The Annals of Statistics, 26(5):1651-1686, 1998.

Boosting the Margin:
A New Explanation for the Effectiveness of Voting Methods

Robert E. Schapire
AT&T Labs
180 Park Avenue, Room A279
Florham Park, NJ 07932-0971 USA
schapire@research.att.com

Yoav Freund
AT&T Labs
180 Park Avenue, Room A205
Florham Park, NJ 07932-0971 USA
yoav@research.att.com

Peter Bartlett
Dept. of Systems Engineering
RSISE, Aust. National University
Canberra, ACT 0200 Australia
Peter.Bartlett@anu.edu.au

Wee Sun Lee
School of Electrical Engineering
University College UNSW
Australian Defence Force Academy
Canberra ACT 2600 Australia
w-lee@ee.adfa.oz.au

May 7, 1998

Abstract. One of the surprising recurring phenomena observed in experiments with boosting is that the test error of the generated classifier usually does not increase as its size becomes very large, and often is observed to decrease even after the training error reaches zero. In this paper, we show that this phenomenon is related to the distribution of *margins* of the training examples with respect to the generated voting classification rule, where the margin of an example is simply the difference between the number of correct votes and the maximum number of votes received by any incorrect label. We show that techniques used in the analysis of Vapnik's support vector classifiers and of neural networks with small weights can be applied to voting methods to relate the margin distribution to the test error. We also show theoretically and experimentally that boosting is especially effective at increasing the margins of the training examples. Finally, we compare our explanation to those based on the bias-variance decomposition.

1 Introduction

This paper is about methods for improving the performance of a *learning algorithm*, sometimes also called a prediction algorithm or classification method. Such an algorithm operates on a given set of *instances* (or cases) to produce a *classifier*, sometimes also called a classification rule or, in the machine-learning literature, a hypothesis. The goal of a learning algorithm is to find a classifier with low *generalization* or *prediction error*, i.e., a low misclassification rate on a separate test set.

In recent years, there has been growing interest in learning algorithms which achieve high accuracy by *voting* the predictions of several classifiers. For example, several researchers have reported significant improvements in performance using voting methods with decision-tree learning algorithms such as C4.5 or CART as well as with neural networks [3, 6, 8, 12, 13, 16, 18, 29, 31, 37].

We refer to each of the classifiers that is combined in the vote as a *base classifier* and to the final voted classifier as the *combined classifier*.

As examples of the effectiveness of these methods, consider the results of the following two experiments using the "letter" dataset. (All datasets are described in Appendix B.) In the first experiment, we used Breiman's bagging method [6] on top of C4.5 [32], a decision-tree learning algorithm similar to CART [9]. That is, we reran C4.5 many times on random "bootstrap" subsamples and combined the computed trees using simple voting. In the top left of Figure 1, we have shown the training and test error curves (lower and upper curves, respectively) of the combined classifier as a function of the number of trees combined. The test error of C4.5 on this dataset (run just once) is 13.8%. The test error of bagging 1000 trees is 6.6%, a significant improvement. (Both of these error rates are indicated in the figure as horizontal grid lines.)

In the second experiment, we used Freund and Schapire's AdaBoost algorithm [20] on the same dataset, also using C4.5. This method is similar to bagging in that it reruns the base learning algorithm C4.5 many times and combines the computed trees using voting. However, the subsamples that are used for training each tree are chosen in a manner which concentrates on the "hardest" examples. (Details are given in Section 3.) The results of this experiment are shown in the top right of Figure 1. Note that boosting drives the test error down even further to just 3.1%. Similar improvements in test error have been demonstrated on many other benchmark problems (see Figure 2).

These error curves reveal a remarkable phenomenon, first observed by Drucker and Cortes [16], and later by Quinlan [31] and Breiman [8]. Ordinarily, as classifiers become more and more complex, we expect their generalization error eventually to degrade. Yet these curves reveal that test error does not increase for either method even after 1000 trees have been combined (by which point, the combined classifier involves more than two million decision-tree nodes). How can it be that such complex classifiers have such low error rates? This seems especially surprising for boosting in which each new decision tree is trained on an ever more specialized subsample of the training set.

Another apparent paradox is revealed in the error curve for AdaBoost. After just five trees have been combined, the training error of the combined classifier has already dropped to zero, but the test error continues to drop[1] from 8.4% on round 5 down to 3.1% on round 1000. Surely, a combination of five trees is much simpler than a combination of 1000 trees, and both perform equally well on the training set (perfectly, in fact). So how can it be that the larger and more complex combined classifier performs so much better on the test set?

The results of these experiments seem to contradict Occam's razor, one of the fundamental principles in the theory of machine learning. This principle states that in order to achieve good test error, the classifier should be as simple as possible. By "simple," we mean that the classifier is chosen from a restricted space of classifiers. When the space is finite, we use its cardinality as the measure of complexity and when it is infinite we use the VC dimension [42] which is often closely related to the number of parameters that define the classifier. Typically, both in theory and in practice, the difference between the training error and the test error increases when the complexity of the classifier increases.

Indeed, such an analysis of boosting (which could also be applied to bagging) was carried out by Freund and Schapire [20] using the methods of Baum and Haussler [4]. This analysis predicts that the test error eventually will increase as the number of base classifiers combined increases. Such a prediction is clearly incorrect in the case of the experiments described above, as was pointed out by Quinlan [31] and Breiman [8]. The apparent contradiction is especially stark in the boosting experiment in which the test error continues to decrease even after the training error has reached zero.

Breiman [8] and others have proposed definitions of bias and variance for classification, and have argued that voting methods work primarily by reducing the variance of a learning algorithm. This explanation is useful for bagging in that bagging tends to be most effective when the variance is large.

[1] Even when the training error of the combined classifier reaches zero, AdaBoost continues to obtain new base classifiers by training the base learning algorithm on different subsamples of the data. Thus, the combined classifier continues to evolve, even after its training error reaches zero. See Section 3 for more detail.

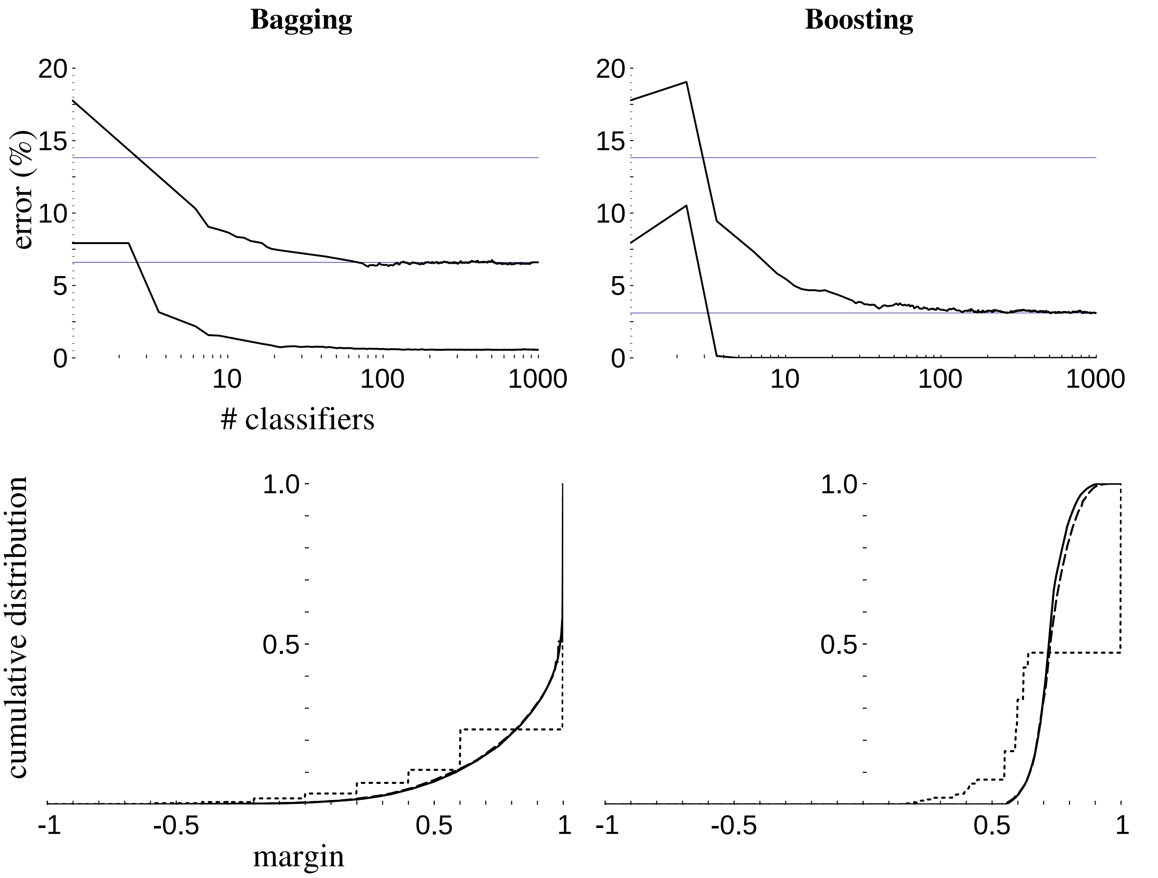

Figure 1: Error curves and margin distribution graphs for bagging and boosting C4.5 on the letter dataset. Learning curves are shown directly above corresponding margin distribution graphs. Each learning-curve figure shows the training and test error curves (lower and upper curves, respectively) of the combined classifier as a function of the number of classifiers combined. Horizontal lines indicate the test error rate of the base classifier as well as the test error of the final combined classifier. The margin distribution graphs show the cumulative distribution of margins of the training instances after 5, 100 and 1000 iterations, indicated by short-dashed, long-dashed (mostly hidden) and solid curves, respectively.

However, for boosting, this explanation is, at best, incomplete. As will be seen in Section 5, large variance of the base classifiers is *not* a requirement for boosting to be effective. In some cases, boosting even increases the variance while reducing the overall generalization error.

Intuitively, it might seem reasonable to think that because we are simply voting the base classifiers, we are not actually increasing their complexity but merely "smoothing" their predictions. However, as argued in Section 5.4, the complexity of such combined classifiers can be much greater than that of the base classifiers and can result in over-fitting.

In this paper, we present an alternative theoretical analysis of voting methods, applicable, for instance, to bagging, boosting, "arcing" [8] and ECOC [13]. Our approach is based on a similar result presented by Bartlett [2] in a different context. We prove rigorous, non-asymptotic upper bounds on the generalization error of voting methods in terms of a measure of performance of the combined classifier on the training set. Our bounds also depend on the number of training examples and the "complexity" of the base classifiers, but do *not* depend explicitly on the *number* of base classifiers. Although too loose

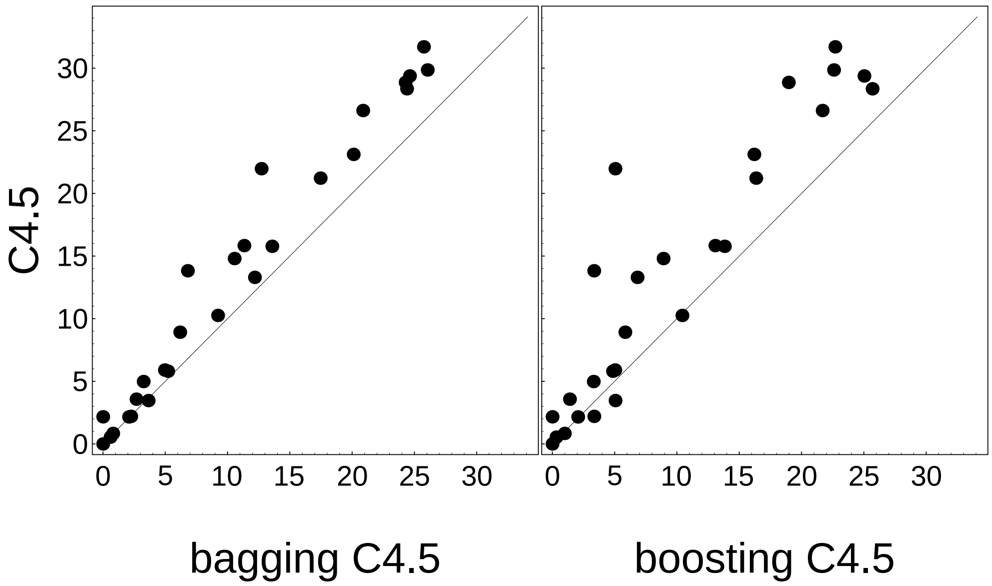

Figure 2: Comparison of C4.5 versus bagging C4.5 and boosting C4.5 on a set of 27 benchmark problems as reported by Freund and Schapire [18]. Each point in each scatter plot shows the test error rate of the two competing algorithms on a single benchmark. The y-coordinate of each point gives the test error rate (in percent) of C4.5 on the given benchmark, and the x-coordinate gives the error rate of bagging (left plot) or boosting (right plot). All error rates have been averaged over multiple runs.

to give practical quantitative predictions, our bounds do give a qualitative explanation of the shape of the observed learning curves, and our analysis may be helpful in understanding why these algorithms fail or succeed, possibly leading to the design of even more effective voting methods.

The key idea of this analysis is the following. In order to analyze the generalization error, one should consider more than just the training error, i.e., the *number* of incorrect classifications in the training set. One should also take into account the *confidence* of the classifications. Here, we use a measure of the classification confidence for which it is possible to prove that an improvement in this measure of confidence *on the training set* guarantees an improvement in the upper bound on the generalization error.

Consider a combined classifier whose prediction is the result of a vote (or a weighted vote) over a set of base classifiers. Suppose that the weights assigned to the different base classifiers are normalized so that they sum to one. Fixing our attention on a particular example, we refer to the sum of the weights of the base classifiers that predict a particular label as the *weight* of that label. We define the classification *margin* for the example as the difference between the weight assigned to the correct label and the maximal weight assigned to any single incorrect label. It is easy to see that the margin is a number in the range $[-1, 1]$ and that an example is classified correctly if and only if its margin is positive. A large positive margin can be interpreted as a "confident" correct classification.

Now consider the distribution of the margin over the whole set of training examples. To visualize this distribution, we plot the fraction of examples whose margin is at most x as a function of $x \in [-1, 1]$. We refer to these graphs as *margin distribution graphs*. At the bottom of Figure 1, we show the margin distribution graphs that correspond to the experiments described above.

Our main observation is that both boosting and bagging tend to increase the margins associated with examples and converge to a margin distribution in which most examples have large margins. Boosting is especially aggressive in its effect on examples whose initial margin is small. Even though the training error remains unchanged (at zero) after round 5, the margin distribution graph changes quite significantly so that after 100 iterations all examples have a margin larger than 0.5. In comparison, on round 5, about 7.7% of the examples have margin below 0.5. Our experiments, detailed later in the paper, show that there is a good correlation between a reduction in the fraction of training examples with small margin and improvements in the test error.

The idea that maximizing the margin can improve the generalization error of a classifier was previously suggested and studied by Vapnik [42] and led to his work with Cortes on support-vector classifiers [10], and with Boser and Guyon [5] on optimal margin classifiers. In Section 6, we discuss the relation between our work and Vapnik's in greater detail.

Shawe-Taylor et al. [38] gave bounds on the generalization error of support-vector classifiers in terms of the margins, and Bartlett [2] used related techniques to give a similar bound for neural networks with small weights. A consequence of Bartlett's result is a bound on the generalization error of a voting classifier in terms of the fraction of training examples with small margin.

In Section 2, we use a similar but simpler approach to give a slightly better bound. Here we give the main intuition behind the proof. This idea brings us back to Occam's razor, though in a rather indirect way. Recall that an example is classified correctly if its margin is positive. If an example is classified by a large margin (either positive or negative), then small changes to the weights in the majority vote are unlikely to change the label. If most of the examples have a large margin then the classification error of the original majority vote and the perturbed majority vote will be similar. Suppose now that we had a small set of weighted majority rules that was fixed ahead of time, called the "approximating set." One way of perturbing the weights of the classifier majority vote is to find a nearby rule within the approximating set. As the approximating set is small, we can guarantee that the error of the approximating rule on the training set is similar to its generalization error, and as its error is similar to that of the original rule, the generalization error of the original rule should also be small. Thus, we are back to an Occam's razor argument in which instead of arguing that the classification rule itself is simple, we argue that the rule is *close* to a simple rule.

Boosting is particularly good at finding classifiers with large margins in that it concentrates on those examples whose margins are small (or negative) and forces the base learning algorithm to generate good classifications for those examples. This process continues even after the training error has reached zero, which explains the continuing drop in test error.

In Section 3, we show that the powerful effect of boosting on the margin is not merely an empirical observation but is in fact the result of a provable property of the algorithm. Specifically, we are able to prove upper bounds on the number of training examples below a particular margin in terms of the training errors of the individual base classifiers. Under certain conditions, these bounds imply that the number of training examples with small margin drops exponentially fast with the number of base classifiers.

In Section 4, we give more examples of margin distribution graphs for other datasets, base learning algorithms and combination methods.

In Section 5, we discuss the relation of our work to bias-variance decompositions. In Section 6, we compare our work to Vapnik's optimal margin classifiers, and in Section 7, we briefly discuss similar results for learning convex combinations of functions for loss measures other than classification error.

2 Generalization Error as a Function of Margin Distributions

In this section, we prove that achieving a large margin on the training set results in an improved bound on the generalization error. This bound does not depend on the number of classifiers that are combined in the vote. The approach we take is similar to that of Shawe-Taylor et al. [38] and Bartlett [2], but the proof here is simpler and more direct. A slightly weaker version of Theorem 1 is a special case of Bartlett's main result.

We give a proof for the special case in which there are just two possible labels $\{-1, +1\}$. In Appendix A, we examine the case of larger finite sets of labels.

Let $\mathcal{H}$ denote the space from which the base classifiers are chosen; for example, for C4.5 or CART, it is the space of decision trees of an appropriate size. A base classifier $h \in \mathcal{H}$ is a mapping from an instance space X to $\{-1, +1\}$. We assume that examples are generated independently at random according to some fixed but unknown distribution $\mathcal{D}$ over $X \times \{-1, +1\}$. The training set is a list of m pairs $S = \langle (x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m) \rangle$ chosen according to $\mathcal{D}$. We use $\mathbf{P}_{(x,y)\sim\mathcal{D}}[A]$ to denote the probability of the event A when the example (x, y) is chosen according to $\mathcal{D}$, and $\mathbf{P}_{(x,y)\sim S}[A]$ to denote probability with respect to choosing an example uniformly at random from the training set. When clear from context, we abbreviate these by $\mathbf{P}_{\mathcal{D}}[A]$ and $\mathbf{P}_S[A]$. We use $\mathbf{E}_{\mathcal{D}}[A]$ and $\mathbf{E}_S[A]$ to denote expected value in a similar manner.

We define the *convex hull* $\mathcal{C}$ of $\mathcal{H}$ as the set of mappings that can be generated by taking a weighted average of classifiers from $\mathcal{H}$:

$$\mathcal{C} \doteq \left\{ f : x \mapsto \sum_{h \in \mathcal{H}} a_h h(x) \;\middle|\; a_h \geq 0; \;\; \sum_h a_h = 1 \right\}$$

where it is understood that only finitely many a_h's may be nonzero.[2] The majority vote rule that is associated with f gives the wrong prediction on the example (x, y) only if $yf(x) \leq 0$. Also, the margin of an example (x, y) in this case is simply $yf(x)$.

The following two theorems, the main results of this section, state that with high probability, the generalization error of any majority vote classifier can be bounded in terms of the number of training examples with margin below a threshold θ, plus an additional term which depends on the number of training examples, some "complexity" measure of $\mathcal{H}$, and the threshold θ (preventing us from choosing θ too close to zero).

The first theorem applies to the case that the base classifier space $\mathcal{H}$ is finite, such as the set of all decision trees of a given size over a set of discrete-valued features. In this case, our bound depends only on $\log |\mathcal{H}|$, which is roughly the description length of a classifier in $\mathcal{H}$. This means that we can tolerate very large classifier classes.

If $\mathcal{H}$ is infinite—such as the class of decision trees over *continuous* features—the second theorem gives a bound in terms of the Vapnik-Chervonenkis dimension[3] of $\mathcal{H}$.

Note that the theorems apply to *every* majority vote classifier, regardless of how it is computed. Thus, the theorem applies to any voting method, including boosting, bagging, etc.

---

[2] A finite support is not a requirement for our proof but is sufficient for the application here which is to majority votes over a finite number of base classifiers.

[3] Recall that the VC-dimension is defined as follows: Let $\mathcal{F}$ be a family of functions $f : X \to Y$ where $|Y| = 2$. Then the VC-dimension of $\mathcal{F}$ is defined to be the largest number d such that there exists $x_1, \ldots, x_d \in X$ for which $|\{\langle f(x_1), \ldots, f(x_d) \rangle : f \in \mathcal{F}\}| = 2^d$. Thus, the VC-dimension is the cardinality of the largest subset S of the space X for which the set of restrictions to S of functions in $\mathcal{F}$ contains all functions from S to Y.

2.1 Finite base-classifier spaces

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

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

Proof: For the sake of the proof we define $\mathcal{C}_N$ to be the set of *unweighted* averages over N elements from $\mathcal{H}$:

$$\mathcal{C}_N \doteq \left\{ f : x \mapsto \frac{1}{N}\sum_{i=1}^{N} h_i(x) \,\middle|\, h_i \in \mathcal{H} \right\} .$$

We allow the same $h \in \mathcal{H}$ to appear multiple times in the sum. This set will play the role of the *approximating set* in the proof.

Any majority vote classifier $f \in \mathcal{C}$ can be associated with a distribution over $\mathcal{H}$ as defined by the coefficients a_h. By choosing N elements of $\mathcal{H}$ independently at random according to this distribution we can generate an element of $\mathcal{C}_N$. Using such a construction we map each $f \in \mathcal{C}$ to a distribution $\mathcal{Q}$ over $\mathcal{C}_N$. That is, a function $g \in \mathcal{C}_N$ distributed according to $\mathcal{Q}$ is selected by choosing $h_1, \ldots, h_N$ independently at random according to the coefficients a_h and then defining $g(x) = (1/N)\sum_{i=1}^{N} h_i(x)$.

Our goal is to upper bound the generalization error of $f \in \mathcal{C}$. For any $g \in \mathcal{C}_N$ and $\theta > 0$ we can separate this probability into two terms:

$$\mathbf{P}_{\mathcal{D}}\left[yf(x) \leq 0\right] \leq \mathbf{P}_{\mathcal{D}}\left[yg(x) \leq \theta/2\right] + \mathbf{P}_{\mathcal{D}}\left[yg(x) > \theta/2, yf(x) \leq 0\right]. \tag{1}$$

This holds because, in general, for two events A and B,

$$\mathbf{P}\left[A\right] = \mathbf{P}\left[B \cap A\right] + \mathbf{P}\left[\overline{B} \cap A\right] \leq \mathbf{P}\left[B\right] + \mathbf{P}\left[\overline{B} \cap A\right]. \tag{2}$$

As Equation (1) holds for any $g \in \mathcal{C}_N$, we can take the expected value of the right hand side with respect to the distribution $\mathcal{Q}$ and get:

$$\begin{aligned}
\mathbf{P}_{\mathcal{D}}\left[yf(x) \leq 0\right] &\leq \mathbf{P}_{\mathcal{D},g\sim\mathcal{Q}}\left[yg(x) \leq \theta/2\right] + \mathbf{P}_{\mathcal{D},g\sim\mathcal{Q}}\left[yg(x) > \theta/2, yf(x) \leq 0\right] \\
&= \mathbf{E}_{g\sim\mathcal{Q}}\left[\mathbf{P}_{\mathcal{D}}\left[yg(x) \leq \theta/2\right]\right] + \mathbf{E}_{\mathcal{D}}\left[\mathbf{P}_{g\sim\mathcal{Q}}\left[yg(x) > \theta/2, yf(x) \leq 0\right]\right] \\
&\leq \mathbf{E}_{g\sim\mathcal{Q}}\left[\mathbf{P}_{\mathcal{D}}\left[yg(x) \leq \theta/2\right]\right] + \mathbf{E}_{\mathcal{D}}\left[\mathbf{P}_{g\sim\mathcal{Q}}\left[yg(x) > \theta/2 \mid yf(x) \leq 0\right]\right]. \qquad (3)
\end{aligned}$$

We bound both terms in Equation (3) separately, starting with the second term. Consider a fixed example (x, y) and take the probability inside the expectation with respect to the random choice of g. It is clear that $f(x) = \mathbf{E}_{g\sim\mathcal{Q}}\left[g(x)\right]$ so the probability inside the expectation is equal to the probability that the average over N random draws from a distribution over $\{-1, +1\}$ is larger than its expected value by more than $\theta/2$. The Chernoff bound yields

$$\mathbf{P}_{g\sim\mathcal{Q}}\left[yg(x) > \theta/2 \mid yf(x) \leq 0\right] \leq e^{-N\theta^2/8}. \tag{4}$$

To upper bound the first term in (3) we use the union bound. That is, the probability over the choice of S that there exists *any* $g \in \mathcal{C}_N$ and $\theta > 0$ for which

$$\mathbf{P}_{\mathcal{D}}\left[yg(x) \leq \theta/2\right] > \mathbf{P}_S\left[yg(x) \leq \theta/2\right] + \epsilon_N$$

is at most $(N+1)|\mathcal{C}_N|e^{-2m\epsilon_N^2}$. The exponential term $e^{-2m\epsilon_N^2}$ comes from the Chernoff bound which holds for any single choice of g and θ. The term $(N+1)|\mathcal{C}_N|$ is an upper bound on the number of such choices where we have used the fact that, because of the form of functions in $\mathcal{C}_N$, we need only consider values of θ of the form $2i/N$ for $i = 0, \ldots, N$. Note that $|\mathcal{C}_N| \le |\mathcal{H}|^N$.

Thus, if we set $\epsilon_N = \sqrt{(1/2m)\ln((N+1)|\mathcal{H}|^N/\delta_N)}$, and take expectation with respect to $\mathcal{Q}$, we get that, with probability at least $1-\delta_N$

$$\mathbf{P}_{\mathcal{D},g\sim\mathcal{Q}}\left[yg(x) \le \theta/2\right] \le \mathbf{P}_{S,g\sim\mathcal{Q}}\left[yg(x) \le \theta/2\right] + \epsilon_N \tag{5}$$

for every choice of θ, and every distribution $\mathcal{Q}$.

To finish the argument we relate the fraction of the training set on which $yg(x) \le \theta/2$ to the fraction on which $yf(x) \le \theta$, which is the quantity that we measure. Using Equation (2) again, we have that

$$\begin{aligned}
\mathbf{P}_{S,g\sim\mathcal{Q}}\left[yg(x) \le \theta/2\right] &\le \mathbf{P}_{S,g\sim\mathcal{Q}}\left[yf(x) \le \theta\right] + \mathbf{P}_{S,g\sim\mathcal{Q}}\left[yg(x) \le \theta/2, yf(x) > \theta\right] \\
&= \mathbf{P}_S\left[yf(x) \le \theta\right] + \mathbf{E}_S\left[\mathbf{P}_{g\sim\mathcal{Q}}\left[yg(x) \le \theta/2, yf(x) > \theta\right]\right] \\
&\le \mathbf{P}_S\left[yf(x) \le \theta\right] + \mathbf{E}_S\left[\mathbf{P}_{g\sim\mathcal{Q}}\left[yg(x) \le \theta/2 \mid yf(x) > \theta\right]\right]. \qquad (6)
\end{aligned}$$

To bound the expression inside the expectation we use the Chernoff bound as we did for Equation (4) and get

$$\mathbf{P}_{g\sim\mathcal{Q}}\left[yg(x) \le \theta/2 \mid yf(x) > \theta\right] \le e^{-N\theta^2/8}. \tag{7}$$

Let $\delta_N = \delta/(N(N+1))$ so that the probability of failure for *any* N will be at most $\sum_{N\ge1}\delta_N = \delta$. Then combining Equations (3), (4), (5), (6) and (7), we get that, with probability at least $1-\delta$, for every $\theta > 0$ and every $N \ge 1$:

$$\mathbf{P}_{\mathcal{D}}\left[yf(x) \le 0\right] \le \mathbf{P}_S\left[yf(x) \le \theta\right] + 2e^{-N\theta^2/8} + \sqrt{\frac{1}{2m}\ln\left(\frac{N(N+1)^2|\mathcal{H}|^N}{\delta}\right)}. \tag{8}$$

Finally, the statement of the theorem follows by setting $N = \lceil (4/\theta^2)\ln(m/\ln|\mathcal{H}|)\rceil$. ∎

2.2 Discussion of the bound

Let us consider the quantitative predictions that can be made using Theorem 1. It is not hard to show that if $\delta > 0$ and $\theta > 0$ are held fixed as $m \to \infty$ the bound given in Equation (8) with the choice of N given in the theorem converges to

$$\mathbf{P}_{\mathcal{D}}\left[yf(x) \le 0\right] \le \mathbf{P}_S\left[yf(x) \le \theta\right] + \sqrt{\frac{2\ln m \ln|\mathcal{H}|}{m\theta^2}} + o\left(\sqrt{\frac{\ln m}{m}}\right). \tag{9}$$

In fact, if $\theta \le 1/2$, $\delta = 0.01$ (1% probability of failure), $|\mathcal{H}| \ge 10^6$ and $m \ge 1000$ then the second term on the right-hand side of Equation (9) is a pretty good approximation of the second and third terms on the right-hand side of Equation (8), as is demonstrated in Figure 3.

From Equation (9) and from Figure 3 we see that the bounds given here start to be meaningful only when the size of the training set is in the tens of thousands. As we shall see in Section 4, the actual performance of AdaBoost is much better than predicted by our bounds; in other words, while our bounds are not asymptotic (i.e., they hold for any size of the training set), they are still very loose. The bounds we give in the next section for infinite base-classifier spaces are even looser. It is an important and challenging open problem to prove tighter bounds.

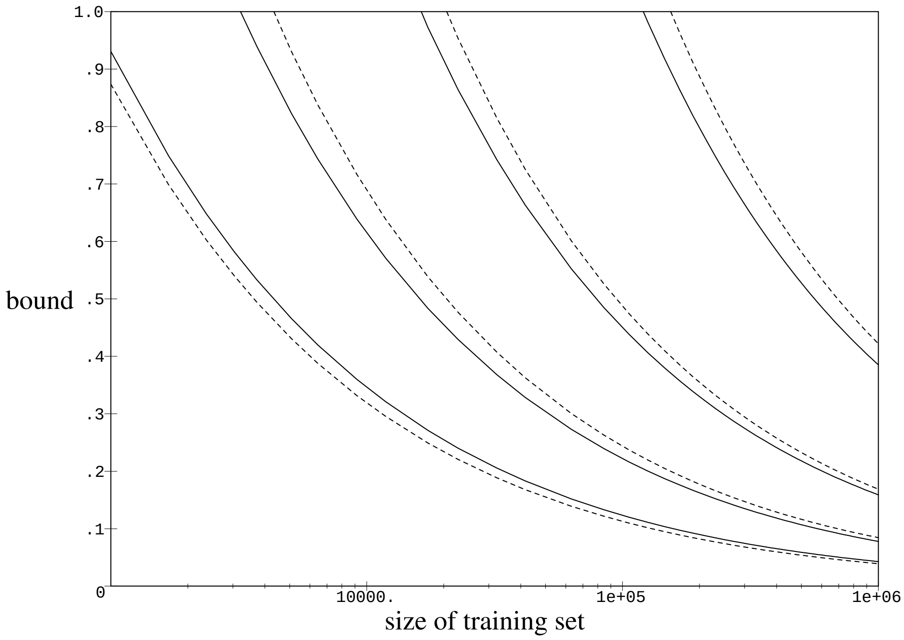

Figure 3: A few plots of the second and third terms in the bound given in Equation (8) (solid lines) and their approximation by the second term in Equation (9) (dotted lines). The horizontal axis denotes the number of training examples (with a logarithmic scale) and the vertical axis denotes the value of the bound. All plots are for $\delta = 0.01$ and $|\mathcal{H}| = 10^6$. Each pair of close lines corresponds to a different value of θ; counting the pairs from the upper right to the lower left, the values of θ are $1/20$, $1/8$, $1/4$ and $1/2$.

From a practical standpoint, the fact that the bounds we give are so loose suggests that there might exist criteria different from the one used in Theorem 1 which are better in predicting the performance of voting classifiers. Breiman [7] and Grove and Schuurmans [23] experimented with maximizing the *minimal* margin, that is, the smallest margin achieved on the training set. The advantage of using this criterion is that maximizing the minimal margin can be done efficiently (if the set of base classifiers is not too large) using linear programming. Unfortunately, their experiments indicate that altering the combined classifier generated by AdaBoost so as to maximize the minimal margin *increases* the generalization error more often than not. In one experiment reported by Breiman, the generalization error increases even though the margins of *all* of the instances are increased (for this dataset, called "ionosphere," the number of instances is 351, much too small for our bounds to apply). While none of these experiments contradict the theory, they highlight the incompleteness of the theory and the need to refine it.

2.3 Infinite base-classifier spaces

Theorem 2 *Let $\mathcal{D}$ be a distribution over $X \times \{-1, 1\}$, and let S be a sample of m examples chosen* *independently at random according to $\mathcal{D}$. Suppose the base-classifier space $\mathcal{H}$ has VC-dimension d, and let $\delta > 0$. Assume that $m \geq d \geq 1$. Then with probability at least $1-\delta$ over the random choice of the training set S, every weighted average function $f \in \mathcal{C}$ satisfies the following bound for all $\theta > 0$:*

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

The proof of this theorem uses the following uniform convergence result, which is a refinement of the Vapnik and Chervonenkis result due to Devroye [11]. Let $\mathcal{A}$ be a class of subsets of a space Z, and define

$$s(\mathcal{A}, m) = \max\left\{|\{A \cap S : A \in \mathcal{A}\}| : S \subseteq Z,\ |S| = m\right\}.$$

Lemma 3 (Devroye) *For any class $\mathcal{A}$ of subsets of Z, and for a sample S of m examples chosen independently at random according to a distribution $\mathcal{D}$ over Z, we have*

$$\mathbf{P}_{S\sim\mathcal{D}^m}\left[\sup_{A\in\mathcal{A}}\left|\mathbf{P}_{z\sim S}\left[z \in A\right] - \mathbf{P}_{z\sim\mathcal{D}}\left[z \in A\right]\right| > \epsilon\right] \leq 4e^8 s(\mathcal{A}, m^2)\exp(-2m\epsilon^2).$$

In other words, the lemma bounds the probability of a significant deviation between the empirical and true probabilities of *any* of the events in the family $\mathcal{A}$.

Proof: (of Theorem 2) The proof proceeds in the same way as that of Theorem 1, until we come to upper bound the first term in (3). Rather than the union bound, we use Lemma 3.

Define

$$\mathcal{A} = \left\{\{(x,y) \in X \times \{-1,1\} : yg(x) > \theta/2\} : g \in \mathcal{C}_N,\ \theta > 0\right\}.$$

Let $x_1, \ldots, x_m \in X$ and $y_1, \ldots, y_m \in \{-1,1\}$. Since the VC-dimension of $\mathcal{H}$ is d, Sauer's lemma [33, 41] states that

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

for $m \geq d \geq 1$. This implies that

$$|\{\langle y_1 g(x_1), \ldots, y_m g(x_m)\rangle : g \in \mathcal{C}_N\}| \leq \left(\frac{em}{d}\right)^{dN}$$

since each $g \in \mathcal{C}_N$ is composed of N functions from $\mathcal{H}$. Since we need only consider $N+1$ distinct values of θ, it follows that $s(\mathcal{A}, m) \leq (N+1)(em/d)^{dN}$. We can now apply Lemma 3 to bound the probability inside the expectation in the first term of (3). Setting

$$\epsilon_N = \sqrt{\frac{1}{2m}\left(dN\ln\left(\frac{em^2}{d}\right) + \ln\left(\frac{4e^8(N+1)}{\delta_N}\right)\right)}$$

and taking expectation with respect to $\mathcal{Q}$, we get that, with probability at least $1-\delta_N$, (5) holds for all θ. Proceeding as in the proof of Theorem 1, we get that, with probability at least $1-\delta$, for all $\theta > 0$ and $N \geq 1$,

$$\mathbf{P}_{\mathcal{D}}\left[yf(x) \leq 0\right] \leq \mathbf{P}_S\left[yf(x) \leq \theta\right] + 2e^{-N\theta^2/8} + \sqrt{\frac{1}{2m}\left(dN\ln\left(\frac{em^2}{d}\right) + \ln\frac{4e^8 N(N+1)^2}{\delta}\right)}.$$

Setting $N = \lceil (4/\theta^2)\ln(m/d)\rceil$ completes the proof. ∎

2.4 Sketch of a more general approach

Instead of the proof above, we can use a more general approach which can also be applied to any class of real-valued functions. The use of an approximating class, such as $\mathcal{C}_N$ in the proofs of Theorems 1 and 2, is central to our approach. We refer to such an approximating class as a sloppy cover. More formally, for a class $\mathcal{F}$ of real-valued functions, a training set S of size m, and positive real numbers θ and ϵ, we say that a function class $\hat{\mathcal{F}}$ is an *ϵ-sloppy θ-cover of $\mathcal{F}$ with respect to S* if, for all f in $\mathcal{F}$, there exists $\hat{f}$ in $\hat{\mathcal{F}}$ with $\mathbf{P}_{x\sim S}\left[|\hat{f}(x) - f(x)| > \theta\right] < \epsilon$. Let $\mathcal{N}(\mathcal{F}, \theta, \epsilon, m)$ denote the maximum, over all training sets S of size m, of the size of the smallest ϵ-sloppy θ-cover of $\mathcal{F}$ with respect to S. Standard techniques yield the following theorem (the proof is essentially identical to that of Theorem 2 in Bartlett [2]).

Theorem 4 *Let $\mathcal{F}$ be a class of real-valued functions defined on the instance space X. Let $\mathcal{D}$ be a distribution over $X \times \{-1, 1\}$, and let S be a sample of m examples chosen independently at random according to $\mathcal{D}$. Let $\epsilon > 0$ and let $\theta > 0$. Then the probability over the random choice of the training set S that there exists any function $f \in \mathcal{F}$ for which*

$$\mathbf{P}_{\mathcal{D}}\left[yf(x) \le 0\right] > \mathbf{P}_S\left[yf(x) \le \theta\right] + \epsilon$$

is at most

$$2\mathcal{N}(\mathcal{F}, \theta/2, \epsilon/8, 2m)\exp(-\epsilon^2 m/32).$$

Theorem 2 can now be proved by constructing a sloppy cover using the same probabilistic argument as in the proof of Theorems 1 and 2, i.e., by choosing an element of $\mathcal{C}_N$ randomly by sampling functions from $\mathcal{H}$. In addition, this result leads to a slight improvement (by log factors) of the main result of Bartlett [2], which gives bounds on generalization error for neural networks with real outputs in terms of the size of the network weights and the margin distribution.

3 The Effect of Boosting on Margin Distributions

We now give theoretical evidence that Freund and Schapire's [20] AdaBoost algorithm is especially suited to the task of maximizing the number of training examples with large margin.

We briefly review their algorithm. We adopt the notation used in the previous section, and restrict our attention to the binary case.

Boosting works by sequentially rerunning a base learning algorithm, each time using a different distribution over training examples. That is, on each round $t = 1, \ldots, T$, a distribution D_t is computed over the training examples, or, formally, over the set of indices $\{1, \ldots, m\}$. The goal of the base learning algorithm then is to find a classifier h_t with small error $\epsilon_t = \mathbf{P}_{i\sim D_t}\left[y_i \ne h_t(x_i)\right]$. The distribution used by AdaBoost is initially uniform ($D_1(i) = 1/m$), and then is updated multiplicatively on each round:

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

Here, $\alpha_t = \frac{1}{2}\ln((1-\epsilon_t)/\epsilon_t)$ and Z_t is a normalization factor chosen so that D_{t+1} sums to one. In our case, Z_t can be computed exactly:

$$Z_t \quad = \quad \sum_{i=1}^{m} D_t(i)\exp(-y_i\alpha_t h_t(x_i))$$

$$
\begin{aligned}
&= \sum_{i:y_i=h_t(x_i)} D_t(i)e^{-\alpha_t} + \sum_{i:y_i\neq h_t(x_i)} D_t(i)e^{\alpha_t} \\
&= (1-\epsilon_t)e^{-\alpha_t} + \epsilon_t e^{\alpha_t} \\
&= 2\sqrt{\epsilon_t(1-\epsilon_t)}.
\end{aligned}
$$

The final combined classifier is a weighted majority vote of the base classifiers, namely, sign(f) where

$$
f(x) = \frac{\sum_{t=1}^{T} \alpha_t h_t(x)}{\sum_{t=1}^{T} \alpha_t}. \tag{10}
$$

Note that, on round t, AdaBoost places the most weight on examples (x, y) for which $y\sum_{t'=1}^{t-1} \alpha_{t'} h_{t'}(x)$ is smallest. This quantity is exactly the margin of the combined classifier computed up to this point.

Freund and Schapire [20] prove that if the training error rates of all the base classifiers are bounded below $1/2$ for all D_t so that $\epsilon_t \leq 1/2 - \gamma$ for some $\gamma > 0$, then the training error of the combined classifier decreases exponentially fast with the number of base classifiers that are combined. The training error is equal to the fraction of training examples for which $yf(x) \leq 0$. It is a simple matter to extend their proof to show that, under the same conditions on ϵ_t, if θ is not too large, then the fraction of training examples for which $yf(x) \leq \theta$ also decreases to zero exponentially fast with the number of base classifiers (or boosting iterations).

Theorem 5 *Suppose the base learning algorithm, when called by AdaBoost, generates classifiers with weighted training errors* $\epsilon_1, \ldots, \epsilon_T$. *Then for any* θ, *we have that*

$$
\mathbf{P}_{(x,y)\sim S}\left[yf(x) \leq \theta\right] \leq 2^T \prod_{t=1}^{T} \sqrt{\epsilon_t^{1-\theta}(1-\epsilon_t)^{1+\theta}}. \tag{11}
$$

Proof: Note that if $yf(x) \leq \theta$ then

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

and so

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

Therefore,

$$
\begin{aligned}
\mathbf{P}_{(x,y)\sim S}\left[yf(x) \leq \theta\right] &\leq \mathbf{E}_{(x,y)\sim S}\left[\exp\left(-y\sum_{t=1}^{T} \alpha_t h_t(x) + \theta\sum_{t=1}^{T} \alpha_t\right)\right] \\
&= \frac{\exp\left(\theta\sum_{t=1}^{T} \alpha_t\right)}{m} \sum_{i=1}^{m} \exp\left(-y_i\sum_{t=1}^{T} \alpha_t h_t(x_i)\right) \\
&= \exp\left(\theta\sum_{t=1}^{T} \alpha_t\right)\left(\prod_{t=1}^{T} Z_t\right)\sum_{i=1}^{m} D_{T+1}(i)
\end{aligned}
$$

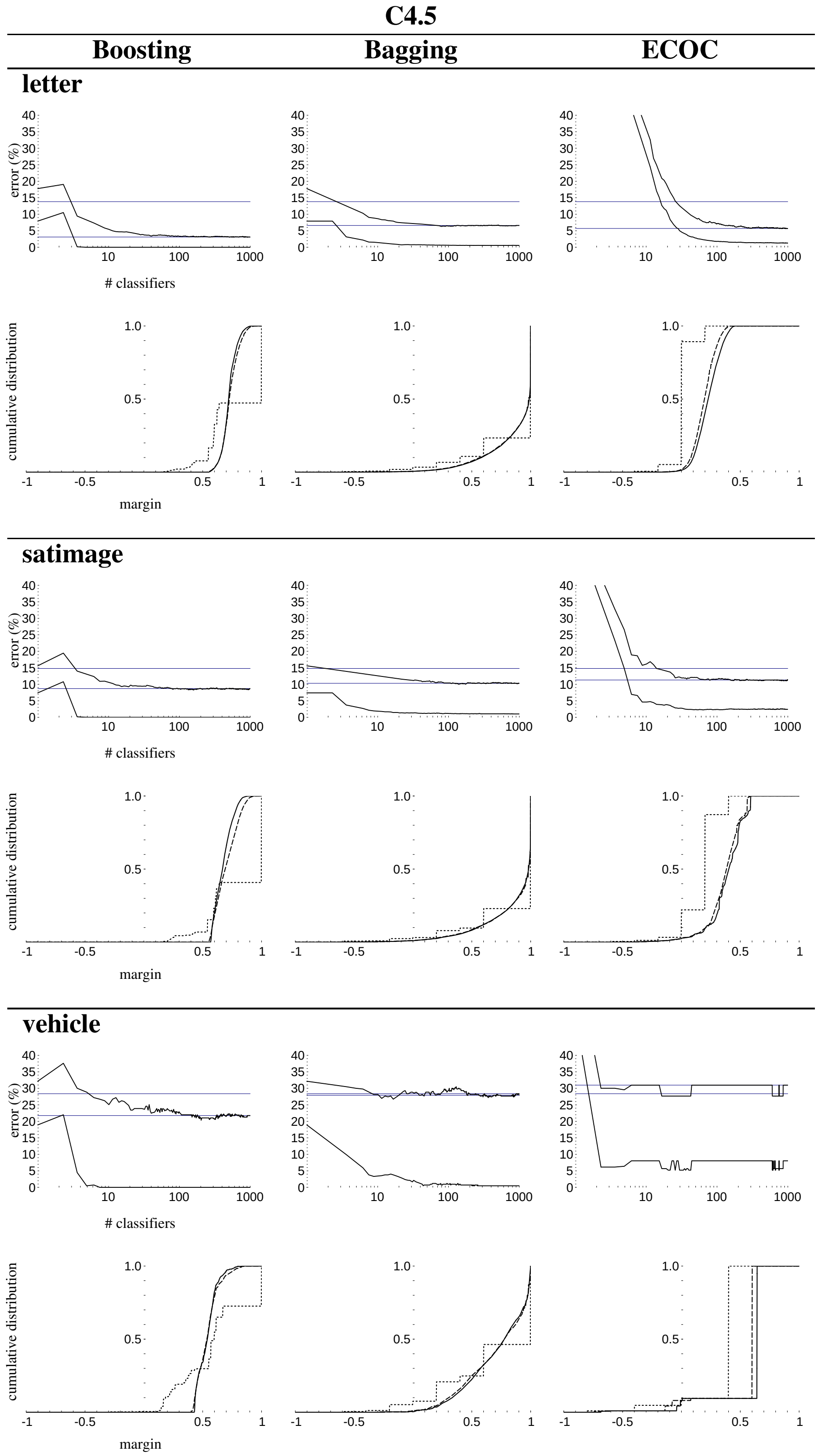

Figure 4: Error curves and margin distribution graphs for three voting methods (bagging, boosting and ECOC) using C4.5 as the base learning algorithm. Results are given for the letter, satimage and vehicle datasets. (See caption under Figure 1 for an explanation of these curves.)

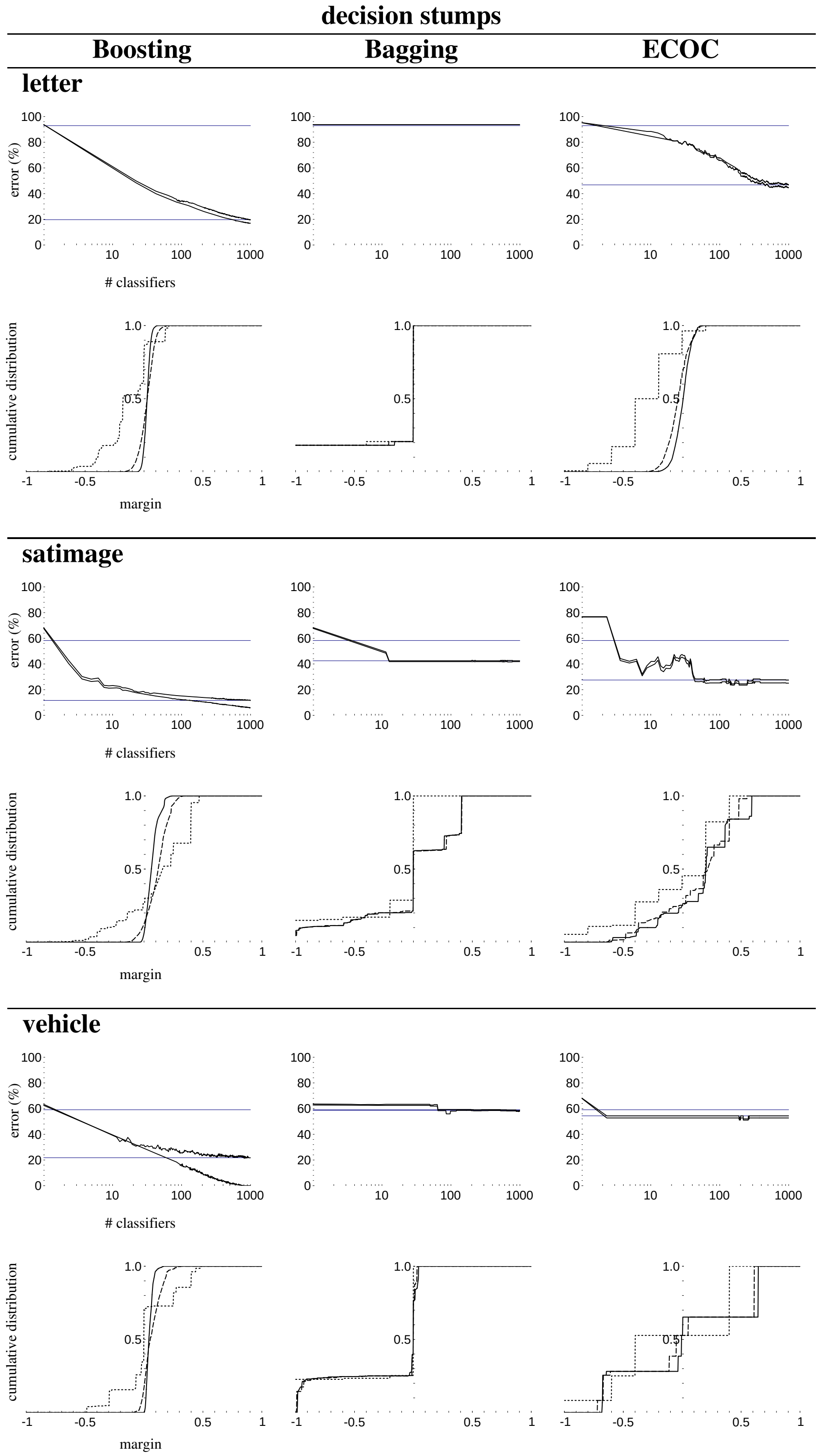

Figure 5: Error curves and margin distribution graphs for three voting methods (bagging, boosting and ECOC) using decision stumps as the base learning algorithm. Results are given for the letter, satimage and vehicle datasets. (See caption under Figure 1 for an explanation of these curves.)

where the last equality follows from the definition of D_{T+1}. Noting that $\sum_{i=1}^{m} D_{T+1}(i) = 1$, and plugging in the values of α_t and Z_t gives the theorem. ∎

To understand the significance of the result, assume for a moment that, for all t, $\epsilon_t \leq 1/2 - \gamma$ for some $\gamma > 0$. Since here we are considering only two-class prediction problems, a random prediction will be correct exactly half of the time. Thus, the condition that $\epsilon_t \leq 1/2 - \gamma$ for some small positive γ means that the predictions of the base classifiers are slightly better than random guessing. Given this assumption, we can simplify the upper bound in Equation (11) to:

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

If $\theta < \gamma$, it can be shown that the expression inside the parentheses is smaller than 1 so that the probability that $yf(x) \leq \theta$ decreases exponentially fast with T.[4] In practice, ϵ_t increases as a function of t, possibly even converging to $1/2$. However, if this increase is sufficiently slow the bound of Theorem 5 is still useful. Characterizing the conditions under which the increase is slow is an open problem.

Although this theorem applies only to binary classification problems, Freund and Schapire [20] and others [35, 36] give extensive treatment to the multiclass case (see also Section 4). All of their results can be extended to prove analogous theorems about margin distributions for this more general case.

4 More Margin Distribution Graphs

In this section, we describe experiments we conducted to produce a series of error curves and margin distribution graphs for a variety of datasets and learning methods.

Datasets. We used three benchmark datasets called "letter," "satimage" and "vehicle." Brief descriptions of these are given in Appendix B. Note that all three of these learning problems are multiclass with 26, 6 and 4 classes, respectively.

Voting methods. In addition to bagging and boosting, we used a variant of Dietterich and Bakiri's [13] method of error-correcting output codes (ECOC), which can be viewed as a voting method. This approach was designed to handle multiclass problems using only a two-class learning algorithm. Briefly, it works as follows: As in bagging and boosting, a given base learning algorithm (which need only be designed for two-class problems) is rerun repeatedly. However, unlike bagging and boosting, the examples are not reweighted or resampled. Instead, on each round, the labels assigned to each example are modified so as to create a new two-class labeling of the data which is induced by a simple mapping from the set of classes into $\{-1, +1\}$. The base learning algorithm is then trained using this relabeled data, generating a base classifier.

The sequence of bit assignments for each of the individual labels can be viewed as a "code word." A given test example is then classified by choosing the label whose associated code word is closest in Hamming distance to the sequence of predictions generated by the base classifiers. This coding-theoretic interpretation led Dietterich and Bakiri to the idea of choosing code words with strong error-correcting properties so that they will be as far apart from one another as possible. However, in our experiments, rather than carefully constructing error-correcting codes, we simply used random output codes which are highly likely to have similar properties.

[4] We can show that if γ is known in advance then an exponential decrease in the probability can be achieved (by a slightly different boosting algorithm) for any $\theta < 2\gamma$. However, we don't know how to achieve this improvement when no nontrivial lower bound on $1/2 - \epsilon_t$ is known a priori.

The ECOC combination rule can also be viewed as a voting method: Each base classifier h_t, on a given instance x, predicts a single bit $h_t(x) \in \{-1, +1\}$. We can interpret this bit as a single vote for each of the labels which were mapped on round t to $h_t(x)$. The combined hypothesis then predicts with the label receiving the most votes overall. Since ECOC is a voting method, we can measure margins just as we do for boosting and bagging.

As noted above, we used three multiclass learning problems in our experiments, whereas the version of boosting given in Section 3 only handles two-class data. Freund and Schapire [20] describe a straightforward adaption of this algorithm to the multiclass case. The problem with this algorithm is that it still requires that the accuracy of each base classifier exceed $1/2$. For two-class problems, this requirement is about as minimal as can be hoped for since random guessing will achieve accuracy $1/2$. However, for multiclass problems in which $k > 2$ labels are possible, accuracy $1/2$ may be much harder to achieve than the random-guessing accuracy rate of $1/k$. For fairly powerful base learners, such as C4.5, this does not seem to be a problem. However, the accuracy $1/2$ requirement can often be difficult for less powerful base learning algorithms which may be unable to generate classifiers with small training errors.

Freund and Schapire [20] provide one solution to this problem by modifying the form of the base classifiers and refining the goal of the base learner. In this approach, rather than predicting a single class for each example, the base classifier chooses a set of "plausible" labels for each example. For instance, in a character recognition task, the base classifier might predict that a particular example is either a "6," "8" or "9," rather than choosing just a single label. Such a base classifier is then evaluated using a "pseudoloss" measure which, for a given example, penalizes the base classifier (1) for *failing* to include the *correct* label in the predicted plausible label set, and (2) for each *in*correct label which *is* included in the plausible set. The combined classifier, for a given example, then chooses the single label which occurs most frequently in the plausible label sets chosen by the base classifiers (possibly giving more or less weight to some of the base classifiers). The exact form of the pseudoloss is under the control of the boosting algorithm, and the base learning algorithm must therefore be designed to handle changes in the form of the loss measure.

Base learning algorithms. In our experiments, for the base learning algorithm, we used C4.5. We also used a simple algorithm for finding the best single-node, binary-split decision tree (a decision "stump"). Since this latter algorithm is very weak, we used the "pseudoloss" versions of boosting and bagging, as described above. (See Freund and Schapire [20, 18] for details.)

Results. Figures 4 and 5 show error curves and margin distribution graphs for the three datasets, three voting methods and two base learning algorithms. Note that each figure corresponds only to a single run of each algorithm.

As explained in the introduction, each of the learning curve figures shows the training error (bottom) and test error (top) curves. We have also indicated as horizontal grid lines the error rate of the base classifier when run just once, as well as the error rate of the combined classifier after 1000 iterations. Note the log scale used in these figures. Margin distribution graphs are shown for 5, 100 and 1000 iterations indicated by short-dashed, long-dashed (sometimes barely visible) and solid curves, respectively.

It is interesting that, across datasets, all of the learning algorithms tend to produce margin distribution graphs of roughly the same character. As already noted, when used with C4.5, boosting is especially aggressive at increasing the margins of the examples, so much so that it is "willing" to suffer significant reductions in the margins of those examples that already have large margins. This can be seen in Figure 4, where we observe that the maximal margin in the final classifier is bounded well away from 1. Contrast this with the margin distribution graphs after 1000 iterations of bagging in which as many

		Kong & Dietterich [26] definitions								Breiman [8] definitions							
		stumps					C4.5			stumps					C4.5		
		error			pseudoloss		error			error			pseudoloss		error		
name		–	boost	bag	boost	bag	–	boost	bag	–	boost	bag	boost	bag	–	boost	bag
waveform	bias	26.0	3.8	22.8	0.8	11.9	1.5	0.5	1.4	19.2	2.6	15.7	0.5	7.9	0.9	0.3	1.4
	var	5.6	2.8	4.1	3.8	8.6	14.9	3.7	5.2	12.5	4.0	11.2	4.1	12.5	15.5	3.9	5.2
	error	44.7	19.6	39.9	17.7	33.5	29.4	17.2	19.7	44.7	19.6	39.9	17.7	33.5	29.4	17.2	19.7
twonorm	bias	2.5	0.6	2.0			0.5	0.2	0.5	1.3	0.3	1.1			0.3	0.1	0.3
	var	28.5	2.3	17.3			18.7	1.8	5.4	29.6	2.6	18.2			19.0	1.9	5.6
	error	33.3	5.3	21.7			21.6	4.4	8.3	33.3	5.3	21.7			21.6	4.4	8.3
threenorm	bias	24.5	6.3	21.6			4.7	2.9	5.0	14.2	4.1	13.8			2.6	1.9	3.1
	var	6.9	5.1	4.8			16.7	5.2	6.8	17.2	7.3	12.6			18.8	6.3	8.6
	error	41.9	22.0	36.9			31.9	18.6	22.3	41.9	22.0	36.9			31.9	18.6	22.3
ringnorm	bias	46.9	4.1	46.9			2.0	0.7	1.7	32.3	2.7	37.6			1.1	0.4	1.1
	var	–7.9	6.6	–7.1			15.5	2.3	6.3	6.7	8.0	2.2			16.4	2.6	6.9
	error	40.6	12.2	41.4			19.0	4.5	9.5	40.6	12.2	41.4			19.0	4.5	9.5
Kong &	bias	49.2	49.1	49.2	7.7	35.1	7.7	5.5	8.9	49.0	49.0	49.0	5.3	29.7	5.1	3.5	6.2
Dietterich	var	0.2	0.2	0.2	5.1	3.5	7.2	6.6	4.3	0.4	0.3	0.5	7.5	8.9	9.8	8.5	6.9
	error	49.5	49.3	49.5	12.8	38.6	14.9	12.1	13.1	49.5	49.3	49.5	12.8	38.6	14.9	12.1	13.1

Table 1: Results of bias-variance experiments using boosting and bagging on five synthetic datasets (described in Appendix B). For each dataset and each learning method, we estimated bias, variance and generalization error rate, reported in percent, using two sets of definitions for bias and variance (given in Appendix C). Both C4.5 and decision stumps were used as base learning algorithms. For stumps, we used both error-based and pseudoloss-based versions of boosting and bagging on problems with more than two classes. Columns labeled with a dash indicate that the base learning algorithm was run by itself.

as half of the examples have a margin of 1.

The graphs for ECOC with C4.5 resemble in shape those for boosting more so than bagging, but tend to have overall lower margins.

Note that, on every dataset, both boosting and bagging eventually achieve perfect or nearly perfect accuracy on the training sets (at least 99%), but the generalization error for boosting is better. The explanation for this is evident from the margin distribution graphs where we see that, for boosting, far fewer training examples have margin close to zero.

It should be borne in mind that, when combining decision trees, the complexity of the trees (as measured, say, by the number of leaves), may vary greatly from one combination method to another. As a result, the margin distribution graphs may not necessarily predict which method gives better generalization error. One must also always consider the complexity of the base classifiers, as explicitly indicated by Theorems 1 and 2.

When used with stumps, boosting can achieve training error much smaller than that of the base learner; however it is unable to achieve large margins. This is because, consistent with Theorem 5, the base classifiers have much higher training errors. Presumably, such low margins do not adversely affect the generalization error because the complexity of decision stumps is so much smaller than that of full decision trees.

5 Relation to Bias-variance Theory

One of the main explanations for the improvements achieved by voting classifiers is based on separating the expected error of a classifier into a *bias* term and a *variance* term. While the details of these definitions differ from author to author [8, 25, 26, 40], they are all attempts to capture the following quantities: The bias term measures the *persistent* error of the learning algorithm, in other words, the error that would remain even if we had an infinite number of independently trained classifiers. The variance term measures the error that is due to *fluctuations* that are a part of generating a single classifier.

The idea is that by averaging over many classifiers one can reduce the variance term and in that way reduce the expected error.

In this section, we discuss a few of the strengths and weaknesses of bias-variance theory as an explanation for the performance of voting methods, especially boosting.

5.1 The bias-variance decomposition for classification.

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

One would naturally hope that this beautiful analysis would carry over from quadratic regression to classification. Unfortunately, as has been observed before us, (see, for instance, Friedman [22]) taking the majority vote over several classification rules can sometimes result in an *increase* in the expected classification error. This simple observation suggests that it may be inherently more difficult or even impossible to find a bias-variance decomposition for classification as natural and satisfying as in the quadratic regression case.

This difficulty is reflected in the myriad definitions that have been proposed for bias and variance [8, 25, 26, 40]. Rather than discussing each one separately, for the remainder of this section, except where noted, we follow the definitions given by Kong and Dietterich [26], and referred to as "Definition 0" by Breiman [8]. (These definitions are given in Appendix C.)

5.2 Bagging and variance reduction.

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

This can happen even when the data distribution is very simple. As a somewhat contrived example, consider data generated according to the following distribution. The label $y \in \{-1, +1\}$ is chosen uniformly at random. The instance $x \in \{-1, +1\}^7$ is then chosen by picking each of the 7 bits to be equal to y with probability 0.9 and $-y$ with probability 0.1. Thus, each coordinate of x is an independent noisy version of y. For our base learner, we use a learning algorithm which generates a classifier that is equal to the single coordinate of x which is the best predictor of y with respect to the training set. It is clear that each coordinate of x has the same probability of being chosen as the classifier on a random training set, so the aggregate predictor over many independently trained samples is the unweighted majority vote over the coordinates of x, which is also the Bayes optimal predictor in this case. Thus, the bias of our learning algorithm is exactly zero. The prediction error of the majority rule is roughly 0.3%, and so a variance of about 9.7% strongly dominates the expected error rate of 10%. In such a favorable case, one would predict, according to the bias-variance explanation, that bagging could get close to the error of the Bayes optimal predictor.

However, using a training set of 500 examples, the generalization error achieved by bagging is 5.6% after 200 iterations. (All results are averaged over many runs.) The reason for this poor performance is that, in any particular random sample, some of the coordinates of x are slightly more correlated with

y and bagging tends to pick these coordinates much more often than the others. Thus, in this case, the behavior of bagging is very different from its expected behavior on truly independent training sets.

Boosting, on the same data, achieved a test error of 0.6%.

5.3 Boosting and variance reduction.

Breiman [8] argued that boosting is primarily a variance-reducing procedure. Some of the evidence for this comes from the observed effectiveness of boosting when used with C4.5 or CART, algorithms known empirically to have high variance. As the error of these algorithms is mostly due to variance, it is not surprising that the reduction in the error is primarily due to a reduction in the variance. However, our experiments show that boosting can also be highly effective when used with learning algorithms whose error tends to be dominated by bias rather than variance.[5]

We ran boosting and bagging on four artificial datasets described by Breiman [8], as well as the artificial problem studied by Kong and Dietterich [26]. Following previous authors, we used training sets of size 200 for the latter problem and 300 for the others. For the base learning algorithm, we tested C4.5. We also used the decision-stump base-learning algorithm described in Section 4. We then estimated bias, variance and average error of these algorithms by rerunning them 1000 times each, and evaluating them on a test set of 10,000 examples. For these experiments, we used both the bias-variance definitions given by Kong and Dietterich [26] and those proposed more recently by Breiman [8]. (Definitions are given in Appendix C.) For multiclass problems, following Freund and Schapire [18], we tested both error-based and pseudoloss-based versions of bagging and boosting. For two-class problems, only the error-based versions were used.

The results are summarized in Table 1. Clearly, boosting is doing more than reducing variance. For instance, on "ringnorm," boosting decreases the overall error of the stump algorithm from 40.6% to 12.2%, but actually increases the variance from -7.9% to 6.6% using Kong and Dietterich's definitions, or from 6.7% to 8.0% using Breiman's definitions. (We did not check the statistical significance of this increase.)

Breiman also tested boosting with a low-variance base learning algorithm—namely, linear discriminant analysis (LDA)—and attributed the ineffectiveness of boosting in this case to the "stability" (low variance) of LDA. The experiments with the fairly stable stump algorithm suggest that stability in itself may not be sufficient to predict boosting's failure.

Our theory suggests a different characterization of the cases in which boosting might fail. Taken together, Theorem 1 and Theorem 5 state that boosting can perform poorly only when either (1) there is insufficient training data relative to the "complexity" of the base classifiers, or (2) the training errors of the base classifiers (the ϵ_t's in Theorem 5) become too large too quickly. Certainly, this characterization is incomplete in that boosting often succeeds even in situations in which the theory provides no guarantees. However, while we hope that tighter bounds can be given, it seems unlikely that there exists a "perfect" theory. By a "perfect" theory we mean here a rigorous analysis of voting methods that, on the one hand, is general enough to apply to *any* base learning algorithm and to *any* i.i.d. source of labeled instances and on the other hand gives bounds that are accurate predictors of the performance of the algorithm in practice. This is because in any practical situation there is structure in the data and in the base learning algorithm that is not taken into account in the assumptions of a general theory.

[5] In fact, the original goal of boosting was to reduce the error of so-called "weak" learning algorithms which tend to have very large bias. [17, 20, 34]

5.4 Why averaging can increase complexity

In this section, we challenge a common intuition which says that when one takes the majority vote over several base classifiers the generalization error of the resulting classifier is likely to be lower than the average generalization error of the base classifiers. In this view, voting is seen as a method for "smoothing" or "averaging" the classification rule. This intuition is sometimes based on the bias-variance analysis of regression described in the previous section. Also, to some, it seems to follow from a Bayesian point of view according to which integrating the classifications over the posterior is better than using any single classifier. If one feels comfortable with these intuitions, there seems to be little point to most of the analysis given in this paper. It seems that because AdaBoost generates a majority vote over several classifiers, its generalization error is, in general, likely to be better than the average generalization error of the base classifiers. According to this point of view, the suggestion we make in the introduction that the majority vote over many classifiers is more complex than any single classifier seems to be irrelevant and misled.

In this section, we describe a base learning algorithm which, when combined using AdaBoost, is likely to generate a majority vote over base classifiers whose training error goes to zero, while at the same time the generalization error does not improve at all. In other words, it is a case in which voting results in over-fitting. This is a case in which the intuition described above seems to break down, while the margin-based analysis developed in this paper gives the correct answer.

Suppose we use classifiers that are delta-functions, i.e., they predict $+1$ on a single point in the input space and -1 everywhere else, or vice versa (-1 on one point and $+1$ elsewhere). (If you dislike delta-functions, you can replace them with nicer functions. For example, if the input space is $\mathbb{R}^n$, use balls of sufficiently small radius and make the prediction $+1$ or -1 inside, and -1 or $+1$, respectively, outside.) To this class of functions we add the constant functions that are -1 everywhere or $+1$ everywhere.

Now, for any training sample of size m we can easily construct a set of at most $2m$ functions from our class such that the majority vote over these functions will always be correct. To do this, we associate one delta function with each training example; the delta function gives the correct value on the training example and the opposite value everywhere else. Letting m_+ and m_- denote the number of positive and negative examples, we next add m_+ copies of the function which predicts $+1$ everywhere, and m_- copies of the function which predicts -1 everywhere. It can now be verified that the sum (majority vote) of all these functions will be positive on all of the positive examples in the training set, and negative on all the negative examples. In other words, we have constructed a combined classifier which exactly fits the training set.

Fitting the training set seems like a good thing; however, the very fact that we can easily fit such a rule to *any* training set implies that we don't expect the rule to be very good on independently drawn points outside of the training set. In other words, the complexity of these average rules is too large, relative to the size of the sample, to make them useful. Note that this complexity is the result of averaging. Each one of the delta rules is very simple (the VC-dimension of this class of functions is exactly 2), and indeed, if we found a single delta function (or constant function) that fit a large sample we could, with high confidence, expect the rule to be correct on new randomly drawn examples.

How would boosting perform in this case? It can be shown using Theorem 5 (with $\theta = 0$) that boosting would slowly but surely find a combination of the type described above having zero training error but very bad generalization error. A margin-based analysis of this example shows that while all of the classifications are correct, they are correct only with a tiny margin of size $O(1/m)$, and so we cannot expect the generalization error to be very good.

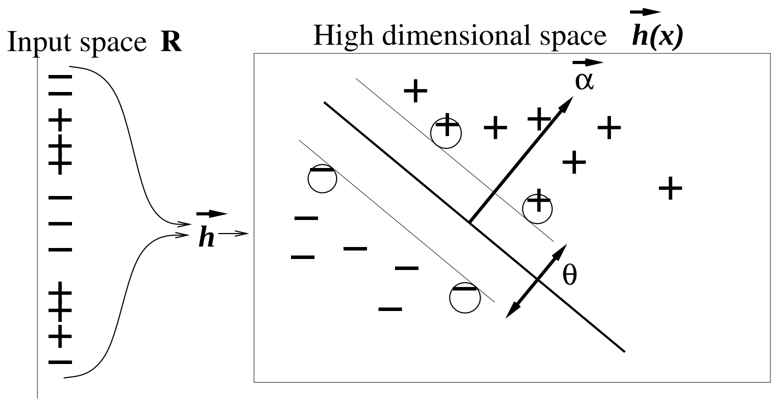

Figure 6: The maximal margins classification method. In this example, the raw data point x is an element of $\mathbb{R}$, but in that space the positive and negative examples are not linearly separable. The raw input is mapped to a point in a high dimensional space (here $\mathbb{R}^2$) by a fixed nonlinear transformation $\vec{h}$. In the high dimensional space, the classes are linearly separable. The vector $\vec{\alpha}$ is chosen to maximize the minimal margin θ. The circled instances are the *support vectors*; Vapnik shows that $\vec{\alpha}$ can always be written as a linear combination of the support vectors.

6 Relation to Vapnik's Maximal Margin Classifiers

The use of the margins of real-valued classifiers to predict generalization error was previously studied by Vapnik [42] in his work with Boser and Guyon [5] and Cortes [10] on optimal margin classifiers.

We start with a brief overview of optimal margin classifiers. One of the main ideas behind this method is that some nonlinear classifiers on a low dimensional space can be treated as linear classifiers over a high dimensional space. For example, consider the classifier that labels an instance $x \in \mathbb{R}$ as $+1$ if $2x^5 - 5x^2 + x > 10$ and -1 otherwise. This classifier can be seen as a linear classifier if we represent each instance by the vector $\vec{h}(x) \doteq (1, x, x^2, x^3, x^4, x^5)$. If we set $\vec{\alpha} = (-10, 1, -5, 0, 0, 2)$ then the classification is $+1$ when $\vec{\alpha} \cdot \vec{h}(x) > 0$ and -1 otherwise. In a typical case, the data consists of about $10,000$ instances in $\mathbb{R}^{100}$ which are mapped into $\mathbb{R}^{1,000,000}$. Vapnik introduced the method of *kernels* which provides an efficient way for calculating the predictions of linear classifiers in the high dimensional space. Using kernels, it is usually easy to find a linear classifier that separates the data perfectly. In fact, it is likely that there are *many* perfect linear classifiers, many of which might have very poor generalization ability. In order to overcome this problem, the prescription suggested by Vapnik is to find the classifier that maximizes the minimal margin. More precisely, suppose that the training sample S consists of pairs of the form (x, y) where x is the instance and $y \in \{-1, +1\}$ is its label. Assume that $\vec{h}(x)$ is some fixed nonlinear mapping of instances into $\mathbb{R}^n$ (where n is typically very large). Then the maximal margin classifier is defined by the vector $\vec{\alpha}$ which maximizes

$$\min_{(x,y)\in S} \frac{y(\vec{\alpha} \cdot \vec{h}(x))}{||\vec{\alpha}||_2}. \tag{12}$$

Here, $||\vec{\alpha}||_2$ is the l_2 or Euclidean norm of the vector $\vec{\alpha}$. A graphical sketch of the maximal margin method is given in Figure 6. For the analysis of this method, Vapnik assumes that all of the vectors $\vec{h}(x)$ are enclosed within a ball of radius R, i.e., they all are within Euclidean distance R from some fixed vector in $\mathbb{R}^n$. Without loss of generality, we can assume that $R = 1$.

Vapnik [42] showed that the VC dimension of all linear classifiers with minimum margin at least θ is upper bounded by $1/\theta^2$. This result implies bounds on the generalization error in terms of the expected minimal margin on test points which do not depend on the dimension n of the space into which the data are mapped. However, typically, the expected value of the minimal margin is not known. Shawe-Taylor et al. [38] used techniques from the theory of learning real-valued functions to give bounds on generalization error in terms of margins on the *training examples*. Shawe-Taylor et al. [39] also gave related results for arbitrary real classes.

Consider the relation between Equation (10) and the argument of the minimum in Equation (12). We can view the coefficients $\{\alpha_t\}_{t=1}^T$ as the coordinates of a vector $\vec{\alpha} \in \mathbb{R}^T$ and the predictions $\{h_t(x)\}_{t=1}^T$ as the coordinates of the vector $\vec{h}(x) \in \{-1,+1\}^T$. Then we can rewrite Equation (10) as

$$f(x) = \frac{\vec{\alpha} \cdot \vec{h}(x)}{||\vec{\alpha}||_1} ,$$

where $||\vec{\alpha}||_1 = \sum_{t=1}^T |\alpha_t|$ is the l_1 norm of $\vec{\alpha}$. In our analysis, we use the fact that all of the components of $\vec{h}(x)$ are in the range $[-1,+1]$, or, in other words that the *max* or l_∞ norm of $\vec{h}(x)$ is bounded by 1: $||\vec{h}(x)||_\infty = \max_{t=1}^T |h_t(x)| \leq 1$.

Viewed this way, the connection between maximal margin classifiers and boosting becomes clear. Both methods aim to find a linear combination in a high dimensional space which has a large margin on the instances in the sample. The norms used to define the margin are different in the two cases and the precise goal is also different—maximal margin classifiers aim to maximize the minimal margin while boosting aims to minimize an exponential weighting of the examples as a function of their margins. Our interpretation for these differences is that boosting is more suited for the case when the mapping $\vec{h}$ maps x into a high dimensional space where all of the coordinates have a similar maximal range, such as $\{-1,+1\}$. On the other hand, the optimal margin method is suitable for cases in which the *Euclidean* norm of $\vec{h}(x)$ is likely to be small, such as is the case when $\vec{h}$ is an orthonormal transformation between inner-product spaces. Related to this, the optimal margin method uses quadratic programming for its optimization, whereas the boosting algorithm can be seen as a method for approximate linear programming [7, 19, 21, 23].

Both boosting and support vector machines aim to find a linear classifier in a very high dimensional space. However, computationally, they are very different: support vector machines use the method of kernels to perform computations in the high dimensional space while boosting relies on a base learning algorithm which explores the high dimensional space one coordinate at a time.

Vapnik [42] gave an alternative analysis of optimal margin classifiers, based on the number of *support vectors*, i.e., the number of examples that define the final classifier. This analysis is preferable to the analysis that depends on the *size* of the margin when only a few of the training examples are support vectors. Previous work [17] has suggested that boosting also can be used as a method for selecting a small number of "informative" examples from the training set. Investigating the relevance of this type of bound when applying boosting to real-world problems is an interesting open research direction.

7 Other loss functions

We describe briefly related work which has been done on loss functions other than the 0-1 loss.

For quadratic loss, Jones [24] and Barron [1] have shown that functions in the convex hull of a class $\mathcal{H}$ of real-valued functions can be approximated by a convex combination of N elements of $\mathcal{H}$ to an accuracy of $O(1/N)$ by iteratively adding the member of $\mathcal{H}$ which minimizes the residual error to the existing convex combination. Lee, Bartlett and Williamson [27] extended this result to show that the procedure will converge to the best approximation in the convex hull of $\mathcal{H}$ even when the target function is not in the convex hull of $\mathcal{H}$. Lee, Bartlett and Williamson [27, 28] also studied the generalization error when this procedure is used for learning. In results analogous to those presented here, they showed that the generalization error can be bounded in terms of the sum of the absolute values of the output weights (when the members of $\mathcal{H}$ are normalized to have output values in the interval $[-1, 1]$), rather than in terms of the number of components in the convex combination.

Similar work on iterative convex approximation in L_p spaces was presented by Donahue et al. [14]. To the best of our knowledge, similar iterative schemes for combining functions have not been studied for the log loss.

Extensions of boosting to solve regression problems have been suggested by Freund [17] and Freund and Schapire [20]. These extensions are yet to be tested in practice. Drucker [15] experimented with a different extension of boosting for regression and reported some encouraging results.

8 Open Problems

The methods in this paper allow us to upper bound the generalization error of a voted classifier based on simple statistics which can be measured using the training data. These statistics are a function of the empirical distribution of the margins. While our bounds seem to explain the experiments qualitatively, their quantitative predictions are greatly over pessimistic. The challenge of coming up with better bounds can be divided into two questions. First, can one give better bounds that are a function of the empirical margins distribution? Second, are there better bounds that are functions of other statistics?

A different approach to understanding the behavior of AdaBoost is to find functions of the training set which predict the generalization error well on all or most of the datasets encountered in practice. While this approach does not give one the satisfaction of a mathematical proof, it might yield good results in practice.

Acknowledgments

Many thanks to Leo Breiman for a poignant email exchange which challenged us to think harder about these problems. Thanks also to all those who contributed to the datasets used in this paper, and to the three anonymous reviewers for many helpful criticisms.

References

[1] Andrew R. Barron. Universal approximation bounds for superposition of a sigmoidal function. *IEEE Transactions on Information Theory*, 39(3):930–945, 1993.

[2] Peter L. Bartlett. The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. *IEEE Transactions on Information Theory*, 1998 (to appear).

[3] Eric Bauer and Ron Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. Unpublished manuscript, 1997.

[4] Eric B. Baum and David Haussler. What size net gives valid generalization? *Neural Computation*, 1(1):151–160, 1989.

[5] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*, pages 144–152, 1992.

[6] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[7] Leo Breiman. Prediction games and arcing classifiers. Technical Report 504, Statistics Department, University of California at Berkeley, 1997.

[8] Leo Breiman. Arcing classifiers. *Annals of Statistics*, to appear.

[9] Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification and Regression Trees*. Wadsworth International Group, 1984.

[10] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, September 1995.

[11] Luc Devroye. Bounds for the uniform deviation of empirical measures. *Journal of Multivariate Analysis*, 12:72–79, 1982.

[12] Thomas G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. Unpublished manuscript, 1998.

[13] Thomas G. Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, January 1995.

[14] M. J. Donahue, L. Gurvits, C. Darken, and E. Sontag. Rates of convex approximation in non-Hilbert spaces. *Constructive Approximation*, 13:187–220, 1997.

[15] Harris Drucker. Improving regressors using boosting techniques. In *Machine Learning: Proceedings of the Fourteenth International Conference*, pages 107–115, 1997.

[16] Harris Drucker and Corinna Cortes. Boosting decision trees. In *Advances in Neural Information Processing Systems 8*, pages 479–485, 1996.

[17] Yoav Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121(2):256–285, 1995.

[18] Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *Machine Learning: Proceedings of the Thirteenth International Conference*, pages 148–156, 1996.

[19] Yoav Freund and Robert E. Schapire. Game theory, on-line prediction and boosting. In *Proceedings of the Ninth Annual Conference on Computational Learning Theory*, pages 325–332, 1996.

[20] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, August 1997.

[21] Yoav Freund and Robert E. Schapire. Adaptive game playing using multiplicative weights. *Games and Economic Behavior*, (to appear).

[22] Jerome H. Friedman. On bias, variance, 0/1-loss, and the curse-of-dimensionality. Available electronically from http://stat.stanford.edu/∼jhf.

[23] Adam J. Grove and Dale Schuurmans. Boosting in the limit: Maximizing the margin of learned ensembles. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, 1998.

[24] Lee K. Jones. A simple lemma on greedy approximation in Hilbert space and convergence rates for projection pursuit regression and neural network training. *Annals of Statistics*, 20(1):608–613, 1992.

[25] Ron Kohavi and David H. Wolpert. Bias plus variance decomposition for zero-one loss functions. In *Machine Learning: Proceedings of the Thirteenth International Conference*, pages 275–283, 1996.

[26] Eun Bae Kong and Thomas G. Dietterich. Error-correcting output coding corrects bias and variance. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 313–321, 1995.

[27] Wee Sun Lee, Peter L. Bartlett, and Robert C. Williamson. Efficient agnostic learning of neural networks with bounded fan-in. *IEEE Transactions on Information Theory*, 42(6):2118–2132, 1996.

[28] Wee Sun Lee, Peter L. Bartlett, and Robert C. Williamson. The importance of convexity in learning with squared loss. *IEEE Transactions on Information Theory*, to appear.

[29] Richard Maclin and David Opitz. An empirical evaluation of bagging and boosting. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, pages 546–551, 1997.

[30] C. J. Merz and P. M. Murphy. UCI repository of machine learning databases, 1998. http://www.ics.uci.edu/∼mlearn/MLRepository.html.

[31] J. R. Quinlan. Bagging, boosting, and C4.5. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 725–730, 1996.

[32] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[33] N. Sauer. On the density of families of sets. *Journal of Combinatorial Theory Series A*, 13:145–147, 1972.

[34] Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990.

[35] Robert E. Schapire. Using output codes to boost multiclass learning problems. In *Machine Learning: Proceedings of the Fourteenth International Conference*, pages 313–321, 1997.

[36] Robert E. Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, 1998.

[37] Holger Schwenk and Yoshua Bengio. Training methods for adaptive boosting of neural networks for character recognition. In *Advances in Neural Information Processing Systems 10*, 1998.

[38] John Shawe-Taylor, Peter L. Bartlett, Robert C. Williamson, and Martin Anthony. A framework for structural risk minimisation. In *Proceedings of the Ninth Annual Conference on Computational Learning Theory*, pages 68–76, 1996.

[39] John Shawe-Taylor, Peter L. Bartlett, Robert C. Williamson, and Martin Anthony. Structural risk minimization over data-dependent hierarchies. Technical Report NC-TR-96-053, Neurocolt, 1996.

[40] Robert Tibshirani. Bias, variance and prediction error for classification rules. Technical report, University of Toronto, November 1996.

[41] V. N. Vapnik and A. Ya. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its applications*, XVI(2):264–280, 1971.

[42] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.

A Generalization Error for Multiclass Problems

In this appendix, we describe how Theorems 1 and 2 can be extended to multiclass problems.

Suppose there are k classes, and define $Y = \{1, 2, \ldots, k\}$ as the output space. We formally view the base classifiers $h \in \mathcal{H}$ as mappings from $X \times Y$ to $\{0, 1\}$, with the interpretation that if $h(x, y) = 1$ then y is predicted by h to be a plausible label for x. This general form of classifier covers the forms of base classifiers used throughout this paper. For simple classifiers, like the decision trees computed by C4.5, only a single label is predicted so that, for each x, $h(x, y) = 1$ for exactly one label y. However, some of the other combination methods — such as pseudoloss-based boosting or bagging, as well as Dietterich and Bakiri's output-coding method — use base classifiers which vote for a *set* of plausible labels so that $h(x, y)$ may be 1 for several labels y.

We define the convex hull $\mathcal{C}$ of $\mathcal{H}$ as

$$\mathcal{C} \doteq \left\{ f : (x, y) \mapsto \sum_{h \in \mathcal{H}} a_h h(x, y) \,\middle|\, a_h \geq 0; \ \sum_h a_h = 1 \right\},$$

so a classifier f in $\mathcal{C}$ predicts label y for input x if $f(x, y) > \max_{y' \neq y} f(x, y')$ (and ties are broken arbitrarily). We define the margin of an example (x, y) for such a function f as

$$\text{margin}(f, x, y) = f(x, y) - \max_{y' \neq y} f(x, y'). \tag{13}$$

Clearly, f gives the wrong prediction on (x, y) only if margin$(f, x, y) \leq 0$. With these definitions, we have the following generalization of Theorems 1 and 2.

Theorem 6 *Let $\mathcal{D}$ be a distribution over $X \times Y$, and let S be a sample of m examples chosen independently at random according to $\mathcal{D}$. Assume that the base-classifier space $\mathcal{H}$ is finite, and let $\delta > 0$. Then with probability at least $1 - \delta$ over the random choice of the training set S, every function $f \in \mathcal{C}$ satisfies the following bound for all $\theta > 0$:*

$$\mathbf{P}_{\mathcal{D}}\left[\text{margin}(f, x, y) \leq 0\right] \leq \mathbf{P}_S\left[\text{margin}(f, x, y) \leq \theta\right] + O\left(\frac{1}{\sqrt{m}}\left(\frac{\log(mk)\log|\mathcal{H}|}{\theta^2} + \log(1/\delta)\right)^{1/2}\right).$$

More generally, for finite or infinite $\mathcal{H}$ with VC-dimension d, the following bound holds as well, assuming that $m \geq d \geq 1$:

$$\mathbf{P}_{\mathcal{D}}\left[\text{margin}(f, x, y) \leq 0\right] \leq \mathbf{P}_S\left[\text{margin}(f, x, y) \leq \theta\right] + O\left(\frac{1}{\sqrt{m}}\left(\frac{d\log^2(mk/d)}{\theta^2} + \log(1/\delta)\right)^{1/2}\right).$$

Proof: The proof closely follows that of Theorem 1, so we only describe the differences. We first consider the case of finite $\mathcal{H}$.

First, we define

$$\mathcal{C}_N \doteq \left\{ f : (x, y) \mapsto \frac{1}{N}\sum_{i=1}^{N} h_i(x, y) \middle| h_i \in \mathcal{H} \right\}.$$

As in the proof of Theorem 1, for any $f \in \mathcal{C}$ we choose an approximating function $g \in \mathcal{C}_N$ according to the distribution $\mathcal{Q}$, and we have

$$\begin{aligned}\mathbf{P}_{\mathcal{D}}\left[\text{margin}(f, x, y) \leq 0\right] \quad \leq \quad & \mathbf{E}_{g\sim\mathcal{Q}}\left[\mathbf{P}_{\mathcal{D}}\left[\text{margin}(g, x, y) \leq \theta/2\right]\right] \\ & + \mathbf{E}_{\mathcal{D}}\left[\mathbf{P}_{g\sim\mathcal{Q}}\left[\text{margin}(g, x, y) > \theta/2 \mid \text{margin}(f, x, y) \leq 0\right]\right].\end{aligned}$$

We bound the second term of the right hand side as follows: Fix f, x and y, and let $y' \neq y$ achieve the maximum in Equation (13) so that

$$\text{margin}(f, x, y) = f(x, y) - f(x, y').$$

Clearly, margin$(g, x, y) \leq g(x, y) - g(x, y')$ and

$$\mathbf{E}_{g\sim\mathcal{Q}}\left[g(x, y) - g(x, y')\right] = f(x, y) - f(x, y')$$

so

$$\begin{aligned}&\mathbf{P}_{g\sim\mathcal{Q}}\left[\text{margin}(g, x, y) > \theta/2 \mid \text{margin}(f, x, y) \leq 0\right] \\ &\quad \leq \quad \mathbf{P}_{g\sim\mathcal{Q}}\left[g(x, y) - g(x, y') > \theta/2 \mid f(x, y) - f(x, y') \leq 0\right] \\ &\quad \leq \quad e^{-N\theta^2/8}\end{aligned}$$

using the Chernoff bound.

Equations (5) and (6) follow exactly as in the proof of Theorem 1 with $yf(x)$ and $yg(x)$ replaced by margin(f, x, y) and margin(g, x, y). We can derive the analog of Equation (7) as follows:

$$\begin{aligned}
&\mathbf{P}_{g\sim\mathcal{Q}}\left[\text{margin}(g, x, y) \le \theta/2 \mid \text{margin}(f, x, y) > \theta\right] \\
&\quad = \mathbf{P}_{g\sim\mathcal{Q}}\left[\exists y' \ne y : g(x, y) - g(x, y') \le \theta/2 \mid \forall y' \ne y : f(x, y) - f(x, y') > \theta\right] \\
&\quad \le \sum_{y' \ne y} \mathbf{P}_{g\sim\mathcal{Q}}\left[g(x, y) - g(x, y') \le \theta/2 \mid f(x, y) - f(x, y') > \theta\right] \\
&\quad \le (k-1)e^{-N\theta^2/8}\,.
\end{aligned}$$

Proceeding as in the proof of Theorem 1, we see that, with probability at least $1 - \delta$, for any $\theta > 0$ and $N \ge 1$,

$$\mathbf{P}_{\mathcal{D}}\left[\text{margin}(f, x, y) \le 0\right] \le \mathbf{P}_S\left[\text{margin}(f, x, y) \le \theta\right] + ke^{-N\theta^2/8} + \sqrt{\frac{1}{2m}\ln\left(\frac{N(N+1)^2|\mathcal{H}|^N}{\delta}\right)}.$$

Setting $N = \lceil (4/\theta^2)\ln(mk^2/\ln|\mathcal{H}|)\rceil$ gives the result.

For infinite $\mathcal{H}$, we follow essentially the same modifications to the argument above as used in the proof of Theorem 2. As before, to apply Lemma 3, we need to derive an upper bound on $s(\mathcal{A}, m)$ where

$$\mathcal{A} = \{\{(x, y) \in X \times Y : \text{margin}(g, x, y) > \theta/2\} : g \in \mathcal{C}_N,\, \theta > 0\}\,.$$

Let $x_1, \ldots, x_m \in X$ and $y_1, \ldots, y_m \in Y$. Then applying Sauer's lemma to the set $\{(x_i, y) : 1 \le i \le m, y \in Y\}$ gives

$$|\{\langle h(x_1, 1), \ldots, h(x_1, k); \ldots; h(x_m, 1), \ldots, h(x_m, k)\rangle : h \in \mathcal{H}\}| \le \sum_{i=0}^{d}\binom{km}{i} \le \left(\frac{emk}{d}\right)^d.$$

This implies that

$$|\{\langle g(x_1, 1), \ldots, g(x_1, k); \ldots; g(x_m, 1), \ldots, g(x_m, k)\rangle : g \in \mathcal{C}_N\}| \le \left(\frac{emk}{d}\right)^{dN},$$

and hence

$$|\{(\text{margin}(g, x_1, y_1), \ldots, \text{margin}(g, x_m, y_m)) : g \in \mathcal{C}_N\}| \le \left(\frac{emk}{d}\right)^{dN}.$$

Thus, $s(\mathcal{A}, m) \le (N+1)(emk/d)^{dN}$. Proceeding as before, we obtain the bound

$$\begin{aligned}
\mathbf{P}_{\mathcal{D}}\left[\text{margin}(f, x, y) \le 0\right] \le\ & \mathbf{P}_S\left[\text{margin}(f, x, y) \le \theta\right] + ke^{-N\theta^2/8} \\
& + \sqrt{\frac{1}{2m}\left(dN\ln\left(\frac{em^2k}{d}\right) + \ln\frac{4e^8N(N+1)^2}{\delta}\right)}.
\end{aligned}$$

Setting N as above completes the proof. ∎

B Brief Descriptions of Datasets

In this appendix, we briefly describe the datasets used in our experiments.

name	# examples train	test	# classes	# features
vehicle	423	423	4	18
satimage	4435	2000	6	36
letter	16000	4000	26	16

Table 2: The three benchmark machine-learning problems used in the experiments.

B.1 Non-synthetic datasets

In Section 4, we conducted experiments on three non-synthetic datasets called "letter," "satimage" and "vehicle." All three are available from the repository at the University of California at Irvine [30].

Some of the basic characteristics of these datasets are given in Table 2. The letter and satimage datasets came with their own test sets. For the vehicle dataset, we randomly selected half of the data to be held out as a test set. All features are continuous (real-valued). None of these datasets have missing values.

The letter benchmark is a letter image recognition task. The dataset was created by David J. Slate. According to the documentation provided with this dataset, "The objective is to identify each of a large number of black-and-white rectangular pixel displays as one of the 26 capital letters in the English alphabet. The character images were based on 20 different fonts and each letter within these 20 fonts was randomly distorted to produce a file of 20,000 unique stimuli. Each stimulus was converted into 16 primitive numerical attributes (statistical moments and edge counts) which were then scaled to fit into a range of integer values from 0 through 15."

The satimage dataset is the statlog version of a satellite image dataset. According to the documentation, "This database consists of the multi-spectral values of pixels in 3×3 neighborhoods in a satellite image, and the classification associated with the central pixel in each neighborhood. The aim is to predict this classification, given the multi-spectral values... The original database was generated from Landsat Multi-Spectral Scanner image data... purchased from NASA by the Australian Center for Remote Sensing, and used for research at The Center for Remote Sensing... The sample database was generated taking a small section (82 rows and 100 columns) from the original data. The binary values were converted to their present ASCII form by Ashwin Srinivasan. The classification for each pixel was performed on the basis of an actual site visit by Ms. Karen Hall, when working for Professor John A. Richards.... Conversion to 3×3 neighborhoods and splitting into test and training sets was done by Alistair Sutherland...."

The purpose of the vehicle dataset, according to its documentation, is "to classify a given silhouette as one of four types of vehicle, using a set of features extracted from the silhouette. The vehicle may be viewed from one of many different angles... This dataset comes from the Turing Institute... The [extracted] features were a combination of scale independent features utilizing both classical moments based measures such as scaled variance, skewness and kurtosis about the major/minor axes and heuristic measures such as hollows, circularity, rectangularity and compactness. Four 'Corgie' model vehicles were used for the experiment: a double decker bus, Chevrolet van, Saab 9000 and an Opel Manta 400.... The images were acquired by a camera looking downwards at the model vehicle from a fixed angle of elevation...."

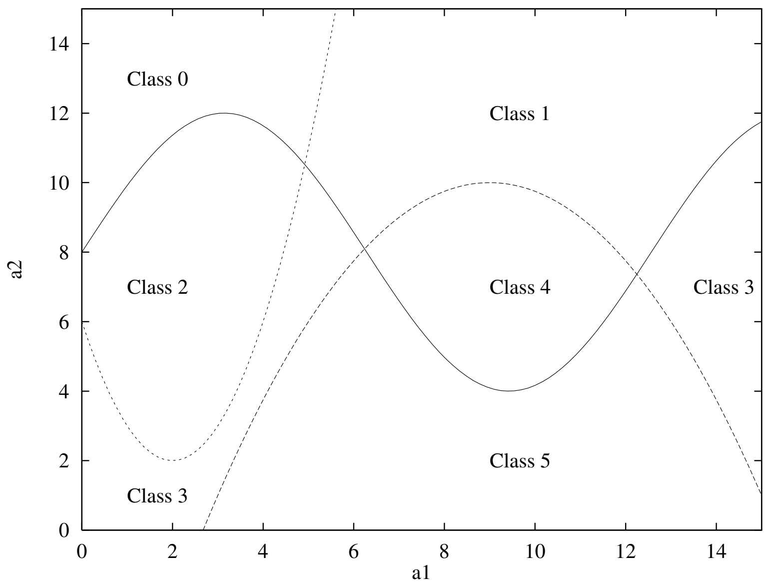

Figure 7: The 2-dimension, 6-class classification problem defined by Kong and Dietterich [26] on the region $[0, 15] \times [0, 15]$.

B.2 Synthetic datasets

In Section 5, we described experiments using synthetically generated data. Twonorm, threenorm and ringnorm were taken from Breiman [8]. Quoting from him:

- Twonorm: This is 20-dimension, 2-class data. Each class is drawn from a multivariate normal distribution with unit covariance matrix. Class #1 has mean $(a, a, ..., a)$ and class #2 has mean $(-a, -a, ..., -a)$ where $a = 2/\sqrt{20}$.
- Threenorm: This is 20-dimension, 2-class data. Class #1 is drawn with equal probability from a unit multivariate normal with mean $(a, a, ..., a)$ and from a unit multivariate normal with mean $(-a, -a, ..., -a)$. Class #2 is drawn from a unit multivariate normal with mean $(a, -a, a, -a, ..., -a)$ where $a = 2/\sqrt{20}$.
- Ringnorm: This is 20-dimension, 2-class data. Class #1 is multivariate normal with mean zero and covariance matrix 4 times the identity. Class #2 has unit covariance matrix and mean $(a, a, ..., a)$ where $a = 1/\sqrt{20}$.

The waveform data is 21-dimension, 3-class data. It is described by Breiman et al. [9]. A program for generating this data is available from the UCI repository [30].

The last dataset was taken from Kong and Dietterich [26]. This is a 2-dimension, 6-class classification problem where the classes are defined by the regions of $[0, 15] \times [0, 15]$ shown in Figure 7.

C Two Definitions of Bias and Variance for Classification

For the sake of completeness, we include here the definitions for bias and variance for classification tasks which we have used in our experiments. The first set of definitions is due to Kong and Dietterich [26] and the second one is due to Breiman [8]. Assume that we have an infinite supply of independent training sets S of size m. Each sample S is drawn i.i.d. from a fixed distribution D over $X \times \{1, \ldots, k\}$ where k is the number of classes. Denote by C_S the classifier that is generated by the base learning algorithm given the sample S. Denote by C_A the classification rule that results from running the base learning algorithm on an infinite number of independent training sets and taking the plurality vote[6] over the resulting classifiers. Finally denote by C^* the Bayes optimal prediction rule for the distribution D. The prediction of a classifier C on an instance $x \in X$ is denoted $C(x)$ and the expected error of a classifier C is denoted

$$PE(C) \doteq \mathbf{P}_{(x,y)\sim D}\left[C(x) \neq y\right] .$$

The definitions of Kong and Dietterich are:

$$\begin{aligned} \text{Bias} &\doteq PE(C_A) - PE(C^*) , \\ \text{Variance} &\doteq \mathbf{E}_{S\sim D^m}\left[PE(C_S)\right] - PE(C_A) . \end{aligned}$$

Breiman defines a partition of the sample space into two sets. The "unbiased" set U consists of all $x \in X$ for which $C_A(x) = C^*(x)$ and the "biased" set B is U's complement. Given these sets the definition of bias and variance are:

$$\begin{aligned} \text{Bias} &\doteq \mathbf{P}_{(x,y)\sim D}\left[C^*(x) = y, x \in B\right] - \mathbf{E}_{S\sim D^m}\left[\mathbf{P}_{(x,y)\sim D}\left[C_S(x) = y, x \in B\right]\right] , \\ \text{Variance} &\doteq \mathbf{P}_{(x,y)\sim D}\left[C^*(x) = y, x \in U\right] - \mathbf{E}_{S\sim D^m}\left[\mathbf{P}_{(x,y)\sim D}\left[C_S(x) = y, x \in U\right]\right] . \end{aligned}$$

[6]The plurality vote outputs the class which receives the largest number of votes, breaking ties uniformly at random. When $k = 2$ the plurality vote is equal to the majority vote.