# Asymmetric Boosting

**Hamed Masnadi-Shirazi** HMASNADI@UCSD.EDU
**Nuno Vasconcelos** NUNO@UCSD.EDU
Statistical Visual Computing Laboratory,University of California San Diego, La Jolla, CA 92039 USA

## Abstract

A cost-sensitive extension of boosting, denoted as asymmetric boosting, is presented. Unlike previous proposals, the new algorithm is derived from sound decision-theoretic principles, which exploit the statistical interpretation of boosting to determine a principled extension of the boosting loss. Similarly to AdaBoost, the cost-sensitive extension minimizes this loss by gradient descent on the functional space of convex combinations of weak learners, and produces large margin detectors. It is shown that asymmetric boosting is fully compatible with AdaBoost, in the sense that it becomes the latter when errors are weighted equally. Experimental evidence is provided to demonstrate the claims of cost-sensitivity and large margin. The algorithm is also applied to the computer vision problem of face detection, where it is shown to outperform a number of previous heuristic proposals for cost-sensitive boosting (AdaCost, CSB0, CSB1, CSB2, asymmetric-AdaBoost, AdaC1, AdaC2 and AdaC3).

## 1. Introduction

Many classification problems, in areas of great practical relevance for machine learning, are naturally cost sensitive. One predominant example is that of detection problems, such as object detection in computer vision (Viola & Jones, 2002), fraud detection (Viaene et al., 2004), or medical diagnosis (Park et al., 2003), where the targets to be detected are rare. For all these problems, where the cost of missing a target is much higher than that of a false-positive, classification algorithms which are optimal under symmetric costs (such as the popular zero-one loss) tend be unsatisfactory. The design of classifiers that are optimal for losses that weigh certain types of errors more heavily than others is denoted as cost-sensitive learning. Current research in this area falls into two main categories. The first attempts to produce generic procedures for making any arbitrary algorithm cost sensitive, by resorting to Bayes risk theory or some other cost minimizing strategy (Zadrozny et al., 2003) (Domingos, 1999), (Chawla et al., 2003) (Guo & Viktor, 2004). The second attempts to extend particular algorithms, so as to produce cost-sensitive generalizations.

One example is the popular AdaBoost algorithm, which is not cost-sensitive but has achieved tremendous practical success in areas such as computer vision (Viola & Jones, 2001). AdaBoost (Freund & Schapire, 1997) produces a strong classifier by combining a voted ensemble of weak classification functions (weak learners). Each weak learner consists of a prediction and a confidence value and each point in the training set has an associated weight. At each round, AdaBoost chooses the weak learner with the smallest error, increases the weights of wrongly classified training points and decreases the weights of correctly classified points. There are multiple interpretations for AdaBoost, including those of a large margin method (Schapire et al., 1998), a gradient descent procedure in the functional space of convex combinations of weak learners (Mason et al., 2000), and a method for step-wise logistic regression (Friedman et al., 2000), among others (Freund & Schapire, 2004). In this work, we build on a combination of these interpretations to derive a sound cost-sensitive extension, which we denote by *asymmetric boosting*.

Various cost-sensitive extensions of boosting have been previously proposed in the literature, including AdaCost (Fan et al., 1999), CSB0, CSB1, CSB2 (Ting, 2000) asymmetric-AdaBoost (Viola & Jones, 2002) and AdaC1, AdaC2, AdaC3 (Sun et al., 2005). All of these algorithms are heuristic in nature, attempting to achieve cost-sensitivity by direct manipulation of the weights and confidence parameters of AdaBoost. In most cases, it is not clear if, or how, these manipulations modify the loss minimized by boosting, or even how they relate to any of the different interpretations of boosting discussed above. We rely on the statistical interpretation of boosting to derive a natural cost-sensitive extension to the boosting loss and show that, similarly to the latter, this loss can be minimized

Appearing in *Proceedings of the $24^{th}$ International Conference on Machine Learning*, Corvallis, OR, 2007. 

by gradient descent in functional space.

We then derive the asymmetric extension of AdaBoost, and show that, like the original, it is a margin maximization method, which increases the margin of the detector even after the training error is exhausted. The only difference is that the margins are now unbalanced, reflecting the cost structure assigned to the different error types by the asymmetric loss function. We present a thorough experimental evaluation on the face detection problem, demonstrating that the new asymmetric boosting algorithm does indeed possess cost sensitive characteristics, and can meet a target detection rate without any form of (sub-optimal) weight tweaking. Finally, asymmetric boosting is shown to outperform the previously proposed cost-sensitive boosting methods, consistently achieving the smallest cost in various experiments.

## 2. AdaBoost and Cost Sensitive Extensions

We start by briefly reviewing the AdaBoost algorithm and previously proposed cost sensitive extensions.

### 2.1. AdaBoost

AdaBoost can be derived under a loss minimization framework (Hastie et al., 2001), with the goal of producing a decision rule of the form

$$f_T(x) = \sum_{m=1}^{T} \alpha_m G_m(x), \tag{1}$$

where $\{\alpha_m\}_{m=1}^T$ and $\{G_m(x)\}_{m=1}^T$ is a sequence of weak learners usually implemented with a decision stump (threshold of the projection of $x$ along the direction of a feature $\phi_m$). This is accomplished through gradient descent, on the functional space $\mathcal{S}$ of convex combinations of weak learners, with respect to the exponential loss function

$$L = \sum_{i=1}^{n} \exp(-y_i f_T(x_i)) \tag{2}$$

where $\{x_i\}_{i=1}^n$ is a set of training examples and $\{y_i\}_{i=1}^n$ the associated sequence of class labels ($y_i \in \{1, -1\}$). Given $\alpha$ the gradient direction at the $m^{th}$ iteration is

$$G_m(x) \quad = \quad \arg\min_G \sum_{i=1}^{N} w_i^{(m)} \exp\left(-y_i \alpha G(x)\right) \tag{3}$$

where

$$w_i^{(m+1)} = w_i^{(m)} e^{-y_i \alpha_m G_m(x_i)}. \tag{4}$$

The gradient $G_m(x)$ can be computed, independently of $\alpha$, with

$$G_m(x) = \arg\min_G \sum_{i=1}^{N} w_i^{(m)} I(y_i \neq G(x)) \tag{5}$$

and the optimal step is found through a line search along direction $G_m(x)$, which can be computed in closed-form by

$$\alpha = \frac{1}{2} \log\left(\frac{1 - err_{(m)}}{err_{(m)}}\right), \tag{6}$$

where $err_{(m)}$ is the total error for the $m^{th}$ feature. An example $x$ is classified as a positive ($y = 1$) if $f_T(x) > 0$ and negative otherwise.

### 2.2. AdaCost

AdaCost (Fan et al., 1999) is a cost sensitive extension of AdaBoost that incorporates a cost adjustment function $\beta_\delta$ in the weight update rule and in the computation of $\alpha$. The weight update rule is modified into

$$w_i^{(m+1)} = w_i^{(m)} e^{-y_i \alpha_m G_m(x_i) \beta_\delta}. \tag{7}$$

and $\alpha$ is computed with

$$err_{(m)} = \sum_{i=1}^{N} w_i^{(m)} \delta \tag{8}$$

$$\alpha = \frac{1}{2} \log\left(\frac{1 + err_{(m)}}{1 - err_{(m)}}\right) \tag{9}$$

where $\delta = -1$ if $y_i = G_m(x_i)$ and $\delta = 1$ otherwise. Finally $\beta_+ = -0.5C_i + 0.5$, and $\beta_- = 0.5C_i + 0.5$ where $C_i$ is the cost assigned to the misclassification of the $i^{th}$ example.

### 2.3. CSB0, CSB1 and CSB2

CSB0, CSB1, CSB2 (Ting, 2000) are cost sensitive extensions of AdaBoost that only alter the weight update rule, relying on (6) for the computation of $\alpha$. Three different cost structures, based on various simplifications of (4), are considered. In CSB0 the modified weight update rule is

$$w_i^{(m+1)} = C_{\delta(i)} w_i^{(m)}, \tag{10}$$

where $C_{-(i)}$ is the misclassification cost of the $i^{th}$ example and $C_{+(i)} = 1$. In CSB1 the weight update rule becomes

$$w_i^{(m+1)} = C_{\delta(i)} w_i^{(m)} e^{-y_i G_m(x_i)}. \tag{11}$$

Finally, CSB2 relies on

$$w_i^{(m+1)} = C_{\delta(i)} w_i^{(m)} e^{-y_i \alpha_m G_m(x_i)}, \tag{12}$$

reducing to AdaBoost if $C_{-(i)} = C_{+(i)} = 1$.

### 2.4. Asymmetric-AdaBoost

Asymmetric-AdaBoost (Viola & Jones, 2002) is another cost sensitive extension of AdaBoost that only alters the

weight update rule, again relying on (6) to compute $\alpha$. The main idea is to increase the weights of positive examples and decrease the weights of negative examples after each iteration. The modified weight update rule is

$$w_i^{(m+1)} = C w_i^{(m)} e^{-y_i \alpha_m G_m(x_i)}, \tag{13}$$

where $C = (\sqrt{K})^{(\frac{1}{N})}$ for positive examples and $C = (\sqrt{K})^{(\frac{-1}{N})}$ for negative ones. $K$ is the cost ratio and $N$ the number of weak learners. We have not considered this method in our experiments due to its similarity with CSB2.

### 2.5. AdaC1, AdaC2 and AdaC3

AdaC1, AdaC2 and AdaC3 (Sun et al., 2005) are cost sensitive extensions that alter both AdaBoost's weight update rule and formula for $\alpha$. Defining $C_i \doteq c_i$ as the misclassification cost of the $i^{th}$ example, the new weight update rule for AdaC1 is

$$w_i^{(m+1)} = \frac{w_i^{(m)} e^{-y_i \alpha_m G_m(x_i) c_i}}{Z_t}. \tag{14}$$

with

$$Z_t = \sum_{i=1}^{N} w_i^{(m)} e^{-y_i \alpha_m G_m(x_i) c_i}$$

and

$$\alpha = \frac{1}{2} \log \frac{1 + \sum_{y_i = G_m(x_i)} c_i w_i^{(m)} - \sum_{y_i \neq G_m(x_i)} c_i w_i^{(m)}}{1 - \sum_{y_i = G_m(x_i)} c_i w_i^{(m)} + \sum_{y_i \neq G_m(x_i)} c_i w_i^{(m)}}.$$

In AdaC2 weights are updated according to

$$w_i^{(m+1)} = \frac{c_i w_i^{(m)} e^{-y_i \alpha_m G_m(x_i)}}{Z_t}, \tag{15}$$

with

$$Z_t = \sum_{i=1}^{N} c_i w_i^{(m)} e^{-y_i \alpha_m G_m(x_i)}$$

and

$$\alpha = \frac{1}{2} \log \frac{\sum_{y_i = G_m(x_i)} c_i w_i^{(m)}}{\sum_{y_i \neq G_m(x_i)} c_i w_i^{(m)}}.$$

Finally, AdaC3 relies on

$$w_i^{(m+1)} = \frac{c_i w_i^{(m)} e^{-y_i \alpha_m G_m(x_i) c_i}}{Z_t} \tag{16}$$

with

$$Z_t = \sum_{i=1}^{N} c_i w_i^{(m)} e^{-y_i \alpha_m G_m(x_i) c_i}$$

and $\alpha = \frac{1}{2} \log \frac{A}{B}$, where

$$A = \sum_{i=1}^{N} c_i w_i^{(m)} + \sum_{y_i = G_m(x_i)} c_i^2 w_i^{(m)} - \sum_{y_i \neq G_m(x_i)} c_i^2 w_i^{(m)}$$

$$B = \sum_{i=1}^{N} c_i w_i^{(m)} - \sum_{y_i = G_m(x_i)} c_i^2 w_i^{(m)} + \sum_{y_i \neq G_m(x_i)} c_i^2 w_i^{(m)}.$$

While various justifications are given to motivate the different proposals for direct manipulation of the AdaBoost equations, none of these are based on the derivation of an optimal solution for the minimization of a cost sensitive loss. To the best of our knowledge, no such derivation has been previously presented in the literature.

## 3. Asymmetric Boosting

To derive the asymmetric boosting algorithm, we start by recalling a statistical interpretation of boosting, first proposed in (Friedman et al., 2000). This interpretation is based on the facts that 1) the boosting loss $L$ is an empirical estimate of the cost $E[\exp(-yf(x))]$, and 2) this cost is minimized by the symmetric logistic transform of $P(y = 1|x)$,

$$f(x) = \frac{1}{2} \log \frac{P(y = 1|x)}{P(y = -1|x)}. \tag{17}$$

It follows that, from a statistical viewpoint, boosting can be interpreted as a stage-wise procedure for fitting additive logistic regression models.

The dependence of logistic regression on the log-odds ratio of (17) follows from the fact that the Bayes decision rule for the detection problem of interest is a threshold on the latter. This, however, only holds for the "0-1" loss, i.e. the loss that assigns equal costs to false-positives and misses. For an asymmetric loss, with a cost of $C_2$ for false-positives and $C_1$ for misses, the parallel with the Bayes decision rule requires an asymmetric logistic transform

$$f_a(x) = \frac{1}{C_1 + C_2} \log \frac{P(y = 1|x) C_1}{P(y = -1|x) C_2}.$$

This can be shown to minimize

$$E\left[ I(y = 1) e^{-y . C_1 f_a(x)} + I(y = -1) e^{-y . C_2 f_a(x)} \right], \tag{18}$$

suggesting an alternative, asymmetric, boosting loss

$$\begin{aligned} L_a = \sum_{i=1}^{n} \big[ & I(y_i = 1) \exp(-C_1 y_i f_T(x_i)) \\ & + I(y_1 = -1) \exp(-C_2 y_i f_T(x_i)) \big]. \end{aligned} \tag{19}$$

Under the statistical interpretation, minimizing this loss is equivalent to fitting the cost-sensitive logistic regression

model associated with the loss function that assigns zero-cost to correct decisions, cost $C_1$ to misses, and cost $C_2$ to false positives. Minimizing (19) for a specific pair $(C_1, C_2)$ is, in general, different from heuristically tuning the threshold on the rule $f_T(x)$ produced by symmetric boosting. However, the general AdaBoost principle of minimizing a loss by gradient descent on the space of convex combinations of weak learners can be extended to the asymmetric loss of (19).

In fact, by combining (19) with (1), and defining two sets

$$\mathcal{I}_+ = \{i|y_i = 1\} \qquad \mathcal{I}_- = \{i|y_i = -1\}, \tag{20}$$

it follows that the gradient direction and step size which minimize the asymmetric boosting loss at iteration $m$ are

$$(\alpha_m, G_m(x)) = \tag{21}$$
$$\arg\min_{\alpha,G} \sum_{i\in\mathcal{I}_+} w_i^{(m)} e^{-C_1\alpha G(x_i)} + \sum_{i\in\mathcal{I}_-} w_i^{(m)} e^{C_2\alpha G(x_i)}$$

with

$$w_i^{(m+1)} = \begin{cases} w_i^{(m)} e^{-C_1\alpha_m G_m(x_i)}, & i \in \mathcal{I}_+ \\ w_i^{(m)} e^{C_2\alpha_m G_m(x_i)}, & i \in \mathcal{I}_-. \end{cases} \tag{22}$$

It can then be shown that, for a given step size $\alpha$, the gradient direction is

$$G_m(x) = \arg\min_{G(x)} \Big[ (e^{C_1\alpha} - e^{-C_1\alpha}) \cdot b + e^{-C_1\alpha} T_+ + (e^{C_2\alpha} - e^{-C_2\alpha}) \cdot d + e^{-C_2\alpha} T_- \Big] \tag{23}$$

and the optimal step size is the solution of

$$2C_1 \cdot b \cdot \cosh(C_1\alpha) + 2C_2 \cdot d \cdot \cosh(C_2\alpha) = \tag{24}$$
$$C_1 \cdot T_+ \cdot e^{-C_1\alpha} + C_2 \cdot T_- \cdot e^{-C_2\alpha}$$

with

$$T_+ = \sum_{i\in\mathcal{I}_+} w_i^{(m)} \tag{25}$$

$$T_- = \sum_{i\in\mathcal{I}_-} w_i^{(m)} \tag{26}$$

$$b = \sum_{i\in\mathcal{I}_+} w_i^{(m)} I(y_i \neq G(x_i)) \tag{27}$$

$$d = \sum_{i\in\mathcal{I}_-} w_i^{(m)} I(y_i \neq G(x_i)) \tag{28}$$

The gradient descent iteration cycles through the weak learners, for each, solving (24). This can be done efficiently with standard scalar search procedures. In the experiments reported in this paper, the optimal $\alpha$ was found in an average of 6 iterations of bisection search. Given $\alpha$, the loss associated with the weak learner can be computed, and the optimal learner selected according to (23).

The complete asymmetric boosting algorithm is as follows:

- Given training set $(x_1, y_1)....(x_n, y_n)$ where $y \in \{+1, -1\}$ is the class label of example $x$.
- Initialize weights to uniform $w_i = \frac{1}{2|\mathcal{I}_+|}, \forall i \in \mathcal{I}_+, w_i = \frac{1}{2|\mathcal{I}_-|} \forall i \in \mathcal{I}_-$.
- Choose positive real $C_1, C_2$ values.
- For $t = 1, ...., T$ (Where $T$ is the total number of weak learners.)
  1. for each $j$, train a weak learner/step-size pair $(G_j(x); \alpha_j)$ using current weights $w_i$. The loss at any classifier threshold is given by (23) with $\alpha$ found by solving (24)
  2. select the weak learner/step-size $(G_m(x), \alpha_m)$ of smallest loss.
  3. update the weights according to (22).
- The final strong classifier implements the decision rule $h(x) = \text{sign}[\sum_{m=1}^{T} \alpha_m G_m(x)]$

It is worth mentioning that the algorithm is fully compatible with AdaBoost, in the sense that it reduces to the latter when $C_1 = C_2 = 1$.

## 4. Properties of Asymmetric Boosting

In this section we present two experiments that demonstrate two important properties of asymmetric boosting: that it 1) is, indeed, cost sensitive, and 2) produces large-margin classifiers.

### 4.1. Cost Sensitive Properties

To verify that asymmetric boosting produces cost sensitive classifiers, and obtain some intuition about its advantages over the existing techniques, we analyzed a simple synthetic experiment. This consisted of a binary scalar classification problem, involving Gaussian classes of equal variance $\sigma^2 = 1$ and means $\mu_- = -1$ ($y = -1$) and $\mu_+ = 1$ ($y = 1$). We then sampled $10K$ examples per class, simulating the scenario where the class probabilities are uniform. For this problem, the optimal (Bayes) decision rule is to choose class 1 if $x$ is larger than the threshold

$$T_{BDR} = -\frac{1}{2} \ln R \tag{29}$$

where $R = \frac{L(0,1)}{L(1,0)} = \frac{\text{Loss of misclassifying class 1}}{\text{Loss of misclassifying class 0}}$. We consider two cases, $R = 20$ and $R = 5$, for which $T_{BDR} = -1.4979$ and $T_{BDR} = -0.8047$, respectively.

Figure 1 presents the evolution of the threshold (decision boundary) produced by the different cost sensitive boosting algorithms, as a function of the boosting iteration. For all

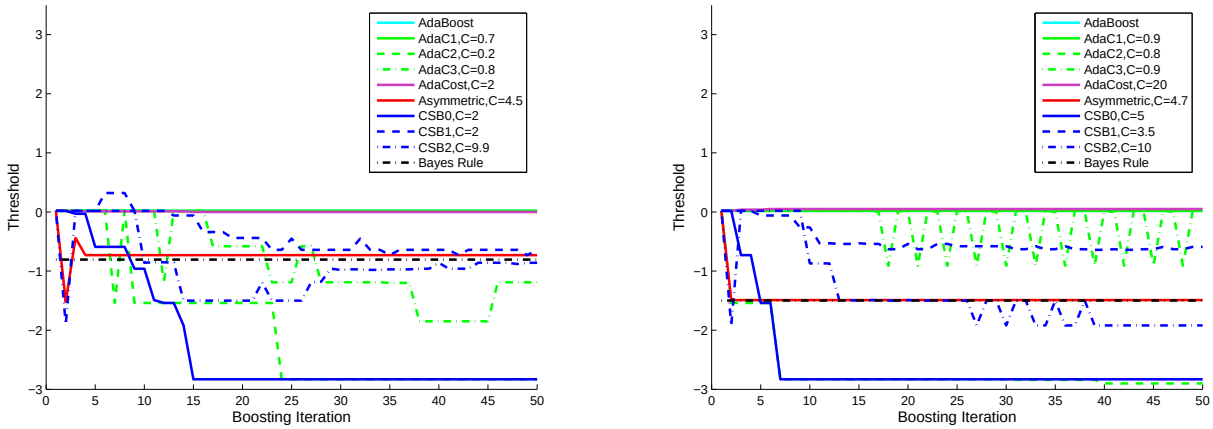



*Figure 1.* Decision boundaries produced by the different boosting algorithms for various cost factors. Left: $R = 5$, right: $R = 20$.

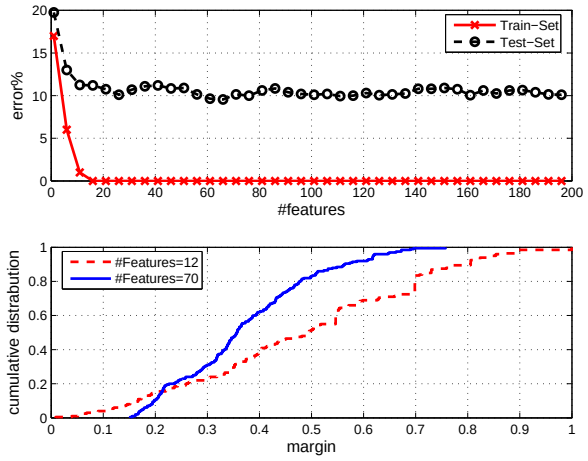



*Figure 2.* Error curves (top) and margin distribution graphs (bottom) for Asymmetric Boosting ($C_1 = 2$) on a face detection problem.

algorithms, we have performed a (rather extensive) search over the range of cost sensitivity parameters (e.g. the ratio between $C_1$ and $C_2$ for asymmetric boosting), so as to achieve the best possible performance after $50$ iterations. Interestingly, this search did not produce good solutions for most of the algorithms. We detected four classes of behaviors. Algorithms in the first class (AdaC1, AdaCost) stubbornly refused to produce any solution other than the cost-insensitive threshold at $0$. This was also the solution produced by AdaBoost, a non-surprising fact given the lack of cost sensitivity of the latter and the symmetry of the problem.

The second class consisted of algorithms (CSB0, AdaC2, AdaC3) that never converged to any meaningful threshold. For this problem, these algorithms did not work at all, in either the cost-sensitive or cost-insensitive sense. The third class consisted of algorithms (CSB1, CSB2) that showed some tendency to converge to the right solution, but were really not able to. While in some cases this was due to a slow convergence rate, in others the algorithms seemed to have converged only to start oscillating, or even diverging. The final class consisted of asymmetric boosting alone. This was the only method that consistently converged to the correct solution in the allotted number of iterations. In particular, asymmetric boosting produced an almost perfect decision boundary of $T_{Asymmetric} = -1.4993$ for $R = 20$, and $T_{Asymmetric} = -0.7352$ for $R = 5$. This was accomplished in only two iterations when $R = 20$ and four when $R = 5$.

The most plausible explanation for the poor performance of all other algorithms appears to be the inappropriate choice of the $\alpha$ parameter: while the weight update rules seemed to produce asymmetric weak learners, the incorrect choice of $\alpha$ frequently gave disproportionate weight to weak learners with poor thresholds. For example, in the case of AdaC1, the first two weak learners have threshold of $0.0152$ and $-0.9186$ but the corresponding values of $\alpha$ are $0.9056$ and $0.2404$. Although the second threshold is close to optimal ($T_{BDR} = -0.8047$), the poor choice of $\alpha = 0.2404$ gives it little weight, much smaller than that of the the first ($\alpha = 0.9056$). This makes the overall decision boundary close to zero. Of all algorithms tested, only CSB1 and CSB2 achieved performance comparable to that of asymmetric boosting, even though the slowness of their convergence in this simple problem appears problematic.

### 4.2. Large Margin Classifier Properties

One of the most important properties of boosting, viewed by many as the reason for the robustness of the resulting

classifiers, is that it tends to continue reducing the test error even after perfect classification is reached on the training set. Schapire et. al used this observation to show that AdaBoost produces large margin classifiers (Schapire et al., 1998), and therefore has good generalization properties. We have applied the same procedure to investigate whether asymmetric boosting maintains this large-margin property. Using a training set of $100$ face images and $100$ nonface images and a test set of $1000$ face images and $1000$ nonface images, we trained asymmetric boosting ($C_1 = 2$) for $200$ iterations. Figure 4.1 (top plot) shows that, while the training error is zero after only $12$ iterations, the test error continues to decrease: from $11.25\%$ after $11$ iterations to $9.55\%$ after $66$. Figure 4.1 (lower plot) depicts the cumulative distribution of the margins after both $12$ and $70$ iterations. While after $12$ iterations the minimum margin of any point is $0.0088$, its value increases to $0.1517$ after $70$ iterations. This considerable increase in the margin is similar to that observed for AdaBoost by Schapire et all, demonstrating that asymmetric boosting maintains the large margin properties of the latter.

### 4.3. Choosing the Cost Parameters

For many cost-sensitive problems, the costs $C_1$ and $C_2$ are naturally specified from domain knowledge. For example, in a fraud detection application, prior experience dictates that there is an average cost of $x$ dollars per false positive, while a false negative (miss) will cost $y > x$ dollars, on average. In this case, the costs are simply the values $x$ and $y$.

For problems where it is more natural to specify desired detection or false-positive rates, the cost parameters $C_1$ and $C_2$ can be determined with resort to the Neyman-Pearson Lemma (Duda et al., 2001). For example, given a specification for a detection rate $\xi$, the optimal cost structure is the one such that

$$\int_{\mathcal{D}} P(x|y=1)dx \quad = \quad \xi \tag{30}$$

with

$$\mathcal{D} = \left\{ x | \frac{P(y=1|x)}{P(y=-1|x)} > \frac{C_2}{C_1} \right\}.$$

Since the optimal decision rule is still the Bayes decision rule, i.e. to decide for class $1$ if $x \in \mathcal{D}$ (and $-1$ otherwise), this does not affect the discussion of Section 3. The only difference is that, rather than specifying the costs, one has to search for the costs that achieve the detection rate of (30). This can be done by cross-validation (note that, because one can always set $C_2$ to the value of one, the search is one-dimensional).

## 5. Evaluation

An important area of application of cost-sensitive learning is the problem of object detection in computer vision, where boosting has recently emerged as the main tool for the design of classifier cascades (Viola & Jones, 2001). These are extremely efficient classifiers, that enable real-time implementation of object detectors, with performance that matches the best results previously available (for non-real time implementation). Since a substantial amount of effort has also been devoted to the design of evaluation protocols in areas like face detection, this is a good domain in which to test cost-sensitive classifiers. We have adopted the protocol of (Viola & Jones, 2001) to compare asymmetric boosting to all previously discussed cost sensitive boosting algorithms. All experiments used a face database of $9832$ positive and $9832$ negative examples, and weak learners based on a combination of decision stumps and Haar wavelet features. $6K$ examples were used, per class, for training, the remaining $3832$ being left for testing. All boosting algorithms were allowed to run for $100$ iterations.

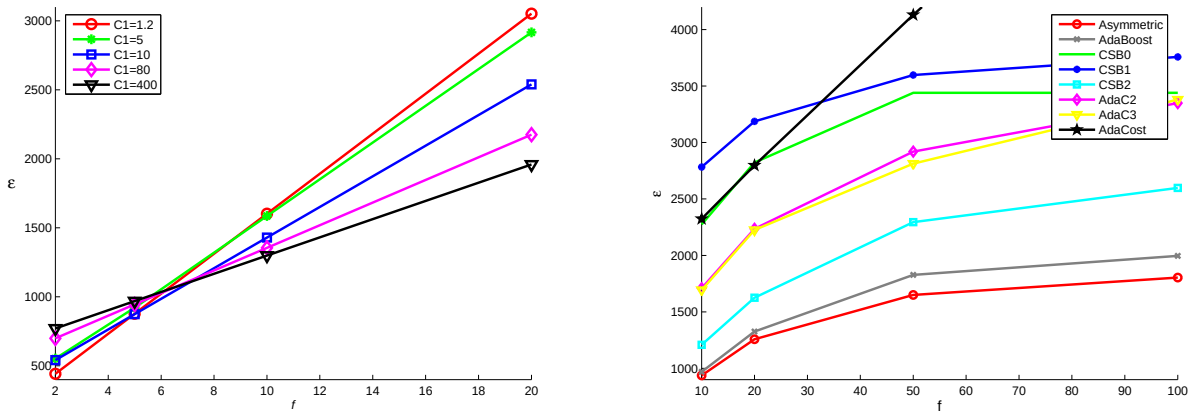
The evaluation of cost-sensitive classification requires a classification metric that weighs some errors more than others. A commonly used metric, which we adopt here, is

$$\epsilon = p + f \times m \tag{31}$$

where $p$ is the number of false-positives of the detector, $m$ the number of misses [1], and $f > 1$ a cost factor that weighs misses more heavily than false positives. Four cost factors ($f = 10, 20, 50, 100$) were considered, and the misclassification cost $\epsilon$ computed for each combination of 1) cost sensitive boosting method, 2) training cost structure, and 3) cost factor $f$ of the classification metric $\epsilon$.

By training cost structure we refer to the ratio between the costs assigned to the different types of errors during training, e.g. the constants $C_1$ and $C_2$ of asymmetric boosting, $C_-$ and $C_+$ of CSB, etc. For each method, we found the range of values of this ratio that spans the operating range achievable by the classifier. This is illustrated, in Figure 3 a) for asymmetric boosting. The figure presents plots of the cost metric $\epsilon$ as a function of the cost factor $f$ for various training cost structures, obtained by setting $C_2 = 1$ and letting $C_1$ take a number of values in the interval $[1.2, 1000]$. Note that detectors trained with larger values of $C_1$ perform better under cost functions with larger $f$, while small ratios lead to best performance when $\epsilon$ weighs the two errors more equally. This confirms the cost-sensitive nature of asymmetric boosting. Note, also, that the slope of the lines $\epsilon(f)$ decreases monotonically with $C_1$, saturating at some point. The operating range of the classifier is the range of slopes that it can achieve.

[1] A miss happens when a positive example is not detected.




(a) (b)

*Figure 3.* (a) Misclassification cost for asymmetric boosting under different training cost structures. (b) Minimum misclassification cost of various cost-sensitive boosting methods on a face detection problem.

Figure 3 b) presents a comparison of the best performance achieved with each of the cost-sensitive boosting methods. The plots were produced by 1) running each method with four cost ratios, within the operating range of the resulting classifiers, and 2) searching for the threshold that achieved the minimum cost at each cost factor $f$. Because AdaBoost does not have a cost parameter (it is equivalent to asymmetric boosting with $C_1 = 1$), it was only subject to a threshold search. It is clear that asymmetric boosting consistently outperforms all other techniques, for all values of $f$.

These results also illustrate the importance of choosing the confidence $\alpha$ optimally, at each iteration. On one hand, methods that do not use $\alpha$ in the weight update rule (CSB0 and CSB1) have very poor performance. On the other, methods that try to be creative with respect to the selection of $\alpha$, but are not provably optimal (AdaC2, AdaC3, and AdaCost), perform worse than simply using the procedure originally proposed in AdaBoost (also used in CSB2). Nevertheless, because AdaBoost is not optimal in the cost-sensitive sense, this is clearly inferior to asymmetric boosting.

Table 1 presents results for the case where a threshold search in not performed for asymmetric boosting (or AdaBoost), but still allowed for the other methods. Despite the extra degree of freedom, none of the latter achieve performance comparable to that of asymmetric boosting. Note that AdaBoost performs well for small cost factors, but fails when these are high. This was expected, since it is the special case of asymmetric boosting with cost-ratio of 1, but may be the reason for previous reports of superior performance by the other methods (when compared with AdaBoost). In terms of these methods, AdaC3 performs better than AdaC2 at high cost factors, confirming the results of (Sun et al., 2005), and CSB2 outperforms CSB0 and CSB1.

*Table 1.* Smallest misclassification costs and corresponding cost ratio $C$ for different methods, cost factors.

| Method | $f = 2$ | $f = 5$ | $f = 10$ | $f = 20$ |
|---|---|---|---|---|
| Asym | **442** $C = 1.2$ | **814** $C = 20$ | **1259** $C = 20$ | **1959** $C = 400$ |
| AdaBoost | 445 $C = 1$ | 877 $C = 1$ | 1597 $C = 1$ | 3037 $C = 1$ |
| CSB0 | 1136 $C = 3$ | 1657 $C = 3$ | 2271 $C = 3$ | 2841 $C = 3$ |
| CSB1 | 1697 $C = 3$ | 2304 $C = 3$ | 2778 $C = 3$ | 3181 $C = 3$ |
| CSB2 | 566 $C = 2$ | 899 $C = 2$ | 1363 $C = 2$ | 1989 $C = 2$ |
| AdaC2 | 810 $C = 0.7$ | 1220 $C = 0.7$ | 1739 $C = 0.7$ | 2264 $C = 0.7$ |
| AdaC3 | 877 $C = 0.7$ | 1338 $C = 0.7$ | 1713 $C = 0.7$ | 2243 $C = 0.7$ |
| AdaCost | 1428 $C = 0.05$ | 1987 $C = 0.05$ | 2324 $C = 0.05$ | 2997 $C = 0.05$ |

AdaCost performs poorly, confirming the results reported in (Ting, 2000). As mentioned in (Ting, 2000), because $\beta_+$ is non-increasing, the reward for correct classification is small when cost is high and vice versa. This is counter intuitive, and could be the source of AdaCost's poor performance. Finally, it should be mentioned that determining $\alpha$ in AdaC1 and AdaCost was especially problematic. In various situations these algorithms are unstable, repeatedly producing meaningless negative, or even imaginary, $\alpha$ values. AdaC1 results are not reported due to this problem.

## 6. Conclusion

In this work, we have presented a novel cost-sensitive boosting algorithm. This algorithm is based on the statisti-

cal interpretation of boosting, and derived with recourse to an asymmetric extension of the logistic transform, which is well motivated from a decision theoretic point of view. The statistical interpretation enables the derivation of a principled asymmetric boosting loss which, similarly to the original AdaBoost algorithm, is then minimized by gradient descent in the functional space of convex combinations of weak learners. The resulting asymmetric boosting algorithm provides a proper combination of 1) cost-sensitive weight update rule, and 2) cost-sensitive method for finding $\alpha$.

Experimental evidence, derived from both a synthetic problem and the (timely) problem of face detection, was presented in support of the cost-sensitive and large margin properties of asymmetric boosting. The performance of the latter was also compared to those of various previous cost-sensitive boosting proposals (CSB0, CSB1, CSB2, AdaC1, AdaC2, AdaC3 and AdaCost), in the face detection problem. Asymmetric boosting was shown to consistently outperform all other methods, achieving the smallest misclassification cost at all cost factors considered.

Previous attempts at producing a cost sensitive boosting algorithm have mostly relied on heuristic alterations of the AdaBoost algorithm, resulting in inconsistencies that where shown to significantly degrade performance. Asymmetric boosting is derived from sound machine learning principles, eliminating these problems. Due to this, we believe that asymmetric boosting is a large margin cost-sensitive boosting algorithm that will find use in many areas of application of machine learning. We are currently pursuing its application to the design of optimal object detection cascades, in computer vision.

## References


Chawla, N. V., Lazarevie, A., Hall, L. O., & Bowyer, K. (2003). Smoteboost: Improving prediction of the minority class in boosting. *In Proceedings of Principles of Knowledge Discovery in Databases*.

Domingos, P. (1999). Metacost: a general method for making classifiers cost-sensitive. *Proceedings of the fifth ACM SIGKDD*. ACM Press.

Duda, R., Hart, P. E., & Stork, D. (2001). *Pattern classification*. New York: Wiley and Sons.

Fan, W., Stolfo, S., Zhang, J., & Chan, P. (1999). Adacost: Misclassification cost-sensitive boosting. *ICML*.

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

Freund, Y., & Schapire, R. (2004). A discussion of "Process consistency for AdaBoost" by Wenxin Jiang, "On the Bayes-risk consistency of regularized boosting methods" by Gabor Lugosi and Nicolas Vayatis, "Statistical behavior and consistency of classification methods based on convex risk minimization" by Tong Zhang. *Annals of Statistics*.

Friedman, J., Hastie, T., & Tibshirani, R. (2000). Additive logistic regression: A statistical view of boosting. *Journal of Annals of Statistics*.

Guo, H., & Viktor, H. L. (2004). Learning from imbalanced data sets with boosting and data generation: the databoost-im approach. *SIGKDD Explor. Newsl.*

Hastie, Tibshirani, & Friedman (2001). *The elements of statistical learning*. New York: Springer-Verlag Inc.

Mason, L., Baxter, J., Bartlett, P., & Frean, M. (2000). Boosting Algorithms as Gradient Descent. *NIPS*.

Park, S.-B., Hwang, S., & Zhang, B.-T. (2003). Mining the risk types of human papillomavirus (hpv) by adacost. *International Conference on Database and expert Systems Applications*.

Schapire, R. E., Freund, Y., Bartlett, P., & Lee, W. S. (1998). Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*.

Sun, Y., Wong, A. K. C., & Wang, Y. (2005). Parameter inference of cost-sensitive boosting algorithms. *Machine Learning and Data Mining in Pattern Recognition,4th International Conference*.

Ting, K. M. (2000). A comparative study of cost-sensitive boosting algorithms. *ICML*.

Viaene, S., Derrig, R. A., & Dedene, G. (2004). Cost-sensitive learning and decision making for massachusetts pip claim fraud data. *International Journal of Intelligent Systems*.

Viola, P., & Jones, M. (2001). Robust real-time object detection. *Proc. 2nd Intl Workshop on Statistical and Computational Theories of Vision Modeling, Learning, Computing and Sampling*. Vancouver, Canada.

Viola, P., & Jones, M. (2002). Fast and robust classification using asymmetric adaboost and a detector cascade. *NIPS*.

Zadrozny, B., Langford, J., & Abe, N. (2003). A simple method for cost-sensitive learning. *Technical Report RC22666, IBM*.