Computational Learning Theory: Fourth European Conference, EuroCOLT'99, pages 1-10, 1999.

Theoretical Views of Boosting

Robert E. Schapire

AT&T Labs, Shannon Laboratory
180 Park Avenue, Room A279, Florham Park, NJ 07932, USA

Abstract. Boosting is a general method for improving the accuracy of any given learning algorithm. Focusing primarily on the AdaBoost algorithm, we briefly survey theoretical work on boosting including analyses of AdaBoost's training error and generalization error, connections between boosting and game theory, methods of estimating probabilities using boosting, and extensions of AdaBoost for multiclass classification problems. We also briefly mention some empirical work.

Background

Boosting is a general method which attempts to "boost" the accuracy of any given learning algorithm. Kearns and Valiant [21, 22] were the first to pose the question of whether a "weak" learning algorithm which performs just slightly better than random guessing in Valiant's PAC model [34] can be "boosted" into an arbitrarily accurate "strong" learning algorithm. Schapire [28] came up with the first provable polynomial-time boosting algorithm in 1989. A year later, Freund [13] developed a much more efficient boosting algorithm which, although optimal in a certain sense, nevertheless suffered from certain practical drawbacks. The first experiments with these early boosting algorithms were carried out by Drucker, Schapire and Simard [12] on an OCR task.

AdaBoost

The AdaBoost algorithm, introduced in 1995 by Freund and Schapire [16], 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 [31]. 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 $Y = \{-1, +1\}$; later, 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

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 weak learner using distribution D_t.
- Get weak hypothesis $h_t : X \to \mathbb{R}$.
- 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 hypothesis:

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

Fig. 1. The boosting algorithm AdaBoost.

examples are increased so that the weak learner is forced to focus on the hard examples in the training set.

The weak learner's job is to find a *weak hypothesis* $h_t : X \to \mathbb{R}$ appropriate for the distribution D_t. In the simplest case, the range of each h_t is binary, i.e., restricted to $\{-1, +1\}$; the weak 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 weak hypothesis h_t has been received, AdaBoost chooses a parameter $\alpha_t \in \mathbb{R}$ which 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

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

More on choosing α_t follows below. The distribution D_t is then updated using the rule shown in the figure. The *final hypothesis* H is a weighted majority vote of the T weak hypotheses where α_t is the weight assigned to h_t.

Analyzing the training error

The most basic theoretical property of AdaBoost concerns its ability to reduce the training error. Specifically, Schapire and Singer [31], in generalizing a theorem of Freund and Schapire [16], show that the training error of the final hypothesis

is bounded as follows:

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

where $f(x) = \sum_t \alpha_t h_t(x)$ so that $H(x) = \mathrm{sign}(f(x))$. 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

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

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

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

where $\epsilon_t = 1/2 - \gamma_t$. This bound was first proved by Freund and Schapire [16]. Thus, if each weak hypothesis is slightly better than random so that γ_t is bounded away from zero, then the training error drops exponentially fast. 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 weak learning algorithm (which can always generate a hypothesis with a weak edge for any distribution) into a strong learning algorithm (which can generate a hypothesis 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 weak hypotheses which attempts to minimize

$$\sum_i \exp(-y_i f(x_i)) = \sum_i \exp\left(-y_i \sum_t \alpha_t h_t(x_i)\right). \tag{3}$$

Essentially, on each round, AdaBoost chooses h_t (by calling the weak learner) and then sets α_t to add one more term to the acculating weighted sum of weak hypotheses 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. (3) where the search is constrained at each step to follow coordinate directions (where we identify coordinates with the weights assigned to weak hypotheses).

Schapire and Singer [31] 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.

Generalization error

Freund and Schapire [16] showed how to bound the generalization error of the final hypothesis in terms of its training error, the size m of the sample, the VC-dimension d of the weak hypothesis space and the number of rounds T of boosting. Specifically, they used techniques from Baum and Haussler [3] to show that the generalization error, with high probability, is at most

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

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 [7, 11, 26] 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 [27] on the "letter" dataset.

In response to these empirical findings, Schapire et al. [30], following the work of Bartlett [1], gave an alternative analysis in terms of the *margins* of the training examples. The margin of example (x, y) is defined to be

$$\frac{y \sum_t \alpha_t h_t(x)}{\sum_t |\alpha_t|}.$$

It is a number in $[-1, +1]$ which 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

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

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

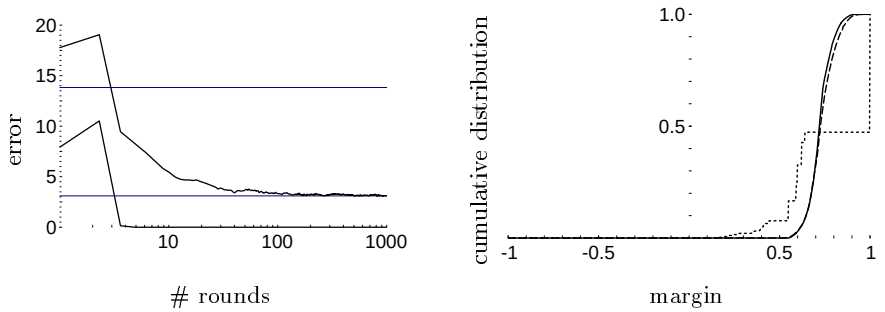

Fig. 2. Error curves and the margin distribution graph for boosting C4.5 on the letter dataset as reported by Schapire et al. [30]. *Left*: the training and test error curves (lower and upper curves, respectively) of the combined classifier as a function of the number of rounds of boosting. The horizontal 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.

after the training error reaches zero, boosting continues to increase the margins of the training examples effecting a corresponding drop in the test error.

Attempts (not always successful) to use the insights gleaned from the theory of margins have been made by several authors [5, 19, 24]. In addition, the margin theory points to a strong connection between boosting and the support-vector machines of Vapnik and others [4, 8, 35] which explicitly attempt to maximize the minimum margin.

A connection to game theory

The behavior of AdaBoost can also be understood in a game-theoretic setting as explored by Freund and Schapire [15, 17] (see also Grove and Schuurmans [19] and Breiman [6]). 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 weak hypotheses are binary, and let $\mathcal{H} = \{h_1, \ldots h_n\}$ be the entire weak hypothesis 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 weak 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 weak learner's choice of a weak hypothesis 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 weak hypothesis with error at most $1/2 - \gamma$, then there exists a convex combination of weak hypotheses with a margin of at least 2γ on all training examples. AdaBoost seeks to find such a final hypothesis with high margin on all examples by combining many weak hypotheses; so in a sense, the minmax theorem tells us that AdaBoost at least has the potential for success since, given a "good" weak learner, there must exist a good combination of weak hypotheses. 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 weak hypotheses in the final hypothesis have game-theoretic intepretations as approximate minmax or maxmin strategies.

Estimating probabilities

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. Recently, Friedman, Hastie and Tibshirani [18] suggested a method for using the output of AdaBoost to make reasonable estimates of such probabilities. Specifically, they suggest using a logistic function, and estimating

$$\Pr_f [y = +1 \mid x] = \frac{e^{f(x)}}{e^{f(x)} + e^{-f(x)}} \tag{4}$$

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

$$\sum_i \ln \left(1 + e^{-2y_i f(x_i)}\right) \tag{5}$$

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

$$\sum_i e^{-y_i f(x_i)}. \tag{6}$$

Specifically, it can be verified that Eq. (5) is upper bounded by Eq. (6). In addition, if we add the constant $1 - \ln 2$ to Eq. (5) (which does not affect its minimization), then it can be verified that the resulting function and the one in Eq. (6) 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 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. (6), as is done by AdaBoost, can be viewed as a method of approximately minimizing the negative log likelihood given in Eq. (5). Therefore, we may expect Eq. (4) to give a reasonable probability estimate.

Friedman, Hastie and Tibshirani also make other connnections between AdaBoost, logistic regression and additive models.

Multiclass classification

There are several methods of extending AdaBoost to the multiclass case. The most straightforward generalization [16], called AdaBoost.M1, is adequate when the weak learner is strong enough to achieve reasonably high accuracy, even on the hard distributions created by AdaBoost. However, this method fails if the weak 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 [31] 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 [16] algorithm AdaBoost.M2 (which is a special case of Schapire and Singer's [31] 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 weak learning algorithm. A different technique [29], which incorporates Dietterich and Bakiri's [10] method of error-correcting output codes, achieves similar provable bounds to those of AdaBoost.MH and AdaBoost.M2, but can be used with any weak learner which can handle simple, binary labeled data. Schapire and Singer [31] give yet another method of combining boosting with error-correcting output codes.

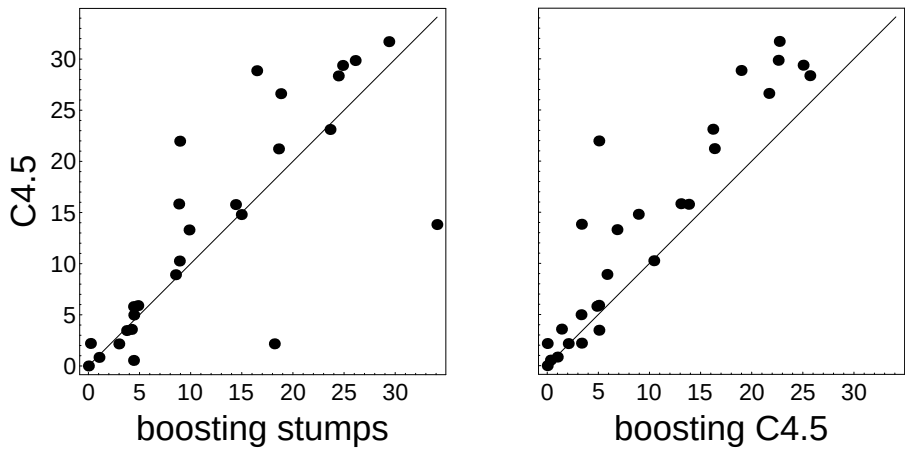

Fig. 3. Comparison of C4.5 versus boosting stumps and boosting C4.5 on a set of 27 benchmark problems as reported by Freund and Schapire [14]. Each point in each scatterplot shows the test error rate of the two competing algorithms on a single benchmark. The y-coordinate of each point gives the test 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.

Experiments and applications

AdaBoost has been tested empirically by many researchers, including [2, 9, 11, 20, 23, 26, 33]. For instance, Freund and Schapire [14] tested AdaBoost on a set of UCI benchmark datasets [25] using C4.5 [27] as a weak learning algorithm, as well as an algorithm which 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 [32] used boosting for text categorization tasks. For this work, weak hypotheses were used which test on the presence or absence of a word or phrase. Some results of these experiments comparing 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.

References

1. 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.

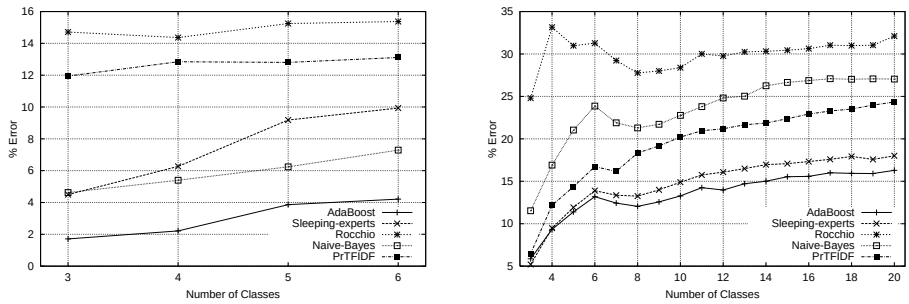

Fig. 4. Comparison of error rates for AdaBoost and four other text categorization methods (naive Bayes, probabilistic TF-IDF, Rocchio and sleeping experts) as reported by Schapire and Singer [32]. The algorithms were tested on two text corpora — Reuters newswire articles (left) and AP newswire headlines (right) — and with varying numbers of class labels as indicated on the x-axis of each figure.

2. Eric Bauer and Ron Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, to appear.
3. Eric B. Baum and David Haussler. What size net gives valid generalization? *Neural Computation*, 1(1):151–160, 1989.
4. 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.
5. Leo Breiman. Arcing the edge. Technical Report 486, Statistics Department, University of California at Berkeley, 1997.
6. Leo Breiman. Prediction games and arcing classifiers. Technical Report 504, Statistics Department, University of California at Berkeley, 1997.
7. Leo Breiman. Arcing classifiers. *The Annals of Statistics*, 26(3):801–849, 1998.
8. Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, September 1995.
9. Thomas G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, to appear.
10. 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.
11. Harris Drucker and Corinna Cortes. Boosting decision trees. In *Advances in Neural Information Processing Systems 8*, pages 479–485, 1996.
12. 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.
13. Yoav Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121(2):256–285, 1995.
14. 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.
15. 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.

16. 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.
17. Yoav Freund and Robert E. Schapire. Adaptive game playing using multiplicative weights. *Games and Economic Behavior*, to appear.
18. Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: a statistical view of boosting. Technical Report, 1998.
19. 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.
20. Jeffrey C. Jackson and Mark W. Craven. Learning sparse perceptrons. In *Advances in Neural Information Processing Systems 8*, pages 654–660, 1996.
21. 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.
22. 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.
23. 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.
24. Llew Mason, Peter Bartlett, and Jonathan Baxter. Direct optimization of margins improves generalization in combined classifiers. Technical report, Deparment of Systems Engineering, Australian National University, 1998.
25. C. J. Merz and P. M. Murphy. UCI repository of machine learning databases, 1998. http://www.ics.uci.edu/~mlearn/MLRepository.html.
26. J. R. Quinlan. Bagging, boosting, and C4.5. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 725–730, 1996.
27. J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.
28. Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990.
29. Robert E. Schapire. Using output codes to boost multiclass learning problems. In *Machine Learning: Proceedings of the Fourteenth International Conference*, pages 313–321, 1997.
30. Robert E. Schapire, Yoav Freund, Peter Bartlett, and Wee Sun Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. In *Machine Learning: Proceedings of the Fourteenth International Conference*, pages 322–330, 1997. To appear, *The Annals of Statistics*.
31. Robert E. Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, pages 80–91, 1998.
32. Robert E. Schapire and Yoram Singer. BoosTexter: A system for multiclass multi-label text categorization. *Machine Learning*, to appear.
33. 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.
34. L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, November 1984.
35. Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.