12 Functional Gradient Techniques for Combining Hypotheses

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
Jon.Baxter@anu.edu.au

Peter L. 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
University of Queensland
Brisbane, QLD, 4072, Australia
marcusf@elec.uq.edu.au

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 abstract algorithms for finding linear and convex 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 these abstract algorithms. 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.

12.1 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, AdaBoost (Freund and Schapire, 1997) and Bagging (Breiman, 1996) have been found to give significant performance improvements over algorithms for the corresponding base classifiers (Drucker and Cortes, 1996; Freund and Schapire, 1996; Quinlan, 1996a; Dietterich, 1998; Schwenk and Bengio, 1998; Bauer and Kohavi, 1997; Maclin and Opitz, 1997), and have led to the study of many related algorithms (Breiman, 1998; Schapire and Singer, 1998; Friedman et al., 1998; Rätsch et al., 1998; Duffy and Helmbold, 1999; Friedman, 1999). Recent theoretical results suggest that the effectiveness of these algorithms is due to their tendency to produce *large margin classifiers*. (See Section 1.4 for a definition of margins and a review of these results.)

Mason, Bartlett, and Baxter (1999) 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 described experiments with an algorithm, DOOM, that modifies the classifier weights of an *existing* combined classifier in order to minimize this cost function. This algorithm exhibits performance improvements over AdaBoost, which suggests that these margin cost functions are appropriate quantities to optimize. Unlike the DOOM algorithm (which does not provide a method for choosing the base classifiers), the DOOM II algorithm presented in this chapter provides an iterative method for choosing both the base classifiers *and* their weights so as to minimize the cost functions suggested by the theoretical analysis of (Mason et al., 1999).

In this chapter, we present a general algorithm, MarginBoost, for choosing a combination of classifiers to optimize the sample average of any cost function of the margin. MarginBoost performs gradient descent in function space, at each iteration choosing a base classifier to include in the combination so as to maximally reduce the cost function. The idea of performing gradient descent in function space in this way is due to Breiman (1998). It turns out that, as in AdaBoost, 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.

There is a simpler and more abstract way to view the MarginBoost algorithm. In Section 12.2, we describe a 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 MarginBoost, this inner product corresponds to the weighted training error of the base classifier.) In Section 12.5, we give convergence results for this class of algorithms. For MarginBoost with a convex cost function, these results show that, with a particular choice of the step-size, if the base classifier minimizes the appropriate weighted error then the algorithm converges to the global minimum of the cost function.

In Section 12.3, we show that this general class of algorithms includes as special cases a number of popular and successful voting methods, including AdaBoost (Freund and Schapire, 1997), an extension of AdaBoost to combinations of real-valued functions (Schapire and Singer, 1998), and LogitBoost (Friedman et al., 1998). That is, all of these algorithms implicitly minimize some margin cost function by gradient descent.

In Section 12.4, we review the theoretical results from (Mason et al., 1999) bounding the error of a combination of classifiers in terms of the sample average of certain cost functions of the margin. The cost functions suggested by these results are significantly different from the cost functions that are implicitly minimized by the methods described in Section 12.3. In Section 12.6, we present experimental results for the MarginBoost algorithm with cost functions that are motivated by the theoretical results. These 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.

Similar techniques for directly optimizing margins (and related quantities) have been described by several authors. Rätsch et al. (1998) show that versions of AdaBoost modified to use regularization are more robust for noisy data. Friedman (1999) describes general "boosting" algorithms for regression and classification using various cost functions and presents specific cases for boosting decision trees. Duffy and Helmbold (1999) describe two algorithms (GeoLev and GeoArc) which attempt to produce combined classifiers with *uniformly* large margins on the training data. Freund (1999) presents a new boosting algorithm which uses example weights similar to those suggested by the theoretical results from (Mason et al., 1999).

12.2 Optimizing cost functions of the margin

We begin with some notation. We assume that examples $(\mathbf{x}, y)$ 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 algorithms of the following section apply to many different machine learning settings, our primary interest in this chapter is voted combinations of classifiers of the form $\operatorname{sgn}(F(\mathbf{x}))$, where

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

$f_t : X \to \{\pm 1\}$ are base classifiers from some fixed class $\mathcal{F}$ and $w_t \in \mathbb{R}$ are the classifier weights. Recall (Definition 1.2) that the *margin* of an example $(\mathbf{x}, y)$ with respect to the classifier $\operatorname{sgn}(F(\mathbf{x}))$ is defined as $yF(\mathbf{x})$.

Given a set $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$ 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 $P_{\mathcal{D}}(\operatorname{sgn}(F(\mathbf{x})) \neq y)$ 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

$$C(F) = \frac{1}{m} \sum_{i=1}^{m} C(y_i F(\mathbf{x}_i)) \tag{12.1}$$

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 $yF(\mathbf{x})$, and the cost *functional* of the function F. Which interpretation is meant should always be clear from the context.

12.2.1 AnyBoost

One way to produce a weighted combination of classifiers which optimizes (12.1) is by gradient descent in function space, an idea first proposed by Breiman (1998). 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, $\mathcal{X}$ 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

$$\langle F, G \rangle := \frac{1}{m} \sum_{i=1}^{m} F(\mathbf{x}_i) G(\mathbf{x}_i) \tag{12.2}$$

for all $F, G \in \operatorname{lin}(\mathcal{F})$. However, the AnyBoost algorithms defined in this section and their convergence properties studied in Section 12.5 are valid for any cost function and inner product. For example, they will hold in the case $\langle F, G \rangle := \int_X F(\mathbf{x})G(\mathbf{x})dP(\mathbf{x})$ where P is the marginal distribution on the input space generated by $\mathcal{D}$.

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. Viewing the cost C as a *functional* on $\operatorname{lin}(\mathcal{F})$, the desired direction is simply $-\nabla C(F)(\mathbf{x})$, the negative of the functional derivative of C at F. Here, $\nabla C(F)$ is the unique function such that for any $f \in \mathcal{X}$,

$$C(F + f) = C(F) + \langle \nabla C(F), f \rangle + o(\|f\|). \tag{12.3}$$

If we assume that C is differentiable everywhere then

$$\nabla C(F)(\mathbf{x}) := \left.\frac{\partial C(F + \alpha 1_{\mathbf{x}})}{\partial \alpha}\right|_{\alpha=0}, \tag{12.4}$$

where $1_{\mathbf{x}}$ if the indicator function of $\mathbf{x}$. 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 (12.3) implies 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 which maximizes $-\langle \nabla C(F), f \rangle$.

The preceding discussion motivates Algorithm 12.1, 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 $-\langle \nabla C(F_t), f_{t+1} \rangle \leq 0$, 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 12.1 : 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 $C\colon \mathrm{lin}(\mathcal{F}) \to \mathbb{R}$.
- A weak learner $\mathcal{L}(F)$ that accepts $F \in \mathrm{lin}(\mathcal{F})$ and returns $f \in \mathcal{F}$ with a large value of $-\langle \nabla C(F), f\rangle$.

Let $F_0(\mathbf{x}) := 0$.
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}.

12.2.2 AnyBoost.$\mathbf{L_1}$

The AnyBoost algorithm can return an arbitrary linear combination of elements of the base hypothesis class. Such flexibility has the potential to cause overfitting. Indeed, Theorem 12.2 in the following section provides guaranteed generalization performance for certain classes of cost functions, provided the algorithm returns elements of $\mathrm{co}(\mathcal{F})$, that is *convex* combinations of elements from the base hypothesis class[1]. This consideration motivates Algorithm 12.2—AnyBoost.L_1—a normalized version of AnyBoost that only returns functions in the convex hull of the base hypothesis class $\mathcal{F}$.

Notice that the stopping criterion of AnyBoost.L_1 is $-\langle \nabla C(F_t), f_{t+1} - F_t\rangle \leq 0$, rather than $-\langle \nabla C(F_t), f_{t+1}\rangle \leq 0$. To see why, notice that at every iteration F_t must lie in $\mathrm{co}(\mathcal{F})$. Hence, in incorporating a new component f_{t+1}, we update F_t to $(1-\alpha)F_t + \alpha f_{t+1}$ for some $\alpha \in [0,1]$. Hence, $F_{t+1} = F_t + \alpha(f_{t+1} - F_t)$ which corresponds to stepping in the direction corresponding to $f_{t+1} - F_t$. Geometrically, $-\langle \nabla C(F_t), f_{t+1} - F_t\rangle \leq 0$ implies that the change $F_{t+1} - F_t$ associated with the addition of f_{t+1} is not within $90°$ of $-\nabla C(F_t)$.

1. For convenience, we assume that the class $\mathcal{F}$ contains the zero function, or equivalently, that $\mathrm{co}(\mathcal{F})$ denotes the convex cone containing convex combinations of functions from $\mathcal{F}$ and the zero function.

Algorithm 12.2 : AnyBoost.L_1

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 $C\colon \mathrm{co}(\mathcal{F}) \to \mathbb{R}$.
- A weak learner $\mathcal{L}(F)$ that accepts $F \in \mathrm{co}(\mathcal{F})$ and returns $f \in \mathcal{F}$ with a large value of $-\langle \nabla C(F), f - F\rangle$.

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

12.2.3 AnyBoost.$\mathbf{L_2}$

AnyBoost.L_1 enforces an L_1 constraint on the size of the combined hypotheses returned by the algorithm. Although for certain classes of cost functionals we have theoretical guarantees on the generalization performance of such algorithms (see Section 12.4), from an aesthetic perspective an L_2 constraint is more natural in an inner product space setting. In particular, we can then ask our algorithm to perform gradient descent on a regularized cost functional of the form

$$C(F) + \lambda \|F\|^2,$$

where λ is a regularization parameter, without needing to refer to the individual weights in the combination F (contrast with AnyBoost.L_1). In future work we plan to investigate the experimental performance of algorithms based on L_2 constraints.

With an L_2 rather than L_1 constraint, we also have the freedom to allow the weak learner to return general linear combinations in the base hypothesis class, not just single hypotheses[2]. In general a linear combination $F \in \mathrm{lin}(\mathcal{F})$ will be closer to the negative gradient direction than any single base hypothesis, hence stepping

2. The optimal direction in which to move for AnyBoost.L_1 is always a *pure* direction $f \in \mathcal{F}$ if the current combined hypothesis F_t is already on the convex hull of $\mathcal{F}$. So a weak learner that produces linear combinations will be no more powerful than a weak learner returning a single hypothesis in the L_1 case. This is not true for the L_2 case.

in the direction of F should lead to a greater reduction in the cost function, while still ensuring the overall hypothesis constructed is an element of $\mathrm{lin}\,(\mathcal{F})$.

A weak learner $\mathcal{L}$ that accepts a direction G and attempts to choose an $f \in \mathcal{F}$ maximizing $\langle G, f\rangle$ can easily be converted to a weak learner $\mathcal{L}'$ that attempts to choose an $H \in \mathrm{lin}\,(\mathcal{F})$ maximizing $\langle G, H\rangle$; the details are given in Algorithm 12.3. $\mathcal{L}'$ would then be substituted for $\mathcal{L}$ in the AnyBoost algorithm.

Algorithm 12.3 : $\mathcal{L}'$: a weak learner returning linear combinations

---

Require :

- An inner product space $(\mathcal{X}, \langle,\rangle)$ (with associated norm $\|F\|^2 := \langle F, F\rangle$) containing functions mapping from X to some set Y.
- A class of base classifiers $\mathcal{F} \subseteq \mathcal{X}$.
- A differentiable cost functional $C\colon \mathrm{lin}\,(\mathcal{F}) \to \mathbb{R}$.
- A weak learner $\mathcal{L}(G)$ that accepts a "direction" $G \in S$ and returns $f \in \mathcal{F}$ with a large value of $\langle G, f\rangle$.
- A starting function $F_t \in \mathrm{lin}\,(\mathcal{F})$.

Let $G_0 := -\nabla C(F_t)/\|\nabla C(F_t)\|$.
Let $H_0 := 0$.
for $t := 0$ to T **do**
  Let $h_{t+1} := \mathcal{L}(G_t)$.
  Let $H_{t+1} := \alpha H_t + \beta h_{t+1}$, with the constraints $\|H_{t+1}\| = 1$ and $\langle H_{t+1}, G_t\rangle$ maximal.
  if $\beta = 0$ **then**
    return H_t.
  end if
  Let $G_{t+1} := G_0 - H_{t+1}$.
end for
return H_{T+1}.

12.2.4 AnyBoost and margin cost functionals

Since the main aim of this chapter is optimization of margin cost functionals, in this section we specialize the AnyBoost and AnyBoost.L_1 algorithms of the previous two sections by restricting our attention to the inner product (12.2), the cost (12.1), and $Y = \{\pm 1\}$. In this case,

$$\nabla C(F)(\mathbf{x}) = \{\, 0 \quad \text{if } \mathbf{x} \neq \mathbf{x}_i, i = 1 \ldots m \frac{1}{m} y_i C'(y_i F(\mathbf{x}_i)) \quad \text{if } \mathbf{x} = \mathbf{x}_i,$$

where $C'(z)$ is the derivative of the margin cost function with respect to z. Hence,

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

Any sensible cost function of the margin will be monotonically decreasing, hence $-C'(y_iF(\mathbf{x}_i))$ will always be positive. Dividing through by $-\frac{1}{m^2}\sum_{i=1}^m C'(y_iF(\mathbf{x}_i))$, we see that finding an f maximizing $-\langle\nabla C(F), f\rangle$ is equivalent to finding an f minimizing

$$-\sum_{i=1}^m y_i f(\mathbf{x}_i)\frac{C'(y_iF(\mathbf{x}_i))}{\sum_{i=1}^m C'(y_iF(\mathbf{x}_i))}. \tag{12.5}$$

Since $Y = \{\pm 1\}$, $y_i f(\mathbf{x}_i)$ is either 1 if $f(\mathbf{x}_i) = y_i$ or -1 if $f(\mathbf{x}_i) \neq y_i$. Hence (12.5) can be rewritten as

$$\sum_{i: f(\mathbf{x}_i)\neq y_i} D(i) - \sum_{i: f(\mathbf{x}_i) = y_i} D(i) = 2\sum_{i: f(\mathbf{x}_i)\neq y_i} D(i) - 1,$$

where $D(1), \ldots, D(m)$ is the distribution

$$D(i) := \frac{C'(y_iF(\mathbf{x}_i))}{\sum_{i=1}^m C'(y_iF(\mathbf{x}_i))}.$$

So finding an f maximizing $-\langle\nabla C(F), f\rangle$ is equivalent to finding f minimizing the weighted error

$$\sum_{i: f(\mathbf{x}_i)\neq y_i} D(i).$$

Making the appropriate substitutions in AnyBoost yields Algorithm 12.4, MarginBoost.

For AnyBoost.L_1 we require a weak learner that maximizes $-\langle\nabla C(F), f - F\rangle$ where F is the current convex combination. In the present setting this is equivalent to minimizing

$$\sum_{i=1}^m [F(\mathbf{x}_i) - f(\mathbf{x}_i)]\, y_i D(i)$$

with $D(i)$ as above. Making the appropriate substitutions in AnyBoost.L_1 yields Algorithm 12.5, MarginBoost.L_1.

12.3 A gradient descent view of voting methods

Many of the most successful voting methods are, for the appropriate choice of cost function and step-size, specific cases of the AnyBoost algorithm described above (or its derivatives).

The AdaBoost algorithm (Freund and Schapire, 1997) is arguably one of the most important developments in practical machine learning in the past decade. Many studies (Freund and Schapire, 1996; Quinlan, 1996a; Drucker and Cortes, 1996; Schwenk and Bengio, 1998) have demonstrated that AdaBoost can produce extremely accurate classifiers from base classifiers as simple as decision stumps or

Algorithm 12.4 : MarginBoost

Require :

- A differentiable cost function $C: \mathbb{R} \to \mathbb{R}$.
- A class of base classifiers $\mathcal{F}$ containing functions $f: X \to \{\pm 1\}$.
- A training set $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$ with each $(\mathbf{x}_i, y_i) \in X \times \{\pm 1\}$.
- A weak learner $\mathcal{L}(S, D)$ that accepts a training set S and a distribution D on the training set and returns base classifiers $f \in \mathcal{F}$ with small weighted error $\sum_{i: f(\mathbf{x}_i) \neq y_i} D(i)$.

Let $D_0(i) := 1/m$ for $i = 1, \ldots, m$.
Let $F_0(\mathbf{x}) := 0$.
for $t := 0$ to T **do**
  Let $f_{t+1} := \mathcal{L}(S, D_t)$.
  if $\sum_{i=1}^m D_t(i) y_i f_{t+1}(\mathbf{x}_i) \leq 0$ **then**
    return F_t.
  end if
  Choose w_{t+1}.
  Let $F_{t+1} := F_t + w_{t+1} f_{t+1}$

$$\text{Let } D_{t+1}(i) := \frac{C'\Big(y_i F_{t+1}(\mathbf{x}_i)\Big)}{\sum_{i=1}^m C'\Big(y_i F_{t+1}(\mathbf{x}_i)\Big)}$$

  for $i = 1, \ldots, m$.
end for
return F_{T+1}

as complex as neural networks or decision trees. The interpretation of AdaBoost as an algorithm which performs a gradient descent optimization of the sample average of a cost function of the margins has been examined by several authors (Breiman, 1998; Frean and Downs, 1998; Friedman et al., 1998; Duffy and Helmbold, 1999).

To see that the AdaBoost algorithm (shown in Table 1.2) is in fact MarginBoost using the cost function $C(\alpha) = e^{-\alpha}$ we need only verify that the distributions and stopping criteria are identical. The distribution D_{t+1} from AdaBoost can be rewritten as

$$\frac{\prod_{s=1}^t e^{-y_i w_s f_s(\mathbf{x}_i)}}{m \prod_{s=1}^t Z_s}. \tag{12.6}$$

Since D_{t+1} is a distribution then

$$m \prod_{s=1}^t Z_s = \sum_{i=1}^m \prod_{s=1}^t e^{-y_i w_s f_s(\mathbf{x}_i)} \tag{12.7}$$

and clearly

$$\prod_{s=1}^t e^{-y_i w_s f_s(\mathbf{x}_i)} = e^{-y_i F_t(\mathbf{x}_i)}. \tag{12.8}$$

Algorithm 12.5 : MarginBoost.L_1

Require :

- A differentiable cost function $C: \mathbb{R} \to \mathbb{R}$.
- A class of base classifiers $\mathcal{F}$ containing functions $f: X \to \{\pm 1\}$.
- A training set $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$ with each $(\mathbf{x}_i, y_i) \in X \times \{\pm 1\}$.
- A weak learner $\mathcal{L}(S, D, F)$ that accepts a training set S, a distribution D on the training set and a combined classifier F, and returns base classifiers $f \in \mathcal{F}$ with small weighted error: $\sum_{i=1}^m [F(\mathbf{x}_i) - f(\mathbf{x}_i)]\, y_i D(i)$.

Let $D_0(i) := 1/m$ for $i = 1, \ldots, m$.
Let $F_0(\mathbf{x}) := 0$.
for $t := 0$ to T **do**
  Let $f_{t+1} := \mathcal{L}(S, D_t, F_t)$.
  if $\sum_{i=1}^m D_t(i) y_i [f_{t+1}(\mathbf{x}_i) - F_t(\mathbf{x}_i)] \leq 0$ **then**
    return F_t.
  end if
  Choose w_{t+1}.

  Let $F_{t+1} := \dfrac{F_t + w_{t+1} f_{t+1}}{\sum_{s=1}^{t+1} |w_s|}$.

  Let $D_{t+1}(i) := \dfrac{C'\Big(y_i F_{t+1}(\mathbf{x}_i)\Big)}{\sum_{i=1}^m C'\Big(y_i F_{t+1}(\mathbf{x}_i)\Big)}$

  for $i = 1, \ldots, m$.
end for
return F_{T+1}

Substituting (12.7) and (12.8) into (12.6) gives the MarginBoost distribution for the cost function $C(\alpha) = e^{-\alpha}$. By definition of ϵ_t, the stopping criterion in AdaBoost is

$$\sum_{i: f_{t+1}(\mathbf{x}_i) \neq y_i} D_t(i) \geq \frac{1}{2}.$$

This is equivalent to

$$\sum_{i: f_{t+1}(\mathbf{x}_i) = y_i} D_t(i) - \sum_{i: f_{t+1}(\mathbf{x}_i) \neq y_i} D_t(i) \leq 0,$$

which is identical to the stopping criterion of MarginBoost.

Given that we have chosen f_{t+1} we wish to choose w_{t+1} to minimize

$$\sum_{i=1}^m C(y_i F_t(\mathbf{x}_i) + y_i w_{t+1} f_{t+1}(\mathbf{x}_i)).$$

Differentiating with respect to w_{t+1}, setting this to 0 and solving for w_{t+1} gives

$$w_{t+1} = \frac{1}{2} \ln \left(\frac{\sum_{i: f_{t+1}(\mathbf{x}_i) = y_i} D_t(i)}{\sum_{i: f_{t+1}(\mathbf{x}_i) \neq y_i} D_t(i)} \right).$$

This is exactly the setting of w_t used in the AdaBoost algorithm. So for this choice of cost function it is possible to find a closed form solution for the line search for optimal step-size at each round. Hence, AdaBoost is performing gradient descent on the cost functional

$$C(F) = \frac{1}{m} \sum_{i=1}^{m} e^{-y_i F(\mathbf{x}_i)}$$

with step-size chosen by a line search.

Schapire and Singer (1998) examine AdaBoost in the more general setting where classifiers can produce real values in $[-1, 1]$ indicating their confidence in $\{\pm 1\}$-valued classification. The general algorithm [3] they present is essentially AnyBoost with the cost function $C(yF(\mathbf{x})) = e^{-yF(\mathbf{x})}$ and base classifiers $f : X \to [-1, 1]$.

The ARC-X4 algorithm due to Breiman (1998) is approximately AnyBoost.L_1 with the cost function $C(\alpha) = (1 - \alpha)^5$ with a constant step-size.

Friedman et al. (1998) examine AdaBoost as an approximation to maximum likelihood. From this viewpoint they develop a more direct approximation (LogitBoost) which exhibits similar performance. LogitBoost is AnyBoost with the cost function $C(\alpha) = \log_2(1 + e^{-2\alpha})$ and step-size chosen via a single Newton-Raphson step.

Lee et al. (1996) describe an iterative algorithm for constructing convex combinations of basis functions to minimize a quadratic cost function. They use a constructive approximation result to prove the rate of convergence of this algorithm to the optimal convex combination. This algorithm can be viewed as gradient descent with a quadratic cost function $C(\alpha) = (1 - \alpha)^2$ and step-size decreasing at the rate $1/t$.

Table 12.1 summarizes the cost function and step-size choices for which AnyBoost and its derivatives approximately reduce to existing voting methods.

12.4 Theoretically motivated cost functions

The following definition from (Mason et al., 1999) gives a condition on a cost function $C_N(\cdot)$ that suffices to prove upper bounds on error probability in terms of sample averages of $C_N(yF(\mathbf{x}))$. The condition requires the cost function $C_N(\alpha)$ to lie strictly above the mistake indicator function, $\operatorname{sgn}(-\alpha)$. How close $C_N(\alpha)$ can be to $\operatorname{sgn}(-\alpha)$ depends on a complexity parameter N.

3. They also present a base learning algorithm for decision trees which directly optimizes the exponential cost function of the margin at each iteration. This variant of boosting does not reduce to a gradient descent optimization.

Algorithm	Cost function	Step-size
AdaBoost (Freund and Schapire, 1996)	$e^{-yF(\mathbf{x})}$	Line search
ARC-X4 (Breiman, 1996)	$(1 - yF(\mathbf{x}))^5$	1
ConfidenceBoost (Schapire and Singer, 1998)	$e^{-yF(\mathbf{x})}$	Line search
LogitBoost (Friedman et al., 1998)	$\ln(1 + e^{-2yF(\mathbf{x})})$	Newton-Raphson
Constructive NN algorithm (Lee et al., 1996)	$(1 - yF(\mathbf{x}))^2$	$1/t$

Table 12.1 Summary of existing voting methods which can be viewed as gradient descent optimizers of margin cost functions.

Definition 12.1
A family $\{C_N : N \in \mathbb{N}\}$ of margin cost functions is B-*admissible* for $B \geq 0$ if for all $N \in \mathbb{N}$ there is an interval $I \subset \mathbb{R}$ of length no more than B and a function $\Psi_N : [-1, 1] \to I$ that satisfies

$$\operatorname{sgn}(-\alpha) \leq \mathbf{E}_Z\left(\Psi_N(Z)\right) \leq C_N(\alpha)$$

for all $\alpha \in [-1, 1]$, where $\mathbf{E}_Z(\cdot)$ denotes the expectation when Z is chosen randomly as $Z = (1/N)\sum_{i=1}^{N} Z_i$ with $Z_i \in \{\pm 1\}$ and $\Pr(Z_i = 1) = (1 + \alpha)/2$.

The following theorem from (Mason et al., 1999) gives a high probability upper bound on the generalization error of any convex combination of classifiers in terms of the sample average of $C_N(yF(\mathbf{x}))$ and a complexity term depending on N.

Theorem 12.2
For any B-admissible family $\{C_N : N \in \mathbb{N}\}$ of margin cost functions, any finite hypothesis class H and any distribution $\mathcal{D}$ on $X \times \{\pm 1\}$, with probability at least $1 - \delta$ over a random sample S of m labelled examples chosen according to $\mathcal{D}$, every N and every F in $\operatorname{co}(\mathcal{F})$ satisfies

$$\Pr\left[yF(\mathbf{x}) \leq 0\right] < \mathbf{E}_S\left[C_N(yF(\mathbf{x}))\right] + \epsilon_N,$$

where

$$\epsilon_N = \sqrt{\frac{B^2}{2m}\left(N \ln |\mathcal{F}| + \ln(N(N+1)/\delta)\right)}.$$

A similar result applies for infinite classes $\mathcal{F}$ with finite VC-dimension.

In this theorem, as the complexity parameter N increases, the sample-based error estimate $\mathbf{E}_S\left[C_N(yF(\mathbf{x}))\right]$ decreases towards the training error (proportion of misclassified training examples). On the other hand, the complexity penalty term ϵ_N increases with N. Hence, in choosing the effective complexity N of the combined classifier, there is a trade-off between these two terms. Smaller cost functions give a more favourable trade-off. The left plot of Figure 12.1 illustrates a family $C_N(\cdot)$ of cost functions that satisfy the B-admissibility condition. Notice that these functions are significantly different from the exponential and logit cost functions that are used in AdaBoost and LogitBoost respectively. Unlike the exponential and logit functions, $C_N(\alpha)$ is nonconvex and for large negative margins the value of $C_N(\alpha)$ is significantly smaller.

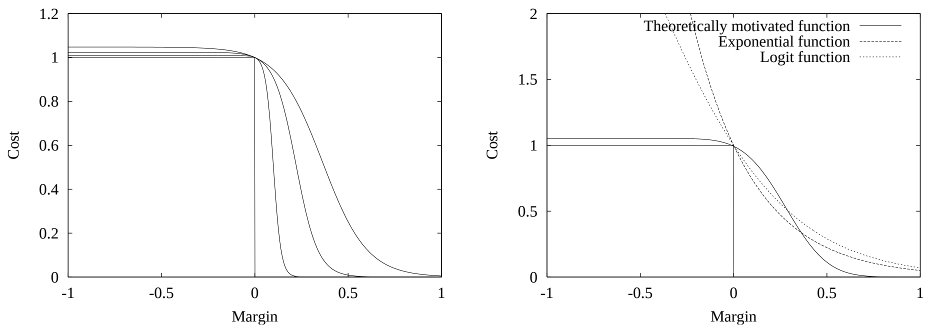

Figure 12.1 Cost functions $C_N(\alpha)$, for $N = 20, 50$ and 100, compared to the function $\mathrm{sgn}(-\alpha)$. Larger values of N correspond to closer approximations to $\mathrm{sgn}(-\alpha)$. The theoretically motivated cost function $C_{40}(\alpha)$ and the exponential and logit cost functions are also plotted together for comparison.

12.5 Convergence results

In this section we prove convergence results for the abstract algorithms AnyBoost and AnyBoost.L_1, 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.

Throughout this section we are interested in the limiting behaviour of AnyBoost (and its derivatives) and thus assume that the algorithms do not terminate after some fixed number of iterations T (although the algorithms can terminate due to internal termination conditions).

12.5.1 Convergence of AnyBoost

The following theorem supplies a specific step-size for AnyBoost and characterizes the limiting behaviour with this step-size.

Theorem 12.3
Let $C\colon \mathrm{lin}(\mathcal{F}) \to \mathbb{R}$ 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 $F, F' \in \mathrm{lin}(\mathcal{F})$). Let $F_0, F_1, \ldots$ be the sequence of combined hypotheses generated

by the AnyBoost algorithm, using step-sizes

$$w_{t+1} := -\frac{\langle \nabla C(F_t), f_{t+1}\rangle}{L\|f_{t+1}\|^2}. \tag{12.9}$$

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

$$\lim_{t\to\infty} \langle \nabla C(F_t), f_{t+1}\rangle = 0.$$

Proof First we need a general lemma.

Lemma 12.4
Let $(\mathcal{X}, \langle,\rangle)$ be an inner product space with squared norm $\|F\|^2 := \langle F, F\rangle$ and let $C: \mathcal{X} \to \mathbb{R}$ be a differentiable functional with $\|\nabla C(F) - \nabla C(F')\| \leq L\|F - F'\|$ for all $F, F' \in \mathcal{X}$. Then for any $w > 0$ and $F, G \in \mathcal{X}$,

$$C(F + wG) - C(F) \leq w \langle \nabla C(F), G\rangle + \frac{Lw^2}{2}\|G\|^2.$$

Proof Define $g: \mathbb{R} \to \mathbb{R}$ by $g(w) := C(F + wG)$. Then $g'(w) = \langle \nabla C(F + wG), G\rangle$ and hence

$$\begin{aligned}
|g'(w) - g'(0)| &= |\langle \nabla C(F + wG) - \nabla C(F), G\rangle| \\
&\leq \|\nabla C(F + wG) - \nabla C(F)\|\|G\| \quad \text{by Cauchy-Schwartz} \\
&\leq Lw\|G\|^2 \quad \text{by Lipschitz continuity of } \nabla C.
\end{aligned}$$

Thus, for $w > 0$,

$$g'(w) \leq g'(0) + Lw\|G\|^2 = \langle \nabla C(F), G\rangle + Lw\|G\|^2$$

which implies

$$\begin{aligned}
g(w) - g(0) &= \int_0^w g'(\alpha)\, d\alpha \\
&\leq \int_0^w \langle \nabla C(F), G\rangle + L\alpha\|G\|^2\, d\alpha \\
&= w \langle \nabla C(F), G\rangle + \frac{Lw^2}{2}\|G\|^2.
\end{aligned}$$

Substituting $g(w) = C(F + wG)$ on the left hand side gives the result. ■

Now we can write:

$$\begin{aligned}
C(F_t) - C(F_{t+1}) &= C(F_t) - C(F_t + w_{t+1} f_{t+1}) \\
&\geq -w_{t+1} \langle \nabla C(F_t), f_{t+1}\rangle - \frac{Lw_{t+1}^2\|f_{t+1}\|^2}{2} \quad \text{by Lemma 12.4.}
\end{aligned}$$

If $\|f_{t+1}\| = 0$ then $\langle \nabla C(F_t), f_{t+1}\rangle = 0$ and AnyBoost will terminate. Otherwise, the greatest reduction occurs when the right hand side is maximized, i.e., when

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

which is the step-size in the statement of the theorem. Thus, for our stated step-size,

$$C(F_t) - C(F_{t+1}) \geq \frac{\langle \nabla C(F_t), f_{t+1} \rangle^2}{2L\|f_{t+1}\|^2}. \tag{12.10}$$

If $-\langle \nabla C(F_t), f_{t+1} \rangle \leq 0$ then AnyBoost terminates. Otherwise, since C is bounded below, $C(F_t) - C(F_{t+1}) \to 0$ which implies $\langle \nabla C(F_t), f_{t+1} \rangle \to 0$.

The next theorem 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 step-sizes given by (12.9) is guaranteed to be a global minimum. It is convenient to assume that the hypothesis class $\mathcal{F}$ is *negation closed*, which means $f \in \mathcal{F}$ implies $-f \in \mathcal{F}$. In this case, a function f_{t+1} that maximizes $-\langle \nabla C(F_t), f_{t+1} \rangle$ always satisfies $-\langle \nabla C(F_t), f_{t+1} \rangle \geq 0$. For ease of exposition, we have assumed that rather than terminating when $-\langle \nabla C(F_T), f_{T+1} \rangle = 0$, AnyBoost simply continues to return F_T for all subsequent time steps t.

Theorem 12.5
Let $C\colon \operatorname{lin}(\mathcal{F}) \to \mathbb{R}$ be a *convex* cost functional with the properties in Theorem 12.3, and let (F_t) be the sequence of combined hypotheses generated by the AnyBoost algorithm with step-sizes given by (12.9). Assume that the weak hypothesis class $\mathcal{F}$ is negation closed 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 the sequence (F_t) satisfies

$$\lim_{t\to\infty} \sup_{f\in\mathcal{F}} -\langle \nabla C(F_t), f \rangle = 0, \tag{12.11}$$

and any accumulation point F of F_t satisfies

$$C(F) = \inf_{G\in\operatorname{lin}(\mathcal{F})} C(G). \tag{12.12}$$

Proof Equation (12.11) follows immediately from Theorem 12.3. For the proof of (12.12) we need the following more general lemma:

Lemma 12.6
Let C be a differentiable convex cost function on an inner product space $(\mathcal{X}, \langle , \rangle)$ with norm $\|F\|^2 = \langle F, F \rangle$. Let $\mathcal{M}$ be any linear subspace of $\mathcal{X}$ and let $\mathcal{M}^\perp$ denote the perpendicular subspace to $\mathcal{M}$ ($\mathcal{M}^\perp = \{G \in \mathcal{X}\colon \langle G, F \rangle = 0 \ \forall F \in \mathcal{M}\}$). If $F \in \mathcal{M}$ satisfies

$$\nabla C(F) \in \mathcal{M}^\perp$$

then

$$C(F) = \inf_{G \in \mathcal{M}} C(G).$$

Proof Consider $G \in \mathcal{M}$. By the convexity of C, for all $0 \leq \epsilon \leq 1$,

$$C\left((1-\epsilon)F + \epsilon G\right) - \left((1-\epsilon)C(F) + \epsilon C(G)\right) \leq 0.$$

Taking the limit as $\epsilon \to 0$ yields,

$\langle G - F, \nabla C(F) \rangle \leq C(G) - C(F)$.

Since $G - F \in \mathcal{M}$ and $\nabla C(F) \in \mathcal{M}^{\perp}$, this implies $C(G) \geq C(F)$. ■

Now let F be an accumulation point of F_t. By Lipschitz continuity of $\nabla C(F)$ and (12.11),

$$\sup_{f \in \mathcal{F}} - \langle \nabla C(F), f \rangle = 0,$$

which by the negation closure of $\mathcal{F}$ implies $\langle \nabla C(F), f \rangle = 0$ for all $f \in \mathcal{F}$, hence $\nabla C(F) \in \operatorname{lin}(\mathcal{F})^{\perp}$. Thus $F \in \operatorname{lin}(\mathcal{F})$ and $\nabla C(F) \in \operatorname{lin}(\mathcal{F})^{\perp}$, which by Lemma 12.6 implies (12.12).

12.5.2 Convergence of AnyBoost.$\mathbf{L_1}$

The following theorem supplies a specific step-size for AnyBoost.L_1 and characterizes the limiting behaviour under this step-size regime.

Theorem 12.7
Let C be a cost function as in Theorem 12.3. Let $F_0, F_1, \ldots$ be the sequence of combined hypotheses generated by the AnyBoost.L_1 algorithm, using step-sizes

$$w_{t+1} := \frac{- \langle \nabla C(F_t), f_{t+1} - F_t \rangle}{L \| f_{t+1} - F_t \|^2 + \langle \nabla C(F_t), f_{t+1} - F_t \rangle} \tag{12.13}$$

Then AnyBoost.L_1 either terminates at some finite time t with $- \langle \nabla C(F_t), f_{t+1} - F_t \rangle \leq 0$, or $C(F_t)$ converges to a finite value C^*, in which case

$$\lim_{t \to \infty} \langle \nabla C(F_t), f_{t+1} - F_t \rangle = 0.$$

Proof Note that the step-sizes w_t are always positive. In addition, if the w_t are such that $\sum_{s=1}^{t} w_s < 1$ for all t then clearly the second case above will apply. So without loss of generality assume $\sum_{s=1}^{t} w_s = 1$. Applying Lemma 12.4, we have:

$$\begin{aligned}
C(F_t) - C(F_{t+1}) &= C(F_t) - C\left(\frac{F_t + w_{t+1} f_{t+1}}{1 + w_{t+1}}\right) \\
&= C(F_t) - C\left(F_t + \frac{w_{t+1}}{1 + w_{t+1}} \left(f_{t+1} - F_t\right)\right) \\
&\geq - \frac{w_{t+1}}{1 + w_{t+1}} \langle \nabla C(F_t), f_{t+1} - F_t \rangle
\end{aligned}$$

$$-\frac{L}{2}\left[\frac{w_{t+1}}{1+w_{t+1}}\right]^2 \|f_{t+1}-F_t\|^2. \tag{12.14}$$

If $-\langle \nabla C(F_t), f_{t+1}-F_t\rangle \leq 0$ then the algorithm terminates. Otherwise, the right hand side of (12.14) is maximized when

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

which is the step-size in the statement of the theorem. Thus, for our stated step-size,

$$C(F_t) - C(F_{t+1}) \geq \frac{\langle \nabla C(F_t), f_{t+1}-F_t\rangle^2}{2L\|f_{t+1}-F_t\|^2},$$

which by the lower-boundedness of C implies $\langle \nabla C(F_t), f_{t+1}-F_t\rangle \to 0$. ∎

The next theorem shows that if the weak learner can always find the best weak hypothesis $f_t \in \mathcal{F}$ on each round of AnyBoost.L_1, and if the cost function C is convex, then AnyBoost.L_1 is guaranteed to converge to the global minimum of the cost. As with Theorem 12.5, we have assumed that rather than terminating when $-\langle f_{T+1}-F_T, \nabla C(F_T)\rangle \leq 0$, AnyBoost.$L_1$ simply continues to return F_T for all subsequent time steps t.

Theorem 12.8
Let C be a *convex* cost function with the properties in Theorem 12.3, and let (F_t) be the sequence of combined hypotheses generated by the AnyBoost.L_1 algorithm using the step-sizes in (12.13). Assume that the weak hypothesis class $\mathcal{F}$ is negation closed and that on each round the AnyBoost.L_1 algorithm finds a function f_{t+1} maximizing $-\langle \nabla C(F_t), f_{t+1}-F_t\rangle$. Then

$$\lim_{t\to\infty} \sup_{f\in\mathcal{F}} -\langle \nabla C(F_t), f-F_t\rangle = 0, \tag{12.15}$$

and any accumulation point F of the sequence (F_t) satisfies

$$C(F) = \inf_{G\in \mathrm{co}\,(\mathcal{F})} C(G), \tag{12.16}$$

where $\mathrm{co}\,(\mathcal{F})$ is the set of all convex combinations of hypotheses from $\mathcal{F}$.

Proof Equation (12.15) follows immediately from Theorem 12.7. Now let F be an accumulation point of F_t. By (12.15) and continuity of $\nabla C(F)$, for all $f \in \mathcal{F}$,

$$\langle \nabla C(F), f-F\rangle = 0,$$

or equivalently $\langle \nabla C(F), f\rangle = \langle \nabla C(F), F\rangle$ for all $f \in \mathcal{F}$. Using the same argument as in the proof of Lemma 12.6, any $G \in \mathrm{co}\,(\mathcal{F})$ has

$$\langle G-F, \nabla C(F)\rangle \leq C(G) - C(F).$$

But because $\mathcal{F}$ is negation closed, we can write $G = \sum w_i f_i$ where all w_i are positive and $\sum w_i = 1$. Then

$$\langle G - F, \nabla C(F)\rangle = \sum w_i \langle f_i, \nabla C(F)\rangle - \langle F, \nabla C(F)\rangle = 0.$$

It follows that $C(G) \geq C(F)$. ■

At this point we should note that for cost functions which are nonconvex (like those motivated by the theoretical result of Section 12.4) we can only guarantee convergence to a local minimum.

12.6 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 (1998) demonstrated that running AdaBoost for hundreds of thousands of rounds can lead to significant overfitting, while a number of authors (Dietterich, 1998; Rätsch et al., 1998; Bauer and Kohavi, 1997; Maclin and Opitz, 1997) 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 Sections 12.4 and 12.5 we propose a new algorithm (DOOM II) based on MarginBoost.L_1 which performs a gradient descent optimization of

$$\frac{1}{m}\sum_{i=1}^{m}(1 - \tanh(\lambda y_i F(\mathbf{x}_i))), \tag{12.17}$$

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 (12.17) 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 family of cost functions (parameterized by N) shown in Figure 12.1. Using the family from Figure 12.1 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.

Choosing a value of λ corresponds to choosing a value of the complexity parameter N in Theorem 12.2. It is a data dependent 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 without overfitting. Conversely, choosing a small value of λ corresponds to a belief that a high complexity classifier can only avoid overfitting if it has large margins.

Algorithm 12.6 : DOOM II

Require :
- A class of base classifiers $\mathcal{F}$ containing functions $f: X \to \{\pm 1\}$.
- A training set $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$ with each $(\mathbf{x}_i, y_i) \in X \times \{\pm 1\}$.
- A weak learner $\mathcal{L}(S, D, F)$ that accepts a training set S, a distribution D on the training set and a combined classifier F, and returns base classifiers $f \in \mathcal{F}$ with small error: $\sum_{i=1}^{m} [F(\mathbf{x}_i) - f(\mathbf{x}_i)] y_i D(i)$.
- A fixed small step-size ϵ.

Let $D_0(i) := 1/m$ for $i = 1, \ldots, m$.
Let $F_0 := 0$.
for $t := 0$ to T **do**
  Let $f_{t+1} := \mathcal{L}(S, D_t, F_t)$.
  if $\sum_{i=1}^{m} D_t(i)[y_i f_{t+1}(\mathbf{x}_i) - y_i F_t(\mathbf{x}_i)] \leq 0$ **then**
    Return F_t.
  end if
  Let $w_{t+1} := \epsilon$.

  Let $F_{t+1} := \dfrac{F_t + w_{t+1} f_{t+1}}{\sum_{s=1}^{t+1} |w_s|}$.

  Let $D_{t+1}(i) := \dfrac{1 - \tanh^2(\lambda y_i F_{t+1}(\mathbf{x}_i))}{\sum_{i=1}^{m} (1 - \tanh^2(\lambda y_i F_{t+1}(\mathbf{x}_i)))}$

  for $i = 1, \ldots, m$.
end for

In the above implementation of DOOM II we are using 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.

It is worth noting that since the l_1-norm of the classifier weights is fixed at 1 for each iteration and the cost function has the property that $C(-\alpha) = 1 - C(\alpha)$, the choice of λ is equivalent to choosing the l_1-norm of the weights while using the cost function $C(\alpha) = 1 - \tanh(\alpha)$.

Given that the normalized sigmoid cost function is nonconvex 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 MarginBoost.$L_1$ algorithm will strike a local minimum at the first step.

Lemma 12.9
Let $C: \mathbb{R} \to \mathbb{R}$ be any cost function satisfying $C(-\alpha) = 1 - C(\alpha)$. If MarginBoost.$L_1$ can find the optimal weak hypothesis f_1 at the first time step, it will terminate at the next time step, returning f_1.

Proof Assume without loss that $C'(0) < 0$. With $F_0 = 0$, $\langle \nabla C(F_0), f \rangle = (C'(0)/m) \sum_{i=1}^m y_i f(\mathbf{x}_i)$ and so by assumption, f_1 will satisfy

$$\sum_{i=1}^m y_i f_1(\mathbf{x}_i) = \inf_{f \in \mathcal{F}} \sum_{i=1}^m y_i f(\mathbf{x}_i)$$

and $F_1 = f_1$. Now $C(-\alpha) = 1 - C(\alpha) \Rightarrow C'(-\alpha) = C'(\alpha)$, and since f_1 only takes the values ± 1, we have for any f:

$$\langle \nabla C(F_1), f - F_1 \rangle = \frac{C'(1)}{m} \sum_{i=1}^m y_i (f(\mathbf{x}_i) - f_1(\mathbf{x}_i)).$$

Thus, for all $f \in \mathcal{F}$, $-\langle \nabla C(F_1), f - F_1 \rangle \leq 0$ and hence MarginBoost.L_1 will terminate, returning f_1. ■

A simple technique for avoiding this local minimum is to apply some notion of randomized initial conditions in the hope that the gradient descent procedure will then avoid this local minimum. Either the initial margins could be randomized or a random initial classifier could be chosen from $\mathcal{F}$. Initial experiments showed that both these techniques are somewhat successful, but could not guarantee avoidance of the single classifier local minimum unless many random initial conditions were tried (a computationally intensive prospect).

A more principled 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 (Blake et al., 1998). 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 produced by 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. The typical behaviour of the test error as DOOM II proceeds is shown in Figure 12.2 for various values of λ. For small values of λ the test error converges to a value much worse than AdaBoost's test error. As λ is increased to the optimal value the test errors decrease. In the case of the `sonar` data set used in Figure 12.2 the test errors for AdaBoost and DOOM II with optimal λ are similar. Of course, with AdaBoost's adaptive step-size it converges much faster than DOOM II (which uses a fixed step-size).

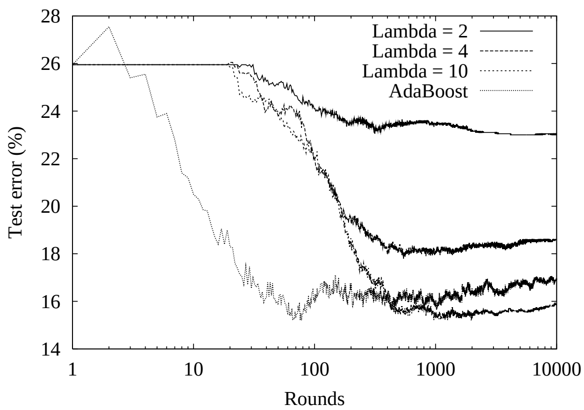

Figure 12.2 Test error for the `sonar` data set over 10000 rounds of AdaBoost and DOOM II with $\lambda = 2, 4$ and 10.

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 provided in Table 12.2. The attained test errors are shown for each data set for a single stump, AdaBoost applied to stumps and DOOM II applied to stumps with 0%, 5% and 15% label noise. A graphical representation of the difference in test error between AdaBoost and DOOM II is shown in Figure 12.3. 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 generally 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 12.4 shows comparisons

		sonar	cleve	ionosphere	vote1	credit	breast-cancer	pima-indians	hypo1	splice
	Examples	208	303	351	435	690	699	768	2514	3190
	Attributes	60	13	34	16	15	9	8	29	60
0%	Stump	26.0	26.9	17.6	6.2	14.5	8.1	27.6	7.0	22.6
Label	AdaBoost	16.0	16.8	10.1	**3.5**	14.1	4.2	25.8	**0.5**	6.4
Noise	DOOM II	**15.8**	**16.5**	**9.7**	4.5	**13.0**	**3.0**	**25.1**	0.7	**5.7**
5%	Stump	30.4	29.0	21.7	10.6	18.0	12.1	29.7	12.4	26.4
Label	AdaBoost	**23.0**	21.6	16.7	9.6	17.5	9.0	**27.9**	8.6	13.9
Noise	DOOM II	23.3	**20.3**	**14.6**	**9.4**	**17.0**	**8.0**	**27.9**	**7.1**	**12.1**
15%	Stump	36.6	33.7	27.7	19.3	25.1	20.3	34.2	21.0	31.1
Label	AdaBoost	33.8	29.8	26.8	**19.0**	25.1	18.6	33.3	18.3	22.2
Noise	DOOM II	**32.6**	**27.6**	**25.9**	**19.0**	**24.7**	**17.6**	**33.1**	**17.1**	**20.3**

Table 12.2 Summary of test errors for a single stump, AdaBoost stumps and DOOM II stumps with varying levels of label noise on nine UCI data sets. The best test error for each data set is displayed in bold face. Note that since DOOM II uses an independent validation set to choose the cost function parameter λ, we are comparing it to a version of AdaBoost modified to use an independent validation set for early stopping.

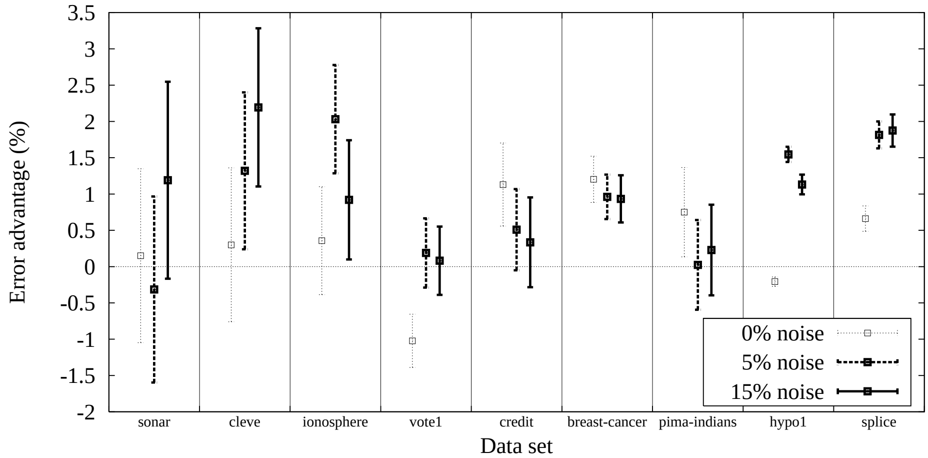

Figure 12.3 Summary of test error advantage (with standard error bars) of DOOM II over AdaBoost with varying levels of noise on nine UCI data sets.

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.

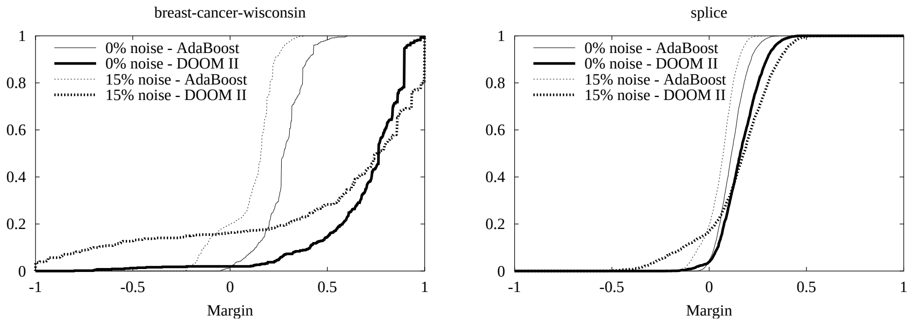

Figure 12.4 Margin distributions for AdaBoost and DOOM II with 0% and 15% label noise for the `breast-cancer` and `splice` data sets.

Given that AdaBoost suffers from overfitting and minimizes 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 with 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 12.5 shows the variation in the normalized sigmoid cost function, the exponential cost function and the test error for AdaBoost for four 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 all four data sets the minimum of AdaBoost's test error and the minimum of the normalized sigmoid cost very nearly coincide. In the `sonar` and `labor` data sets AdaBoost's test error converges and overfitting does not occur. For these data sets both the normalized sigmoid cost and the exponential cost converge, although in the case of the `sonar` data set the exponential cost converges significantly later than the test error. In the `cleve` and `vote1` data sets AdaBoost initially decreases the test error and then increases the test error (as overfitting occurs). For these data sets the normalized sigmoid cost mirrors this behaviour, while the exponential cost converges to 0.

To examine the effect of step-size we compare AdaBoost to a modified version using fixed step-sizes, called ε-AdaBoost. In ε-AdaBoost, the first classifier is given weight 1 and all others thereafter are given weight ε. A comparison of the test errors of both of these algorithms for various values of ε is shown in Figure 12.6.

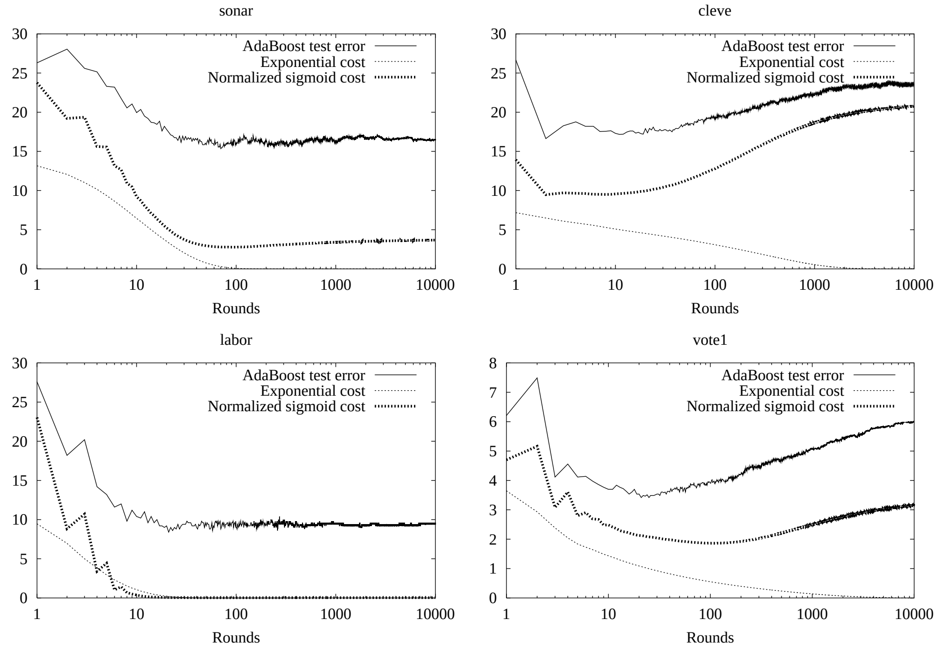

Figure 12.5 AdaBoost test error, exponential cost and normalized sigmoid cost over 10000 rounds of AdaBoost for the `sonar`, `cleve`, `labor` and `vote1` data sets. Both costs have been scaled in each case for easier comparison with test error.

As expected, changing the value of the fixed step size ε simply translates the test error curve on the log scale and does not significantly alter the minimum test error.

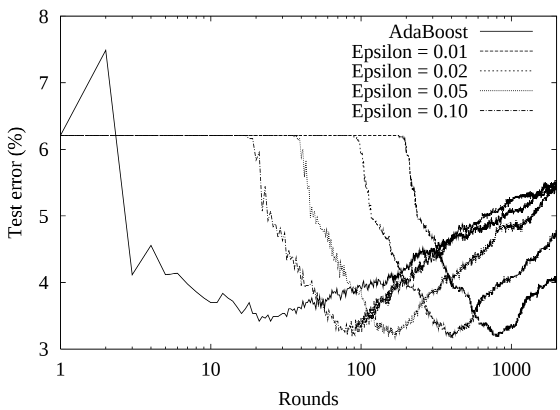

Figure 12.6 Test error for the `vote1` data set over 2000 rounds of AdaBoost and ε-AdaBoost for $\varepsilon = 0.01, 0.02, 0.05$ and 0.10.

12.7 Conclusions

We have shown how most 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, and a related algorithm, AnyBoost.L_1, for generating convex combinations from the base hypothesis class. Prescriptions for the step-sizes in these algorithms guaranteeing convergence to the optimal linear or convex combination were given.

For cost functions depending only upon the *margins* of the classifier on the training set, AnyBoost and AnyBoost.L_1 become MarginBoost and MarginBoost.L_1. We showed that many existing algorithms for combining classifiers can be viewed as special cases of MarginBoost.L_1; each algorithm differing only in its choice of margin cost function and step-size. In particular, AdaBoost is MarginBoost.L_1 with e^{-z} as the cost function of the margin z, and with a step-size equal to the one that would be found by a line search.

The main theoretical result from (Mason et al., 1999) provides bounds on the generalization performance of a convex combination of classifiers in terms of training sample averages of certain, sigmoid-like, cost functions of the margin. This suggests that algorithms such as Adaboost that optimize an exponential margin cost function are placing too much emphasis on examples with large negative margins, and that this is a likely explanation for overfitting, particularly in the presence of label noise.

Motivated by this result, we derived DOOM II—a further specialization of MarginBoost.L_1—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.

In future we plan to investigate the properties of AnyBoost.L_2, mentioned in Sec. 12.2.3. Although we do not have theoretical results on the generalization performance of this algorithm, viewed in the inner product space setting an L_2 constraint on the combined hypothesis is considerably more natural than an L_1 constraint. In addition, the inner product perspective on boosting can be applied to any inner product space, not just spaces of functions as done here. This opens up the possibility of applying boosting in many other machine learning settings.

Acknowledgments

This research was supported by the ARC. 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.