Boosting Algorithms as Gradient Descent

Llew Mason
Research School of Information Sciences and Engineering
Australian National University
Canberra, ACT, 0200, Australia
lmason@syseng.anu.edu.au

Jonathan Baxter
Research School of Information Sciences and Engineering
Australian National University
Canberra, ACT, 0200, Australia
Jonathan.Baxter@anu.edu.au

Peter Bartlett
Research School of Information Sciences and Engineering
Australian National University
Canberra, ACT, 0200, Australia
Peter.Bartlett@anu.edu.au

Marcus Frean
Department of Computer Science and Electrical Engineering
The University of Queensland
Brisbane, QLD, 4072, Australia
marcusf@elec.uq.edu.au

Abstract

Much recent attention, both experimental and theoretical, has been focussed on classification algorithms which produce voted combinations of classifiers. Recent theoretical work has shown that the impressive generalization performance of algorithms like AdaBoost can be attributed to the classifier having large margins on the training data.

We present an abstract algorithm for finding linear combinations of functions that minimize arbitrary cost functionals (i.e functionals that do not necessarily depend on the margin). Many existing voting methods can be shown to be special cases of this abstract algorithm. Then, following previous theoretical results bounding the generalization performance of convex combinations of classifiers in terms of general cost functions of the margin, we present a new algorithm (DOOM II) for performing a gradient descent optimization of such cost functions.

Experiments on several data sets from the UC Irvine repository demonstrate that DOOM II generally outperforms AdaBoost, especially in high noise situations. Margin distribution plots verify that DOOM II is willing to 'give up' on examples that are too hard in order to avoid overfitting. We also show that the overfitting behavior exhibited by AdaBoost can be quantified in terms of our proposed cost function.

1 Introduction

There has been considerable interest recently in *voting methods* for pattern classification, which predict the label of a particular example using a weighted vote over a set of base classifiers. For example, Freund and Schapire's AdaBoost algorithm [11] and Breiman's Bagging algorithm [3] have been found to give significant performance improvements over algorithms for the corresponding base classifiers [7, 10, 18, 6], and have led to the study of many related algorithms [4, 21, 13, 19, 8, 12, 9]. Recent theoretical results suggest that the effectiveness of these algorithms is due to their tendency to produce *large margin classifiers*. The *margin* of an example is defined as the difference between the total weight assigned to the correct label and the largest weight assigned to an incorrect label. We can interpret the value of the margin as an indication of the confidence of correct classification: an example is classified correctly if and only if it has a positive margin, and a larger margin can be viewed as a confident correct classification. Results in [1] and [20] show that, loosely speaking, if a combination of classifiers correctly classifies most of the training data with a large margin, then its error probability is small.

In [16], Mason, Bartlett and Baxter have presented improved upper bounds on the misclassification probability of a combined classifier in terms of the average over the training data of a certain *cost function* of the margins. That paper also describes experiments with an algorithm that directly minimizes this cost function through the choice of weights associated with each base classifier. This algorithm exhibits performance improvements over AdaBoost, which suggests that these margin cost functions are appropriate quantities to optimize.

In this paper, we present a general class of algorithms (called AnyBoost) which are gradient descent algorithms for choosing linear combinations of elements of an inner product space so as to minimize some cost functional. Each component of the linear combination is chosen to maximize a certain inner product. (In the specific case of choosing a combination of classifiers to optimize the sample average of a cost function of the margin, the choice of the base classifier corresponds to a minimization problem involving weighted classification error. That is, for a certain weighting of the training data, the base classifier learning algorithm attempts to return a classifier that minimizes the weight of misclassified training examples.) In Section 4, we give convergence results for this class of algorithms.

In Section 3, we show that this general class of algorithms includes as special cases a number of popular and successful voting methods, including Freund and Schapire's AdaBoost [11], Schapire and Singer's extension of AdaBoost to combinations of real-valued functions [21], and Friedman, Hastie and Tibshirani's LogitBoost [13]. That is, all of these algorithms implicitly minimize some margin cost function by gradient descent.

In Section 5, we present experimental results for a particular implementation of the AnyBoost algorithm using cost functions of the margin that are motivated by the theoretical results presented in [16]. The cost functions suggested by these results are significantly different from the cost functions that are implicitly minimized by the methods described in Section 3. The experiments show that the new algorithm typically outperforms AdaBoost, and that this is especially true with label noise. In addition, the theoretically-motivated cost functions provide good estimates of the error of AdaBoost, in the sense that they can be used to predict its overfitting behaviour.

2 Optimizing cost functions of the margin

We begin with some notation. We assume that examples (x, y) are randomly generated according to some unknown probability distribution $\mathcal{D}$ on $X \times Y$ where X is the space of measurements (typically $X \subseteq \mathbb{R}^N$) and Y is the space of labels (Y is usually a discrete set or some subset of $\mathbb{R}$).

Although the abstract algorithm of the following section apply to many different machine learning settings, our primary interest in this paper is voted combinations of classifiers of the form $\operatorname{sgn}(F(x))$, where

$$F(x) = \sum_{t=1}^{T} w_t f_t(x),$$

$f_t : X \to \{\pm 1\}$ are base classifiers from some fixed class $\mathcal{F}$ and $w_t \in \mathbb{R}$ are the classifier weights. The *margin* of an example (x, y) with respect to the classifier $\operatorname{sgn}(F(x))$ is defined as $yF(x)$.

Given a set $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ of m labelled examples generated according to $\mathcal{D}$ we wish to construct a voted combination of classifiers of the form described above so that $P_{\mathcal{D}}(\operatorname{sgn}(F(x)) \neq y)$ is small. That is, the probability that F incorrectly classifies a random example is small. Since $\mathcal{D}$ is unknown and we are only given a training set S, we take the approach of finding voted classifiers which minimize the sample average of some cost function of the margin. That is, for a training set S we want to find F such that

$$C(F) = \frac{1}{m} \sum_{i=1}^{m} C(y_i F(x_i)) \tag{1}$$

is minimized for some suitable cost function $C : \mathbb{R} \to \mathbb{R}$. Note that we are using the symbol C to denote both the cost function of the real margin $yF(x)$, and the cost *functional* of the function F. Which interpretation is meant should always be clear from the context.

2.1 AnyBoost

One way to produce a weighted combination of classifiers which optimizes (1) is by gradient descent in function space, an idea first proposed by Breiman [4]. Here we present a more abstract treatment that shows how many existing voting methods may be viewed as gradient descent in a suitable *inner product* space.

At an abstract level we can view the base hypotheses $f \in \mathcal{F}$ and their combinations F as elements of an inner product space $(\mathcal{X}, \langle,\rangle)$. In this case, S is a linear space of functions that contains $\operatorname{lin}(\mathcal{F})$, the set of all linear combinations of functions in $\mathcal{F}$, and the inner product is defined by

$$\langle F, G \rangle := \frac{1}{m} \sum_{i=1}^{m} F(x_i) G(x_i) \tag{2}$$

for all $F, G \in \operatorname{lin}(\mathcal{F})$. However, the AnyBoost algorithm defined in this section and its convergence properties studied in Section 4 are valid for any cost function and inner product.

Now suppose we have a function $F \in \operatorname{lin}(\mathcal{F})$ and we wish to find a new $f \in \mathcal{F}$ to add to F so that the cost $C(F + \epsilon f)$ decreases, for some small value of ϵ. Viewed in function space terms, we are asking for the "direction" f such that $C(F+\epsilon f)$ most rapidly decreases. The desired direction is simply the negative of the functional derivative of C at F, $-\nabla C(F)(x)$, where:

$$\nabla C(F)(x) := \left.\frac{\partial C(F+\alpha 1_x)}{\partial \alpha}\right|_{\alpha=0}, \tag{3}$$

where 1_x if the indicator function of x. Since we are restricted to choosing our new function f from $\mathcal{F}$, in general it will not be possible to choose $f = -\nabla C(F)$, so instead we search for an f with greatest inner product with $-\nabla C(F)$. That is, we should choose f to maximize $-\langle \nabla C(F), f\rangle$. This can be motivated by observing that, to first order in ϵ, $C(F+\epsilon f) = C(F) + \epsilon \langle \nabla C(F), f\rangle$ and hence the greatest reduction in cost will occur for the f maximizing $-\langle \nabla C(F), f\rangle$.

The preceding discussion motivates Algorithm 1, an iterative algorithm for finding linear combinations F of base hypotheses in $\mathcal{F}$ that minimize the cost $C(F)$. Note that we have allowed the base hypotheses to take values in an arbitrary set Y, we have not restricted the form of the cost or the inner product, and we have not specified what the step-sizes should be. Appropriate choices for these things will be made when we apply the algorithm to more concrete situations. Note also that the algorithm terminates when $-\langle \nabla C(F_t), f_{t+1}\rangle \le 0$, i.e when the weak learner $\mathcal{L}$ returns a base hypothesis f_{t+1} which *no longer points in the downhill direction* of the cost function $C(F)$. Thus, the algorithm terminates when, to first order, a step in function space in the direction of the base hypothesis returned by $\mathcal{L}$ would increase the cost.

Algorithm 1 : AnyBoost

Require :
- An inner product space $(\mathcal{X}, \langle,\rangle)$ containing functions mapping from X to some set Y.
- A class of base classifiers $\mathcal{F} \subseteq \mathcal{X}$.
- A differentiable cost functional $C\colon \operatorname{lin}(\mathcal{F}) \to \mathbb{R}$.
- A weak learner $\mathcal{L}(F)$ that accepts $F \in \operatorname{lin}(\mathcal{F})$ and returns $f \in \mathcal{F}$ with a large value of $-\langle \nabla C(F), f\rangle$.

Let $F_0(x) := 0$.
for $t := 0$ to T **do**
  Let $f_{t+1} := \mathcal{L}(F_t)$.
  if $-\langle \nabla C(F_t), f_{t+1}\rangle \le 0$ **then**
    return F_t.
  end if
  Choose w_{t+1}.
  Let $F_{t+1} := F_t + w_{t+1} f_{t+1}$
end for
return F_{T+1}.

3 A gradient descent view of voting methods

Since the main aim of this paper is optimization of margin cost functionals, we restrict our attention to the inner product (2), the cost (1), and $Y = \{\pm 1\}$. For these choices,

$$-\langle \nabla C(F), f\rangle = -\frac{1}{m^2}\sum_{i=1}^{m} y_i f(x_i) C'(y_i F(x_i)).$$

Any sensible cost function of the margin will be monotonically decreasing, hence $-C'(y_i F(x_i))$ will always be positive. Dividing through by $-\sum_{i=1}^{m} C'(y_i F(x_i))$, we see that finding an f maximizing $-\langle \nabla C(F), f\rangle$ is equivalent to finding an f minimizing the weighted error

$$\sum_{i\colon f(x_i)\neq y_i} D(i) \quad \text{where} \quad D(i) := \frac{C'(y_i F(x_i))}{\sum_{i=1}^{m} C'(y_i F(x_i))} \quad \text{for } i = 1, \dots, m.$$

In this case, many of the most successful voting methods are, for the appropriate choice of cost function and step-size, specific cases of the AnyBoost algorithm. Table 3 summarizes the AnyBoost cost function and step-size settings needed to obtain the AdaBoost [11], confidence-rated AdaBoost [21], ARC-X4 [4] and LogitBoost [13] algorithms. A more detailed analysis of these algorithms as specific cases of AnyBoost can be found in the full version of this paper [17].

4 Convergence of AnyBoost

In this section we provide convergence results for the abstract AnyBoost algorithm, under quite weak conditions on the cost functional C. The prescriptions given for the step-sizes w_t in these results are for

Table 1: Existing voting methods viewed as gradient descent optimizers of margin cost functions.

Algorithm	**Cost function**	**Step size**
AdaBoost [10]	$e^{-yF(x)}$	Line search
ARC-X4 [3]	$(1-yF(x))^5$	$1/t$
ConfidenceBoost [21]	$e^{-yF(x)}$	Line search
LogitBoost [13]	$\ln(1+e^{-yF(x)})$	Newton-Raphson

convergence guarantees only: in practice they will almost always be smaller than necessary, hence fixed small steps or some form of line search should be used.

The following theorem (proof omitted, see [17]) supplies a specific step-size for AnyBoost and characterizes the limiting behaviour with this step-size.

Theorem 1. *Let $C\colon \operatorname{lin}(\mathcal{F}) \to \mathbb{R}$ be any lower bounded, Lipschitz differentiable cost functional (that is, there exists $L > 0$ such that $\|\nabla C(F) - \nabla C(F')\| \le L\|F - F'\|$ for all $F, F' \in \operatorname{lin}(\mathcal{F})$). Let $F_0, F_1, \ldots$ be the sequence of combined hypotheses generated by the AnyBoost algorithm, using step-sizes*

$$w_{t+1} := -\frac{\langle \nabla C(F_t), f_{t+1} \rangle}{L\|f_{t+1}\|^2}. \tag{4}$$

Then AnyBoost either halts on round T with $-\langle \nabla C(F_T), f_{T+1} \rangle \le 0$, or $C(F_t)$ converges to some finite value C^*, in which case $\lim_{t\to\infty} \langle \nabla C(F_t), f_{t+1} \rangle = 0$.

The next theorem (proof omitted, see [17]) shows that if the weak learner can always find the best weak hypothesis $f_t \in \mathcal{F}$ on each round of AnyBoost, and if the cost functional C is convex, then any accumulation point F of the sequence (F_t) generated by AnyBoost with the step sizes (4) is a global minimum of the cost. For ease of exposition, we have assumed that rather than terminating when $-\langle \nabla C(F_T), f_{T+1} \rangle \le 0$, AnyBoost simply continues to return F_T for all subsequent time steps t.

Theorem 2. *Let $C\colon \operatorname{lin}(\mathcal{F}) \to \mathbb{R}$ be a* convex *cost functional with the properties in Theorem 1, and let (F_t) be the sequence of combined hypotheses generated by the AnyBoost algorithm with step sizes given by (4). Assume that the weak hypothesis class $\mathcal{F}$ is negation closed ($f \in \mathcal{F} \implies -f \in \mathcal{F}$) and that on each round the AnyBoost algorithm finds a function f_{t+1} maximizing $-\langle \nabla C(F_t), f_{t+1} \rangle$. Then any accumulation point F of the sequence (F_t) satisfies*

$$\sup_{f\in\mathcal{F}} -\langle \nabla C(F), f \rangle = 0, \quad \text{and} \quad C(F) = \inf_{G\in\operatorname{lin}(\mathcal{F})} C(G). \tag{5}$$

5 Experiments

AdaBoost had been perceived to be resistant to overfitting despite the fact that it can produce combinations involving very large numbers of classifiers. However, recent studies have shown that this is not the case, even for base classifiers as simple as decision stumps. Grove and Schuurmans [14] demonstrated that running AdaBoost for hundreds of thousands of rounds can lead to significant overfitting, while a number of authors [6, 19] showed that, by adding label noise, overfitting can be induced in AdaBoost even with relatively few classifiers in the combination.

Given the theoretical motivations described in [16] we propose a new algorithm (DOOM II) which is a specific case of AnyBoost using the cost functional

$$C(F) = \frac{1}{m}\sum_{i=1}^{m} 1 - \tanh(\lambda y_i F(x_i)), \tag{6}$$

where F is restricted to be a *convex* combination of classifiers from some base class $\mathcal{F}$ and λ is an adjustable parameter of the cost function. Henceforth we will refer to (6) as the *normalized sigmoid cost function* (normalized because the weights are normalized so F is a convex combination). This family of cost functions (parameterized by λ) is qualitatively similar to the theoretically motivated family of cost functions used in [16]. Using the family from [16] in practice may cause difficulties for the gradient descent procedure because the functions are very flat for negative margins and for margins close to 1. Using the normalized sigmoid cost function alleviates this problem.

Following the theoretical analysis in [16], λ can be viewed as a data dependent complexity parameter which measures the resolution at which we examine the margins. A large value of λ corresponds to a high resolution and hence high effective complexity of the convex combination. Thus, choosing a large value of λ amounts to a belief that a high complexity classifier can be used to obtain large margins without overfitting. Conversely, choosing a small value of λ corresponds to a belief that a high complexity classifier can only obtain large margins by overfitting.

In our implementation of DOOM II we use a fixed small step-size ϵ (for all of the experiments $\epsilon = 0.05$). In practice the use of a fixed ϵ could be replaced by a line search for the optimal step-size at each round. For full details of the algorithm the reader is referred to the full version of this paper [17].

Given that the normalized sigmoid cost function is non-convex the DOOM II algorithm will suffer from problems with local minima. In fact, the following result shows that for cost functions satisfying $C(-\alpha) = 1 - C(\alpha)$, the algorithm will strike a local minimum at the first step.

Lemma 3. *Let $C\colon \mathbb{R} \to \mathbb{R}$ be any cost function satisfying $C(-\alpha) = 1 - C(\alpha)$. If DOOM II can find the optimal weak hypothesis f_1 at the first time step, it will terminate at the next time step, returning f_1.*

One way of avoiding this local minimum is to remove f_1 from $\mathcal{F}$ after the first round and then continue the algorithm returning f_1 to $\mathcal{F}$ only when the cost goes below that of the first round. Since f_1 is a local minimum the cost is guaranteed to increase after the first round. However, if we continue to step in the best available direction (the flattest uphill direction) we should eventually 'crest the hill' defined by the basin of attraction of the first classifier and then start to decrease the cost. Once the cost decreases below that of the first classifier we can safely return the first classifier to the class of available base classifiers. Of course, we have no guarantee that the cost will decrease below that of the first classifier at any round after the first. Practically however, this does not seem to be a problem except for very small values of λ where the cost function is almost linear over $[-1, 1]$ (in which case the first classifier corresponds to a global minimum anyway).

In order to compare the performance of DOOM II and AdaBoost a series of experiments were carried out on a selection of data sets taken from the UCI machine learning repository [5]. To simplify matters, only binary classification problems were considered. All of the experiments were repeated 100 times with 80%, 10% and 10% of the examples randomly selected for training, validation and test purposes respectively. The results were then averaged over the 100 repeats. For all of the experiments axis orthogonal hyperplanes (also known as decision stumps) were used as the weak learner. This fixed the complexity of the weak learner and thus avoided any problems with the complexity of the combined classifier being dependent on the actual classifiers produced by the weak learner.

For AdaBoost, the validation set was used to perform early stopping. AdaBoost was run for 2000 rounds and then the combined classifier from the round corresponding to minimum error on the validation set was chosen. For DOOM II, the validation set was used to set the data dependent complexity parameter λ. DOOM II was run for 2000 rounds with $\lambda = 2, 4, 6, 10, 15$ and 20 and the optimal λ was chosen to correspond to minimum error on the validation set after 2000 rounds.

AdaBoost and DOOM II were run on nine data sets to which varying levels of label noise had been applied. A summary of the experimental results is shown in Figure 1. The improvement in test error exhibited by DOOM II over AdaBoost (with standard error bars) is shown for each data set and noise level. These results show that DOOM II generally outperforms AdaBoost and that the improvement is more pronounced in the presence of label noise.

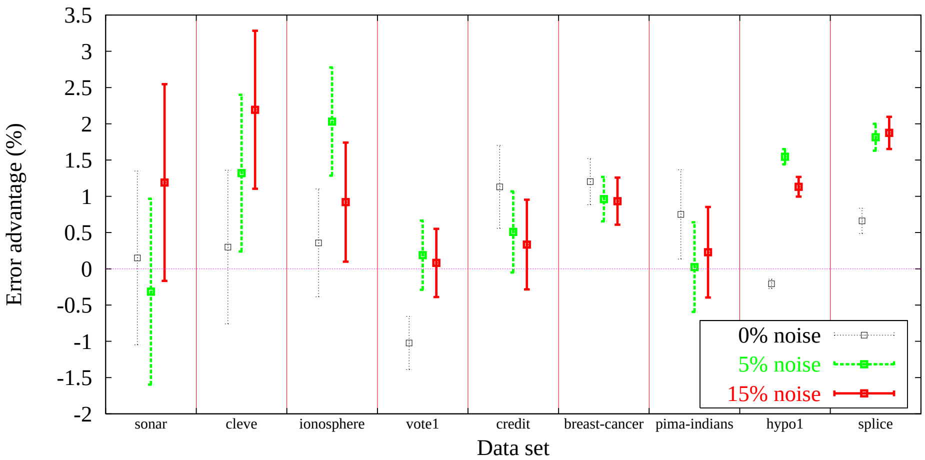

Figure 1: Summary of test error advantage (with standard error bars) of DOOM II over AdaBoost with varying levels of noise on nine UCI data sets.

The effect of using the normalized sigmoid cost function rather than the exponential cost function is best illustrated by comparing the cumulative margin distributions generated by AdaBoost and DOOM II. Figure 2 shows comparisons for two data sets with 0% and 15% label noise applied. For a given margin, the value on the curve corresponds to the proportion of training examples with margin less than or equal to this value. These curves show that in trying to increase the margins of negative examples AdaBoost is willing to sacrifice the margin of positive examples significantly. In contrast, DOOM II 'gives up' on examples with large negative margin in order to reduce the value of the cost function.

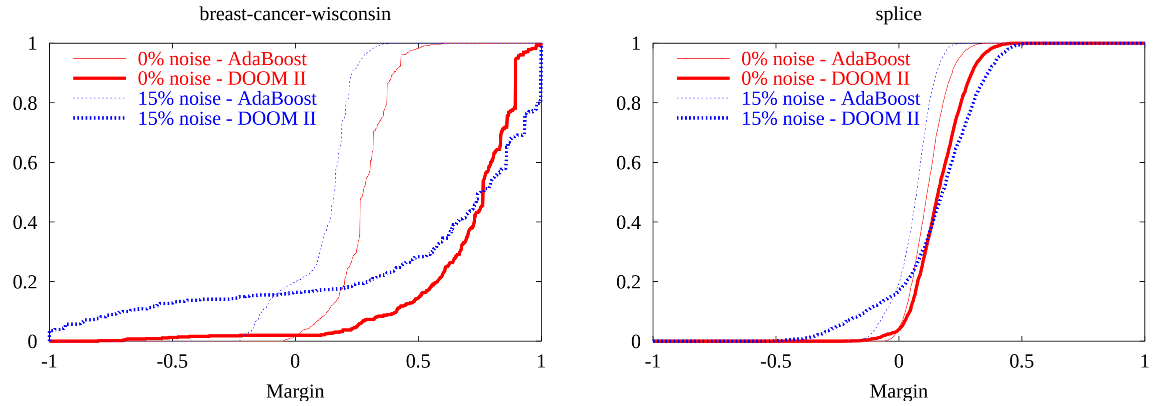

Figure 2: Margin distributions for AdaBoost and DOOM II with 0% and 15% label noise for the `breast-cancer` and `splice` data sets.

Given that AdaBoost does suffer from overfitting and is guaranteed to minimize an exponential cost function of the margins, this cost function certainly does not relate to test error. How does the value of our proposed cost function correlate against AdaBoost's test error? The theoretical bound suggests that for the 'right' value of the data dependent complexity parameter λ our cost function and the test error should be closely correlated. Figure 3 shows the variation in the normalized sigmoid cost function, the exponential cost function and the test error for AdaBoost for two UCI data sets over 10000 rounds. As before, the values of these curves were averaged over 100 random train/validation/test splits. The value of λ used in each case was chosen by running DOOM II for various values of λ and choosing the λ corresponding to minimum error on the validation set. These curves show that there is a strong correlation between the normalized sigmoid cost (for the right value of λ) and AdaBoost's test error. In both data sets the minimum of AdaBoost's test error and the minimum of the normalized sigmoid cost very nearly coincide. In the `labor` data set AdaBoost's test error converges and overfitting does not occur. For this data set both the normalized sigmoid cost and the exponential cost converge. In the `vote1` data set AdaBoost initially decreases the test error and then increases the test error (as overfitting set in). For this data set the normalized sigmoid cost mirrors this behaviour, while the exponential cost converges to 0.

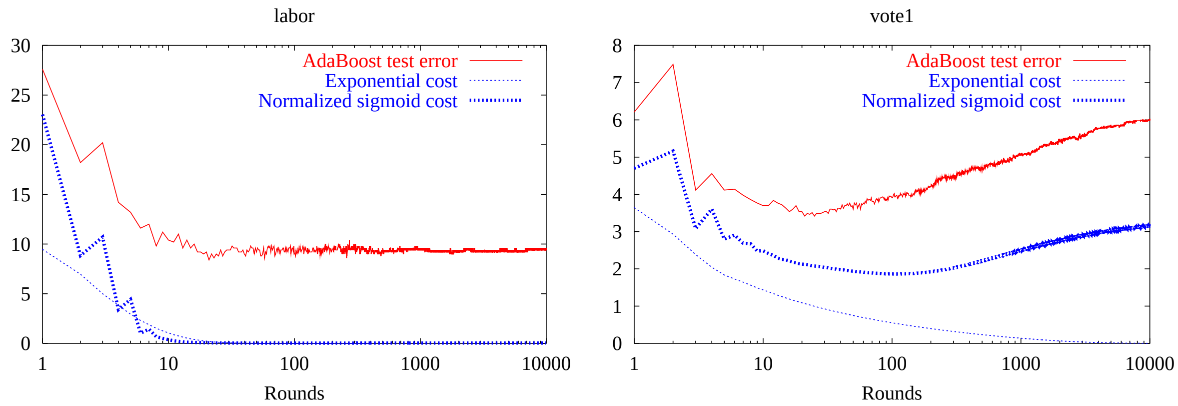

Figure 3: AdaBoost test error, exponential cost and normalized sigmoid cost over 10000 rounds of AdaBoost for the `labor` and `vote1` data sets. Both costs have been scaled in each case for easier comparison with test error.

6 Conclusions

We have shown how many existing "boosting-type" algorithms for combining classifiers can be viewed as gradient descent on an appropriate cost functional in a suitable inner product space. We presented "AnyBoost", an abstract algorithm of this type for generating general linear combinations from some base hypothesis class. A prescription for the step-sizes in this algorithm guaranteeing convergence to the optimal cost were given.

Motivated by the main theoretical result from [16], we derived DOOM II—a specialization of AnyBoost—that used $1 - \tanh(z)$ as its cost function of the margin z. Experimental results on the UCI datasets verified that DOOM II generally outperformed AdaBoost when boosting decision stumps, particularly in the presence of label noise. We also found that DOOM II's cost on the training data was a very reliable predictor of test error, while AdaBoost's exponential cost was not.

The inner product perspective on boosting can be applied to any inner product space, not just spaces of functions as we have done here. In future work we will explore the possibility of applying boosting in other machine learning settings.

Acknowledgments

This research was supported by the Australian Research Council. Llew Mason was supported by an Australian Postgraduate Research Award. Jonathan Baxter was supported by an Australian Postdoctoral Fellowship. Peter Bartlett and Marcus Frean were supported by an Institute of Advanced Studies/Australian Universities Collaborative grant. Thanks to Shai Ben-David for a stimulating discussion.

References

[1] P. 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.

[2] E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 1997. (to appear).

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

[4] L. Breiman. Prediction games and arcing algorithms. Technical Report 504, Department of Statistics, University of California, Berkeley, 1998.

[5] E. Keogh C. Blake and C. J. Merz. UCI repository of machine learning databases, 1998. http://www.ics.uci.edu/~mlearn/MLRepository.html.

[6] T.G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. Technical report, Computer Science Department, Oregon State University, 1998.

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

[8] N. Duffy and D. Helmbold. A geometric approach to leveraging weak learners. In *Computational Learning Theory: 4th European Conference*, 1999. (to appear).

[9] Y. Freund. An adaptive version of the boost by majority algorithm. In *Proceedings of the Twelfth Annual Conference on Computational Learning Theory*, 1999. (to appear).

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

[11] Y. Freund and R. 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.

[12] J. Friedman. Greedy function approximation : A gradient boosting machine. Technical report, Stanford University, 1999.

[13] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression : A statistical view of boosting. Technical report, Stanford University, 1998.

[14] A. Grove and D. Schuurmans. Boosting in the limit: Maximizing the margin of learned ensembles. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 692–699, 1998.

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

[16] L. Mason, P. L. Bartlett, and J. Baxter. Improved generalization through explicit optimization of margins. *Machine Learning*, 1999. (to appear).

[17] L. Mason, J. Baxter, P. L. Bartlett, and M. Frean. Boosting algorithms as gradient descent in function space. Technical report, RSISE, Australian National University, 1999.

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

[19] G. Rätsch, T. Onoda, and K.-R. Müller. Soft margins for AdaBoost. Technical Report NC-TR-1998-021, Department of Computer Science, Royal Holloway, University of London, Egham, UK, 1998.

[20] R. E. Schapire, Y. Freund, P. L. Bartlett, and W. S. Lee. Boosting the margin : A new explanation for the effectiveness of voting methods. *Annals of Statistics*, 26(5):1651–1686, October 1998.

[21] R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, pages 80–91, 1998.

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