Asymmetric Gradient Boosting with Application to Spam Filtering

Jingrui He
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213 USA
jingruih@cs.cmu.edu

Bo Thiesson
Microsoft Research
One Microsoft Way
Redmond, WA 98052-6399 USA
thiesson@microsoft.com

ABSTRACT

In this paper, we propose a new asymmetric boosting method, Boosting with Different Costs. Traditional boosting methods assume the same cost for misclassified instances from different classes, and in this way focus on good performance with respect to overall accuracy. Our method is more generic, and is designed to be more suitable for problems where the major concern is a low false positive (or negative) rate, such as spam filtering. Experimental results on a large scale email spam data set demonstrate the superiority of our method over state-of-the-art techniques.

1. INTRODUCTION

Classification is a very important field in machine learning, and has been well studied over the past years. Many different classification methods have been proposed, such as logistic regression, decision trees, neural networks, SVMs, etc. Recent developments in classification methodology combines individual classifiers into a powerful ensemble classifier, which predicts the label of a particular instance using a weighted vote over the ensemble.

Boosting methods are ensemble methods in which one adds new base classifiers to the ensemble in a sequential way (e.g., Freund & Schapire, 1997). The new classifiers are trained on the basis of reweighed versions of the data, where instances that are not predicted well by the current ensemble are given a higher weight. This simple strategy of combining classifiers has shown to improve results dramatically in many cases, and thus has been the focus of much subsequent research. See, e.g., Friedman, Hastie & Tibshirani (2000), Mason, Baxter, Bartlett & Frean (1999), and Friedman (2001) to mention a few.

The loss function is one of the key factors in a standard classification framework. In boosting, this loss function affects the reweighing of the data used to train the next classifier in the ensemble. Most loss functions are *symmetric* in the sense that they assume the same cost for misclassified instances with different class labels. In many application domains, however, this assumption is not appropriate. For example, in an email spam filtering setting it is much more concerning to lose a piece of good mail (ham) than it is to receive a few unsoliced mails (spam).

CEAS 2007 - Fourth Conference on Email and Anti-Spam, August 2-3, 2007, Mountain View, California USA

In this paper we focus on email spam filtering. Recently, Androutsopoulos, Paliouras & Michelakis (2004) and Carreras & Marquez (2001) introduced boosting with symmetric cost functions for building spam filtering classifiers. We propose a new *asymmetric* boosting method, Boosting with Different Costs (BDC). This method is generally applicable to any problems, where it is important to achieve good classification at low false positive (or low false negative) rates.

Despite the slightly mind-bending connotation, we will follow tradition and denote the mis-classification of good emails as false positives. For an email spam filter, false positives are therefore more expensive than false negatives—or in other words, misclassifying good email as spam is more concerning than the other way around. To accommodate this asymmetry, a commonly used method is stratification with more emphasis on ham than spam emails during the training of the classifier. For boosting, this stratification can be achieved by multiplying the loss associated with a ham email by a utility value greater than one.

One possible problem with this simple method is that all the spam emails are de-emphasized in the same way no matter if they are important for constructing the classifier or not. If the training data are noisy, either due to mis-labeling or due to very uncharacteristic spam examples—which is often the case—we will not be able to differentiate between the noisy and the characteristic spam examples. In this way, the performance of the subsequent classifier may be largely affected by the noise.

The proposed BDC method is designed to solve this problem. It is based on the MarginBoost framework (Mason, Baxter, Bartlett & Frean, 1999). In our approach, we design two different cost functions, one for ham and one for spam. For ham, the cost gradually increases as an instance moves away from the decision boundary on the wrong side of its label; while for spam, the cost remains unchanged after it reaches a point far enough from the decision boundary. As will become clear in Section 3, the effect of this difference in the cost functions is that we are trading off difficult spam for the ability to better classify all ham and easier, more characteristic spam. We are in this way skewing the classifier towards better classification at low false positive rates.

As straw men for comparisons, we will—with and without stratification—consider symmetric boosting with logistic loss, as well as logistic regression, which is one of the most popular machine learning algorithms for email spam filtering. See, e.g., Yih, Goodman, & Hulten (2006). As noticed by Friedman, Hastie & Tibshirani (2000) and Collins, Schapire & Singer (2000), using logistic loss, boosting and logistic regression are quite similar. To further bolster this boosting view on logistic regression and in this way better understand similarities and differences to our BDC method, we demonstrate how logistic regression can be considered as a special implementation of the MarginBoost framework in Mason *et al.* (1999).

Our spam filtering application involves a large number of features, and it turns out that if we use decision stumps as the base classifiers, boosting is similar to logistic regression with a smart feature selection method. In addition, boosting with decision trees of depth two, for example, can be used to add more complicated co-occurrence features to existing models, which is not possible for logistic regression without an explosion in the number of parameters.

To summarize, the possible advantages of BDC include:

1. BDC is tailored for spam filtering by introducing different cost functions for ham and spam.
2. Unlike stratification, spam is de-emphasized only when hard to classify correctly. If the training data is noisy, BDC will filter out noisy spam, and focus on all the other instances (ham and characteristic spam). Thus it produces a classifier with better performance at low false positive rate.
3. Like other boosting algorithms, BDC can be used to select features, which will save space for storing the model and have faster runtime when deployed; it can also be used to add more complicated features into existing models without an explosion in the number of parameters.

The rest of the paper is organized as follows. In Section 2, we briefly review the MarginBoost framework (Mason *et al.*, 1999), and view logistic regression as a special implementation of this framework. In Section 3, we introduce the cost functions used in BDC, present the associated boosting algorithm, and discuss its behavior in the low false positive region. Section 4 studies the parameter setting in BDC. To prove the effectiveness of BDC in spam filtering, we conduct experiments on a large scale spam data set, and summarize the results in Section 5. In Section 6 we discuss related work, followed by conclusions in Section 7.

2. MARGINBOOST

MarginBoost is a special variation of a more general class of boosting algorithms based on gradient decent in function space. A detailed derivation of this general class of boosting algorithms can be found in Mason *et al.* (1999) and Friedman (2001). In this Section, we will first review the MarginBoost algorithm, as described in Mason *et al.* (1999), who also coined this name. Then, we will consider particular specializations of MarginBoost and demonstrate their relations to logistic regression.

2.1 The Framework

In a binary classification problem, given a training set $D = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ of m labeled data points generated according to some unknown distribution, where $x_i \in R^d$ and $y_i \in \{\pm 1\}$, we wish to construct a voted combination of classifiers, or strong classifier:

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

where $f_t(x) : R^d \to \{\pm 1\}$ are base classifiers, and $w_t \in R$ are the weights for each base classifier in the combination. A data point (x, y) is classified according to the sign of $F(x)$, and its *margin* is defined by $yF(x)$. A positive value for the margin therefore corresponds to a correct classification and a negative value corresponds to a misclassification.

To obtain the classifier, MarginBoost minimizes the sample average of some suitable cost function of the margin $C : R \to R$. That is, we find a classifier F that minimizes the loss functional

$$S(F) = \frac{1}{m} \sum_{i=1}^{m} C\left(y_i F(x_i)\right) \tag{2}$$

We emphasize here that MarginBoost does not perform traditional parameter optimization. As formulated in Friedman (2001), we instead consider F evaluated at each point x to be a "parameter", and seek to minimize (2) directly with respect to F evaluated at each x.

Taking a numerical approach, the minimization of (2) can be achieved by gradient descent in function space. In other words, given the current classifier F_t, at iteration t of the algorithm, we are looking for a new direction f_{t+1} such that $S(F_t + \epsilon f_{t+1})$ decreases most rapidly, for small values of ϵ. In function space, the desired direction is simply the negative functional derivative of S at F_t, $-\nabla S(F_t)(x)$, defined by

$$\begin{aligned}\nabla S(F_t)(x) &= \left.\frac{\partial S(F_t + \epsilon 1_x)}{\partial \epsilon}\right|_{\epsilon=0} \\ &= \frac{1}{m} \sum_{i=1}^{m} \delta(x = x_i) y_i C'\left(y_i F_t(x_i)\right) \qquad (3)\end{aligned}$$

where 1_x is the indicator function at x , $\delta(x = x_i) = 1$ iff $x = x_i$, and $C'(z)$ is the derivative of the cost function with respect to the margin z.

This derivative is defined only at the data points and do not generalize to other values. We achieve generalization (or smoothing) by restricting the direction of f_{t+1} to a base classifier in some fixed parameterized class $\mathcal{F}$. In general it will therefore not be possible to choose $f_{t+1} = -\nabla S(F_t)(x)$, so instead we find the base classifier that has the greatest inner product with $-\nabla S(F_t)(x)$. Letting a dot denote inner product, and inserting the expression for $\nabla S(F_t)$ from (3), the new direction f_{t+1} should in this case maximize

$$-\langle \nabla S(F_t), f_{t+1}\rangle = -\frac{1}{m}\sum_{i=1}^{m} \nabla S(F_t)(x_i)\cdot f_{t+1}(x_i)$$
$$= -\frac{1}{m^2}\sum_{i=1}^{m} y_i f_{t+1}(x_i) C'(y_i F_t(x_i)) \quad (4)$$

For later discussions, it is important to notice from (4) that the new base classifier f_{t+1} is found by considering this classifier's weighted margins for all the data points, $y_i f_{t+1}(x_i)$, where the weights are in proportion to minus the derivative of the cost function evaluated at the current margins, $D_t(i) \propto -C'(y_i F_t(x_i))$.

Given the new base classifier f_{t+1}, we can now use a line search to find the best coefficient w_{t+1} for f_{t+1}. The iterative process of adding $w_{t+1} f_{t+1}$ to the strong classifier in (1) can be terminated when the dot product between the negative functional derivative of S at F_t and any base classifier is less than or equal to zero. Another stopping criterion commonly used in practice is to set a fixed maximum number of iterations.

2.2 MarginBoost Specializations

As has been noted in the literature (e.g., Mason *et al.*, 1999 and Friedman, 2001), many important boosting algorithms can be reformulated in the MarginBoost framework. For example, if we use the exponential loss $C(yF(x)) = \exp(-yF(x))$, and w_{t+1} is obtained by a line search, we will get AdaBoost (Freund & Schapire, 1997); if we use the logistic loss $C(yF(x)) = \ln(1+\exp(-yF(x)))$, and w_{t+1} is chosen by a single Newton-Raphson step, we get LogitBoost (Friedman *et al.*, 2000). In general, the MarginBoost framework will work for any differentiable cost function, and in most situations, a monotonically decreasing cost function of the margin is a sensible choice.

In order to demonstrate that linear logistic regression can also be viewed as a special implementation of MarginBoost, we will concentrate on the logistic loss function. Let us first consider the class of linear base classifiers of the form

$$\mathcal{F} = \{f(x) = a^\top x + b, a \in R^d, b \in R, \| a \|^2 + b^2 = 1\},$$

where a is the coefficient vector, b is the intercept, and $\top$ denotes transpose. The new direction f_{t+1} is obtained by choosing a base classifier in $\mathcal{F}$ that maximizes (4) with a logistic loss for C

$$-\langle \nabla S(F_t), f_{t+1}\rangle \propto \sum_{i=1}^{m} y_i(a_{t+1}^\top x_i + b_{t+1}) \frac{\exp(-y_i F_t(x_i))}{1+\exp(-y_i F_t(x_i))}$$

This constrained optimization yields a classifier in $\mathcal{F}$ with coefficients and intercept

$$a_{t+1} \propto \sum_{i=1}^{m} \frac{y_i x_i \exp(-y_i F_t(x_i))}{1+\exp(-y_i F_t(x_i))} \quad (5)$$

$$b_{t+1} \propto \sum_{i=1}^{m} \frac{y_i \exp(-y_i F_t(x_i))}{1+\exp(-y_i F_t(x_i))} \quad (6)$$

Iteration $t+1$ therefore updates the strong classifier as

$$F_{t+1}(x) = F_t(x) + w_{t+1}(a_{t+1}^\top x + b_{t+1}),$$

where a_{t+1} and b_{t+1} are determined as in (5) and (6), and w_{t+1} is a small positive parameter, which is chosen by line search or based on simple heuristics.

It is an interesting observation that for a logistic regression model, $P(y|x) = 1/\big(1+\exp(-y(a^\top x + b))\big)$ with parameters a and b, the gradient of the log-likelihood equals exactly the same expressions as on the right-hand sides of (5) and (6). Hence, the particular boosting algorithm, described above, is equivalent to logistic regression with a gradient ascent parameter optimization method. [1]

Another interesting observation will follow from choosing decision stumps as the class of base classifiers. For the spam data set that we will consider later in this paper, all features are binary, i.e., $x \in \{0,1\}^d$. In this case, the learned decision stump at iteration t can be defined as $f_t(x) = cx^t - d$, where x^t is the most discriminating feature at that iteration, and $c, d \in R$. The voted combination $F(x)$ will therefore also result in a linear classifier of the form $F(x) = a^\top x + b$, where $a \in R^d$ and $b \in R$. Since $F(x)$ and logistic regression are minimizing the same concave cost function, upon convergence, the two classifiers would therefore again be the same, although they are achieved in two different ways. It is also noteworthy that by making use of a rougher stopping criterion, MarginBoost may end up with a sparse coefficient vector a. In this case, MarginBoost implicitly performs feature selection, which will reduce both the storage space and the response time for the deployed classifier.

3. BOOSTING WITH DIFFERENT COSTS

Existing cost functions used in the MarginBoost framework (including the one used in logistic regression) are symmetric in the sense that they assume the same cost function for misclassified instances from different classes. However, in spam filtering, the cost for misclassifying ham is much higher than that for misclassifying spam, which requires cost functions that are asymmetric for the two classes. In this section, we therefore present the Boosting with Different Costs method. It uses two different cost functions for ham and spam, with an emphasis on classifying ham correctly. In each round of boosting, *moderately* misclassified spam (the absolute value of the margin is small) have large weights, whereas *extremely* misclassified spam (the absolute value of the margin is large) will have small weights instead, since they tend to be noise. In contrast, all the misclassified ham have large weights. In this way, the base classifier trades off correct classification of extremely misclassified spam for better classification of all ham and the remaining spam, thus ensuring a low false positive rate of the strong ensemble classifier.

3.1 Cost Functions in BDC

Suppose that a data point is labeled as $y_i = -1$ for ham, and as $y_i = 1$ for spam. The cost functions used in BDC are

[1] A logistic regression model is often regularized by adding a term to the model in the form of a penalizing distribution on the parameters in the model. This type of regularization will not fit into the MarginBoost framework, because we cannot in this case express a suitable cost function of the margin only.

as follows

$$\text{Ham:} \quad C_h(yF(x)) = \ln(1 + \exp(-yF(x))) \tag{7}$$

$$\text{Spam:} \quad C_s(yF(x)) = \frac{\alpha}{1 + \exp(\gamma y F(x))} \tag{8}$$

where α and γ are two positive parameters that control the shape of the cost function for spam relative to that for ham. Note that here the two cost functions are designed for spam filtering. In general, BDC could work with any two cost functions that satisfy the differentiability condition mentioned in Section 2.2. Figure 1 compares the two costs as functions of the margin ($\alpha = 1$ and $\gamma = 1$). From this figure, we can see that when the margin is positive, which corresponds to correct classification, both functions will output a small cost. On the other hand, when the margin is negative, the cost for ham increases at a higher and higher rate as the margin becomes more negative, until it approximates a linear function for extremely misclassified instances. In contrast, the cost for spam almost remains unchanged at very negative margins. That is, if a spam message is extremely misclassified, the further decrease in the margin will not increase the cost.

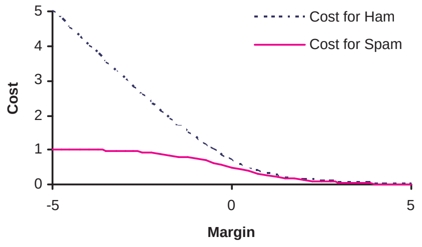


Figure 1: Cost functions for ham and spam

For iteration t of BDC, as in MarginBoost, we need to calculate the weight for each training instance. As mentioned in Subsection 2.1, this weight is in proportion to the negative first derivatives of the two cost functions, i.e.

$$\text{Ham:} D_t(i) \propto -C'_h(y_i F(x_i)) = \frac{\exp(-y_i F(x_i))}{1 + \exp(-y_i F(x_i))} \tag{9}$$

$$\text{Spam:} D_t(i) \propto -C'_s(y_i F(x_i)) = \frac{\alpha\gamma \exp(\gamma y_i F(x_i))}{(1+\exp(\gamma y_i F(x_i)))^2} \tag{10}$$

In Figure 2, we compare the negative first derivatives as functions of the margin ($\alpha = 1$ and $\gamma = 1$). From the figure, we can see that misclassified ham always have a large weight; while misclassified spam get a large weight iff the margin is close to zero.

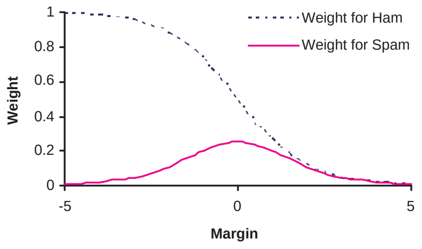


Figure 2: Weight functions for ham and spam

The advantages of the weighing scheme in BDC are two-fold. Firstly, in spam filtering, the training set is always noisy to some extent, and the outliers tend to be difficult to classify as their given labels. If a training instance is extremely misclassified (the margin is very negative), it is quite likely to be an outlier. In existing boosting algorithms, as more and more base classifiers are combined into the strong classifier, the outliers will have larger and larger weights, and the subsequent base classifiers will end up fitting the noise. From this perspective, since BDC assigns small weights to extremely misclassified spam, it is able to discard some noisy spam. Secondly, in spam filtering, people would rather receive a few pieces of spam than losing a single piece of ham. In BDC, the weight of misclassified ham is always high, which ensures that the combined classifier will have a low false positive rate.

3.2 BDC Algorithm

Based on the MarginBoost framework, we summarize the BDC algorithm in Figure 3. Initially, each training instance is assigned the same weight. In each iteration, the weak learner selects a base classifier that maximizes the inner product in (4). After the coefficient w_{t+1} is chosen by line search, the weights of the training instances are updated based on the negative first derivative of the ham and spam cost functions C_h and C_s with respect to the margin for the newly combined classifier. The iteration process is stopped when the inner product in (4) is less than or equal to zero, or when we reach a prespecified maximum number of iterations T.

```
Let $D_0(i) = 1/m$ for $i = 1, \ldots, m$.
Let $F_0(x) = 0$.
For $t = 0$ to $T$ do
    Let $f_{t+1} = \arg\max_{f \in \mathcal{F}} \sum_{i=1}^{m} y_i f(x_i) D_t(i)$.
    If $\sum_{i=1}^{m} y_i f_{t+1} D_t(i)(x_i) \le 0$ then
        Return $F_t$
    End if
    Choose $w_{t+1}$ by line search.
    Let $F_{t+1} = F_t + w_{t+1} f_{t+1}$
    For $i = 1, \ldots, m$ do
        Let $D_{t+1}(i) = \begin{cases} -C'_h(y_i F_{t+1}(x_i)) & \text{for } y_i = -1 \\ -C'_s(y_i F_{t+1}(x_i)) & \text{for } y_i = 1 \end{cases}$
    End for
    Let $D_{t+1}(i) = \frac{D_{t+1}(i)}{\sum_{i=1}^{m} D_{t+1}(i)}$ for $i = 1, \ldots, m$.
End for
Return $F_{t+1}$
```

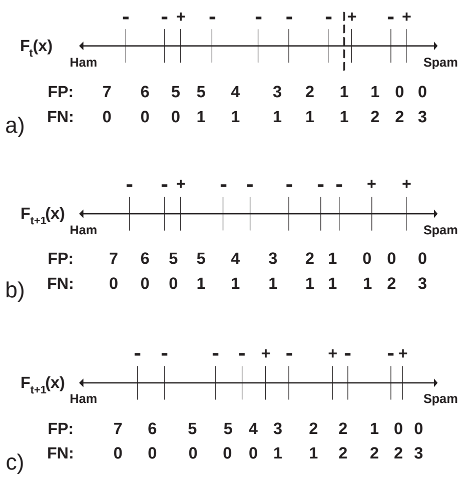
Figure 3: BDC algorithm

3.3 BDC at Low False Positive Region

A commonly used performance measure in spam filtering is the ROC curve, which is obtained by varying the threshold of the classifier. In the ROC curve, false negative rates at the low false positive region are of particular interest. Compared to boosting algorithms with symmetric cost functions, or symmetric boosting, BDC tends to have lower false negative rates in the low false positive region.

Figure 4 is an illustrative example of the process causing this phenomenon. The ticks on the axis correspond to the training instances. The +/– above each tick labels an instance as spam (+) or ham (–). According to some decision threshold on the axis, instances to the left of this threshold are classified as ham and instances to the right as spam. The further an instance is to one side of the threshold, the more confidence the classifier has in the decision. The dashed line perpendicular to the axis in Figure 4(a) marks the decision threshold for $F_t(x)$. The left most + in this figure therefore represents a noisy spam message. The numbers following FP (FN) are the false positive rates (false negative rates) in terms of the number of false positive (false negative) instances if the decision threshold of the classifier is drifted to the corresponding position on the axis. If we apply BDC, the misclassified ham (the instance marked as – on the right-hand side of the threshold) will have a large weight, but the noisy spam (the instance marked as + far out on the left-hand side of the threshold) will have a small weight. After one base classifier is added to $F_t(x)$, the classifier is able to better classify ham and all but the noisy spam (Figure 4(b)). In contrast, if we apply symmetric boosting, the misclassified spam will have the largest weight of all instances, and the classifier may end up correctly classifying this instance or just lowering the confidence in the misclassification. This improvement may come at a slight expense in the confidence of correctly classified instances (Figure 4(c)).

Comparing Figure 4(b) and Figure 4(c), at FP value zero, FN using BDC decreases from two to one, while FN using symmetric boosting stays at two. This suggests the advantage of BDC over symmetric boosting in the low FP region, which is a desirable property in spam filtering. On the other hand, at FP value four, FN using BDC is one, whereas FN using symmetric boosting is zero, which suggests that BDC may not be as good as symmetric boosting in the high FP region.



$F_t(x)$
Ham
Spam
FP: 7 6 5 5 4 3 2 1 1 0 0
FN: 0 0 0 1 1 1 1 1 2 2 3
a)
$F_{t+1}(x)$
Ham
Spam
FP: 7 6 5 5 4 3 2 1 0 0 0
FN: 0 0 0 1 1 1 1 1 1 2 3
b)
$F_{t+1}(x)$
Ham
Spam
FP: 7 6 5 5 4 3 2 2 1 0 0
FN: 0 0 0 0 0 1 1 2 2 2 3
c)

Figure 4: a) Current classifier. b) After one iteration in BDC. c) After one iteration in symmetric boosting.

4. PARAMETER STUDY IN BDC

In BDC, we need to set the value for the positive parameters α and γ in the cost function for spam. From (8), we can see that α is the maximum cost for spam, and that the cost for a spam message with $F(x) = 0$ is $\alpha/2$. By setting the value of α, we are implicitly performing stratification. The smaller α is, the less important spam is compared to ham.

If we approximate $C_s(yF(x))$ in (8) by a first order Taylor expansion around $yF(x) = 0$, we have

$$C_s(yF(x)) \approx \alpha(0.5 - 0.25\gamma yF(x)) \qquad (11)$$

Thus, with α fixed at a certain value, the γ parameter controls the slope of the cost. The larger γ is, the faster the cost changes with the margin around $yF(x) = 0$, and vice versa.

We study the effect of the two parameters on the performance of BDC using three noisy data sets, constructed from the synthetic data set illustrated in Figure 5. In this data

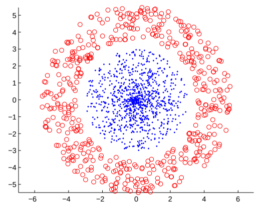


Figure 5: Synthetic data.

set, the inner ball and the outer circle represent the positive and negative classes, respectively. The noisy data sets are constructed by assigning an incorrect class label to each of the data points with a small probability. We have gradually increased the noise level, and generated three data sets with noise probabilities 0.03, 0.05, and 0.1. For different settings of the parameters α and γ, we generate the corresponding ROC curves by varying the threshold of the classifier. Since we are only interested in classification results at low false positive rates, we only compare FN rates at FP rates 3%, 5%, and 10% in the ROC curves, which will provide a gauge to how the ROC curve will behave in the low false positive region. The averaged results for 4-fold cross validation are summarized in Figure 6. The first row of experiments shows FN rates with varying α and fixed $\gamma = 1$ for the three noisy data sets, and the second row shows similar results with varying γ and fixed $\alpha = 1$.

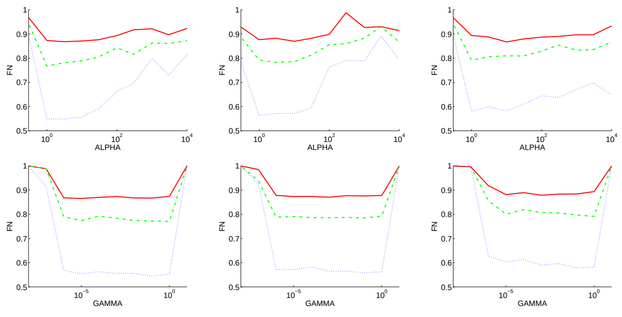


Figure 6: In each of the six experiments, the three curves correspond to: upper curve: FN at FP 3%; middle curve: FN at FP 5%; lower curve: FN at FP 10%. The first row illustrates the effect of α with $\gamma = 1$, and the second row, the effect of γ with $\alpha = 1$. The three columns correspond to the three data sets with noise probabilities 0.03, 0.05, and 0.1, respectively.

From the figure we see that the performance of BDC is close to optimal over a wide range of the parameter values. In other words, the results are not very sensitive to specific parameter values as long as they are within a reasonable range. This observation suggests a simple way of selecting parameter values in real applications: using a hold-out test set or via cross validation, perform a simple line search for one parameter while fixing the other, and iterate this process until performance does not improve further. The line search could be as simple as just sampling a number of parameters within an allowable range, and then pick the best one. In our empirical experiments, this simple scheme always converges to a good parameter setting.

5. EXPERIMENTAL RESULTS

In this section, we use a large spam corpus to compare BDC with state-of-the-art spam filtering techniques. Both the training and test sets are from the Hotmail Feedback Loop data, which is collected by polling over 100,000 Hotmail volunteers daily. In this feedback loop, each user is provided with a special copy of a message that was addressed to him, and is asked to label this message as ham or spam. The training set consists of messages received between July 1st, 2005 and August 9th, 2005. We randomly picked 5,000 messages from each day, which results in a total number of 200,000 email messages used for training. The test set contains messages received between December 1st, 2005 and December 6th, 2005. 10,000 messages were drawn from each day, resulting in a collection of 60,000 messages. From each message, we extracted features consisting of subject keywords and body keywords that occurred at least three times in the training set. The total number of binary features is over 2 million. Please refer to Yih, Goodman & Hulten (2006) for a detailed description of a similar corpus.

The methods used for comparison include: logistic regression (LR), regularized logistic regression, regularized logistic regression with stratification, LogitBoost[2] (LB), and LogitBoost with stratification. We regularized LR the standard way by multiplying the likelihood with a Normal prior $\mathcal{N}(0, \sigma^2)$ on the parameters in the model. To determine the parameter values in each method—σ for regularization in LR, the stratification factor for LR and LB, and α, γ in BDC—we split the training set into two sets of 100,000 messages each: one for training the classifier with different parameter settings, and the other for validating the performance of the trained classifier. The values leading to the lowest FN at low FP regions are assigned to the parameters. With the chosen parameter values, we train the classifiers again, this time using the whole training set, and compare the results for the different methods on the test set.

With BDC, LogitBoost, and LogitBoost with stratification, we have considered decision trees of depth one (decision stumps) and depth two (which generates features similar to co-occurrence features) as the base classifiers. We have also tried two criteria for stopping the iteration procedure in the boosting algorithms: 1) setting the maximum number of base classifiers, and 2) setting the minimum decrease in the loss function between two consecutive iteration steps. The results using these two methods are quite similar, so we only present the results obtained with the first stopping criterion. Experiments with decision stumps are stopped after including 4000 base classifiers in the ensemble, and experiments with decision trees of depth two are stopped after including 2000 base classifiers. Finally, recall from Subsection 2.2 that although LB and LR optimize parameters in different ways, the models obtained by LB with decision stumps are similar to the models obtained by LR with a smart selection of the most dominant features (< 4000).

Figures 7 and 8 compare the ROC curves at the low false positive region from zero to 0.2 for all of the considered methods by varying the threshold of the associated classifier. The two figures show the resulting curves for the boosting methods with decision stumps and with decision trees of depth two as base classifiers, respectively. The curves for the logistic regression methods are the same in these figures.

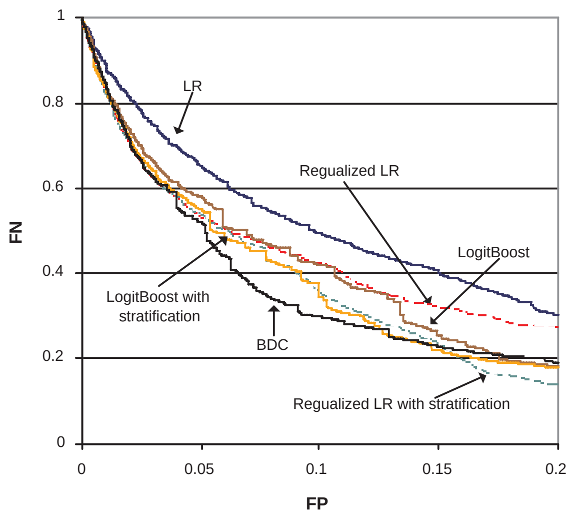


Figure 7: ROC curves for BDC, LB, and LR methods. The base classifiers in BDC and LB methods are decision stumps.

From Figures 7 and 8, we can make the following conclusions: 1) As is desirable for spam filtering, at the low false positive region, BDC with simple base classifiers (decision stumps) outperforms BDC with complicated ones (decision trees of depth 2), and actually performs the best of all the methods reported here; 2) in both of the base classifier settings, BDC is among the best methods in the low false positive region; 3) the improvement of BDC over LB or LB with stratification is more obvious with complicated decision trees than with simple ones, showing that BDC is more resistant to overfitting than LB based methods; 4) no matter with LR or LB, stratification always improves over the original methods in the low false positive region.

Considering the storage space and the runtime for a deployed classifier, the size of the classifier trained by BDC is a lot smaller than that trained by LR based methods. For instance, with 4000 decision stumps, the number of coefficients used in the classifier trained by BDC is at most 4000 (different base classifiers may involve the same feature), compared

[2]Different from LogitBoost in Friedman (2001), here, the weight w_t is obtained by line search.

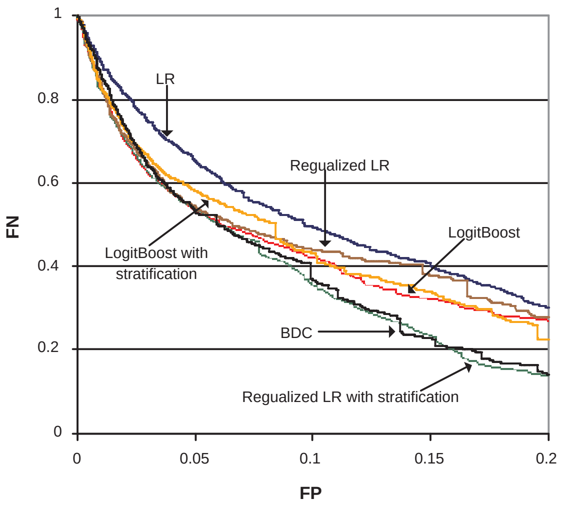


Figure 8: ROC curves for BDC, LB, and LR methods. The base classifiers in BDC and LB methods are decision trees of depth two.

with more than 2 million coefficients used in the classifier trained by LR based methods.

Finally, it is worth mentioning that the above results are for a spam classifier solely based on text features, and does not reflect the performance of a full commercial system. In commercial systems, a text based classifier is only one component of a larger system involving IP blocklists, IP safelists, user supplied blocklists and safelists, etc., where each component helps driving the spam system to even better performance. Better performance obviously creates more satisfied users, but it also results in significant storage savings for the email provider. For example, the throughput on Hotmail is well over one billion messages per day, with an average spam message size of 14Kb. If messages classified as spam (at a fixed low false positive rate) are deleted right away, storage of more than 140Gb daily spam messages are avoided per percent improvement.

6. RELATED WORK

Recently, boosting has been applied to spam filtering. In Carreras and Marquez (2001), the authors consider several variants of the AdaBoost algorithm with confidence-rated predictions, and compare with Naive Bayes and induction of decision trees, reaching the conclusion that the boosting-based methods clearly outperform the baseline learning algorithms on the PU1 corpus. The authors of Androutsopoulos *et al.* (2004) also apply LogitBoost, as well as Support Vector Machines, Naive Bayes, and Flexible Bayes on four spam data sets, so as to examine various aspects of the design of learning-based anti-spam filters.

On the issue of good classification of spam at low false positive (spam) rates, Yih, Goodman and Hulten (2006) propose a two-stage cascade classifier. The first-stage classifier is trained on all data, and classifications as ham by this classifier are not further contested. Classifications as spam, on the other hand, are sent to the second-stage classifier, which is trained from cases classified as spam during the training of the first-stage classifier. The suggested cascading technique can be combined with any type of classifiers, including BDC, and may even result in further improvement.

Viola and Jones (2002) also suggests a cascade of classifiers. Starting with very simple classifiers at early stages of the cascade and gradually increasing the complexity, the classifier is very fast, because most cases in the domain they consider are easy negatives. Their classifier further ensures good performance at low false positive regions by training each classifier in the cascade by an asymmetic variation of AdaBoost that weighs false negatives differently than false positives during training. - Very much in the spirit of stratification.

Recently, there has also been great interest in algorithms for ranking. In particular Rudin (2006) has derived a very flexible boosting-style algorithm, designed to perform well on the top of the ranking list at the expense of possible misrankings further down the list. A parameterized cost function controls this tradeoff. We may consider the asymmetric spam classification problem as a special instance of this framework. By pushing ham to the top of the list at the expense of rankings further down the list, this skewed ranking is in effect concentrating on the low false positive region of the ROC.

Finally, the proposed BDC combines boosting and cost-sensitive learning in a natural way. A closely related work is AdaCost proposed in Fan *et al.* (1999). It uses the cost of misclassifications to update the training distribution on successive boosting rounds. The purpose is to reduce the cumulative misclassification cost more than AdaBoost. The difference between AdaCost and BDC is that, the former relies on a pre-specified cost for each training instance, while the latter provides a reasonable way of obtaining those costs.

7. CONCLUSION

In this paper, we have proposed a new asymmetric boosting method, Boosting with Different Costs, and applied it to email spam filtering. In BDC we define two cost functions, one for ham and one for spam, according to practical needs. In each round of boosting, misclassified ham always have large weights. Moderately misclassified spam also have large weights, but extremely misclassified spam will have small weights. In this way, BDC is able to focus on the ham messages and the more characteristic spam messages at the expense of the more difficult or noisy spam messages. BDC will therefore improve the false negative rates at the low false positive region in the ROC curve, as demonstrated for the spam filtering data investigated in this paper.

Acknowledgements

We thank Paul Viola, Joshua Goodman, and Wen-tau Yih for helpful discussions regarding this paper.

References

Androutsopoulos, I., Paliouras, G., & Michelakis, E. (2004). *Learning to filter unsolicited commercial e-mail* (Technical Report 2004/2). National Centre for Scientific Research ”Demokritos”.

Carreras, X., & Márquez, L. (2001). Boosting trees for anti-spam email filtering. *Proceedings of RANLP-01, 4th International Conference on Recent Advances in Natural Language Processing.* Tzigov Chark, BG.

Collins, M., Schapire, R. E., & Singer, Y. (2000). Logistic regression, adaboost and bregman distances. *Proceedings of Thirteenthth Annual Conference on Computational Learing Theory* (pp. 158–169).

Fan, W., Stolfo, S. J., Zhang, J., & Chan, P. K. (1999). AdaCost: misclassification cost-sensitive boosting. *Proceedings of the Sixteenth International Conference on Machine Learning* (pp. 97–105). Morgan Kaufmann.

Freund, Y., & Schapire, R. (1997). A decision-theoretic generalization of on-line learning and application to boosting. *Journal of Computer and System Sciences*, *55*, 119–139.

Friedman, J. (2001). Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, *29*, 1189–1232.

Friedman, J., Hastie, T., & Tibshirani, R. (2000). Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, *28*, 337–407.

Mason, L., Baxter, J., Bartlett, P., & Frean, M. (1999). *Boosting algorithms as gradient descent in function space* (Technical Report). RSISE, Australian National University.

Rudin, C. (2006). Ranking with a p-norm push. *Proceedings of the Nineteenth Annual Conference on Computational Learning Theory* (pp. 589–604).

Viola, P., & Jones, M. (2002). Fast and robust classification using asymmetric AdaBoost and a detector cascade. *Advances in Neural Information Processing Systems 14*. MIT Press, Cambridge, MA.

Yih, W., Goodman, J., & Hulten, G. (2006). Learning at low false positive rates. *Proceedings of the Third Conference on Email and Anti-Spam.*