MSRI Workshop on Nonlinear Estimation and Classification, 2002.

The Boosting Approach to Machine Learning
An Overview

Robert E. Schapire
AT&T Labs − Research
Shannon Laboratory
180 Park Avenue, Room A203
Florham Park, NJ 07932 USA
www.research.att.com/∼schapire

December 19, 2001

Abstract

Boosting is a general method for improving the accuracy of any given learning algorithm. Focusing primarily on the AdaBoost algorithm, this chapter overviews some of the recent work on boosting including analyses of AdaBoost's training error and generalization error; boosting's connection to game theory and linear programming; the relationship between boosting and logistic regression; extensions of AdaBoost for multiclass classification problems; methods of incorporating human knowledge into boosting; and experimental and applied work using boosting.

1 Introduction

Machine learning studies automatic techniques for learning to make accurate predictions based on past observations. For example, suppose that we would like to build an email filter that can distinguish spam (junk) email from non-spam. The machine-learning approach to this problem would be the following: Start by gathering as many examples as posible of both spam and non-spam emails. Next, feed these examples, together with labels indicating if they are spam or not, to your favorite machine-learning algorithm which will automatically produce a classification or prediction rule. Given a new, unlabeled email, such a rule attempts to predict if it is spam or not. The goal, of course, is to generate a rule that makes the most accurate predictions possible on new test examples.

Building a highly accurate prediction rule is certainly a difficult task. On the other hand, it is not hard at all to come up with very rough rules of thumb that are only moderately accurate. An example of such a rule is something like the following: "If the phrase 'buy now' occurs in the email, then predict it is spam." Such a rule will not even come close to covering all spam messages; for instance, it really says nothing about what to predict if 'buy now' does not occur in the message. On the other hand, this rule will make predictions that are significantly better than random guessing.

Boosting, the machine-learning method that is the subject of this chapter, is based on the observation that finding many rough rules of thumb can be a lot easier than finding a single, highly accurate prediction rule. To apply the boosting approach, we start with a method or algorithm for finding the rough rules of thumb. The boosting algorithm calls this "weak" or "base" learning algorithm repeatedly, each time feeding it a different subset of the training examples (or, to be more precise, a different distribution or weighting over the training examples[1]). Each time it is called, the base learning algorithm generates a new weak prediction rule, and after many rounds, the boosting algorithm must combine these weak rules into a single prediction rule that, hopefully, will be much more accurate than any one of the weak rules.

To make this approach work, there are two fundamental questions that must be answered: first, how should each distribution be chosen on each round, and second, how should the weak rules be combined into a single rule? Regarding the choice of distribution, the technique that we advocate is to place the most weight on the examples most often misclassified by the preceding weak rules; this has the effect of forcing the base learner to focus its attention on the "hardest" examples. As for combining the weak rules, simply taking a (weighted) majority vote of their predictions is natural and effective.

There is also the question of what to use for the base learning algorithm, but this question we purposely leave unanswered so that we will end up with a general boosting procedure that can be combined with any base learning algorithm.

Boosting refers to a general and provably effective method of producing a very accurate prediction rule by combining rough and moderately inaccurate rules of thumb in a manner similar to that suggested above. This chapter presents an overview of some of the recent work on boosting, focusing especially on the AdaBoost algorithm which has undergone intense theoretical study and empirical testing.

[1]A distribution over training examples can be used to generate a subset of the training examples simply by sampling repeatedly from the distribution.

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

- Train base learner using distribution D_t.
- Get base classifier $h_t : X \to \mathbb{R}$.
- Choose $\alpha_t \in \mathbb{R}$.
- Update:

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

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

Output the final classifier:

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

Figure 1: The boosting algorithm AdaBoost.

2 AdaBoost

Working in Valiant's PAC (probably approximately correct) learning model [75], Kearns and Valiant [41, 42] were the first to pose the question of whether a "weak" learning algorithm that performs just slightly better than random guessing can be "boosted" into an arbitrarily accurate "strong" learning algorithm. Schapire [66] came up with the first provable polynomial-time boosting algorithm in 1989. A year later, Freund [26] developed a much more efficient boosting algorithm which, although optimal in a certain sense, nevertheless suffered like Schapire's algorithm from certain practical drawbacks. The first experiments with these early boosting algorithms were carried out by Drucker, Schapire and Simard [22] on an OCR task.

The AdaBoost algorithm, introduced in 1995 by Freund and Schapire [32], solved many of the practical difficulties of the earlier boosting algorithms, and is the focus of this paper. Pseudocode for AdaBoost is given in Fig. 1 in the slightly generalized form given by Schapire and Singer [70]. The algorithm takes as input a training set $(x_1, y_1), \ldots, (x_m, y_m)$ where each x_i belongs to some *domain* or *instance space* X, and each *label* y_i is in some label set Y. For most of this paper, we assume $Y = \{-1, +1\}$; in Section 7, we discuss extensions to the multiclass case. AdaBoost calls a given *weak* or *base learning algorithm* repeatedly in a series of rounds $t = 1, \ldots, T$. One of the main ideas of the algorithm is to maintain a distribution or set of weights over the training set. The weight of this distribution on training example i on round t is denoted $D_t(i)$. Initially, all weights are set equally, but on each round, the weights of incorrectly classified examples are increased so that the base learner is forced to focus on the hard examples in the training set.

The base learner's job is to find a *base classifier* $h_t : X \to \mathbb{R}$ appropriate for the distribution D_t. (Base classifiers were also called rules of thumb or weak prediction rules in Section 1.) In the simplest case, the range of each h_t is binary, i.e., restricted to $\{-1, +1\}$; the base learner's job then is to minimize the *error*

$$\epsilon_t = \Pr_{i \sim D_t}\left[h_t(x_i) \neq y_i\right].$$

Once the base classifier h_t has been received, AdaBoost chooses a parameter $\alpha_t \in \mathbb{R}$ that intuitively measures the importance that it assigns to h_t. In the figure, we have deliberately left the choice of α_t unspecified. For binary h_t, we typically set

$$\alpha_t = \tfrac{1}{2} \ln\left(\frac{1 - \epsilon_t}{\epsilon_t}\right) \tag{1}$$

as in the original description of AdaBoost given by Freund and Schapire [32]. More on choosing α_t follows in Section 3. The distribution D_t is then updated using the rule shown in the figure. The *final* or *combined classifier* H is a weighted majority vote of the T base classifiers where α_t is the weight assigned to h_t.

3 Analyzing the training error

The most basic theoretical property of AdaBoost concerns its ability to reduce the training error, i.e., the fraction of mistakes on the training set. Specifically, Schapire and Singer [70], in generalizing a theorem of Freund and Schapire [32], show that the training error of the final classifier is bounded as follows:

$$\frac{1}{m}\left|\{i : H(x_i) \neq y_i\}\right| \;\leq\; \frac{1}{m}\sum_i \exp(-y_i f(x_i)) \;=\; \prod_t Z_t \tag{2}$$

where henceforth we define

$$f(x) = \sum_t \alpha_t h_t(x) \tag{3}$$

so that $H(x) = \mathrm{sign}(f(x))$. (For simplicity of notation, we write $\sum_i$ and $\sum_t$ as shorthand for $\sum_{i=1}^m$ and $\sum_{t=1}^T$, respectively.) The inequality follows from the fact that $e^{-y_i f(x_i)} \geq 1$ if $y_i \neq H(x_i)$. The equality can be proved straightforwardly by unraveling the recursive definition of D_t.

Eq. (2) suggests that the training error can be reduced most rapidly (in a greedy way) by choosing α_t and h_t on each round to minimize

$$Z_t = \sum_i D_t(i) \exp(-\alpha_t y_i h_t(x_i)). \tag{4}$$

In the case of binary classifiers, this leads to the choice of α_t given in Eq. (1) and gives a bound on the training error of

$$\prod_t Z_t = \prod_t \left[2\sqrt{\epsilon_t(1-\epsilon_t)}\right] = \prod_t \sqrt{1-4\gamma_t^2} \leq \exp\left(-2\sum_t \gamma_t^2\right) \tag{5}$$

where we define $\gamma_t = 1/2 - \epsilon_t$. This bound was first proved by Freund and Schapire [32]. Thus, if each base classifier is slightly better than random so that $\gamma_t \geq \gamma$ for some $\gamma > 0$, then the training error drops exponentially fast in T since the bound in Eq. (5) is at most $e^{-2T\gamma^2}$. This bound, combined with the bounds on generalization error given below prove that AdaBoost is indeed a boosting algorithm in the sense that it can efficiently convert a true weak learning algorithm (that can always generate a classifier with a weak edge for any distribution) into a strong learning algorithm (that can generate a classifier with an arbitrarily low error rate, given sufficient data).

Eq. (2) points to the fact that, at heart, AdaBoost is a procedure for finding a linear combination f of base classifiers which attempts to minimize

$$\sum_i \exp(-y_i f(x_i)) = \sum_i \exp\left(-y_i \sum_t \alpha_t h_t(x_i)\right). \tag{6}$$

Essentially, on each round, AdaBoost chooses h_t (by calling the base learner) and then sets α_t to add one more term to the accumulating weighted sum of base classifiers in such a way that the sum of exponentials above will be maximally reduced. In other words, AdaBoost is doing a kind of steepest descent search to minimize Eq. (6) where the search is constrained at each step to follow coordinate directions (where we identify coordinates with the weights assigned to base classifiers). This view of boosting and its generalization are examined in considerable detail by Duffy and Helmbold [23], Mason et al. [51, 52] and Friedman [35]. See also Section 6.

Schapire and Singer [70] discuss the choice of α_t and h_t in the case that h_t is real-valued (rather than binary). In this case, $h_t(x)$ can be interpreted as a "confidence-rated prediction" in which the sign of $h_t(x)$ is the predicted label, while the magnitude $|h_t(x)|$ gives a measure of confidence. Here, Schapire and Singer advocate choosing α_t and h_t so as to minimize Z_t (Eq. (4)) on each round.

4 Generalization error

In studying and designing learning algorithms, we are of course interested in performance on examples *not* seen during training, i.e., in the generalization error, the topic of this section. Unlike Section 3 where the training examples were arbitrary, here we assume that all examples (both train and test) are generated i.i.d. from some unknown distribution on $X \times Y$. The generalization error is the probability of misclassifying a new example, while the test error is the fraction of mistakes on a newly sampled test set (thus, generalization error is expected test error). Also, for simplicity, we restrict our attention to binary base classifiers.

Freund and Schapire [32] showed how to bound the generalization error of the final classifier in terms of its training error, the size m of the sample, the VC-dimension[2] d of the base classifier space and the number of rounds T of boosting. Specifically, they used techniques from Baum and Haussler [5] to show that the generalization error, with high probability, is at most[3]

$$\hat{\Pr}\left[H(x) \neq y\right] + \tilde{O}\left(\sqrt{\frac{Td}{m}}\right)$$

where $\hat{\Pr}\left[\cdot\right]$ denotes empirical probability on the training sample. This bound suggests that boosting will overfit if run for too many rounds, i.e., as T becomes large. In fact, this sometimes does happen. However, in early experiments, several authors [8, 21, 59] observed empirically that boosting often does *not* overfit, even when run for thousands of rounds. Moreover, it was observed that AdaBoost would sometimes continue to drive down the generalization error long after the training error had reached zero, clearly contradicting the spirit of the bound above. For instance, the left side of Fig. 2 shows the training and test curves of running boosting on top of Quinlan's C4.5 decision-tree learning algorithm [60] on the "letter" dataset.

In response to these empirical findings, Schapire et al. [69], following the work of Bartlett [3], gave an alternative analysis in terms of the *margins* of the training examples. The margin of example (x, y) is defined to be

$$\text{margin}_f(x, y) = \frac{yf(x)}{\sum_t |\alpha_t|} = \frac{y \sum_t \alpha_t h_t(x)}{\sum_t |\alpha_t|}.$$

[2]The Vapnik-Chervonenkis (VC) dimension is a standard measure of the "complexity" of a space of binary functions. See, for instance, refs. [6, 76] for its definition and relation to learning theory.

[3]The "soft-Oh" notation $\tilde{O}(\cdot)$, here used rather informally, is meant to hide all logarithmic and constant factors (in the same way that standard "big-Oh" notation hides only constant factors).

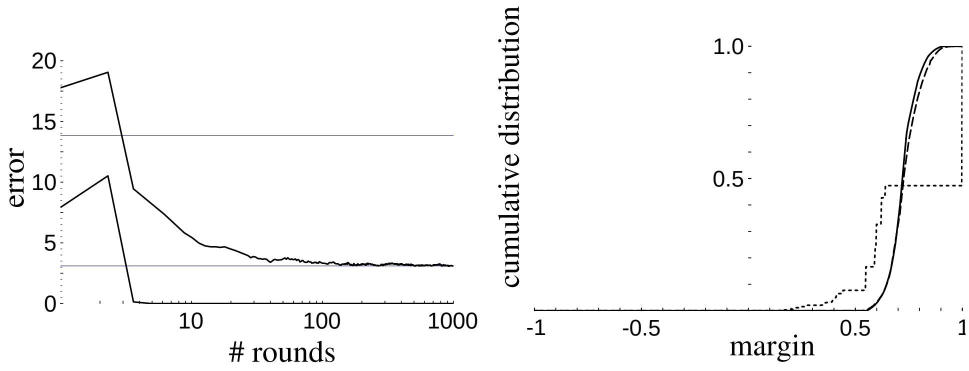

Figure 2: Error curves and the margin distribution graph for boosting C4.5 on the letter dataset as reported by Schapire et al. [69]. *Left*: the training and test error curves (lower and upper curves, respectively) of the combined classifier as a function of the number of rounds of boosting. The horizontal lines indicate the test error rate of the base classifier as well as the test error of the final combined classifier. *Right*: The cumulative distribution of margins of the training examples after 5, 100 and 1000 iterations, indicated by short-dashed, long-dashed (mostly hidden) and solid curves, respectively.

It is a number in $[-1, +1]$ and is positive if and only if H correctly classifies the example. Moreover, as before, the magnitude of the margin can be interpreted as a measure of confidence in the prediction. Schapire et al. proved that larger margins on the training set translate into a superior upper bound on the generalization error. Specifically, the generalization error is at most

$$\hat{\Pr}\left[\mathrm{margin}_f(x,y) \le \theta\right] + \tilde{O}\left(\sqrt{\frac{d}{m\theta^2}}\right)$$

for any $\theta > 0$ with high probability. Note that this bound is entirely independent of T, the number of rounds of boosting. In addition, Schapire et al. proved that boosting is particularly aggressive at reducing the margin (in a quantifiable sense) since it concentrates on the examples with the smallest margins (whether positive or negative). Boosting's effect on the margins can be seen empirically, for instance, on the right side of Fig. 2 which shows the cumulative distribution of margins of the training examples on the "letter" dataset. In this case, even after the training error reaches zero, boosting continues to increase the margins of the training examples effecting a corresponding drop in the test error.

Although the margins theory gives a qualitative explanation of the effectiveness of boosting, quantitatively, the bounds are rather weak. Breiman [9], for instance, shows empirically that one classifier can have a margin distribution that is uniformly better than that of another classifier, and yet be inferior in test accuracy. On the other hand, Koltchinskii, Panchenko and Lozano [44, 45, 46, 58] have recently proved new margin-theoretic bounds that are tight enough to give useful quantitative predictions.

Attempts (not always successful) to use the insights gleaned from the theory of margins have been made by several authors [9, 37, 50]. In addition, the margin theory points to a strong connection between boosting and the support-vector machines of Vapnik and others [7, 14, 77] which explicitly attempt to maximize the minimum margin.

5 A connection to game theory and linear programming

The behavior of AdaBoost can also be understood in a game-theoretic setting as explored by Freund and Schapire [31, 33] (see also Grove and Schuurmans [37] and Breiman [9]). In classical game theory, it is possible to put any two-person, zero-sum game in the form of a matrix $\mathbf{M}$. To play the game, one player chooses a row i and the other player chooses a column j. The loss to the row player (which is the same as the payoff to the column player) is $\mathbf{M}_{ij}$. More generally, the two sides may play randomly, choosing distributions $\mathbf{P}$ and $\mathbf{Q}$ over rows or columns, respectively. The expected loss then is $\mathbf{P}^{\mathrm{T}}\mathbf{M}\mathbf{Q}$.

Boosting can be viewed as repeated play of a particular game matrix. Assume that the base classifiers are binary, and let $\mathcal{H} = \{h_1, ..., h_n\}$ be the entire base classifier space (which we assume for now to be finite). For a fixed training set $(x_1, y_1), \ldots, (x_m, y_m)$, the game matrix $\mathbf{M}$ has m rows and n columns where

$$\mathbf{M}_{ij} = \begin{cases} 1 & \text{if } h_j(x_i) = y_i \\ 0 & \text{otherwise.} \end{cases}$$

The row player now is the boosting algorithm, and the column player is the base learner. The boosting algorithm's choice of a distribution D_t over training examples becomes a distribution $\mathbf{P}$ over rows of $\mathbf{M}$, while the base learner's choice of a base classifier h_t becomes the choice of a column j of $\mathbf{M}$.

As an example of the connection between boosting and game theory, consider von Neumann's famous minmax theorem which states that

$$\max_{\mathbf{Q}} \min_{\mathbf{P}} \mathbf{P}^{\mathrm{T}}\mathbf{M}\mathbf{Q} = \min_{\mathbf{P}} \max_{\mathbf{Q}} \mathbf{P}^{\mathrm{T}}\mathbf{M}\mathbf{Q}$$

for any matrix $\mathbf{M}$. When applied to the matrix just defined and reinterpreted in the boosting setting, this can be shown to have the following meaning: If, for any distribution over examples, there exists a base classifier with error at most $1/2-\gamma$, then there exists a convex combination of base classifiers with a margin of at least 2γ on all training examples. AdaBoost seeks to find such a final classifier with high margin on all examples by combining many base classifiers; so in a sense, the minmax theorem tells us that AdaBoost at least has the potential for success since, given a "good" base learner, there must exist a good combination of base classifiers. Going much further, AdaBoost can be shown to be a special case of a more general algorithm for playing repeated games, or for approximately solving matrix games. This shows that, asymptotically, the distribution over training examples as well as the weights over base classifiers in the final classifier have game-theoretic intepretations as approximate minmax or maxmin strategies.

The problem of solving (finding optimal strategies for) a zero-sum game is well known to be solvable using linear programming. Thus, this formulation of the boosting problem as a game also connects boosting to linear, and more generally convex, programming. This connection has led to new algorithms and insights as explored by Rätsch et al. [62], Grove and Schuurmans [37] and Demiriz, Bennett and Shawe-Taylor [17].

In another direction, Schapire [68] describes and analyzes the generalization of both AdaBoost and Freund's earlier "boost-by-majority" algorithm [26] to a broader family of repeated games called "drifting games."

6 Boosting and logistic regression

Classification generally is the problem of predicting the label y of an example x with the intention of minimizing the probability of an incorrect prediction. However, it is often useful to estimate the *probability* of a particular label. Friedman, Hastie and Tibshirani [34] suggested a method for using the output of AdaBoost to make reasonable estimates of such probabilities. Specifically, they suggested using a logistic function, and estimating

$$\mathrm{Pr}_f\left[y=+1 \mid x\right] = \frac{e^{f(x)}}{e^{f(x)}+e^{-f(x)}} \tag{7}$$

where, as usual, $f(x)$ is the weighted average of base classifiers produced by AdaBoost (Eq. (3)). The rationale for this choice is the close connection between the log loss (negative log likelihood) of such a model, namely,

$$\sum_i \ln\left(1+e^{-2y_i f(x_i)}\right) \tag{8}$$

and the function that, we have already noted, AdaBoost attempts to minimize:

$$\sum_i e^{-y_i f(x_i)}. \tag{9}$$

Specifically, it can be verified that Eq. (8) is upper bounded by Eq. (9). In addition, if we add the constant $1 - \ln 2$ to Eq. (8) (which does not affect its minimization), then it can be verified that the resulting function and the one in Eq. (9) have identical Taylor expansions around zero up to second order; thus, their behavior near zero is very similar. Finally, it can be shown that, for any distribution over pairs (x, y), the expectations

$$\mathrm{E}\left[\ln\left(1 + e^{-2yf(x)}\right)\right]$$

and

$$\mathrm{E}\left[e^{-yf(x)}\right]$$

are minimized by the same (unconstrained) function f, namely,

$$f(x) = \tfrac{1}{2} \ln\left(\frac{\Pr\left[y = +1 \mid x\right]}{\Pr\left[y = -1 \mid x\right]}\right).$$

Thus, for all these reasons, minimizing Eq. (9), as is done by AdaBoost, can be viewed as a method of approximately minimizing the negative log likelihood given in Eq. (8). Therefore, we may expect Eq. (7) to give a reasonable probability estimate.

Of course, as Friedman, Hastie and Tibshirani point out, rather than minimizing the exponential loss in Eq. (6), we could attempt instead to directly minimize the logistic loss in Eq. (8). To this end, they propose their LogitBoost algorithm. A different, more direct modification of AdaBoost for logistic loss was proposed by Collins, Schapire and Singer [13]. Following up on work by Kivinen and Warmuth [43] and Lafferty [47], they derive this algorithm using a unification of logistic regression and boosting based on Bregman distances. This work further connects boosting to the maximum-entropy literature, particularly the iterative-scaling family of algorithms [15, 16]. They also give unified proofs of convergence to optimality for a family of new and old algorithms, including AdaBoost, for both the exponential loss used by AdaBoost and the logistic loss used for logistic regression. See also the later work of Lebanon and Lafferty [48] who showed that logistic regression and boosting are in fact solving the same constrained optimization problem, except that in boosting, certain normalization constraints have been dropped.

For logistic regression, we attempt to minimize the loss function

$$\sum_i \ln\left(1 + e^{-y_i f(x_i)}\right) \tag{10}$$

which is the same as in Eq. (8) except for an inconsequential change of constants in the exponent. The modification of AdaBoost proposed by Collins, Schapire and Singer to handle this loss function is particularly simple. In AdaBoost, unraveling the definition of D_t given in Fig. 1 shows that $D_t(i)$ is proportional (i.e., equal up to normalization) to

$$\exp\left(-y_i f_{t-1}(x_i)\right)$$

where we define

$$f_t(x) = \sum_{t'=1}^{t} \alpha_{t'} h_{t'}(x).$$

To minimize the loss function in Eq. (10), the only necessary modification is to redefine $D_t(i)$ to be proportional to

$$\frac{1}{1 + \exp\left(y_i f_{t-1}(x_i)\right)}.$$

A very similar algorithm is described by Duffy and Helmbold [23]. Note that in each case, the weight on the examples, viewed as a vector, is proportional to the negative gradient of the respective loss function. This is because both algorithms are doing a kind of functional gradient descent, an observation that is spelled out and exploited by Breiman [9], Duffy and Helmbold [23], Mason et al. [51, 52] and Friedman [35].

Besides logistic regression, there have been a number of approaches taken to apply boosting to more general regression problems in which the labels y_i are real numbers and the goal is to produce real-valued predictions that are close to these labels. Some of these, such as those of Ridgeway [63] and Freund and Schapire [32], attempt to reduce the regression problem to a classification problem. Others, such as those of Friedman [35] and Duffy and Helmbold [24] use the functional gradient descent view of boosting to derive algorithms that directly minimize a loss function appropriate for regression. Another boosting-based approach to regression was proposed by Drucker [20].

7 Multiclass classification

There are several methods of extending AdaBoost to the multiclass case. The most straightforward generalization [32], called AdaBoost.M1, is adequate when the base learner is strong enough to achieve reasonably high accuracy, even on the hard distributions created by AdaBoost. However, this method fails if the base learner cannot achieve at least 50% accuracy when run on these hard distributions.

For the latter case, several more sophisticated methods have been developed. These generally work by reducing the multiclass problem to a larger binary problem. Schapire and Singer's [70] algorithm AdaBoost.MH works by creating a set of binary problems, for each example x and each possible label y, of the form: "For example x, is the correct label y or is it one of the other labels?" Freund and Schapire's [32] algorithm AdaBoost.M2 (which is a special case of Schapire and Singer's [70] AdaBoost.MR algorithm) instead creates binary problems, for each example x with correct label y and each *incorrect* label y' of the form: "For example x, is the correct label y or y'?"

These methods require additional effort in the design of the base learning algorithm. A different technique [67], which incorporates Dietterich and Bakiri's [19] method of error-correcting output codes, achieves similar provable bounds to those of AdaBoost.MH and AdaBoost.M2, but can be used with any base learner that can handle simple, binary labeled data. Schapire and Singer [70] and Allwein, Schapire and Singer [2] give yet another method of combining boosting with error-correcting output codes.

8 Incorporating human knowledge

Boosting, like many machine-learning methods, is entirely data-driven in the sense that the classifier it generates is derived exclusively from the evidence present in the training data itself. When data is abundant, this approach makes sense. However, in some applications, data may be severely limited, but there may be human knowledge that, in principle, might compensate for the lack of data.

In its standard form, boosting does not allow for the direct incorporation of such prior knowledge. Nevertheless, Rochery et al. [64, 65] describe a modification of boosting that combines and balances human expertise with available training data. The aim of the approach is to allow the human's rough judgments to be refined, reinforced and adjusted by the statistics of the training data, but in a manner that does not permit the data to entirely overwhelm human judgments.

The first step in this approach is for a human expert to construct by hand a rule p mapping each instance x to an estimated probability $p(x) \in [0, 1]$ that is interpreted as the guessed probability that instance x will appear with label $+1$. There are various methods for constructing such a function p, and the hope is that this difficult-to-build function need not be highly accurate for the approach to be effective.

Rochery et al.'s basic idea is to replace the logistic loss function in Eq. (10)

with one that incorporates prior knowledge, namely,

$$\sum_i \ln\left(1 + e^{-y_i f(x_i)}\right) + \eta \sum_i \mathrm{RE}\left(p(x_i) \parallel \frac{1}{1 + e^{-f(x_i)}}\right)$$

where $\mathrm{RE}\,(p \parallel q) = p\ln(p/q) + (1-p)\ln((1-p)/(1-q))$ is binary relative entropy. The first term is the same as that in Eq. (10). The second term gives a measure of the distance from the model built by boosting to the human's model. Thus, we balance the conditional likelihood of the data against the distance from our model to the human's model. The relative importance of the two terms is controlled by the parameter η.

9 Experiments and applications

Practically, AdaBoost has many advantages. It is fast, simple and easy to program. It has no parameters to tune (except for the number of round T). It requires no prior knowledge about the base learner and so can be flexibly combined with *any* method for finding base classifiers. Finally, it comes with a set of theoretical guarantees given sufficient data and a base learner that can reliably provide only moderately accurate base classifiers. This is a shift in mind set for the learning-system designer: instead of trying to design a learning algorithm that is accurate over the entire space, we can instead focus on finding base learning algorithms that only need to be better than random.

On the other hand, some caveats are certainly in order. The actual performance of boosting on a particular problem is clearly dependent on the data and the base learner. Consistent with theory, boosting can fail to perform well given insufficient data, overly complex base classifiers or base classifiers that are too weak. Boosting seems to be especially susceptible to noise [18] (more on this in Sectionsec:exps).

AdaBoost has been tested empirically by many researchers, including [4, 18, 21, 40, 49, 59, 73]. For instance, Freund and Schapire [30] tested AdaBoost on a set of UCI benchmark datasets [54] using C4.5 [60] as a base learning algorithm, as well as an algorithm that finds the best "decision stump" or single-test decision tree. Some of the results of these experiments are shown in Fig. 3. As can be seen from this figure, even boosting the weak decision stumps can usually give as good results as C4.5, while boosting C4.5 generally gives the decision-tree algorithm a significant improvement in performance.

In another set of experiments, Schapire and Singer [71] used boosting for text categorization tasks. For this work, base classifiers were used that test on the presence or absence of a word or phrase. Some results of these experiments comparing

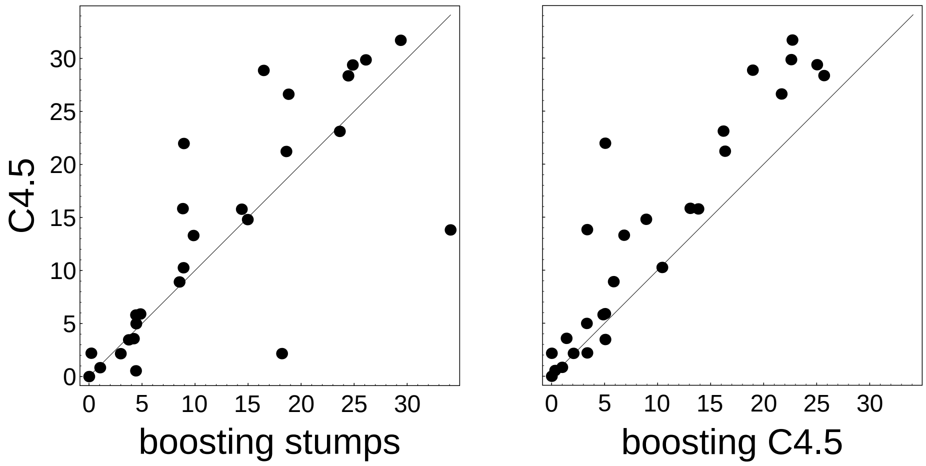

Figure 3: Comparison of C4.5 versus boosting stumps and boosting C4.5 on a set of 27 benchmark problems as reported by Freund and Schapire [30]. Each point in each scatterplot shows the test error rate of the two competing algorithms on a single benchmark. The y-coordinate of each point gives the test error rate (in percent) of C4.5 on the given benchmark, and the x-coordinate gives the error rate of boosting stumps (left plot) or boosting C4.5 (right plot). All error rates have been averaged over multiple runs.

AdaBoost to four other methods are shown in Fig. 4. In nearly all of these experiments and for all of the performance measures tested, boosting performed as well or significantly better than the other methods tested. As shown in Fig. 5, these experiments also demonstrated the effectiveness of using confidence-rated predictions [70], mentioned in Section 3 as a means of speeding up boosting.

Boosting has also been applied to text filtering [72] and routing [39], "ranking" problems [28], learning problems arising in natural language processing [1, 12, 25, 38, 55, 78], image retrieval [74], medical diagnosis [53], and customer monitoring and segmentation [56, 57].

Rochery et al.'s [64, 65] method of incorporating human knowledge into boosting, described in Section 8, was applied to two speech categorization tasks. In this case, the prior knowledge took the form of a set of hand-built rules mapping keywords to predicted categories. The results are shown in Fig. 6.

The final classifier produced by AdaBoost when used, for instance, with a decision-tree base learning algorithm, can be extremely complex and difficult to comprehend. With greater care, a more human-understandable final classifier can be obtained using boosting. Cohen and Singer [11] showed how to design a base

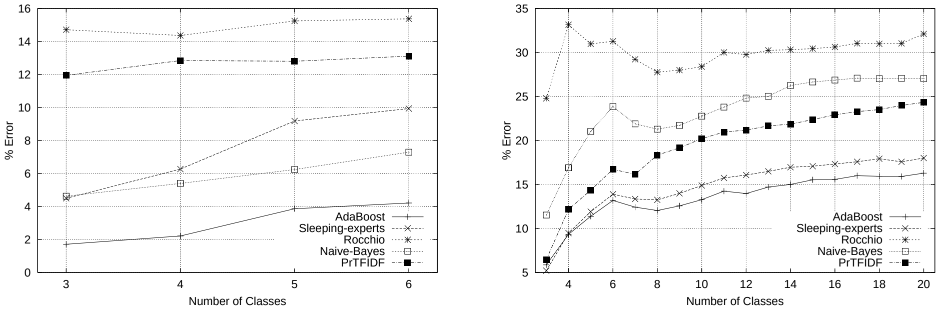

Figure 4: Comparison of error rates for AdaBoost and four other text categorization methods (naive Bayes, probabilistic TF-IDF, Rocchio and sleeping experts) as reported by Schapire and Singer [71]. The algorithms were tested on two text corpora — Reuters newswire articles (left) and AP newswire headlines (right) — and with varying numbers of class labels as indicated on the x-axis of each figure.

learning algorithm that, when combined with AdaBoost, results in a final classifier consisting of a relatively small set of rules similar to those generated by systems like RIPPER [10], IREP [36] and C4.5rules [60]. Cohen and Singer's system, called SLIPPER, is fast, accurate and produces quite compact rule sets. In other work, Freund and Mason [29] showed how to apply boosting to learn a generalization of decision trees called "alternating trees." Their algorithm produces a single alternating tree rather than an ensemble of trees as would be obtained by running AdaBoost on top of a decision-tree learning algorithm. On the other hand, their learning algorithm achieves error rates comparable to those of a whole ensemble of trees.

A nice property of AdaBoost is its ability to identify *outliers*, i.e., examples that are either mislabeled in the training data, or that are inherently ambiguous and hard to categorize. Because AdaBoost focuses its weight on the hardest examples, the examples with the highest weight often turn out to be outliers. An example of this phenomenon can be seen in Fig. 7 taken from an OCR experiment conducted by Freund and Schapire [30].

When the number of outliers is very large, the emphasis placed on the hard examples can become detrimental to the performance of AdaBoost. This was demonstrated very convincingly by Dietterich [18]. Friedman, Hastie and Tibshirani [34] suggested a variant of AdaBoost, called "Gentle AdaBoost" that puts less emphasis on outliers. Rätsch, Onoda and Müller [61] show how to regularize AdaBoost to handle noisy data. Freund [27] suggested another algorithm, called "BrownBoost," that takes a more radical approach that de-emphasizes outliers when it seems clear that they are "too hard" to classify correctly. This algorithm, which is an adaptive

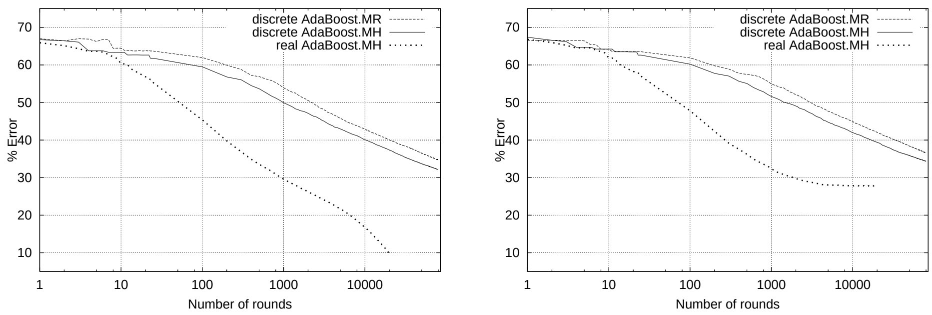

Figure 5: Comparison of the training (left) and test (right) error using three boosting methods on a six-class text classification problem from the TREC-AP collection, as reported by Schapire and Singer [70, 71]. Discrete AdaBoost.MH and discrete AdaBoost.MR are multiclass versions of AdaBoost that require binary ($\{-1, +1\}$-valued) base classifiers, while real AdaBoost.MH is a multiclass version that uses "confidence-rated" (i.e., real-valued) base classifiers.

version of Freund's [26] "boost-by-majority" algorithm, demonstrates an intriguing connection between boosting and Brownian motion.

10 Conclusion

In this overview, we have seen that there have emerged a great many views or interpretations of AdaBoost. First and foremost, AdaBoost is a genuine boosting algorithm: given access to a true weak learning algorithm that always performs a little bit better than random guessing on every distribution over the training set, we can prove arbitrarily good bounds on the training error and generalization error of AdaBoost.

Besides this original view, AdaBoost has been interpreted as a procedure based on functional gradient descent, as an approximation of logistic regression and as a repeated-game playing algorithm. AdaBoost has also been shown to be related to many other topics, such as game theory and linear programming, Bregman distances, support-vector machines, Brownian motion, logistic regression and maximum-entropy methods such as iterative scaling.

All of these connections and interpretations have greatly enhanced our understanding of boosting and contributed to its extension in ever more practical directions, such as to logistic regression and other loss-minimization problems, to multiclass problems, to incorporate regularization and to allow the integration of prior background knowledge.

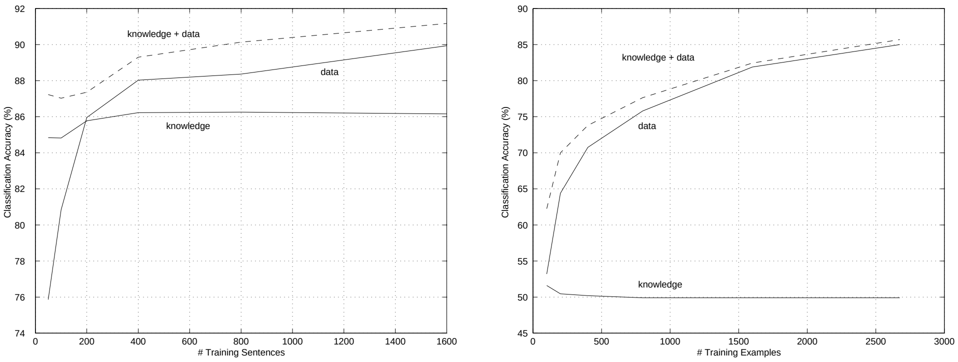

Figure 6: Comparison of percent classification accuracy on two spoken language tasks (“How may I help you” on the left and “Help desk” on the right) as a function of the number of training examples using data and knowledge separately or together, as reported by Rochery et al. [64, 65].

We also have discussed a few of the growing number of applications of AdaBoost to practical machine learning problems, such as text and speech categorization.

References

[1] Steven Abney, Robert E. Schapire, and Yoram Singer. Boosting applied to tagging and PP attachment. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, 1999.

[2] Erin L. Allwein, Robert E. Schapire, and Yoram Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. *Journal of Machine Learning Research*, 1:113–141, 2000.

[3] 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*, 44(2):525–536, March 1998.

[4] Eric Bauer and Ron Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36(1/2):105–139, 1999.

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

[6] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the Association for Computing Machinery*, 36(4):929–965, October 1989.

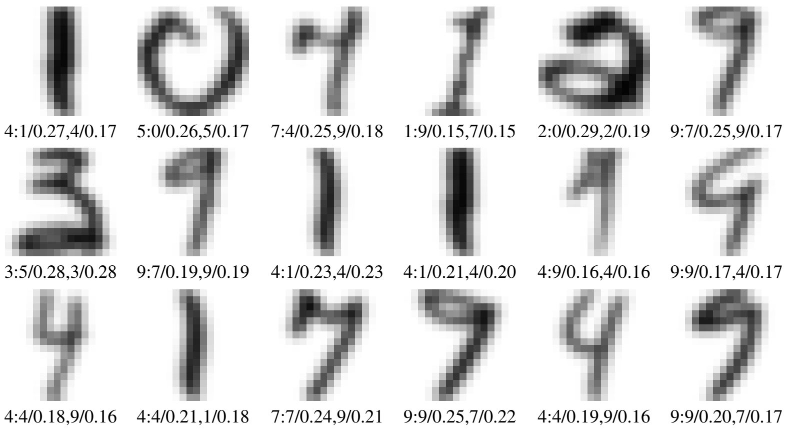

Figure 7: A sample of the examples that have the largest weight on an OCR task as reported by Freund and Schapire [30]. These examples were chosen after 4 rounds of boosting (top line), 12 rounds (middle) and 25 rounds (bottom). Underneath each image is a line of the form d:ℓ_1/w_1,ℓ_2/w_2, where d is the label of the example, ℓ_1 and ℓ_2 are the labels that get the highest and second highest vote from the combined classifier at that point in the run of the algorithm, and w_1, w_2 are the corresponding normalized scores.

[7] 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.

[8] Leo Breiman. Arcing classifiers. *The Annals of Statistics*, 26(3):801–849, 1998.

[9] Leo Breiman. Prediction games and arcing classifiers. *Neural Computation*, 11(7):1493–1517, 1999.

[10] William Cohen. Fast effective rule induction. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 115–123, 1995.

[11] William W. Cohen and Yoram Singer. A simple, fast, and effective rule learner. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, 1999.

[12] Michael Collins. Discriminative reranking for natural language parsing. In *Proceedings of the Seventeenth International Conference on Machine Learning*, 2000.

[13] Michael Collins, Robert E. Schapire, and Yoram Singer. Logistic regression, AdaBoost and Bregman distances. *Machine Learning*, to appear.

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

[15] J. N. Darroch and D. Ratcliff. Generalized iterative scaling for log-linear models. *The Annals of Mathematical Statistics*, 43(5):1470–1480, 1972.

[16] Stephen Della Pietra, Vincent Della Pietra, and John Lafferty. Inducing features of random fields. *IEEE Transactions Pattern Analysis and Machine Intelligence*, 19(4):1–13, April 1997.

[17] Ayhan Demiriz, Kristin P. Bennett, and John Shawe-Taylor. Linear programming boosting via column generation. *Machine Learning*, 46(1/2/3):225–254, 2002.

[18] Thomas G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40(2):139–158, 2000.

[19] 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.

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

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

[22] Harris Drucker, Robert Schapire, and Patrice Simard. Boosting performance in neural networks. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(4):705–719, 1993.

[23] Nigel Duffy and David Helmbold. Potential boosters? In *Advances in Neural Information Processing Systems 11*, 1999.

[24] Nigel Duffy and David Helmbold. Boosting methods for regression. *Machine Learning*, 49(2/3), 2002.

[25] Gerard Escudero, Lluís Màrquez, and German Rigau. Boosting applied to word sense disambiguation. In *Proceedings of the 12th European Conference on Machine Learning*, pages 129–141, 2000.

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

[27] Yoav Freund. An adaptive version of the boost by majority algorithm. *Machine Learning*, 43(3):293–318, June 2001.

[28] Yoav Freund, Raj Iyer, Robert E. Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. In *Machine Learning: Proceedings of the Fifteenth International Conference*, 1998.

[29] Yoav Freund and Llew Mason. The alternating decision tree learning algorithm. In *Machine Learning: Proceedings of the Sixteenth International Conference*, pages 124–133, 1999.

[30] 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.

[31] 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.

[32] 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.

[33] Yoav Freund and Robert E. Schapire. Adaptive game playing using multiplicative weights. *Games and Economic Behavior*, 29:79–103, 1999.

[34] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: A statistical view of boosting. *The Annals of Statistics*, 38(2):337–374, April 2000.

[35] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5), October 2001.

[36] Johannes Fürnkranz and Gerhard Widmer. Incremental reduced error pruning. In *Machine Learning: Proceedings of the Eleventh International Conference*, pages 70–77, 1994.

[37] 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.

[38] Masahiko Haruno, Satoshi Shirai, and Yoshifumi Ooyama. Using decision trees to construct a practical parser. *Machine Learning*, 34:131–149, 1999.

[39] Raj D. Iyer, David D. Lewis, Robert E. Schapire, Yoram Singer, and Amit Singhal. Boosting for document routing. In *Proceedings of the Ninth International Conference on Information and Knowledge Management*, 2000.

[40] Jeffrey C. Jackson and Mark W. Craven. Learning sparse perceptrons. In *Advances in Neural Information Processing Systems 8*, pages 654–660, 1996.

[41] Michael Kearns and Leslie G. Valiant. Learning Boolean formulae or finite automata is as hard as factoring. Technical Report TR-14-88, Harvard University Aiken Computation Laboratory, August 1988.

[42] Michael Kearns and Leslie G. Valiant. Cryptographic limitations on learning Boolean formulae and finite automata. *Journal of the Association for Computing Machinery*, 41(1):67–95, January 1994.

[43] Jyrki Kivinen and Manfred K. Warmuth. Boosting as entropy projection. In *Proceedings of the Twelfth Annual Conference on Computational Learning Theory*, pages 134–144, 1999.

[44] V. Koltchinskii and D. Panchenko. Empirical margin distributions and bounding the generalization error of combined classifiers. *The Annals of Statistics*, 30(1), February 2002.

[45] Vladimir Koltchinskii, Dmitriy Panchenko, and Fernando Lozano. Further explanation of the effectiveness of voting methods: The game between margins and weights. In *Proceedings 14th Annual Conference on Computational Learning Theory and 5th European Conference on Computational Learning Theory*, pages 241–255, 2001.

[46] Vladimir Koltchinskii, Dmitriy Panchenko, and Fernando Lozano. Some new bounds on the generalization error of combined classifiers. In *Advances in Neural Information Processing Systems 13*, 2001.

[47] John Lafferty. Additive models, boosting and inference for generalized divergences. In *Proceedings of the Twelfth Annual Conference on Computational Learning Theory*, pages 125–133, 1999.

[48] Guy Lebanon and John Lafferty. Boosting and maximum likelihood for exponential models. In *Advances in Neural Information Processing Systems 14*, 2002.

[49] 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.

[50] Llew Mason, Peter Bartlett, and Jonathan Baxter. Direct optimization of margins improves generalization in combined classifiers. In *Advances in Neural Information Processing Systems 12*, 2000.

[51] Llew Mason, Jonathan Baxter, Peter Bartlett, and Marcus Frean. Functional gradient techniques for combining hypotheses. In Alexander J. Smola, Peter J. Bartlett, Bernhard Schölkopf, and Dale Schuurmans, editors, *Advances in Large Margin Classifiers*. MIT Press, 1999.

[52] Llew Mason, Jonathan Baxter, Peter Bartlett, and Marcus Frean. Boosting algorithms as gradient descent. In *Advances in Neural Information Processing Systems 12*, 2000.

[53] Stefano Merler, Cesare Furlanello, Barbara Larcher, and Andrea Sboner. Tuning cost-sensitive boosting and its application to melanoma diagnosis. In *Multiple Classifier Systems: Proceedings of the 2nd International Workshop*, pages 32–42, 2001.

[54] C. J. Merz and P. M. Murphy. UCI repository of machine learning databases, 1999. www.ics.uci.edu/∼mlearn/MLRepository.html.

[55] Pedro J. Moreno, Beth Logan, and Bhiksha Raj. A boosting approach for confidence scoring. In *Proceedings of the 7th European Conference on Speech Communication and Technology*, 2001.

[56] Michael C. Mozer, Richard Wolniewicz, David B. Grimes, Eric Johnson, and Howard Kaushansky. Predicting subscriber dissatisfaction and improving retention in the wireless telecommunications industry. *IEEE Transactions on Neural Networks*, 11:690–696, 2000.

[57] Takashi Onoda, Gunnar Rätsch, and Klaus-Robert Müller. Applying support vector machines and boosting to a non-intrusive monitoring system for household electric appliances with inverters. In *Proceedings of the Second ICSC Symposium on Neural Computation*, 2000.

[58] Dmitriy Panchenko. New zero-error bounds for voting algorithms. Unpublished manuscript, 2001.

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

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

[61] G. Rätsch, T. Onoda, and K.-R. Müller. Soft margins for AdaBoost. *Machine Learning*, 42(3):287–320, 2001.

[62] Gunnar Rätsch, Manfred Warmuth, Sebastian Mika, Takashi Onoda, Steven Lemm, and Klaus-Robert Müller. Barrier boosting. In *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*, pages 170–179, 2000.

[63] Greg Ridgeway, David Madigan, and Thomas Richardson. Boosting methodology for regression problems. In *Proceedings of the International Workshop on AI and Statistics*, pages 152–161, 1999.

[64] M. Rochery, R. Schapire, M. Rahim, N. Gupta, G. Riccardi, S. Bangalore, H. Alshawi, and S. Douglas. Combining prior knowledge and boosting for call classification in spoken language dialogue. Unpublished manuscript, 2001.

[65] Marie Rochery, Robert Schapire, Mazin Rahim, and Narendra Gupta. BoosTexter for text categorization in spoken language dialogue. Unpublished manuscript, 2001.

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

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

[68] Robert E. Schapire. Drifting games. *Machine Learning*, 43(3):265–291, June 2001.

[69] Robert E. Schapire, Yoav Freund, Peter Bartlett, and Wee Sun Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5):1651–1686, October 1998.

[70] Robert E. Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, December 1999.

[71] Robert E. Schapire and Yoram Singer. BoosTexter: A boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168, May/June 2000.

[72] Robert E. Schapire, Yoram Singer, and Amit Singhal. Boosting and Rocchio applied to text filtering. In *Proceedings of the 21st Annual International Conference on Research and Development in Information Retrieval*, 1998.

[73] Holger Schwenk and Yoshua Bengio. Training methods for adaptive boosting of neural networks. In *Advances in Neural Information Processing Systems 10*, pages 647–653, 1998.

[74] Kinh Tieu and Paul Viola. Boosting image retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2000.

[75] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, November 1984.

[76] 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.

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

[78] Marilyn A. Walker, Owen Rambow, and Monica Rogati. SPoT: A trainable sentence planner. In *Proceedings of the 2nd Annual Meeting of the North American Chapter of the Associataion for Computational Linguistics*, 2001.