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

We provide an abstract characterization of boosting algorithms as gradient decsent on cost-functionals in an inner-product function space. We prove convergence of these functional-gradient-descent algorithms under quite weak conditions. 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, and that the overfitting behaviour of AdaBoost is predicted by our cost functions.

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 [10, 2, 6, 9, 16, 5, 3, 19, 12, 17, 7, 11, 8]. Recent theoretical results suggest that the effectiveness of these algorithms is due to their tendency to produce *large margin classifiers* [1, 18]. 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 [14] we gave 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 described DOOM, an algorithm for directly minimizing the margin cost function by adjusting the weights associated with each base classifier (the base classifiers are suppiled to DOOM). DOOM exhibits performance improvements over AdaBoost, even when using the same base hypotheses, which provides additional empirical evidence 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 function space so as to minimize some cost functional. The normal operation of a weak learner is shown to be equivalent to maximizing a certain inner product. We prove convergence of AnyBoost under weak conditions. In Section 3, we show that this general class of algorithms includes as special cases nearly all existing voting methods. In Section 5, we present experimental results for a special case of AnyBoost that minimizes a theoretically-motivated margin cost functional. 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 AnyBoost

Let (x, y) denote *examples* from $X \times Y$, 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 $\mathbb{R}$). Let $\mathcal{F}$ denote some class of functions (the base hypotheses) mapping $X \to Y$, and $\operatorname{lin}(\mathcal{F})$ denote the set of all linear combinations of functions in $\mathcal{F}$. Let $\langle , \rangle$ be an *inner product* on $\operatorname{lin}(\mathcal{F})$, and

$$C\colon \operatorname{lin}(\mathcal{F}) \to \Re$$

a cost *functional* on $\operatorname{lin}(\mathcal{F})$.

Our aim is to find a function $F \in \operatorname{lin}(\mathcal{F})$ minimizing $C(F)$. We will proceed iteratively via a gradient descent procedure.

Suppose we have some $F \in \operatorname{lin}(\mathcal{F})$ 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, $-\nabla C(F)$, where:

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

where 1_x is 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 $-\langle \nabla C(F), f \rangle$.

For reasons that will become obvious later, an algorithm that chooses f attempting to maximize $-\langle \nabla C(F), f \rangle$ will be described as a *weak learner*.

The preceding discussion motivates Algorithm 1 (AnyBoost), an iterative algorithm for finding linear combinations F of base hypotheses in $\mathcal{F}$ that minimize the cost functional $C(F)$. 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 $-\langle \nabla C(F_t), f_{t+1}\rangle \leq 0$, 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 $-\langle \nabla C(F_t), f_{t+1}\rangle \leq 0$ **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

We now restrict our attention to base hypotheses $f \in \mathcal{F}$ mapping to $Y = \{\pm 1\}$, and the inner product

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

for all $F, G \in \operatorname{lin}(\mathcal{F})$, where $S = \{x_1, y_1), \ldots, (x_n, y_n)\}$ is a set of training examples generated according to some unknown distribution $\mathcal{D}$ on $X \times Y$. Our aim now is to find $F \in \operatorname{lin}(\mathcal{F})$ such that $\Pr_{(x,y)\sim\mathcal{D}} \operatorname{sgn}(F(x)) \neq y$ is minimal, where $\operatorname{sgn}(F(x)) = -1$ if $F(x) < 0$ and $\operatorname{sgn}(F(x)) = 1$ otherwise. In other words, $\operatorname{sgn} F$ should minimize the misclassification probability.

The *margin* of $F\colon X \to \Re$ on example (x, y) is defined as $yF(x)$. Consider *margin cost-functionals* defined by

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

where $c\colon \Re \to \Re$ is any differentiable real-valued function of the margin. With these definitions, a quick calculation shows:

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

Since positive margins correspond to examples correctly labelled by $\operatorname{sgn} F$ and negative margins to incorrectly labelled examples, any sensible cost function of the

Table 1: Existing voting methods viewed as AnyBoost on margin cost functions.

Algorithm	Cost function	Step size
AdaBoost [9]	$e^{-yF(x)}$	Line search
ARC-X4 [2]	$(1-yF(x))^5$	$1/t$
ConfidenceBoost [19]	$e^{-yF(x)}$	Line search
LogitBoost [12]	$\ln(1+e^{-yF(x)})$	Newton-Raphson

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

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

Many of the most successful voting methods are, for the appropriate choice of margin cost function c and step-size, specific cases of the AnyBoost algorithm (see Table 3). A more detailed analysis can be found in the full version of this paper [15].

4 Convergence of AnyBoost

In this section we provide convergence results for the 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 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 [15]) 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 $\|\nabla C(F) - \nabla C(F')\| \leq L\|F - F'\|$ 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*

$$w_{t+1} := -\frac{\langle \nabla C(F_t), f_{t+1} \rangle}{L\|f_{t+1}\|^2}. \tag{3}$$

Then AnyBoost either halts on round T with $-\langle \nabla C(F_T), f_{T+1} \rangle \leq 0$, 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 [15]) 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 (3) is a global minimum of the cost. For ease of exposition, we have assumed that rather than terminating when $-\langle \nabla C(F_T), f_{T+1} \rangle \leq 0$, 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* (3)*. 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* $\sup_{f \in \mathcal{F}} - \langle \nabla C(F), f \rangle = 0$, *and* $C(F) = \inf_{G \in \mathrm{lin}\,(\mathcal{F})} C(G)$.

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 [13, 5, 17]. This overfitting can be attributed to the use of exponential margin cost functions (recall Table 3).

The results in in [14] showed that overfitting may be avoided by using margin cost functionals of a form qualitatively similar to

$$C(F) = \frac{1}{m} \sum_{i=1}^{m} 1 - \tanh(\lambda y_i F(x_i)), \tag{4}$$

where λ is an adjustable parameter controlling the steepness of the margin cost function $c(z) = 1 - \tanh(\lambda z)$. For the theoretical analysis of [14] to apply, F must be a *convex* combination of base hypotheses, rather than a general linear combination. Henceforth (4) will be referred to as the *normalized sigmoid cost functional.* AnyBoost with (4) as the cost functional and (2) as the inner product will be referred to as **DOOM II**. In our implementation of DOOM II we use a fixed small step-size ϵ (for all of the experiments $\epsilon = 0.05$). For all details of the algorithm the reader is referred to the full version of this paper [15].

We compared the performance of DOOM II and AdaBoost on a selection of nine data sets taken from the UCI machine learning repository [4] to which various levels of label noise had been applied. To simplify matters, only binary classification problems were considered. For all of the experiments axis orthogonal hyperplanes (also known as decision stumps) were used as the weak learner. Full details of the experimental setup may be found in [15]. A summary of the experimental results is shown in Figure 1. The improvement in test error exhibited by DOOM II over AdaBoost is shown for each data set and noise level. DOOM II generally outperforms AdaBoost and the improvement is more pronounced in the presence of label noise.

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.

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? 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. There is a strong correlation between the normalized sigmoid cost and AdaBoost's test error. In both data sets the minimum

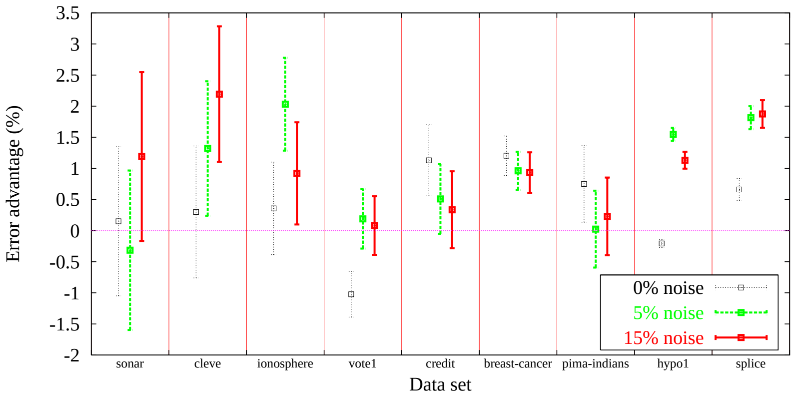

Figure 1: Summary of test error advantage (with standard error bars) of DOOM II over AdaBoost with varying levels of noise on nine UCI data sets.

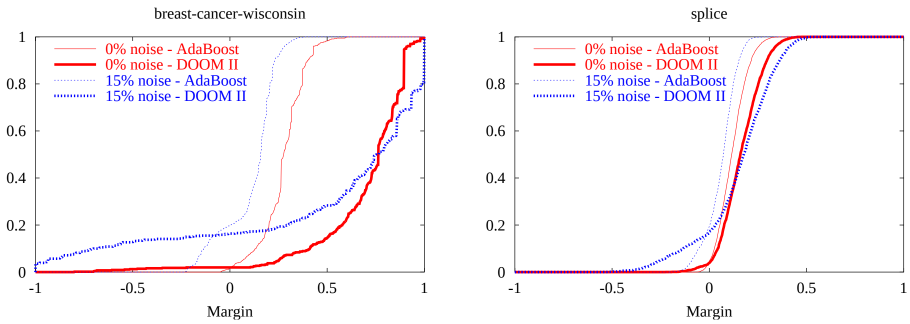

Figure 2: Margin distributions for AdaBoost and DOOM II with 0% and 15% label noise for the `breast-cancer` and `splice` data sets.

of AdaBoost's test error and the minimum of the normalized sigmoid cost very nearly coincide, showing that the sigmoid cost function predicts when AdaBoost will start to overfit.

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

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

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

[5] 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.

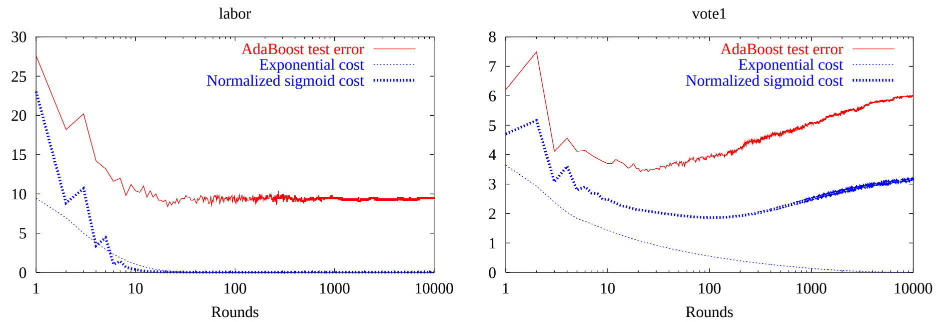

Figure 3: AdaBoost test error, exponential cost and normalized sigmoid cost over 10000 rounds of AdaBoost for the `labor` and `vote1` data sets. Both costs have been scaled in each case for easier comparison with test error.

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

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

[8] 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).

[9] 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.

[10] 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.

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

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

[13] 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.

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

[15] Llew Mason, Jonathan Baxter, Peter Bartlett, and Marcus Frean. Functional Gradient Techniques for Combining Hypotheses. In Alex Smola, Peter Bartlett, Bernard Schölkopf, and Dale Schurmanns, editors, *Large Margin Classifiers*. MIT Press, 1999. To appear.

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

[17] 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.

[18] 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.

[19] 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.