training set which are unrepresentative of the instance space. This phenomenon is called *overfitting*; the testing error of a hypothesis suffers when the hypothesis overfits, even though the training error may be very low. In principle and practice, overly complex hypotheses tend to overfit.

A surprising property of AdaBoost is its resilience to overfitting. Even after thousands of rounds of training weak learners, and after having achieved zero training error, AdaBoost’s testing error still continues to decrease [6]. This shows that training error is not the best measure of a boosting hypothesis’ predictive capabilities. Hence the concept of a *margin* was introduced. A margin is the difference in the weights assigned to each label (positive or negative) by the underlying weak learners. A large difference implies a strong vote in favor of a particular label. Margins are thus a measure of AdaBoost’s ‘certainty’ in its prediction. Margins improve as the number of boosting rounds increase, and this is linked to the continued improvement in the testing error.

Recent work has shown that AdaBoost fits an additive logistic regression model [3]. Its hypothesis can thus be rephrased as an estimate of the probability of either label being correct given an example. Using this distribution form of AdaBoost’s hypothesis allows us to perform an information-theoretic analysis, because information-theory involves properties of distributions. Margins, being a measure of certainty, are very closely related to entropy, an information-theoretic measure of uncertainty. Our analysis shows that AdaBoost’s training error is upper bounded by the entropy of the probability of noise in the data (some of the training data may be labeled incorrectly), plus the relative entropy between the predicted and actual distributions, which is a measure of the quality of the prediction. Our analysis further provides an alternative upper bound on testing error which is independent of the size of the weak-learner space.

In section 2, we provide background on information-theory, including: the notions of

entropy and relative entropy; the method of types, a technique used in our testing error proof; an information-theoretic analysis of gambling on horse races. We also give more detailed background on PAC learning, boosting, and AdaBoost.

In section 3, we outline the connection between boosting and logistic regression, and between boosting and gambling. This will give us the proper perspective with which to approach the alternative training and testing error bounds.

In section 4, we prove alternative upper bounds on the training and testing error of boosting. We also compare our new testing error bound to the original bound in [6].

2 Background

2.1 Entropy

The fundamental quantity in information theory is the *entropy* of a probability distribution. The entropy of a distribution P , denoted $H(P)$, is defined as

$$H(P) = - \sum_x p(x) \log p(x)$$

Alternatively, if X is a random variable with corresponding distribution P , then the entropy $H(X)$ of X is simply $H(P)$. Entropy is typically viewed as a measure of the “randomness” of a distribution. That is, if values of X are generated independently according to P , then the sequence of values will appear more random as the entropy of P increases. For example, suppose X can take on the values $\{0, 1\}$ with probabilities $P(0) = 0$ and $P(1) = 1$. In this case the entropy of P is 0, and the sequence will correspondingly display no randomness at all since it will consist solely of 1’s. On the other hand, the distribution $(P(0) = \frac{1}{2}, P(1) = \frac{1}{2})$ will generate a completely random sequence of 0’s and 1’s. Its entropy is 1, the maximum possible entropy for a binary distribution.

Entropy can also be viewed as a measure of “certainty” (“confidence” would also be an appropriate term, but we will use it in a different setting). Suppose we are observing a sequence of 0’s and 1’s, and we are asked to predict the next symbol. We can express our certainty in our prediction of the next label by specifying a distribution on the labels. For example, by specifying the distribution $(P(0) = \frac{1}{4}, P(1) = \frac{3}{4})$, we are saying that we are 75% certain that the next symbol will be a 1. The higher our certainty, the lower the entropy of our distribution.

Note that $\log_b p = \log_b a \log_a p$. Thus we are free to use any logarithm base in our formula for entropy—the entropy using base b will differ from the entropy using base a by a constant factor of $\log_b a$. We will use base 2, in which case the entropy units are *bits*.

An important property of the entropy function $H(P)$ is that it is concave in P . A function f is said to be *concave* over an interval (a, b) if for every $x_1, x_2 \in (a, b)$ and $0 \leq \lambda \leq 1$,

$$f(\lambda x_1 + (1 - \lambda)x_2) \geq \lambda f(x_1) + (1 - \lambda)f(x_2)$$

A function f is *convex* if $-f$ is concave. The concavity of entropy is important because it allows us to apply the following inequality to entropy functions:

Theorem 1 (*Jensen's inequality*): If f is a concave function and X is a random variable, then

$$E[f(X)] \leq f(E[X])$$

Cover and Thomas [1] provide an elegant proof by induction of Jensen's inequality.

Another important quantity is the *relative entropy* between two distributions. It is also called the *Kullback-Leibler distance*, because it can be viewed as a measure of the distance between two distributions.

Definition 1 The relative entropy between two probability mass functions $p(x)$ and $q(x)$ is defined as

$$\begin{aligned} D(p \parallel q) &= \sum_x p(x) \log \frac{p(x)}{q(x)} \\ &= E_p \left[\log \frac{p(X)}{q(X)} \right] \end{aligned}$$

One of the main properties of relative entropy is that it is non-negative.

Theorem 2 (*Information inequality*): Let $p(x)$, $q(x)$ be two probability mass functions over the same space. Then

$$D(p \parallel q) \geq 0$$

with equality if and only if for all x ,

$$p(x) = q(x)$$

The proof involves Jensen's inequality and can be found in Cover and Thomas [1].

2.2 The Method of Types

Consider an alphabet of symbols $\mathcal{A} = \{a_1, a_2, \dots, a_r\}$ and a distribution Q over this alphabet. Let X_1, X_2, \dots, X_n be a sequence of n symbols from \mathcal{A} drawn according to Q . We will denote sequences of length n by x^n . The method of types is a technique used to determine the probabilities of sets of such sequences. The following definitions are due to Cover and Thomas [1].

Definition 2 The type P_{x^n} (or empirical probability distribution) of a sequence x^n is the relative proportion of occurrences of each symbol. For example, if a appears i times in x^n , then $P_{x^n}(a) = i/n$.

Definition 3 Let \mathcal{P}_n denote the set of types with denominator n (i.e. derived from sequences of length n). If $P \in \mathcal{P}_n$, then the set of sequences of length n and type P is called the type class of P , denoted $T(P)$; that is, $T(P) = \{x^n \in \mathcal{A}^n : P_{x^n} = P\}$

A number of interesting results follow from these definitions. For example, it can be shown that the probability of a sequence x^n depends only on its type. For our purposes, the most important point is that the size of the set \mathcal{P}_n is at most polynomial in n

Theorem 3 For an alphabet of size r , $|\mathcal{P}_n| \leq (n+1)^r$. To be exact, the size of \mathcal{P}_n is

$$\binom{n+r-1}{r-1}$$

Proof: For each of the r probabilities in a type from \mathcal{P}_n , there are $(n+1)$ choices for the numerator. So there are at most $(n+1)^r$ choices for the type, which gives us the upper bound. Of course the numerators are not independent—they must sum to n . To find the exact number of types in \mathcal{P}_n , consider the following scenario. We have $(n+r-1)$ red balls lined up. By choosing $(r-1)$ of these to color black, only n red balls remain. Meanwhile, the $r-1$ black balls now serve as delimiters, and segment the remaining red balls into r groups. These groups of red balls are directly analogous to the numerators of a type from \mathcal{P}_n . So the number of types in \mathcal{P}_n equals the number of ways to choose $(r-1)$ balls from a set of $(n+r-1)$. \square

2.3 Gambling

Cover and Thomas [1] present an interesting application of information-theory to gambling. Suppose two horses run in a race. If horse i wins, the payoff is o_i for 1, where $i = 1$ or 2 . This means that a bet of one dollar on horse 1 yields a return of o_i dollars if horse i wins, and 0 dollars if horse i loses. Suppose a gambler has a certain amount of money that he will distribute over the two horses. Let b_i be the fraction of wealth bet on horse i . We thus have $b_1 \geq 0$, $b_2 \geq 0$, and $b_1 + b_2 = 1$. If horse i wins, the gambler will have multiplied his wealth by a factor of $b_i o_i$. Let X be the winning horse (1 or 2).

Definition 4 The wealth relative $S(X) = b_X o_X$ is the factor by which the gambler's wealth grows if horse X wins the race.

Now suppose the gambler bets on a series of races, each between two horses. Let S_n be the factor by which the gambler's wealth has grown after n races.

$$S_n = \prod_{j=1}^n S(X_j)$$

Each horse has a certain probability of winning p_i before the race. Let $\mathbf{p} = (p_1, p_2)$. With this distribution, we can determine the expected factor by which the gambler's wealth will grow. However, a more interesting quantity is the expected log of the factor by which the gambler's wealth will grow.

Definition 5 The doubling rate of a horse race is

$$W(\mathbf{b}, \mathbf{p}) = E[\log S(X)] = p_1 \log b_1 o_1 + p_2 \log b_2 o_2.$$

By maximizing the doubling rate, the gambler maximizes his expected winnings. The gambling rate is maximized when $\mathbf{b} = \mathbf{p}$. That is, proportional gambling (where the gambler distributes his wealth according to the horses' probabilities of winning) is optimal. This is proven in the following theorem.

Theorem 4 *The optimum doubling rate is given by*

$$W^*(\mathbf{p}) = \sum p_i \log o_i - H(\mathbf{p})$$

and is achieved by the proportional gambling scheme $\mathbf{b}^ = \mathbf{p}$.*

Proof:

$$\begin{aligned} W(\mathbf{b}, \mathbf{p}) &= \sum p_i \log b_i o_i \\ &= \sum p_i \log \left(\frac{b_i}{p_i} p_i o_i \right) \\ &= \sum p_i \log o_i - H(\mathbf{p}) - D(\mathbf{p} \parallel \mathbf{b}) \\ &\leq \sum p_i \log o_i - H(\mathbf{p}) \end{aligned} \tag{1}$$

with equality if and only if $\mathbf{p} = \mathbf{b}$. □

We will later show that there is a direct connection between gambling and boosting, and more precisely between proportional betting and boosting.

2.4 PAC Learning

Learning per se is a vague concept, so there are many ways to model the process of learning. The model we use is the Probably Approximately Correct (PAC) learning model, first introduced by Leslie Valiant [8]. The definitions that follow are from Kearns and Vazirani [4].

Let \mathcal{X} be a set called the *instance space*. \mathcal{X} is an encoding of objects or instances in the learner's world. A *concept* over \mathcal{X} is a subset $c \subseteq \mathcal{X}$ of the instance space. A concept can also equivalently be defined as a Boolean mapping $c : \mathcal{X} \rightarrow \{0, 1\}$, with $c(x) = 1$ indicating that x is a positive example of c and vice-versa. For example, suppose the instance space consisted of encodings of various hand-written digits zero through nine. Then all representations of the number eight form a concept. For our purposes, the type of encoding will not be relevant; it does not affect the accuracy of a learning algorithm, only its efficiency. A *concept class* \mathcal{C} over \mathcal{X} is a collection of concepts over \mathcal{X} . In this example, the concept class is the collection of the digits zero through nine. The learner outputs a *hypothesis concept*, which is its estimate of the *target concept* to be learned.

The learner has access to a procedure $EX(c, \mathcal{D})$, sometimes called an *oracle*, which on each call will return a labeled example $\langle x, c(x) \rangle$ according to some distribution \mathcal{D} over the examples. After repeated exposures to labeled examples from this oracle, the learning algorithm is expected to correctly classify instances it has never encountered before.

More rigorously, given $0 < \epsilon < 1/2$ and $0 < \delta < 1/2$, the algorithm must be able to output a hypothesis $h(x)$ which will classify examples $x \in X$ with error at most ϵ , with probability at least $(1 - \delta)$. Hence the name Probably Approximately Correct: with high probability, the error of the hypothesis will be low. The error of h is taken with respect to D :

$$error(h) = P_{\mathcal{D}}[c(x) \neq h(x)]$$

The parameters ϵ and δ are called the *accuracy* and *confidence* of the hypothesis, respectively. The exact definition of the PAC model, from Kearns and Vazirani [4], is as follows.

Definition 6 (*The PAC Model*): Let \mathcal{C} be a concept class over \mathcal{X} . We say that \mathcal{C} is PAC learnable if there exists an algorithm L with the following property: for every concept $c \in \mathcal{C}$, for every distribution \mathcal{D} on \mathcal{X} , and for all $0 < \epsilon < 1/2$ and $0 < \delta < 1/2$, if L is given access to $EX(c, \mathcal{D})$ and inputs ϵ and δ , then with probability at least $(1 - \delta)$, L outputs a hypothesis concept $h \in \mathcal{C}$ satisfying $\text{error}(h) \leq \epsilon$. This probability is taken over the random examples drawn by calls to $EX(c, \mathcal{D})$.

In practice, a sample of labeled examples is drawn beforehand according to \mathcal{D} , and the learner is then presented each instance in this sample (note that an example may appear more than once in the sample, and so will be presented more than once). This sample is called the *training sample*. The error of the learner's outputted hypothesis h on the instances from this sample is called the *training error* of h . The error of h over the entire instance space is called the *testing error*. The learner's objective is to have low testing error with high probability.

Testing and training error are naturally related. Intuitively, a learner with low training error should also have low testing error. However, note that the learner could easily have a training error of 0 by memorizing all the labeled examples. In this case, the learner will actually exhibit poor testing error, since it will not have learned any special characteristics of the target concept. When the learner begins to memorize training examples rather than deducing patterns from them, its testing error tends to suffer. This phenomenon is called *overfitting* the training data. Boosting is very resilient towards overfitting, so we will later be able to derive a significant relationship between training and testing error in boosting.

We end this section with two bounds from probability theory that are very useful in PAC analyses.

Theorem 5 (*The Union Bound*): For any probability space, and for any events A_1, A_2, \dots, A_m over that space, where m is finite,

$$P[\cup_{i=1}^m A_i] \leq \sum_{i=1}^m P[A_i]$$

Theorem 6 (*The Chernoff Bound*): Let X_1, \dots, X_m be a sequence of m independent Bernoulli trials, each with probability of success $E[X_i] = p$. Let $\hat{p} = (\sum_{i=1}^m X_i)/m$ be an estimate of p . Note that $E[\hat{p}] = p$. Then for $0 \leq \epsilon \leq 1$, the following bound holds.

$$P[\hat{p} > p + \epsilon] \leq e^{-2m\epsilon^2}$$

2.5 Boosting

Before explaining the notion of boosting, we need to make a distinction between *strong* and *weak* PAC learning. The PAC model presented in 2.4 is known as strong PAC learning. In this model, the learning algorithm must be able to achieve arbitrarily low values of ϵ and δ . A *weak learner*, on the other hand, only promises to meet the PAC criteria for fixed,

Given: $(x_1, y_1), \dots, (x_m, y_m); x_i \in \mathcal{X}, y_i \in \{-1, +1\}$
 Initialize $D_1(i) = 1/m$.
 For $t = 1, \dots, T$:

1. Train weak learner using distribution D_t .
2. Get weak hypothesis $h_t : \mathcal{X} \rightarrow \mathbb{R}$.
3. Choose $\alpha_t = \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$, where ϵ_t is the error of h_t .
4. Update:

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis:

$$H(x) = \text{sign}(f(x)), \text{ where}$$

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

Figure 1: Schapire and Singer's AdaBoost [7]

constant values ϵ_0 and δ_0 . A boosting algorithm uses a weak learner as a subroutine to achieve arbitrarily low accuracy and confidence. Boosting is therefore a means of generating a strong learner from a weak learner. The fact that this is at all possible is a remarkable result, derived by Schapire [5].

Boosting proceeds by training different versions of the weak learner using different distributions on the training data. Thus some versions can be made to specialize in classifying certain examples by putting more weight on these examples. The boosting algorithm then takes a weighted majority of each of the weak learners' outputs. The result of this weighted majority "vote" is the algorithm's prediction.

Currently, one of the most popular boosting algorithms is AdaBoost. A version of Adaboost is shown in Figure 1. Here training examples are taken from an instance space \mathcal{X} . Our label space is $\mathcal{Y} = \{-1, +1\}$, so we are dealing with a binary classification problem. AdaBoost proceeds in rounds. On each round t , a weak learner is trained using a local distribution D_t on the training examples, thus generating a weak hypothesis h_t . This version of AdaBoost allows the hypotheses to output any real number, according to the hypotheses' confidence (a high confidence in $+1$ is expressed by a high positive number, and vice-versa). AdaBoost then determines a performance assessment α_t based on the error of h_t . This parameter is used to reweight the distribution for the next round. The distribution is modified so that incorrectly classified examples are reweighted more heavily, while correctly classified examples are reweighted more lightly. The final hypothesis is a linear combination of the outputs of each weak learner, each weighted by their corresponding performance assessment α_t .

A peculiar property AdaBoost is its resilience towards overfitting. The testing error of AdaBoost tends to improve as the rounds increase indefinitely, whereas other learning algorithms usually reach an overfitting threshold, at which point their testing error worsens. To explain this property, the notion of a *margin* was introduced by Schapire et al. [6]. The margin of an example is simply the difference between the total weight on correct votes and the maximum weight of votes received by any incorrect label. In our binary label case, the margin of an example x with correct label is y is:

$$\text{margin}(x) = \sum_{h_t: h_t(x)=y} \alpha_t h_t(x) - \sum_{h_t: h_t(x) \neq y} \alpha_t h_t(x) = \sum_t y \alpha_t h_t(x) = y f(x)$$

A margin is thus a measure of the algorithm's *certainty* in the label predicted by $H(x)$. I use the term 'certainty' to distinguish the concept from the 'confidence' in the PAC model. Schapire et al. show that boosting improves the margins of training examples even after the training error has been reduced to zero. They also show that improved margins imply better testing error; our work will closely model this analysis.

3 Connection to other fields

3.1 Connection between Adaboost and Logistic Regression

AdaBoost minimizes the exponential loss criterion,

$$\frac{1}{m} \sum_{i=1}^m \exp(-y_i f(x_i))$$

which is an estimate of $E_D[\exp(-yf(x))]$, where D is the distribution on $\mathcal{X} \times \mathcal{Y}$. This expected value is minimized at

$$f^*(x) = \frac{1}{2} \log \frac{P(y = +1 | x)}{P(y = -1 | x)}$$

by Lemma 1 in Friedman et al. [2]. According to Propositions 1 and 2 in this same paper, AdaBoost fits an additive logistic regression model by stage-wise optimization of $E[\exp(-yf(x))]$. Thus the hypothesis $f(x)$ output by AdaBoost is an estimate of $f^*(x)$ (recall that $f(x) \in \mathbb{R}$, while AdaBoost's ultimate prediction is $H(x) = \text{sign}(f(x))$). From this $f(x)$ we can thus obtain an estimate b of the true distribution on the labels.

$$b(y = +1 | x) = \frac{1}{1 + \exp(-2f(x))}$$

Hastie et al. [3] mention that another loss criterion with the *same* population minimizer is the negative binomial log-likelihood, or "deviance". Let $y' = (y + 1)/2$, so that we have the mapping $+1 \rightarrow 1$, $-1 \rightarrow 0$. The binomial log-likelihood and deviance are respectively

defined as

$$\begin{aligned} l(Y, p(y|x)) &= y' \log p(y|x) + (1 - y') \log(1 - p(y|x)) \\ -l(Y, p(y|x)) &= -\log \frac{1}{1 + \exp(-2yf(x))} \\ &= \log(1 + \exp(-2yf(x))) \end{aligned}$$

Each criterion leads to the same solution for $f(x)$. Both exponential loss and deviance are upper bounds on training error. Schapire and Singer [7] prove the exponential loss bound. By a similar analysis we will later prove the deviance bound, which has a nice information-theoretic interpretation.

3.2 Connection to gambling

There is a direct analogy between the classification problem a boosting algorithm faces and gambling. We can think of each individual classification problem on a label Y given an example x as a “horse race”. The labels $+1$ and -1 are analogous to horses that, given the side information x , each have a certain probability of “winning” the race, i.e. being correct.

Now suppose without loss of generality that the true label of x is $+1$. Let η be the probability that x is labeled incorrectly in the training data (we have noisy training data). Then

$$\begin{aligned} p(+1 | x) &= (1 - \eta) \\ p(-1 | x) &= \eta \end{aligned}$$

is the probability mass function associated with each of the labels given x . That is, label $+1$ “wins” the race (is the label of x in the training data) with probability $(1 - \eta)$. As explained above, AdaBoost attempts to estimate this distribution given the example x . In other words, it tries to bet proportionally, which as we have seen is optimal in an information-theoretic sense.

4 New Bounds on Training and Testing Error

As shown previously, AdaBoost’s real-valued output $f(x)$ can be converted into a probability by the formula

$$b(y|x) = \frac{1}{1 + \exp(-2yf(x))}$$

and the predicted label is wrong if and only if $b(y|x) \leq \frac{1}{2}$, where y is the correct label for x . The distribution b is an estimate of the true probability distribution p .

4.1 Training Error

In this section we prove an upper bound on the distribution of the average margin of AdaBoost’s examples. An upper bound on AdaBoost’s training error will follow directly from this result.

Theorem 7 Let $\gamma = (1 + e^{-2\theta})^{-1}$. As the number of training example increases, we approach the following upper bound on the distribution of AdaBoost's average margin.

$$P_S[yf(x) \leq \theta] \leq (H(\mathbf{p}) + D(\mathbf{p}||\mathbf{b})) \cdot \log_{1/\gamma} 2$$

Proof: First note that

$$\begin{aligned} y_i f(x_i) \leq \theta &\Leftrightarrow (1 + \exp(-2y_i f(x_i)))^{-1} \leq (1 + e^{-2\theta})^{-1} \\ &\Leftrightarrow b(y_i|x_i) \leq \gamma \end{aligned}$$

Let $[\pi]$ be an indicator variable which is 1 if the predicate π is true and 0 otherwise. Note that $0 \leq b(y_i|x_i) \leq 1$, since $b(y_i|x_i)$ is a probability. We then have the following set of implications.

$$\begin{aligned} 0 \leq b(y_i|x_i) \leq \gamma &\Rightarrow \log_{1/\gamma} b(y_i|x_i) \leq -1 \\ &\Rightarrow -\log_{1/\gamma} b(y_i|x_i) \geq 1 \\ &\Rightarrow [b(y_i|x_i) \leq \gamma] \leq -\log_{1/\gamma} b(y_i|x_i), \text{ and} \\ 1 \geq b(y_i|x_i) \geq \gamma &\Rightarrow 0 \geq \log_{1/\gamma} b(y_i|x_i) \geq -1 \\ &\Rightarrow 0 \leq -\log_{1/\gamma} b(y_i|x_i) \leq 1 \\ &\Rightarrow [b(y_i|x_i) \leq \gamma] \leq -\log_{1/\gamma} b(y_i|x_i) \end{aligned}$$

In both cases we have $[b(y_i|x_i) \leq \gamma] \leq -\log_{1/\gamma} b(y_i|x_i)$. This in turn implies that

$$\begin{aligned} &\frac{1}{m} \sum_i [b(y_i|x_i) \leq \gamma] \leq \frac{1}{m} \sum_i -\log_{1/\gamma} b(y_i|x_i) \\ \Rightarrow &P_S[b(y|x) \leq \gamma] \leq \frac{1}{m} \sum_i -\log_{1/\gamma} b(y_i|x_i) \\ \Rightarrow &P_S[yf(x) \leq \theta] \leq \frac{1}{m} \sum_i -\log_{1/\gamma} b(y_i|x_i) \end{aligned} \tag{2}$$

The right-hand side approaches $E[-\log_{1/\gamma} b(y|x)]$ in probability, by the weak law of large numbers. Finally, we have

$$\begin{aligned} &E[-\log_{1/\gamma} b(y_i|x_i)] \\ &= -E[\log b(y|x)] \cdot \log_{1/\gamma} 2 \\ &= -W(\mathbf{p}, \mathbf{b}) \cdot \log_{1/\gamma} 2 \tag{3} \\ &= (H(\mathbf{p}) + D(\mathbf{p}||\mathbf{b})) \cdot \log_{1/\gamma} 2 \tag{4} \end{aligned}$$

where (3) follows by definition, and (4) from section 2.3. Substituting this into (2) yields the statement of the theorem. \square

By setting $\theta = 0$, and thus $\gamma = 1/2$, we obtain the following corollary.

Corollary 1 As the number of training example increases, we approach the following upper bound on AdaBoost's training error.

$$P_S[yf(x) \leq 0] \leq H(\mathbf{p}) + D(\mathbf{p}||\mathbf{b})$$

4.2 Generalization Error

In this section, we prove a new bound on the generalization error of boosting that is independent of the size of the weak-learner space. It therefore applies to both finite and infinite weak-learner spaces. Our proof is modeled on that of Schapire et al. [6].

Let \mathcal{H} denote the space from which the weak learners are chosen. \mathcal{H} may be finite or infinite. The weak learners are mappings from the instance space \mathcal{X} to \mathbb{R} —that is, we allow the weak learners to output confidence-rated hypotheses. Let $\mathcal{Y} = \{0, 1\}$ be the label space. Our proof applies only to the special case in which there are only two possible labels. Our proof applies to any majority vote classifier whose output can be translated into a probability distribution as in AdaBoost’s case.

Theorem 8 *Let D be a distribution over $\mathcal{X} \times \{0, 1\}$, and let S be a sample of m examples chosen independently at random according to D . Then with probability at least $1 - \delta$ over the random choice of the training set S , and for all $0 < \beta \leq \frac{1}{2}$, the error of any majority vote classifier whose output can be translated into a probability b over the label space is bounded as follows.*

$$P_D[b(y|x) \leq 1/2] \leq P_S[b(y|x) \leq 1/2 + \beta] + O\left(\frac{1}{\sqrt{m}} \sqrt{\log \log m + \log \frac{1}{\beta} + \log \frac{1}{\delta}}\right)$$

Proof: Let n be an integer greater than 0. Given the estimated distribution b , we can find a related distribution q by generating a random sequence of 0’s and 1’s of length n according to b , and letting q be the type of this sequence. Note that by this procedure $E_b[q] = b$. Now, for any two events A and B ,

$$P(A) = P(B \cap A) + P(\bar{B} \cap A) \leq P(B) + P(\bar{B} \cap A) \leq P(B) + P(\bar{B}|A) \quad (5)$$

Let A be the event that $P_D[b(y|x) \leq 1/2]$, and B the event that $P_D[q(y|x) \leq 1/2 + \beta/2]$. Substituting these into 5, we obtain:

$$P_D[b(y|x) \leq 1/2] \leq P_D[q(y|x) \leq 1/2 + \beta/2] + P_D[q(y|x) > 1/2 + \beta/2 | b(y|x) \leq 1/2] \quad (6)$$

Since (6) holds for any q , we can take the expected value of the right-hand side with respect to the distribution b .

$$\begin{aligned} P_D[b(y|x) \leq 1/2] &\leq P_{D,b}[q(y|x) \leq 1/2 + \beta/2] + P_{D,b}[q(y|x) > 1/2 + \beta/2 | b(y|x) \leq 1/2] \\ &= E_b[P_D(q(y|x) \leq 1/2 + \beta/2)] + E_D[P_b(q(y|x) > 1/2 + \beta/2 | b(y|x) \leq 1/2)] \end{aligned} \quad (7)$$

We bound both term in (7) separately. Consider the probability inside the second expectation (taken with respect to b). For any fixed example (x, y) , this is the probability that $q(y|x)$ will be larger than $b(y|x)$ by more than $\beta/2$. The distribution q is an estimate of b determined from the outcomes of n independent Bernoulli trials, so the Chernoff bound applies.

$$P_b[b(y|x) > 1/2 + \beta/2 | b(y|x) \leq 1/2] \leq e^{-n\beta^2/2} \quad (8)$$

Now consider the probability inside the first expectation. For fixed q and β , the probability with respect to the random choice of S that

$$P_D[q(y|x) \leq 1/2 + \beta/2] > P_S[q(y|x) \leq 1/2 + \beta/2] + \epsilon \quad (9)$$

is at most $e^{-2m\epsilon^2}$, again by the Chernoff bound. Since each probability in the distribution q has denominator n , we need only consider values of β such that $\beta = \frac{2i}{n}$, for $i = 0, 1, 2, \dots, \frac{n}{2}$. There are no more than $(n+1)$ such values. Meanwhile, $q \in \mathcal{P}_n$ and the size of our label space is 2, so there are $(n+1)$ choices for q . So by the union bound, the probability that equation 9 holds is at most $(n+1)^2 e^{-2m\epsilon^2}$. Setting δ equal to this value, we have

$$\delta = (n+1)^2 e^{-2m\epsilon^2} \Rightarrow \epsilon = \sqrt{\frac{1}{2m} \ln \frac{(n+1)^2}{\delta}}$$

With this ϵ we then have with probability at least $(1 - \delta)$ that

$$P_D[q(y|x) \leq 1/2 + \beta/2] \leq P_S[q(y|x) \leq 1/2 + \beta/2] + \epsilon \quad (10)$$

for every choice of β and q .

We can now relate $P_S[q(y|x) \leq 1/2 + \beta/2]$ to $P_S[b(y|x) \leq 1/2 + \beta]$ by exactly the same procedure.

$$P_{S,q}[q(y|x) \leq 1/2 + \beta/2] \leq E_q[P_S[b(y|x) \leq 1/2 + \beta]] + E_S[P_q[q(y|x) \leq 1/2 + \beta/2 \mid b(y|x) > 1/2 + \beta]] \quad (11)$$

The probability inside the first expectation is a constant with respect to q , so this expected value is simply $P_S[b(y|x) \leq 1/2 + \beta]$. For the probability inside the second expected value, we again use the Chernoff bound.

$$P_q[q(y|x) \leq 1/2 + \beta/2 \mid b(y|x) > 1/2 + \beta] \leq e^{-n\beta^2/2} \quad (12)$$

Combining (7), (8), (10), and (12), we obtain our final result.

$$P_D[b(y|x) \leq 1/2] \leq P_S[b(y|x) \leq 1/2 + \beta] + 2e^{-n\beta^2/2} + \sqrt{\frac{1}{2m} \ln \frac{(n+1)^2}{\delta}} \quad (13)$$

Setting $n = \lceil (\ln m)/\beta^2 \rceil$ yields the statement of the theorem. \square

4.3 Comparison with Original Bound

In this section, we compare the alternative upper-bound on generalization error just derived to the original bound in Schapire [6]. To reiterate, the two bounds are as follows.

Theorem 9 (*Original Bound*): *with probability at least $1 - \delta$ over the random choice of the training set S , and for all $\theta > 0$,*

$$P_D[yf(x) \leq 0] \leq P_S[yf(x) \leq \theta] + O\left(\frac{1}{\sqrt{m}} \sqrt{\frac{\log m \log |\mathcal{H}|}{\theta^2}} + \log \frac{1}{\delta}\right)$$

Theorem 10 (*New Bound*): with probability at least $1 - \delta$ over the random choice of the training set S , and for all $0 < \beta \leq \frac{1}{2}$,

$$P_D[b(y|x) \leq 1/2] \leq P_S[b(y|x) \leq 1/2 + \beta] + O\left(\frac{1}{\sqrt{m}} \sqrt{\log \log m + \log \frac{1}{\beta} + \log \frac{1}{\delta}}\right)$$

We want to consider what happens to these bounds as θ and β both approach zero, because as this happens the margin distribution in the bounds approaches the training error. In the second term, the new bound has an advantage because it is logarithmic in β , whereas the original bound is linear in θ . This is a little misleading, however, because a linear decrease in θ implies an exponential decrease in β by the way they are related:

$$\beta = \frac{1}{1 + e^{-2\theta}} - 1/2$$

It remains to be seen how these effects balance out, and at this point we cannot assert that our bound is any better with respect to the margin parameter (either β or θ , depending on the case). Our bound does have the distinct advantage of being independent of the size of the weak-learner space, however.

5 Conclusion

We have described the analogy between optimal gambling and boosting. This allowed us to derive a new upper bound on training error which explicitly contains the effect of noise in the training data. We have provided an alternative upper bound on the generalization error of boosting, using the method of types from information-theory. This upper bound is independent of the size of the weak-learner space.

At this point, it would be interesting to further compare the original and new generalization error bounds both theoretically and empirically to see how significant the size of the weak-learner space really is in the original bound. Another open problem would be to determine a means of calculating the relative entropy term in the training error upper bound. Finally, our paper did not investigate AdaBoost's update mechanism, which is fundamental to its success. This aspect should also lend itself well to information-theoretic analysis.

6 Acknowledgements

I would like to thank Professor Aslam for being extremely generous with his time in introducing me to PAC Learning, boosting, and information-theory, and for his guidance and support throughout this project. I would also like to thank Professors Mackey and Khaneja of Harvard for sparking my interest in probability and statistics.

References

- [1] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley & Sons, 1991.
- [2] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28:307–337, 2000.
- [3] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer, 2001.
- [4] Michael J. Kearns and Umesh V. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, 1994.
- [5] Robert Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–226, 1990.
- [6] Robert Schapire, Yoav Freund, Peter Bartlett, and Wee Sun Lee. Boosting the margin: a new explanation for the effectiveness of the voting method. *Annals of Statistics*, 1998.
- [7] Robert Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, 1998.
- [8] Leslie Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, November 1984.