Multiple Randomized Classifiers: MRCL

Yali Amit [*] and Gilles Blanchard [†]

November 5, 2001

[*]Department of Statistics, University of Chicago, Chicago, IL, 60637. Supported in part by the Army Research Office under grant DAAH04-96-1-0061 and MURI grant DAAH04-96-1-0445,

[†]Department of Statistics, University of Chicago, Chicago, IL, 60637; Supported by the Army Research Office under grant DAAH04-96-1-0061.

Abstract

We discuss a family of protocols for producing and aggregating multiple classifiers based on sequential reweighting of the data and randomization, a particular example of which is the AdaBoost algorithm. We point out that all successful protocols are producing weakly dependent classifiers conditional on class and provide empirical evidence that this leads both to a decrease in the bias of the aggregate classifier relative to the optimal individual classifier, as well as a decrease in variance with respect to the training sample. We argue that all protocols can be viewed as producing a sample from some distribution on the space of classifiers. There is a simple tradeoff in terms of two key expectations with respect to this distribution: the average and variance of the margin conditional on class. The second expectation is a measure of the dependence between pairs of classifiers drawn at random from the distribution. We illustrate these points using a simple artificial example in the plane and the NIST Standard Database 2, on which we report error rates of .7%.

1 Introduction

In recent years a multitude of algorithms have been suggested in which a large number of classifiers are trained on the same training sample and then combined to provide an improved *aggregate classifier*, Amit et al. (1995), Amit and Geman (1997), Blanchard (1999), Breiman (1996), Breiman (1999), Breiman (1998), Dietterich (1998), Dietterich and Bakiri (1995), Friedman et al. (1998), Ho et al. (1994), Schapire et al. (1998). Mostly the ideas have been presented in the context of classification trees but similar procedures can be implemented with other classifiers. We argue that in all cases the aggregate classifiers can be viewed as a random sample from a distribution $\mathbf{Q}$ on the space of classifiers, which depends on the *protocol* and on the *training set,* hence the acronym Multiple Randomized Classifiers (MRCL). In many cases the sample is truly i.i.d. For example in Amit and Geman (1997) the split at each node of a tree is chosen as the best from among a small random sample of all possible predictors, this is revisited in Breiman (1999). The protocols in Blanchard (1999), Breiman (1996), Breiman (1998), Dietterich (1998), Ho (1998), suggest other forms of randomization. We also include in our setting the AdaBoost algorithm of Schapire et al. (1998), and other sequential reweighting schemes. Each of these schemes by definition produces a convex combination of classifiers, which can be considered as a probability distribution on the space of classifiers. Furthermore these schemes typically produce a dynamical process in the space of classifiers which asymptotically stabilizes at a limiting distribution on this space and does not depend on the initial point or on the first iterations of the process.

Intuitively randomization is a method to obtain multiple points of view or "experts", which are then aggregated using some form of voting. The intuition behind sequential reweighting schemes is the need for experts which concentrate on the difficult examples, or alternatively experts who actually "disagree" on these difficult examples.

The success of MRCL is striking. Using a simple set of binary local features (described in section 4.2) one classification tree can achieve at most 5% error on the NIST data base with 100,000 training data points. On the other hand 100 trees, trained in under one hour, when aggregated yield an error rate under .7%. We believe that this stems from the fact that a sample from a very rich and diverse set of classifiers produces on average *weakly dependent classifiers conditional on class.* We will show that this weak conditional dependence, measured in terms of covariances among the classifiers drawn from $\mathbf{Q}$, leads to reductions in both bias and variance. There is a tradeoff in terms of two expectations calculated with respect to $\mathbf{Q}$. The first, called the *average conditional margin* (ACM), is the difference between the expected weight on a class c and the expected weight on the other classes, conditional class c. The second, called the *average conditional covariance* (ACC), can be viewed on one hand as the degree to which two classifiers sampled independently from $\mathbf{Q}$ have a low conditional covariance, and on the other hand as the degree of concentration of the aggregate classifier around the average conditional mean.

Experiments lead us to the following conclusions:

- A variety of sequential reweighting schemes combined with randomization achieve significant reductions in ACC, which despite the accompanying reductions in ACM yield significant improvements in classification rates.

- The particular details of these protocols, or a detailed analysis of their behavior on the training set is of little importance. They perform equally well when initialized with random weights on the training data, and when the initial classifiers of the sequence are omitted. The crucial aspect appears to be that these schemes are implicitly aiming to reduce the covariance of the new classifier with respect to the current aggregate classifier, while maintaining a reasonable classification rate. Asymptotically this leads to low values of the ACC.

- Randomization in this context serves as a smoothing mechanism so that the training procedure does not focus too much on a small number of problematic examples. Furthermore, with large datasets, randomization allows us to produce 100's of trees in minutes as opposed to several days.

- The reduction in ACC also contributes to greater stability of the algorithms. The variability in quantities such as the error rate, or the ACM and ACC, across randomly sampled training sets is very small when the ACC is small.

The paper is organized as follows. In section 2 we present an overview of some recent work on the subject. In section 3 we offer a description of the various protocols. In section 4 we describe two datasets with which we provide empirical evidence for our conclusions. In section 5 we define the average conditional margin (ACM) and average conditional covariance (ACC), and bound the classification error with an elementary Chebychev-type inequality making use of these two quantities. In section 6 we present experimental results, focusing on the role played by these two quantities, in terms of classification error and stability with respect to training data.

In section 7 we discuss the sequential reweighting schemes as dynamic processes, demonstrating their stability with respect to initial conditions, and point to an implicit covariance minimization which take place in all such schemes.

2 Related work

The emphasis on weak conditional dependence presented in this paper has appeared in Amit et al. (1995), Amit and Geman (1997), Breiman (1998). In these papers the role of weak conditional dependence in bias reduction is discussed in the context of shape recognition. This concept has not been addressed in most of the recent discussions of aggregation, e.g. Breiman (1996), Breiman (1998), Friedman et al. (1998), Schapire et al. (1998). Only recently has it been studied in Dietterich (1998), Breiman (1999) and Murua (1999). In the first, this dependence is measured in an *unconditioned* way. It is therefore unclear what this type of dependence is measuring and how it connects to the performance of the aggregation methods. In Breiman (1999) a measure of dependence is suggested which overcomes this limitation and in the two class case is very similar to a measure of conditional dependence. The conclusions are very similar to those of Amit and Geman (1997). In Murua (1999) the basic paradigm is that of modeling the conditional dependence between the classifiers, but there is an attempt to define this in terms conditional exponential moments. Given these moments one obtains more precise error bounds, however the moments are very hard to estimate robustly.

We review here other explanations for the success of aggregation methods. The emphasis in Breiman (1996) and Breiman (1998), is on the reduction of variance with respect to the training set. In some sense it is argued that multiple classifier methods are supposed to mimic a semi-ideal situation where one has a large number of independent training sets of the same size, each of which would is used to produce one classifier. These would then be aggregated. This motivated the idea of bootstrap resampling ("bagging") for generating multiple classifiers. However if the base classifier is very stable, this variance reduction scheme alone will yield negligible improvement. The classifiers are not sufficiently independent conditional on class, in other words the ACC is too high (see e.g. table 4). Nonetheless we agree that there is a significant element of variance reduction, in other words improved stability with respect to training data, as discussed in section 6.2.

Concerning the now popular AdaBoost algorithm, all analyses of this method proposed in the literature clearly point to the fact that it is effectively reducing bias by constructing a compound classifier in the space of convex combinations of base classifiers, which is of course richer than the space of individual classifiers themselves. This appears in the initial motivations for AdaBoost in Freund and Shapire (1997) as well as in recent attempts to provide an interpretation in the framework of generalized additive models in Friedman et al. (1998). Although there are pretty clear theoretical reasons why AdaBoost dramatically decreases the classification error on training, there is still a lack of theoretical explanation as to why this algorithm also does so well on the test set.

In particular, Friedman et al. (1998) provide a more transparent derivation of the sequential reweighting scheme of AdaBoost. In the two class case where the two classes are labeled $-1, 1$ and the classifier outputs a positive or negative value the protocol can be viewed as a procedure to iteratively minimize a cost function of the form

$$C_{n+1} = E \exp[-Y(H^n(X) + \beta_{n+1} h_{n+1}(X))].$$

Here H^n is the current aggregate, h_{n+1} is the new classifier, Y is the class label, and β_{n+1} is a free parameter chosen to yield maximum decrease of the cost once h_{n+1} has been chosen. The expectation is with respect to the empirical distribution defined by the *training set*. It has more recently be proven in Collins et al. (2000) that, in the ideal case where the base classifier having least weighted error is picked at each iteration, the AdaBoost protocol indeed leads to asymptotic minimization of the cost function $\mathcal{C}(H) = E[\exp(-YH(X))]$, where H is any linear combination of base classifiers with positive coefficients, and again the expectation is the empirical expectation on the training set.

The problem with these analyses is that they are entirely dedicated to the training set. In the context of a simple example used below we show that C_n actually diverges exponentially on the test set while converging exponentially to zero on the training set. This is demonstrated graphically in figure 1 where $\log(C_n)$ is plotted both for the training sample and the test sample in a run of AdaBoost for the 2-dimensional dataset defined in section 4.1. We emphasize that this phenomenon is observed in all experiments with a variety of data sets. Perhaps more importantly, in the above cost function $\mathcal{C}(H)$, the aggregate classifier H is not normalized, i.e. the weights of the convex combination are not required to sum to 1. This means that in the case of a separable training set, *any* aggregate classifier H_0 that perfectly fits the data is such that $\mathcal{C}(\lambda H_0) \to 0$ when $\lambda \to \infty$. Therefore this property does not give much information about the asymptotic aggregate classifier. Thus we argue in section 7 that the sole minimization of the above cost function cannot explain the success of AdaBoost and that it is the *dynamical* behavior which is more important.

The analysis in Schapire et al. (1998) does deal with the generalization properties

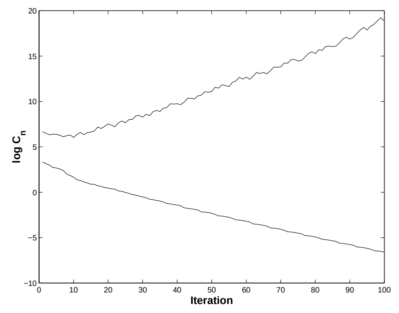


Figure 1: $\log(C_n)$ as estimated from training data (lower curve) and from test data.

of multiple classifier schemes. It is based on an extension of a Vapnik-type inequality in the space of convex combinations of classifiers, and on the empirical cumulative distribution function of the margins. The margin of an example is the difference between the weight on the correct class and the weight on the largest among all other classes. The bounds obtained are very conservative and depend on the lower tails of the empirical distribution of the margin. The empirical evidence given below suggests that simple statistics such as the ACM and ACC, may already explain many of the phenomena and even allow us to predict the relative performance of various protocols in a number of cases. The tradeoff between the individual performance of each classifier and the conditional dependence among the classifiers is expressed more directly through these statistics, though the bounds in Schapire et al. (1998) can be seen as pointing to a tradeoff between obtaining larger margins and reducing the spread of their distribution. Furthermore, we also observe a noticeable improvement in terms of the stability of the different algorithms when the ACC is low, which does not seem directly explained by the type of bounds cited above.

Thus, although we agree with the analysis of Schapire et al. (1998) that sequential reweighting schemes are constructing an aggregate classifier by making use of the rich structure of convex combinations of base classifiers, the alternative and in a sense more "naive" point of view we emphasize in this paper is to study this convex combination as a probability distribution on the space of classifiers. We then concentrate only on elementary, but robust, properties of this distribution and show that these properties are common to all successful algorithms used to build multiple classifiers. The comparable performance of the different sequential reweighting schemes means that the explanation must lie in something common to all protocols, not the particular details of any one of them. We believe it is due to the effective sampling from a distribution in the space of classifiers with a good balance between the ACM and ACC.

3 Protocols for generating multiple classifiers

The notations used are the following: X denotes an object to be classified, Y is its associated class belonging to the set $\{1, \ldots, K\}$. We assume that we have at hand a (finite or infinite) set $\mathcal{H}$ of classifiers. A classifier $h \in \mathcal{H}$ is a function associating to every object X a probability distribution over $\{1, \ldots, K\}$, that is, a vector of sum one belonging to $[0,1]^K$, that will be denoted by $h(X,d), d = 1, \ldots, K$. Also we set $C_h(X) = \text{Argmax}_{d=1,\ldots,K}\, h(X,d)$ and $e_h(X) = \mathbf{1}_{\{C_h(X) \neq Y\}}$, namely the indicator of an error. In the experiments we will concentrate on classification trees but will refer to general classifiers whenever the explicit form of the classifiers is unimportant.

Sequential reweighting schemes (SRS)

Given a training set $\mathcal{L} = \{(X_1, Y_1), \ldots, (X_L, Y_L)\}$ a sequence of classifiers $h_1, \ldots, h_N$ can be produced using one of several sequential reweighting schemes.

All sequential schemes start with an initial weight vector W_1 for the training data, typically, but not necessarily, $W_1(l) = 1/L, l = 1, \ldots, L$. A classifier h_1 is built to fit the training data. The training error ε_1 of the classifier is obtained. The misclassified points in the training set have their weight increased according to some prescribed formula depending on the error ε_1, and the correctly classified points have their weight decreased. The weights of the training data are renormalized to sum to 1 and produce the new weight vector W_2. After $n-1$ classifiers $h_1, \ldots, h_{n-1}$ have been produced and given an updated weight vector W_n, a new classifier h_n is built to fit the weighted training sample. The weighted training error ε_n of h_n is evaluated in terms of the weighted training data. The particular formulas for updating the weights are as follows. Let $e_n(l) = e_{h_n}(X_l)$ denote the error of classifier h_n on example l.

1. **AdaBoost, (AB).**

$$W_{n+1}(l) = \begin{cases} \frac{(1-\varepsilon_n)}{\varepsilon_n} W_n, \text{ if } e_n(l) = 1 \\ W_n \text{ otherwise.} \end{cases} \tag{1}$$

2. **Modified AdaBoost, (MAB).** For multi-class problems it is often the case that $\varepsilon_n > .5$ and the weight on misclassified points as prescribed in item 1 would decrease. We therefore use a modified updating scheme of the form

$$W_{n+1}(l) = \begin{cases} \frac{1}{\varepsilon_n} W_n, \text{ if } e_n(l) = 1 \\ W_n, \text{ otherwise,} \end{cases} \tag{2}$$

3. **Additive reweighting, (AR).** Instead of updating the weights by a multiplicative factor we use an additive update with a non-linearity to maintain positivity. This protocol has to our knowledge not been proposed elsewhere, and we designed it for the mainly for comparison with existing protocols.

$$W_{n+1} = \max(0, W_n + e_n(l) - \varepsilon_n), \tag{3}$$

4. **Blackwell, (BL).** In addition to the weights a cost $R_n(l)$ is updated at each step for each data point from which the weights are computed.

$$R_{n+1}(l) = R_n(l) + e_n(l) - \varepsilon_n, \text{ and } W_{n+1} = \max(0, R_{n+1}(l)). \tag{4}$$

This reweighting scheme arises in game theory where it is called Blackwell's strategy. In the case when the best classifier in $\mathcal{H}$ is chosen at each step (in terms of the weighted training data), it can be proved that the aggregate classifier converges towards a convex combination that maximizes the minimum attained margin on the training data. For a more detailed account see to Blanchard (2001). Links between AdaBoost, classification and the theory of games are also discussed in Schapire and Freund (1996) and Breiman (1997).

Randomization

It is possible to introduce randomization in the training procedure of each classifier. In the context of neural networks the architecture can be randomized (see Amit and

Mascaro (2001)). In the present context we focus on trees and randomization is achieved as follows:

Randomized splits. At each node a random subset of size **S** of all candidate predictors is entertained. The best split from this subset is chosen. See for example Amit and Geman (1997).

At the two extremes are

- *Deterministic sequential reweighting*, where no randomization is performed in producing the classifiers, namely **S** is of the size of the predictor vector. This will be denoted as $\mathbf{S} = \mathit{all}$.
- *Randomization* (**RAN**) where multiple randomized classifiers are produced independently, namely no reweighting is performed.

Other protocols for producing multiple classifiers have been suggested in the literature, including error correcting codes (ECOC), see Dietterich and Bakiri (1995), Bayesian sampling, see Blanchard (1999), Chipman et al. (1998) and bootstrap resampling (Bagging), see Breiman (1996).

Aggregating the classifiers

In all the protocols, the aggregate classifier is produced by averaging the output vectors $h_n(X, \cdot)$ over the sequence of classifiers. Given a set of coefficients $(\beta_i)_{i=1,\dots,N}$, we define the aggregated distribution vector $H^N(X, \cdot)$ and classifier $C^N(X)$ as

$$H^N(X,d) = \sum_{n=1}^{N} \beta_n h_n(X,d) \quad \text{and} \quad C^N(X) = \mathrm{Argmax}_d H^N(X,d). \tag{5}$$

In the original AdaBoost protocol the coefficients are defined as

$$\beta_n = \frac{1}{2} \log\left((1-\varepsilon_n)/\varepsilon_n\right), \tag{6}$$

where ε_n is the weighted empirical error of h_n. For all other protocols we have simply used $\beta_i \equiv 1/N$.

4 Data sets

The empirical evidence presented below will be illustrated on two datasets. The first is a simple two class problem in the plane with a very limited number of linear classifiers. The goal primary goal here is visualizing the behavior of the distribution $\mathbf{Q}$. The second involves a real problem of hand written digit recognition, using the NIST Standard Database 2.

4.1 Artificial example

In order to make the problem non-trivial we choose two classes in the plane distributed according to a mixture of Gaussians. The first class has means at $(2, 2)$ and $(-2, -2)$, with weights $1/4, 3/4$ respectively, and the second has means at $(-2, 2)$ and $(2, -2)$ with weights $1/2, 1/2$ respectively. The skewed weights on one of the classes are specifically chosen to demonstrate certain properties of sequential reweighting. We purposely pick a small size training set, 20 points per class to study variance properties of these classifiers with respect to the training set. Figure 2 shows a training sample of 20 points per-class alongside a test sample of 500 points per class.

The classifiers employed are stumps (depth one trees) determined by a finite number of separating lines in the plane. A line is defined by a couple (θ, τ), where θ is an angle and τ a scalar. Setting v_θ the unit vector in direction θ, the associated classifier predicts class 1 for the "positive" half-plane $\{x \in R^2 | \langle v_\theta, x \rangle - \tau > 0\}$ and class 2 for the "negative" half-plane. Specifically we define 10 uniformly spaced angles with 11 uniformly spaced thresholds. Thus in total there are 110 possible features or splits. Note that this model can also be interpreted as a perceptron architecture.

4.2 NIST handwritten character database

The NIST Standard Database 2 contains 223000 binary images of handwritten digits. We use this data set to demonstrate the power of a wide range of multiple classifier protocols for real data. No sophisticated pre-processing is performed prior to the feature extraction on the data. Each image which is of dimension higher than 32×32 is subsampled to that size. Smaller images are not modified. A collection of 62 binary local features $X_\alpha, \alpha = 1, \ldots, 62$ are defined. These represent a coarse to fine

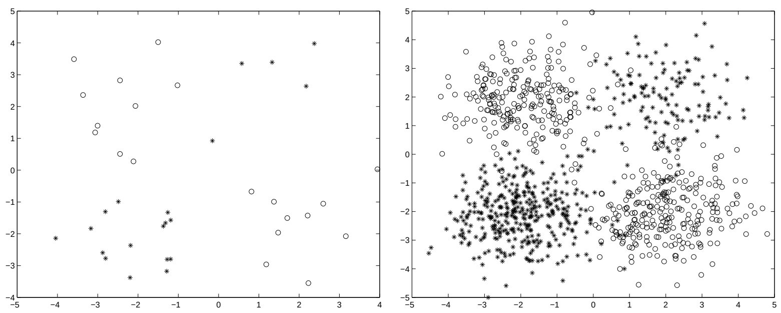


Figure 2: Left: A training sample of 20 data points per class. The first class is a mixture of two Gaussians centered at $(-2,-2),(2,2)$ with weights $3/4, 1/4$ respectively. The second class is an equal mixture with centers at $(-2,2),(2,-2)$. Right: A test sample of 500 per class.

quantization of the population of 4×4 *binary sub-images*. A detailed description of these features can be found in Amit and Geman (1997). The response of a feature at a point x in the image is denoted $X_\alpha(x)$. Several features could be 'on' (i.e. $X_\alpha(x) = 1$) at the same point. In order to ensure invariance of the classifier to a wide range of deformations the features are 'spread' by defining disjunctions over large neighborhoods. Specifically let $\tilde{X}_\alpha(x) = \max_{y \in N_s(x)} X_\alpha(y)$, where $N_s(x)$ is an $s \times s$ neighborhood of x, here we use $s = 15$.

The predictor vector used for the classification trees is then

$$\{\tilde{X}_\alpha(x), \alpha = 1, \dots, 62, x \in G\},$$

where G is the 32×32 image grid. Thus the predictor vector here is of size $62 \cdot 32 \cdot 32 = 63,488$. In growing classification trees for this data set we have varied two parameters: **S** - the size of the random sample of predictors from which the optimal predictor is chosen at each node of the tree, and **m** - the minimal weight on the *second* largest class to allow a node to split. Note that this is the only stopping rule used in tree growing, the larger **m** the shallower the trees.

5 Asymptotic analysis in the number of classifiers

In this section we present an analysis of the sequential reweighting schemes, asymptotically in the number of classifiers. In other words we assume that the empirical average of the process $h_1, h_2, \dots, h_N, \dots$ as given in equation (5), can be replaced by an expected value with respect to a limiting distribution **Q** on the space of classifiers which is determined by the protocol and by the training sample $\mathcal{L}$.

We therefore replace H^N defined in equation (5) and the associated classifier by

$$H_{\mathbf{Q}}(X) = E_{\mathbf{Q}}[h(X)], \text{ and } C_{\mathbf{Q}}(X) = \text{Argmax}_d H_{\mathbf{Q}}(X, d). \tag{7}$$

Note however that the same analysis can be applied to the empirical distribution $\mathbf{Q}^N$ determined by the first N classifiers.

5.1 A variance decomposition

In the analysis to come we will make use of the population conditional probability distribution of a point X given $Y = c$, denoted by P_c, and the associated conditional expectation and variance operators will be denoted E_c and Var_c. Define the vectors of average aggregates conditional on class c as

$$M_c(d) = E_c\left[H_{\mathbf{Q}}(X, d)\right] = E\left[H_{\mathbf{Q}}(X, d)|Y = c\right], \tag{8}$$

for $d = 1, \ldots, K$. The *average conditional margin* (ACM) for class c is defined as

$$\theta_c = \min_{d \neq c}(M_c(c) - M_c(d)). \tag{9}$$

We assume that $\theta_c > 0$. This assumption is very weak since it involves only the average over the population of class c. It is quite natural since one would not expect good classification results when it is violated. Indeed as shown below it is satisfied in all cases.

Given that $\theta_c > 0$, the error rate for class c depends on the extent to which the aggregate classifier $H_{\mathbf{Q}}(X, d)$ is concentrated around $M_c(d)$ for each $d = 1, \ldots, K$. The simplest measure of concentration is the variance of $H_{\mathbf{Q}}(X, d)$ with respect to the distribution P_c. Using Chebyshev's inequality we write a coarse bound on the misclassification probability with respect to P_c as follows.

$$\begin{aligned} P_c(C_{\mathbf{Q}}(X) \neq c) &\leq P_c(H_{\mathbf{Q}}(X, c) < M_c(c) - \theta_c/2) \\ &\quad + \sum_{d \neq c} P_c(H_{\mathbf{Q}}(X, d) > M_c(d) + \theta_c/2) \\ &\leq \sum_{d=1}^{K} P_c(|H_{\mathbf{Q}}(X, d) - M_c(d)| > \theta_c/2) \\ &\leq \frac{4}{\theta_c^2} \sum_{d=1}^{K} Var_c[H_{\mathbf{Q}}(X, d)]. \end{aligned} \tag{10}$$

Of course Chebyshev's inequality is coarse and will not give very sharp results in itself, be we state it here as a landmark pointing to the relative importance of margin and variance, and to the tradeoff between the two quantities.

We rewrite each of the variance terms of the last equation as

$$\begin{aligned} Var_c[E_{\mathbf{Q}}h(X,d)] &= E_c[E_{\mathbf{Q}}h(X,d)]^2 - [E_c E_{\mathbf{Q}}h(X,d)]^2 \\ &= E_{\mathbf{Q}\otimes\mathbf{Q}}E_c[h_1(X,d)h_2(X,d)] - E_{\mathbf{Q}\otimes\mathbf{Q}}\big[E_c[h_1(X,d)]E_c[h_2(X,d)]\big] \\ &= E_{\mathbf{Q}\otimes\mathbf{Q}}Cov_c[h_1(X,d),h_2(X,d)] \doteq \gamma_{c,d}, \qquad (11) \end{aligned}$$

where the notation $E_{\mathbf{Q}\otimes\mathbf{Q}}$ means that h_1, h_2 are two classifiers sampled *independently* from the distribution $\mathbf{Q}$. We can therefore interpret this variance term as the conditional covariance of two classifiers independently sampled from $\mathbf{Q}$. We call this quantity the *average conditional covariance* (ACC). Even if $\mathbf{Q}$ is a discrete distribution, such as that provided by a particular run of N classifiers, when it is supported on a moderate number of classifiers, it is dominated by the conditional covariances of which there are order N^2, and not the conditional variances of which there are order N.

5.2 Conditional and unconditional dependence

It should be emphasized that two classifiers, provided that they achieve reasonable classification rate (that is, better than just picking a class at random) *will not be unconditionally independent.* If we do not know the class label of a point, and vector $(h_1(X,i))_i$ is large at class c, then we actually change our expectations regarding $(h_2(X,i))_i$. On the other hand if we were given in advance the class label Y, then knowing $h_1(X)$ would hardly affect our guess about $h_2(X)$. This is the motivation behind the notion of weak conditional dependence.

This is in contrast to the measure of dependence introduced in Dietterich (1998), which involves the unconditional covariance. The κ statistic used there is

$$\kappa(h_1,h_2) = \frac{\sum_d Cov[h_1(X,d),h_2(X,d)]}{1-\sum_d Eh_1(X,d)Eh_2(X,d)},$$

A simple decomposition of the numerator yields:

$$Cov[h_1(X,d),h_2(X,d)] = ECov[h_1(X,d),h_2(X,d)|Y] + Cov[E[h_1(X,d)|Y],E[h_2(X,d)|Y]]. \qquad (12)$$

The first term is related to equation (11) and should therefore be small on average over classifiers. However the second term can be relatively large since on average we expect both classifiers to produce similar results in terms of the conditional expectation $E_c h_i(X,d), d = 1, \ldots, K$. Also this second term in no way affects classification rates.

In Breiman (1999) a different measure of dependence is used which although unconditional is very closely related to the conditional one defined here. It is defined as

$$\begin{aligned} &Var\left[H_{\mathbf{Q}}(X,Y) - \max_{d \neq Y} H_{\mathbf{Q}}(X,d)\right] \\ &= E_{\mathbf{Q}\otimes\mathbf{Q}} Cov\Big(h_1(X,Y) - \max_{d\neq Y} E_{\mathbf{Q}} h(X,d)\ ;\ h_2(X,Y) - \max_{d \neq Y} E_{\mathbf{Q}} h(X,d)\Big). \end{aligned} \tag{13}$$

Here $\mathbf{Q}$ denotes some distribution on classifiers. In this case the second term in the covariance decomposition of equation (12) will be very small, because all that is observed is the expected difference between the weight on the true class and the mode outside the true class, this should be more or less the same for all classes. Another attraction of this measure of dependence is that the quantity

$$P\left[H_{\mathbf{Q}}(X,Y) - \max_{d\neq Y} H_{\mathbf{Q}}(X,d) < 0\right]$$

is precisely the misclassification probability. However this condition is not given directly in terms of the relationship between two independently sampled classifiers. It depends in a complex way on the expected classifier.

6 Experimental results

6.1 The ACC-ACM tradeoff

For the experimental results we use the classifier H^N defined in equation 5, which for large N is taken as an approximation to $H_{\mathbf{Q}}$. The data in the test set are used to approximate population means. Let $\mathcal{L}_c$ denote the points of class c in the test set, Define

$$\hat{M}_c(d) = \frac{1}{|\mathcal{L}_c|} \sum_{\mathcal{L}_c} H^N(X,d), \tag{14}$$

Protocol	N=1	N=10	N=20	N=50	N=100	N=1000	N=1000	N=1000
	S=2	**S**=2	**S**=2	**S**=2	**S**=2	**S**=2	**S**=50	**S**=all
$\hat{c}$	.590	.762	.840	.893	.903	.900	.895	.877
$\hat{\theta}$	.178	.197	.189	.175	.163	.135	.223	.234
$\hat{V}$	.261	.035	.017	.008	.006	.004	.013	.021
Std. $\hat{\theta}$	.171	.043	.034	.020	.027	.022	.038	.046

AdaBoost

Protocol	N=1	N=1	N=20	N=1000	N=1000	N=1000
	S=all	**S**=2	**S**=2	**S**=2	**S**=10	**S**=20
$\hat{c}$	.744	.590	.639	.642	.743	.752
$\hat{\theta}$	.488	.197	.178	.176	.347	.399
$\hat{V}$	.320	.262	.037	.027	.109	.148
Std. $\hat{\theta}$	.100	.171	.036	.028	.040	.043

RAN

Table 1: Results for various protocols. 20 samples per class in training, and 500 per class in testing. **S** is the number of randomized splits (maximum number is 110). All values are averages over 50 training sets, except for Std. $\hat{\theta}$ which is the standard deviation of $\hat{\theta}$ over the 50 training sets.

and

$$\hat{V}_c(d) = \frac{1}{|\mathcal{L}_c|}\sum_{\mathcal{L}_c}[H^N(X,d)]^2 - [M_c(d)]^2. \tag{15}$$

These are simply the empirical versions of the quantities defined in equations 8 and 11 respectively. Averaging over the K classes define

$$\hat{\theta} = \frac{1}{K}\sum_{c=1}^{K}[\hat{M}_c(c) - \operatorname*{Argmax}_{d\neq c}\hat{M}_c(d)],$$

and

$$\hat{V} = \frac{1}{K}\sum_{c=1}^{K}\sum_{d=1}^{K}\hat{V}_c(d).$$

The quantities $\tilde{V}$ and $\tilde{\theta}$ will denote the same quantities estimated from the training set. Finally $\hat{c}$ will denote the classification rate estimated from the test set.

For the synthetic example table 1 provides $\hat{c}, \hat{\theta}, \hat{V}$ for a variety of protocols averaged over 50 randomly generated training sets of size 20 points per class. The last row gives the standard deviation of $\hat{\theta}$ over the 50 training sets and is relevant for quantifying the stability with respect to training data (see below). The important observation is $\hat{V}$ decrease dramatically as a function of the number N of classifiers. Let $\mathbf{Q}^N$ denote the empirical distribution on classifiers obtained after N iterations (the average of N classifiers). For a small number of classifiers the distribution $\mathbf{Q}^N$ is highly concentrated and the variance of the aggregate classifier is dominated by the variances of the individual classifiers. The magnitude of these can be read off the first columns of table 1, where only 1 classifier is produced (the variance is of the order of .3). As the number of classifiers increases, $\mathbf{Q}^N$ becomes more spread out and stabilizes. The variance of the aggregate becomes dominated by the covariances, and drops to an asymptotic value (which depends on the protocol) — as low as .004 for certain protocols.

The same phenomenon is observed in experiments on the NIST database. In the first set of experiments the number of training samples is 100,000 and the results are tested on another 50,000 samples. For a particular choice of the parameters, $\mathbf{S} = 1000$ and $\mathbf{m} = 100$ and using the modified AdaBoost protocol (MAB), in figure 3 we plot $\hat{c}, \hat{\theta}, \hat{V}$ as a function of the number of trees. It is interesting to note that $\hat{\theta}$ *decreases* with the number of trees until it more or less reaches the asymptotic value at $N = 100$. The decrease in $\hat{V}$ compensates for this and the classification rate increases and asymptotes at .993. This is a highly competitive classification rate on this data set.

The outcomes for a number of other protocols where both $\mathbf{m}$ and $\mathbf{S}$ are varied with pure randomization (RAN) and 3 forms of sequential reweighting (MAB, AR, BL), are shown in table 2. Again one notices the tradeoff between $\hat{\theta}$ and $\hat{V}$, and it is clear that the best performance is achieved with relatively low values of $\hat{\theta}$. The poor performance of the protocols using randomization alone is due to the relatively high value of $\hat{V}$. Although randomization goes a long way towards decreasing the average covariance it seems to be consistently missing some problematic points and hence maintaining a higher conditional correlation between the trees. On the other hand for randomization with no reweighting the training error rate is a very reliable predictor of the test error rate, in sharp contrast to the reweighting protocols where the training error rate is typically zero. Higher classification rates can be obtained with randomization alone if deeper trees are grown. In section 6.3 we revisit the

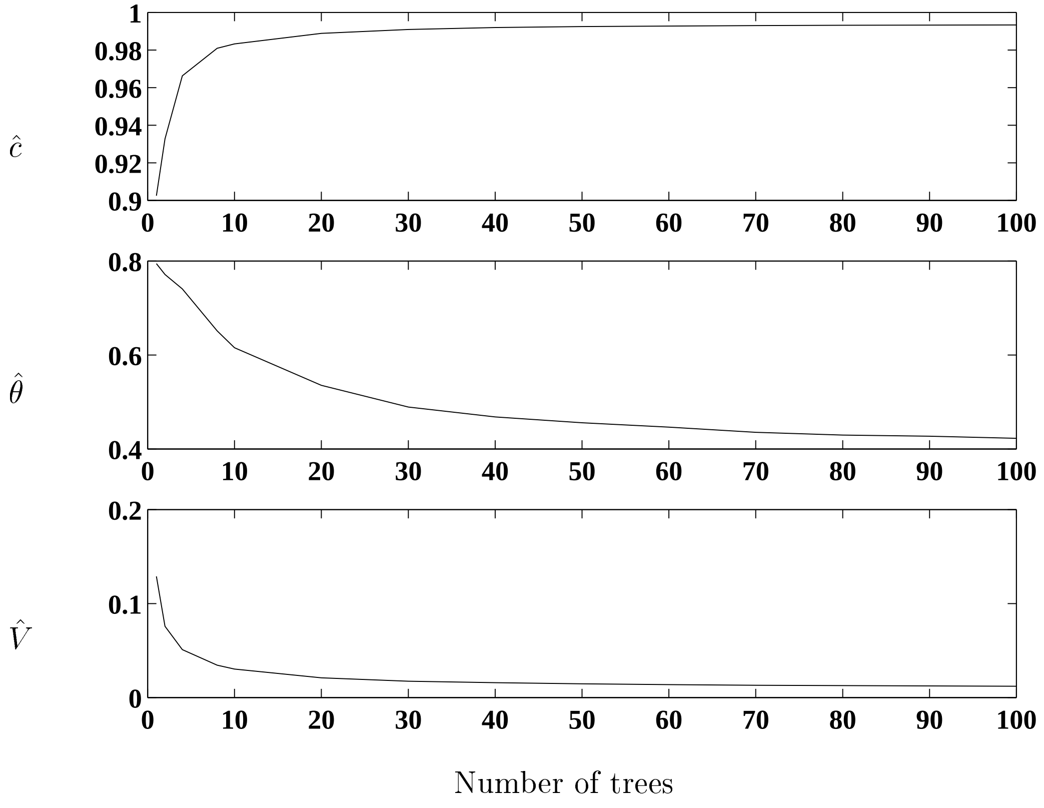


Figure 3: Behavior of $\hat{c}, \hat{\theta}, \hat{V}$ as a function of the number of trees. MAB, **m**=100, **R**=1000

comparison between different protocols.

Protocol	**m**=100 **S**=1000 (RAN)	**m**=20 **S**=20 (RAN)	**m**=100 **S**=1000 (MAB)	**m**=20 **S**=20 (MAB)	**m**=100 **S**=1000 (AR)	**m**=20 **S**=20 (AR)	**m**=100 **S**=1000 (BL)	**m**=20 **S**=20 (BL)
$\hat{c}$	.9735	.9840	.9933	.9929	.9923	.9919	.9925	.9927
$\hat{\theta}$	.799	.669	.422	.586	.563	.667	.461	.619
$\hat{V}$	.042	.032	.012	.020	.017	.022	.012	.019

Table 2: Classification rates, M and V for a number of sequential reweighting protocols

In the NIST experiment an actual functional relationship emerges. A log-linear regression was performed of the test error rate $\hat{e} = 1 - \hat{c}$ on $\tilde{\theta}$ and $\tilde{V}$, i.e. the average margin and average conditional variance *estimated from training.* The 8 experiments reported in table 2 were used and some additional ones, all in all 24 experiments. The regression of $\log \hat{e}$ against $\log \tilde{V}$ and $\log \tilde{\theta}$ yields a .942 R^2 statistic with the following

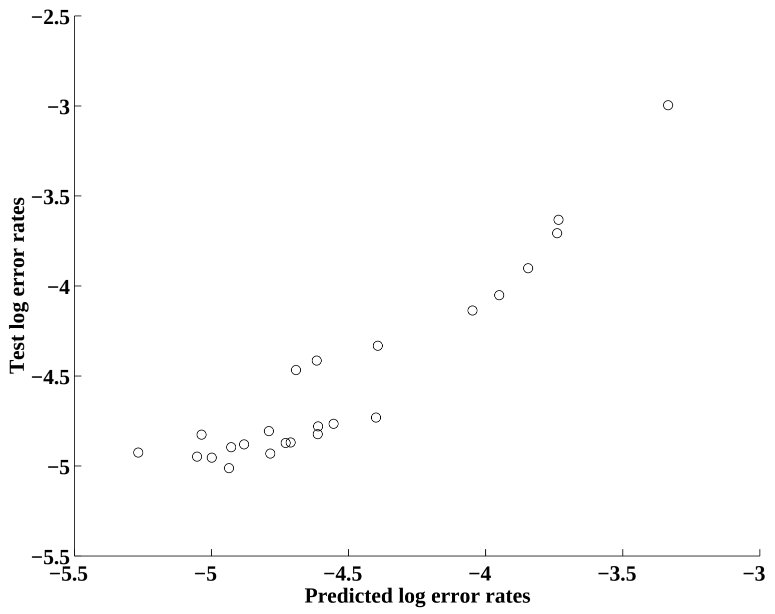


Figure 4: Scatter plot of predicted log error rates vs. real log error rates.

regression equation:

$$\log(\hat{e}) = -.024 - 1.00 \log(\tilde{\theta}) + 1.19 \log(\tilde{V}).$$

The scatter plot of predicted vs. real error rates for the 24 protocols is shown in figure 4 with varying values of **S** various stopping rules on the growth of the trees and different types of sequential reweighting.

Of course this relationship does not depend only on training statistics, since the coefficients have been estimated using the test error; however once the coefficients are estimated it is quite reliable in evaluating the relative performance of two competing protocols.

6.2 Stability

The variability with respect to the training set, expressed for example in the variability of the classification rate or in the average margin is greatly reduced by the aggregation protocols. For the synthetic data set table 1 shows the standard deviation of the margins, which are estimated on a large test set, with respect to classifiers produced with 50 random training sets. These clearly decrease with the number of classifiers in the aggregate, reaching a small asymptotic value. Furthermore, although the error in estimating those small values is probably too large to draw strong conclusions, it appears that this asymptotic value is smaller when the ACC is smaller, suggesting that protocols reducing the asymptotic ACC also lead to greater stability.

	std. $\hat{c}$	std. $\hat{\theta}$	std. $\hat{V}$
1 tree	.0239	.0334	.0074
5 trees	.0041	.0134	.0015
10 trees	.0021	.0099	.0009
100 trees	.0008	.0057	.0004

Table 3: Standard deviation of $\hat{e}$, $\hat{V}$, and $\hat{\theta}$ over 100 training sets, computed for one tree and aggregate classifiers of 5, 10 and 100 trees. Training sets size was 5000. 100 independent training sets were sampled. Protocol used. $\mathbf{S} = 20$, $\mathbf{m} = 20$, modified AdaBoost.

The same experiment was performed on the NIST data set. Random samples of size 5000, were produced 100 times from the larger 100,000 point training data set. For each training sample we grew 100 trees with a fixed protocol. In table 3 we show the decrease in the standard deviation over the 100 experiments of $\hat{e}, \hat{V}, \hat{\theta}$. between one tree and aggregating over 5, 10 and all 100 trees. The same phenomenon appears in all protocols.

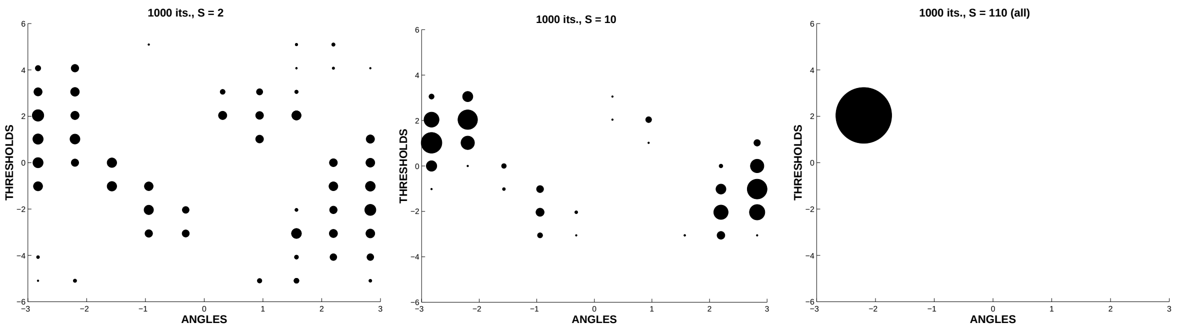
6.3 Comparing the different protocols

The RAN protocol which uses small random subsets of the predictors at each node of the tree, without any sequential reweighting, is discussed in Amit and Geman (1997) in a shape recognition context. There however there is a virtually infinite number of predictors, each corresponding to a spatial arrangement of local image features. Two random predictors drawn from this huge pool will typically be very weakly correlated. Also, at a given node of the tree there is a variety of possible predictors achieving good performance, so that the RAN protocol will naturally build very different — and hence not very correlated — trees. This seems the basis of the success of simple randomization in that context. This is also discussed in Breiman (1999). There it is suggested that in many other contexts one should generate large numbers of more complex features from the initial input features in order to create such a pool of very weakly correlated splits.

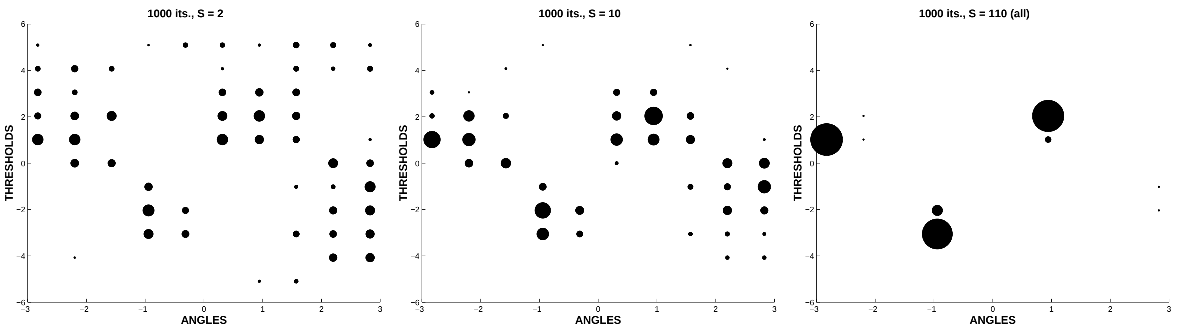
By contrast, the 110 features we use in the synthetic example are not weakly correlated. The linear classifier that separates the lower left hand cluster of class 1 from class 2 clearly outperforms all the other base classifiers in terms of (individual) classification error. Therefore, whenever more than one predictor is sampled those close to the optimal one tend to be chosen. As a consequence, the upper right hand cluster of class 1 is ignored and the aggregate classifier performs poorly. This is expressed in the high variance of the aggregate as summarized in the lower panel of table 1. As mentioned above one could create more complex features, or grow depth 2 trees and then the problem would become trivial. There is still however much to be learned from this very constrained situation. In other contexts it is not always possible to create more complex decision boundaries within the framework of the base classifier. Interestingly, we observe that in this same setup AdaBoost, in particular with randomization, produces good classification rates, and the variance of the aggregate is much smaller than with randomization alone, see table 1 top panel. The corresponding decision boundaries are shown in figure 6.

The distributions produced by the different protocols on the set of 110 planes are shown in figure 5, all using the same training set. The first row shows the distribution produced by RAN with $\mathbf{S} = 2, 10, 110$. Note how the supports differ for each value of $\mathbf{S}$. The second row shows the same for AdaBoost. Note how the latter puts more weight on the upper right hand region, which apparently takes care of the smaller cluster of the first class.

However, AdaBoost does not always produce better classifiers; in particular, the *deterministic* version can produce over fitting, and is especially sensitive to small training sets (see also Dietterich (1998)). At times, strange decision boundaries are produced, where the process concentrates on a cycle in the space of classifiers as exhibited in the right hand panel in the second row of figure 5. The boosting process eventually cycles between the three planes shown in the figure; the corresponding decision boundaries are shown in the left hand panel of 6. In less extreme cases $\mathbf{Q}$ is concentrated on a small number of planes. In all our experiments, with a variety of data sets, better classification rates are obtained with the randomized sequential reweighting protocols. It is of interest that this also corresponds to lower values of the average conditional covariance and of the training variance: both decrease as $\mathbf{S}$



RAN, **S**=2,10,110



AB, **S**=2,10,110

Figure 5: Distributions on the space of planes given by different protocols. The x-axis is the angle θ and the y-axis is the threshold τ. The area of the circles is proportional to the probability of the corresponding point. Decision regions associated to panels 2,4 and 6 are shown in fig. 6.

decreases (see table 1), and the best rates are observed for $\mathbf{S} = 2$.

Similar phenomena are observed in the NIST experiment. From table 2 we see that pure randomization does not perform as well as sequential reweighting with randomization, when the conservative stopping rules of $\mathbf{m} \geq 20$ are used. However if deeper trees are produced, with say $\mathbf{m} = 3$, the performance improves and reaches just over a classification rate of .99. We have not been able to achieve rates over .992 with randomization alone using the predictors described above.

In the NIST context we carried out some more experiments with a smaller training set of 1000 training points. Here again we observe that sequential reweighting performs much better with randomization, see table 4. Deterministic sequential reweighting is using all predictors to choose the best split so that the individual trees have higher classification rates and consequently the value of $\hat{\theta}$ is larger. However this is accompanied by an excessive increase in $\hat{V}$ and ultimately the error rate is

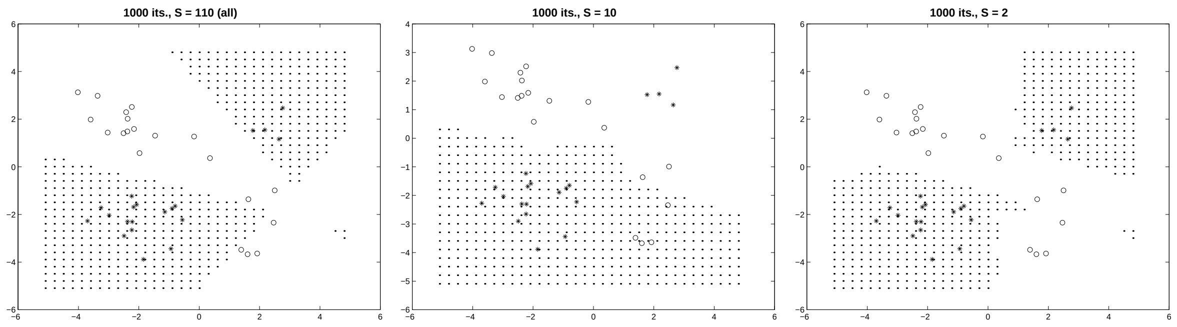


Figure 6: Left: Decision boundaries obtained after 1000 iterations of (deterministic) AdaBoost. A 3-cycle has been reached. Middle: Decision boundaries obtained after 1000 iterations by the randomized splits algorithm **S** = 10: the small cluster of class one is 'ignored'. Right: Decision boundaries obtained by the randomized AdaBoost algorithm **S** = 2. These decision regions correspond to distributions 6,2 and 4 shown on fig. 5.

Protocol	**m**=20 **S**=all MAB	**m**=20 **S**=all BAG	**m**=50 **S**=20 MAB	**m**=20 **S**=20 MAB	**m**=20 **S**=500 MAB	**m**=5 **S**=500 MAB
$\hat{c}$	.93	.776	.964	.966	.962	.940
$\hat{\theta}$	.448	.557	.188	.591	.331	.07
$\hat{V}$	.046	.17	.01	.04	.020	.004

Table 4: Classification for various protocols using 1000 training data.

larger. We also show the outcome of a Bagging experiment which suffers from similar drawbacks. This however is a property of the smaller data sets. On the larger data sets deterministic sequential reweighting performed nearly as well, with one major drawback. Training one tree with 100,000 training data and using all 62,000 predictors at each node, takes close to an hour on a PENTIUM III 500Mhz. The other randomized protocols take 10-20 minutes to train all 100 trees, depending on the number of questions used at each node.

7 Sequential reweighting schemes

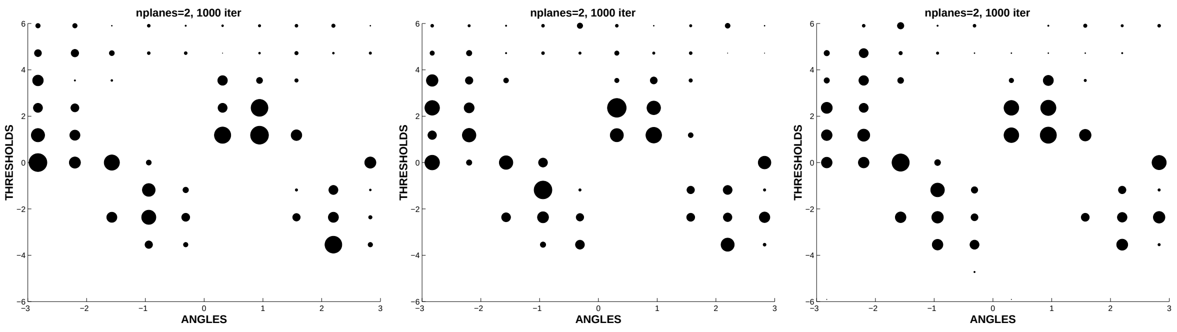
We now take a closer look at the behavior of the sequential reweighting schemes as dynamic processes on the space of classifiers, and try to provide an explanation as to how they achieve reductions in ACC.

7.1 Sequential reweighting as a dynamical process

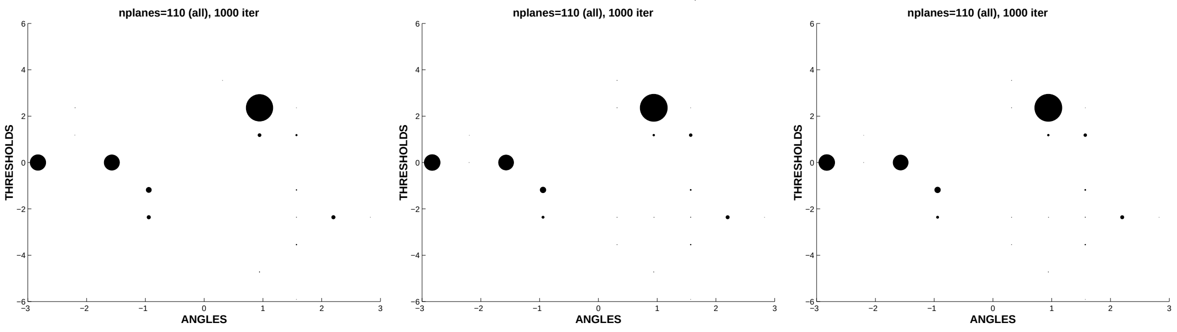
Let $\mathcal{H}$ denote the set of all base classifiers, and $\mathcal{W}$ be the set of all probability measures on the training set. Any of the deterministic sequential reweighting protocols define a dynamical process on the product space $\mathcal{H} \times \mathcal{W}$. A given couple $(h_n, W_n) \in \mathcal{H} \times \mathcal{W}$ produces a new couple (h_{n+1}, W_{n+1}), which is completely determined by the data set and the tree growing protocol. When randomization is introduced the process becomes a Markov chain on $\mathcal{H} \times \mathcal{W}$. This raises the question as to whether some kind of ergodicity could be observed in these processes. More precisely, for a fixed training set we looked at two specific properties: the dependence of the final aggregate classifier on the initial conditions (the initial weights given to the examples), and how the aggregate classifier is modified when an initial part of the process is deleted.

For both the synthetic classification problem and for the NIST dataset, under all protocols, we have observed that *sequential reweighting does not depend on the initial condition.* We start with randomized weights on the training set, drawing independently from the uniform distribution on $[0, 1]$, and then normalize. The final classification rates are extremely close, independent of the initial choice of weights. In the context of the synthetic example we can visualize the asymptotic distribution on the set of classifiers, see fig. 7. The top three panels show $\mathbf{Q}$, with $\mathbf{S} = 2$, for the same training set on three different runs with 1000 iterations. Both the initial conditions are randomized (each point is assigned a uniformly sampled weight in $(0, 1)$) and the random planes in each run are sampled with different seeds. The bottom three panels show the same results for deterministic AdaBoost, i.e. $\mathbf{S} = 110$, where randomization is only in the initial weights.

An additional important observation is that the initial classifiers in a sequence can be dropped with no effect on the classification rate. In experiments with deterministic AdaBoost on the synthetic example, we observed that dropping the initial 1000 classifiers out of a sequence of length 2000 hardly alters the asymptotic distribution on classifiers, and has no effect on the final classification rate. This should of course not be surprising given the independence with respect to the initial weights observed above. These facts hint at some form of ergodic behavior of the different protocols. This should be taken in a broad sense, since we have observed degenerate asymptotic behaviors such as limit cycles with AdaBoost (see section 6.3,) however even there



Randomized AdaBoost, $\mathbf{S} = 2$



Deterministic AdaBoost, $\alpha = 1, \mathbf{S} = 110$

Figure 7: Top: Invariant distributions of the AdaBoost protocol on the same training set with three randomized initial conditions. The x-axis is the angle θ and the y-axis is the threshold τ. The area of the circles is proportional to the probability of the corresponding point.

the limit cycle itself is of independent of the initial conditions.

The same phenomenon emerges in the NIST experiment. Using a small training set of 1000 data points, and a coarser set of features, we produced ten sets of 300 trees with deterministic MAB. The only difference between the ten runs was the random assignment of the initial weights. In table 5 we present some numbers which suggest that very similar decision boundaries are achieved by these different runs. For 1 tree and aggregating over 10, 100 and 300 trees respectively, we provide the average classification rate over the 10 runs on a test set of 10000. We then report the proportion of test points which are classified the *same way* by a pair of runs, averaged over all 90 pairs. Note that asymptotically (e.g. 300 trees) these proportions are near 99%, although the classification rates are only at 92.5%. We also provide the average over the test data of the distance between the aggregates:

$$\frac{1}{10000}\sum_{i=1}^{10000}\frac{1}{10}\sum_{c=1}^{10}|H^a(X_i,c)-H^b(X_i,c)]|,$$

No. of Trees	1	10	100	300
Cl. rate	.76 (.01)	.89 (.005)	.924 (.003)	.926 (.002)
Agreement	.89 (.04)	.92 (.005)	.97 (.002)	.985 (.001)
Dist.	.03 (.01)	.03 (.001)	.01 (.0004)	.007 (.0002)

Table 5: Comparison of ten different runs of deterministic MAB with randomized initial weights. First row: number of trees in aggregate. Second row: Average and std. of Classification rate over ten runs. Third row: Proportion of pairwise agreement in classification, average and std. over all 90 pairs of runs. Fourth row: Average distance between aggregate output - average and std. over all 90 pairs of runs.

averaged over all 90 pairs of runs, $1 \leq a < b \leq 10$. Note how this average distance decreases with the number of trees and reaches the very low value of .007. We also note that the average classification rate using the *last* 100 trees of each sequence is *indistinguishable* from the first 100.

These experimental results may seem surprising if one has in mind that deterministic AdaBoost aims at minimizing a certain cost function. As mentioned in section 2, it can be shown that in a 2-class problem, assuming Y and the classifiers $h \in \mathcal{H}$ take their values in $\{-1; 1\}$, AdaBoost is minimizing the cost function $\mathcal{C}(H) = E[\exp(-YH(X))]$, where H is any combination with positive coefficients of base classifiers and the expectation is the empirical expectation on the training set. This fact was already suggested in Schapire et al. (1998), then in Friedman et al. (1998) it was shown that AdaBoost is similar to a Newton algorithm for minimizing this empirical cost iteratively; finally it has been shown Collins et al. (2000) that AdaBoost will always asymptotically drive the cost function to its infimum $\mathcal{C}^*$. This "gradient-descent" interpretation of Boosting seems at odds with our previous claims: the initial weights would seem to be important, (they implicitly appear in the empirical expectation in the definition of $\mathcal{C}$, where they are assumed to be uniform); one would think that dropping the first iterations would completely change the aggregate classifier obtained asymptotically.

To cast some light on these apparent contradictions, first note that, when there exists *some* convex combination of base classifiers that perfectly separates the training data, the minimum of the cost function $\mathcal{C}(H)$ defined above is $\mathcal{C}^* = 0$. Since the aggregate classifier in the cost function is *unnormalized*, one can take any convex combination separating the data and multiply it by an arbitrarily large constant to make the cost function tend to 0. Thus, in this case the convergence of the cost function to $\mathcal{C}^*$ does not say much about the asymptotic behavior of the aggregate classifier apart from the fact that the training error will eventually drop to zero. On the other hand, one of the main reasons for the success of AdaBoost is that even when the training error has reached zero for the aggregate, the *test* error continues to decrease, (see e.g. Schapire et al. (1998)). This phenomenon can not be attributed only to the asymptotic convergence of $\mathcal{C}(H_n)$ to 0.

Our opinion is that the crucial factor here is the *dynamical* behavior of AdaBoost, all the more in light of the fact that all the other sequential reweighting schemes behave the same way, and do not necessarily correspond to a minimization procedure of any cost function. In the next section we suggest that perhaps these different schemes are yielding lower average conditional variances because of an implicit sequential covariance minimization.

7.2 Sequential covariance minimization

For a two-class problem where the two classes are labeled $-1, 1$, and the outputs of the base classifiers are accordingly negative or positive, it is known that the AdaBoost procedure reweights a training example X in proportion to $e^{-YH(X)}$, where H is the current (unnormalized) aggregate classifier:

$$H = H^N = \sum_{i=1}^{N} \beta_i h_i.$$

Let h be a new candidate classifier. We assume that the output $h(X)$ of classifier h is either 1 or -1. Consider a slightly more general reweighting scheme, in which h is chosen in order to minimize (at least approximately) a weighted error function

$$\mathcal{E}(h) = E[W(-HY)(1 - hY)],$$

where the expectation is taken with respect to the empirical distribution determined by the training data, and W a strictly increasing and convex weight function. (Note that $\mathcal{E}$ is not a normalized error.) For AdaBoost we have $W(x) = \exp(x)$.

Now this can be rewritten as

$$\mathcal{E}(h) = Cov[-W(-HY), hY] + e(h)E[W(-HY)]. \tag{16}$$

where $e(h)$ is the empirical error of classifier h. Thus the normalized weighted error function $\alpha(h)$ is given by

$$\alpha(h) = Cov\left[\frac{-W(-HY)}{E[W(-HY)]}, hY\right] + e(h).$$

At each step these types of reweighting schemes try to minimize a criterion which is a compromise between the error of the next classifier and its covariance with some convex deformation of the current aggregate.

Suppose we now take uniform coefficients β_i of sum one for the aggregate at step n, and choose the linear weight function $W_n(x) = x + C$. In this case we obtain

$$\mathcal{E}(h) = Cov[HY, hY] + e(h)\,(C - E[HY]) = Cov[HY, hY] + \gamma(H)e(h). \tag{17}$$

This motivates an *explicit* covariance minimization protocol which tries to minimize the covariance of the new classifier with the existing aggregate subject to a constraint on the classification rate of the new classifier. Specifically, in the simple setting of the synthetic dataset, select $\mathbf{S}$ random planes $f^1, \ldots, f^{\mathbf{S}}$, and let $h^1, \ldots, h^{\mathbf{S}}$ denote the corresponding classifiers Evaluate

$$j^* = \mathrm{Argmax}_j \left\{Cov[H^N Y, h^j Y] + \gamma e(h^j)\right\}, \tag{18}$$

The next classifier h_{N+1} is taken to be h^{j^*}. Experiments with this protocol on the synthetic data set, using a fixed suitable γ for the entire sequence yields very similar results to the AdaBoost protocol.

Qualitatively it appears from equations 16 through 18 that all sequential reweighting schemes, in aiming to reduce some form of covariance with the current aggregate, all lead to asymptotically low ACC.

8 Discussion

We believe the main reason behind the success of various sequential reweighting protocols lies in the reduction of the average conditional covariance with respect to the limit distribution $\mathbf{Q}$ produced on the space of classifiers. Such a limit distribution appears to exist even in the context of entirely deterministic protocols. The low average conditional covariance reduces the variance of the margin around its mean and is also related to lower estimation errors, namely lower variability with respect to the training set.

The role of randomization is to produce a large variety of weakly dependent classifiers, using randomly selected subsets of features. We have discussed randomization in the particular context of decision trees, however randomization can also be applied to other classification methods. It is always possible to produce the base classifier using a random subset of features. In the context of feed-forward neural nets the connections can be randomly selected. Sequential reweighting appears to be more actively reducing covariances through estimates obtained on the training set, and in the absence of randomization risks over-fitting with smaller training sets. In this sense randomization tends to produce 'different' experts and sequential reweighting produces experts which 'disagree'. Randomized sequential reweighting seems to perform better and be more stable than any of the two extremes: deterministic sequential reweighting or only randomization without reweighting.

References

Amit, Y. and Geman, D. (1997), 'Shape quantization and recognition with randomized trees', *Neural Computation* **9**, 1545–1588.

Amit, Y. and Mascaro, M. (2001), 'Attractor networks for shape recognition', *Neural Computation*.

Amit, Y., Geman, D. and Wilder, K. (1995), Recognizing shapes from simple queries about geometry, Technical report, Department of Statistics, University of Chicago, `http://galton.uchicago.edu/~amit/Papers/digits.ps.gz`.

Blanchard, G. (1999), 'The "progressive mixture" estimator for regression trees', *Annales de l'I.H.P.* **35**, 793–820.

Blanchard, G. (2001), Mixture and aggregation of estimators for pattern recognition. Application to decision trees., PhD thesis, Université Paris Nord, Available at `http://www.dma.ens.fr/~gblancha`.

Breiman, L. (1996), 'Bagging predictors', *Machine Learning* **26**, 123–140.

Breiman, L. (1997), Prediction games and arcing algorithms, Technical report, Statistics department, University of California at Berkeley, Available at `ftp://ftp.stat.berkeley.edu/pub/users/breiman/games-.ps.Z`.

Breiman, L. (1998), 'Arcing classifiers (with discussion)', *The Annals of Statistics* **26**, 801–849.

Breiman, L. (1999), Random forests - random features, Technical report, Statistics department, University of California at Berkeley.

Chipman, H. A., George, E. I. and McCulloch, R. E. (1998), 'Bayesian cart model search', *JASA* **93**, 935–948.

Collins, M., Shapire, R. E. and Singer, Y. (2000), Logistic regression, AdaBoost and Bregman distances, *in* 'Proceedings of the thirteenth annual conference on computational learning theory'.

Dietterich, T. G. (1998), An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting and randomization, Submitted for publication to Machine Learning.

Dietterich, T. G. and Bakiri, G. (1995), 'Solving multiclass learning problems via error-correcting output codes', *J. Artificial Intell. Res.* **2**, 263–286.

Freund, Y. and Shapire, R. E. (1997), 'A decision-theoretic generalization of on-line learning and an application to boosting', *Journal of computer and system sciences* **55**, 119–139.

Friedman, J., Hastie, T. and Tibshirani, R. (1998), Additive logistic regression: a statistical view of boosting, Technical report, Department of Statistics, Stanford University.

Ho, T. K. (1998), 'The random subspace method for constructing decision forests', *IEEE Trans. PAMI* **20**(8), 832–844.

Ho, T. K., Hull, J. J. and Srihari, S. N. (1994), 'Decision combination in multiple classifier systems', *IEEE Trans. PAMI* **16**, 66–75.

Murua, A. (1999), Upper bounds for error rates associated to linear combination of classifiers., Technical Report 354, Department of Statistics, University of Washington.

Schapire, R. E. and Freund, Y. (1996), Game theory, on-line prediciton and boosting, *in* 'Proceedings of the 9th annual conference on computational learning theory'.

Schapire, R. E., Freund, Y., Bartlett, P. and Lee, W. S. (1998), 'Bossting the margins: a new explanation for the effectiveness of voting methods', *Annals of Statistics* **26**(5), 1651–1686.