Cost-Sensitive Boosting

Hamed Masnadi-Shirazi and Nuno Vasconcelos

SVCL-TR 2007/06

June 2007

Cost-Sensitive Boosting

Hamed Masnadi-Shirazi and Nuno Vasconcelos
Statistical Visual Computing Lab
Department of Electrical and Computer Engineering
University of California, San Diego

June 2007

Abstract

A novel framework, based on the statistical interpretation of boosting, is proposed for the design of cost sensitive boosting algorithms. It is argued that, although predictors produced with boosting converge to the ratio of posterior class probabilities that also appears in Bayes decision rule, this convergence only occurs in a small neighborhood of the optimal cost-insensitive classification boundary. This is due to a combination of the cost-insensitive nature of current boosting losses, and boosting's sample reweighing mechanism. It is then shown that convergence in the neighborhood of a target cost-sensitive boundary can be achieved through boosting-style minimization of extended, cost-sensitive, losses. The framework is applied to the design of specific algorithms, by introduction of cost-sensitive extensions of the exponential and binomial losses. Minimization of these losses leads to cost sensitive extensions of the popular AdaBoost, RealBoost, and LogitBoost algorithms. Experimental validation, on various UCI datasets and the computer vision problem of face detection, shows that the new algorithms substantially improve performance over what was achievable with previous cost-sensitive boosting approaches.

Author email: hmasnadi@ucsd.edu

SVCL Technical reports are available on the SVCL's web page at
http://www.svcl.ucsd.edu

University of California, San Diego
Statistical Visual Computing Laboratory
9500 Gilman Drive, Mail code 0407
EBU 1, Room 5512
La Jolla, CA 92093-0407

1 Introduction

Classification problems such as fraud detection [21, 41], medical diagnosis [15, 20, 23, 44, 46], or object detection in computer vision [42], are naturally cost sensitive [6]. In these problems the cost of missing a target is much higher than that of a false-positive, and classifiers that are optimal under symmetric costs (such as the popular zero-one loss) tend to under perform. The design of optimal classifiers with respect to losses that weigh certain types of errors more heavily than others is denoted as cost-sensitive learning [6]. Current research in this area falls into two main categories. The first aims for generic procedures that can make arbitrary classifiers cost sensitive, by resorting to Bayes risk theory or some other cost minimization strategy [4, 24, 48, 49]. The second attempts to extend particular algorithms, so as to produce cost-sensitive generalizations.

Of interest to this work are classifiers obtained by thresholding a continuous function, here denoted as a *predictor*, and therefore similar to the Bayes decision rule (BDR) [5, 45], which is well known to be optimal for both cost-insensitive and cost-sensitive classification. In particular, we consider learning algorithms in the boosting family [2, 8, 12, 36], such as the popular AdaBoost [8, 10], which is not cost-sensitive but has achieved tremendous practical success in important areas of application, such as computer vision [43]. Like all other boosting algorithms, AdaBoost learns a predictor by composing an ensemble of weak classification rules (weak learners), and relies on a sample re-weighting mechanism to place greater emphasis on a neighborhood of the classification boundary. This guarantees a large classification margin and good (cost-insensitive) generalization with small amounts of training data. There are multiple interpretations for Adaboost, including those of a large margin method [33, 35], a gradient descent procedure in the functional space of convex combinations of weak learners [13, 25, 50], and a method for step-wise logistic regression [3, 12], among others [2, 9, 11].

This work builds on a combination of these interpretations to derive a cost-sensitive boosting extension. We start with the observation, by Friedman et al. [12], that the predictor which minimizes the exponential loss used by AdaBoost (and many other boosting algorithms) is the ratio of posterior distributions that also appears in the BDR. Given the optimality of the latter, this offers an explanation for the excellent performance of boosted detectors in cost-insensitive classification problems. It is, however, at odds with various empirical observations of boosting's 1) poor cost-sensitive performance [7, 27, 37, 38, 42], and 2) inability to produce well calibrated estimates of class posterior probabilities [12, 19, 26, 27, 30]. We argue that this is an intrinsic limitation of the large-margin nature of boosting: due to the emphasis (sample reweighing) on the classification border, the predictor produced by boosting only converges to the BDR in a small neighborhood of that border. Outside this neighborhood, it has identical sign to the BDR (a sufficient condition for cost-insensitive classification) but does not necessarily approximate it well (a necessary condition for good cost-sensitive performance).

Two conditions are identified as necessary for optimal cost-sensitive boosting: 1) that the predictor does converge to the BDR in the neighborhood of a classification boundary, but 2) that the latter is the target cost-sensitive boundary, rather than the one optimal in the cost-insensitive sense. We propose that this is best accomplished

by modifying the loss function minimized by boosting, so that boosting-style gradient descent can satisfy the two conditions. This leads to a general framework for the cost-sensitive extension of boosting algorithms. We introduce cost-sensitive versions of the exponential and binomial losses, which underly some of the most popular boosting algorithms, including AdaBoost [8], RealBoost [12, 36], and LogitBoost [12]. Cost-sensitive extensions of these algorithms are then derived, and shown to satisfy the two necessary conditions for cost-sensitive optimality.

Various cost-sensitive extensions of boosting have been previously proposed in the literature, including AdaCost [7], CSB0, CSB1, CSB2 [38] asymmetric-AdaBoost [42] and AdaC1, AdaC2, AdaC3 [37]. 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. This is unlike the framework now proposed, which relies on the statistical interpretation of boosting to derive cost-sensitive extensions of the boosting loss. Due to this, the algorithms now proposed inherit all the properties of classical, cost-insensitive, boosting. They simply shift boosting's emphasis from the neighborhood of the cost-insensitive boundary to the neighborhood of the target cost-sensitive boundary.

The performance of the proposed cost-sensitive boosting algorithms is evaluated empirically, through experiments on both synthetic classification problems (which provide insight) and standard datasets from the UCI repository and computer vision (face detection). These experiments show that the algorithms do indeed possess cost sensitive optimality, and can meet target detection rates without (sub-optimal) weight or threshold manipulation. They are also shown to outperform the previously available cost-sensitive boosting methods, consistently achieving the best results in all experiments.

The paper is organized as follows. In Section 2 we review the main principles of cost-sensitive classification. Section 3 then presents a brief review of the standard boosting algorithms and previous attempts at cost-sensitive extensions, discussing their limitations for optimal cost-sensitive classification. The new framework for cost-sensitive boosting is introduced in Section 4, where the extensions of AdaBoost, RealBoost, and LogitBoost, are also derived. Finally, the empirical evaluation is discussed in Section 5, and some conclusions are drawn in Section 6.

2 Cost-sensitive classification

We start by reviewing the fundamental concepts of cost-sensitive classification. Although most of these apply to multi-way classification problems, in this work we only consider the binary case, usually referred to as the *detection* problem.

2.1 Detection

A detector, or binary classifier, is a function $h : \mathcal{X} \rightarrow \{-1, 1\}$ that maps a feature vector $\mathbf{x} = (x_1, \ldots, x_N)^T \in \mathcal{X} \subset \mathbb{R}^N$ into a class label $y \in \{-1, 1\}$. This mapping

is implemented as

$$h(\mathbf{x}) = sgn[f(\mathbf{x})] \tag{1}$$

where $f : \mathcal{X} \rightarrow \mathbb{R}$ is a predictor, and $sgn[x] = 1$ if $x \geq 0$, and $sgn[x] = -1$ otherwise. Feature vectors are samples from a random process $\mathbf{X}$ that induces a probability distribution $P_{\mathbf{X}}(\mathbf{x})$ on $\mathcal{X}$, and labels are samples from a random variable Y that induces a probability distribution $P_Y(y)$ in $\{-1, 1\}$.

The detector is optimal if it minimizes the risk

$$R = E_{\mathbf{X},Y}[L(\mathbf{x}, y)],$$

where $L(\mathbf{x}, y)$ is a loss function. We consider losses of the form

$$L(\mathbf{x}, y) = \begin{cases} 0, & \text{if } h(\mathbf{x}) = y \\ C_2 & \text{if } y = -1 \text{ and } h(\mathbf{x}) = 1 \\ C_1 & \text{if } y = 1 \text{ and } h(\mathbf{x}) = -1 \end{cases}, \tag{2}$$

with $C_i > 0$. When $C_1 = C_2$ the detector is said to be cost-insensitive, otherwise it is cost-sensitive. The three scenarios accounted by $L(\mathbf{x}, y)$ are denoted as correct decisions ($h(\mathbf{x}) = y$), false positives ($y = -1$ and $h(\mathbf{x}) = 1$), and false-negatives or misses ($y = 1$ and $h(\mathbf{x}) = -1$).

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 C_2 dollars per false positive, while a false negative (miss) will cost $C_1 > C_2$ dollars, on average. In this case, the costs are simply the values C_2 and C_1. There are, nevertheless, other problems in which it is more natural to specify target detection or false-positive rates than to specify costs. The two types of problems can be addressed within a common optimal detection framework.

2.2 Optimal detection

We start by considering the case where the costs C_1 and C_2 are specified. In this case, it is well known that the optimal predictor is given by the BDR [5, 45], i.e.

$$f^* = \arg\min_f E_{\mathbf{X},Y}[L(\mathbf{x}, y)]$$

if and only if

$$f^*(\mathbf{x}) = \frac{P_{Y|\mathbf{X}}(1|\mathbf{x})C_1}{P_{Y|\mathbf{X}}(-1|\mathbf{x})C_2}. \tag{3}$$

When the specification is in terms of error rates, this result still holds, but the cost structure (C_1, C_2) that meets the specified rates must be determined. This can be done with resort to the Neyman-Pearson Lemma [29]. For example, given the specification of detection rate ξ, the optimal cost structure is the one such that

$$\int_{\mathcal{H}} P(\mathbf{x}|y = 1)d\mathbf{x} \quad = \quad \xi \tag{4}$$

with

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

Note that the optimal decision rule is still the BDR, i.e. to decide for class 1 if $\mathbf{x} \in \mathcal{H}$ (and -1 otherwise). The only difference is that, rather than specifying the costs, one has to search for the costs that achieve the detection rate of (4). This can be done by cross-validation. Note that, because all that matters is the ratio C_1/C_2, C_2 can be set to one and the search is one-dimensional.

In any case, the optimal detector can be written as

$$h_T^*(\mathbf{x}) = sgn\left[\log\left(f_0^*(\mathbf{x})\right) - T\right] \tag{5}$$

where

$$f_0^*(\mathbf{x}) = \frac{P_{Y|\mathbf{X}}(1|\mathbf{x})}{P_{Y|\mathbf{X}}(-1|\mathbf{x})}, \tag{6}$$

is the optimal cost-insensitive predictor and

$$T = \log \frac{C_1}{C_2}. \tag{7}$$

Hence, for any cost structure (C_1, C_2), cost-sensitive optimality differs from cost-insensitive optimality only through the threshold T: given $f_0^*(\mathbf{x})$ all optimal cost-sensitive rules can be obtained by simple threshold manipulation. Furthermore, from (4), different thresholds correspond to different detection rates, and threshold manipulation can produce the optimal decision functions at any desired detection (or false-positive) rate. This is the motivation for the widespread use of receiver operating curves (ROCs) [1, 14, 16, 18, 39], and the tuning of error rates by threshold manipulation.

2.3 Practical detection

In practice, the posterior probabilities of (6) are unknown, and a learning algorithm is used to estimate the predictor

$$\hat{f}(\mathbf{x}) \approx f_0^*(\mathbf{x}), \tag{8}$$

enabling the implementation of approximately optimal cost-sensitive rules

$$\hat{h}_T(\mathbf{x}) = sgn[\hat{f}(\mathbf{x}) - T]. \tag{9}$$

While this is a commonly used strategy to obtain cost-sensitive rules, it does not necessarily guarantee good cost-sensitive performance. In fact, there are no guarantees of the latter *even when the cost-insensitive detector is optimal*, i.e. when

$$\hat{h}_0(\mathbf{x}) = sgn[f_0^*(\mathbf{x})]. \tag{10}$$

While the necessary and sufficient conditions for (10) are that

$$\hat{f}(\mathbf{x}) \quad = \quad f_0^*(\mathbf{x}) = 0, \ \ \forall \mathbf{x} \in \mathcal{C} \tag{11}$$

$$sgn[\hat{f}(\mathbf{x})] \quad = \quad sgn[f_0^*(\mathbf{x})], \ \ \forall \mathbf{x} \notin \mathcal{C}, \tag{12}$$

where

$$\mathcal{C} = \left\{ \mathbf{x} \left| \frac{P_{Y|\mathbf{X}}(1|\mathbf{x})}{P_{Y|\mathbf{X}}(-1|\mathbf{x})} = 1 \right. \right\}$$

is the optimal cost-insensitive classification boundary, the optimality of (9) requires that

$$\hat{f}(\mathbf{x}) \quad = \quad f_0^*(\mathbf{x}) = T, \ \ \forall \mathbf{x} \in \mathcal{C}_T \tag{13}$$

$$sgn[\hat{f}(\mathbf{x}) - T] \quad = \quad sgn[f_0^*(\mathbf{x}) - T], \ \ \forall \mathbf{x} \notin \mathcal{C}_T \tag{14}$$

with

$$\mathcal{C}_T = \left\{ \mathbf{x} \left| \frac{P_{Y|\mathbf{X}}(1|\mathbf{x})}{P_{Y|\mathbf{X}}(-1|\mathbf{x})} = T \right. \right\}.$$

Hence, for any $\mathbf{x} \in \mathcal{C}_T$, the necessary condition for cost-sensitive optimality

$$\hat{f}(\mathbf{x}) = f_0^*(\mathbf{x}) \tag{15}$$

is much tighter than the sufficient condition for cost-insensitive optimality

$$sgn[\hat{f}(\mathbf{x})] = sgn[f_0^*(\mathbf{x})]. \tag{16}$$

It follows that threshold manipulation can only produce optimal cost-sensitive detectors for all values of T if $\hat{f}(\mathbf{x}) = f_0^*(\mathbf{x}), \forall \mathbf{x} \in \mathcal{X}$. Since this is a much more restrictive constraint than the necessary and sufficient conditions, (11) and (12), for cost-insensitive optimality there is, in general, no reason for a cost-insensitive learning algorithm to enforce it. This is, in fact, Vapnik's argument against generative solutions to the classification problem: that there is no point in attempting to learn the optimal predictor everywhere, when it is sufficient to do so on the classification boundary [40]. In summary, manipulating the threshold of an optimal cost-insensitive detector provides no guarantees of optimal cost-sensitive performance.

3 Boosting

We consider cost-sensitive extensions of boosting algorithms. Such algorithms learn a predictor $f(\mathbf{x})$ by linear combination of simple decision rules $G_m(\mathbf{x})$, known as weak learners [34],

$$f(\mathbf{x}) = \sum_{m=1}^{M} G_m(\mathbf{x}). \tag{17}$$

Predictor optimality is defined with respect to some loss function $l[y, f(\mathbf{x})]$, such as the exponential loss

$$l_e[y, f(\mathbf{x})] = E_{\mathbf{X},Y}[\exp(-yf(\mathbf{x}))], \tag{18}$$

or the expected negative binomial log-likelihood

$$l_b[y', f(\mathbf{x})] \quad = \quad -E_{\mathbf{X},Y}[y' \log(p(\mathbf{x})) + (1 - y') \log(1 - p(\mathbf{x}))] \tag{19}$$

where $y' = (y+1)/2 \in \{0,1\}$ is a re-parametrization of y and

$$p(\mathbf{x}) = \frac{e^{f(\mathbf{x})}}{e^{f(\mathbf{x})} + e^{-f(\mathbf{x})}}. \tag{20}$$

Learning is based on a training sample of feature vectors $\{\mathbf{x}_i\}_{i=1}^n$ and labels $\{y_i\}_{i=1}^n$, empirical estimates of these losses, and the iterative selection of weak learners. At iteration m, a weight $w_i^{(m)}$ is assigned to example $(\mathbf{x}_i, y_i)$ and the sample is reweighed so as to amplify the importance of points that are poorly classified with the current ensemble predictor of (17). We next review some popular examples of algorithms in this family, whose cost-sensitive extensions will be introduced in later sections. In all these cases, boosting can be interpreted as gradient descent on a functional space of linear combinations of weak learners, with respect to one of the losses above [13, 25, 50].

3.1 AdaBoost

AdaBoost [8, 10] produces combinations of scaled binary classifiers

$$G_m^{Ada}(\mathbf{x}) = \alpha_m g_m(\mathbf{x}), \tag{21}$$

where $\{\alpha_m\}_{m=1}^M$ is a weight sequence and $\{g_m(\mathbf{x})\}_{m=1}^M$ a sequence of binary rules, $g_m(\mathbf{x}) : \mathcal{X} \to \{-1, 1\}$, usually implemented with a decision stump

$$g_m(\mathbf{x}) = sgn[\phi_m(\mathbf{x}) - t_m]$$

where $\phi_m(\mathbf{x})$ is a feature response (usually the projection of $\mathbf{x}$ along the direction of a basis function ϕ_m) and t_m a threshold. The ensemble predictor of (17) is learned by gradient descent with respect to the exponential loss, for which the gradient at the m^{th} iteration is [17, 25]

$$g_m(\mathbf{x}) = \arg\min_g \sum_{i=1}^n w_i^{(m)}[1 - I(y_i = g(\mathbf{x}_i))], \tag{22}$$

where $I(\cdot)$ is the indicator function

$$I(y = x) = \begin{cases} 1 & y = x \\ 0 & y \neq x. \end{cases} \tag{23}$$

α_m is the optimal step size in the direction of the gradient, found by a line search with closed-form solution

$$\alpha_m = \frac{1}{2}\log\left(\frac{1 - err_{(m)}}{err_{(m)}}\right), \tag{24}$$

where

$$err_{(m)} = \sum_{i=1}^n w_i^{(m)}[1 - I(y_i = g_m(\mathbf{x}_i))], \tag{25}$$

is the total error of $g_m(\mathbf{x})$. The weights are updated according to

$$w_i^{(m+1)} = w_i^{(m)} e^{-y_i G_m^{Ada}(\mathbf{x}_i)}. \tag{26}$$

3.2 RealBoost

RealBoost [12, 36] is an extension of AdaBoost that produces better estimates of the optimal predictor $f_0^*(\mathbf{x})$ by using real-valued weak learners in (17). In this case, the gradient of the exponential loss is a (re-weighted) log-odds ratio

$$G_m^{real}(\mathbf{x}) = \frac{1}{2}\log\frac{P_{Y|\mathbf{X}}^{(w)}(1|\phi_m(\mathbf{x}))}{P_{Y|\mathbf{X}}^{(w)}(-1|\phi_m(\mathbf{x}))}, \tag{27}$$

where, as before, $\phi_m(\mathbf{x})$ is a feature response to $\mathbf{x}$, and the superscript w indicates that the probability distribution is that of the re-weighted sample. Weights are updated according to

$$w_i^{(m+1)} = w_i^{(m)} e^{-y_i G_m^{real}(\mathbf{x}_i)}. \tag{28}$$

3.3 LogitBoost

Logitboost is motivated by the following observation, initially made by Friedman et al. [12].

Lemma 1. (Statistical interpretation of boosting.)

The loss $E[\exp(-yf(\mathbf{x}))]$ *is minimized by the symmetric logistic transform of* $P_{Y|\mathbf{X}}(1|\mathbf{x})$,

$$f_0^*(x) = \frac{1}{2}\log\frac{P_{Y|\mathbf{X}}(1|\mathbf{x})}{P_{Y|\mathbf{X}}(-1|\mathbf{x})}. \tag{29}$$

Proof. See [12]. ∎

This implies that both Ada and RealBoost can be interpreted as step-wise procedures for fitting an additive logistic regression model. Friedman et al. argued that this is more naturally accomplished by step-wise minimization of the classical logistic regression losses, namely the expected negative binomial log-likelihood of (19). At the m^{th} boosting iteration, the optimal step with respect to the binomial loss can be found by solving a weighted least squares regression for the weak learner $G_m^{logit}(\mathbf{x})$ that best fits a set of working responses

$$z_i^{(m)} = \frac{y_i' - p^{(m)}(\mathbf{x}_i)}{p^{(m)}(\mathbf{x}_i)(1-p^{(m)}(\mathbf{x}_i))},$$

where $p^{(m)}(\mathbf{x})$ is the probability of (20) based on the ensemble predictor of (17) after $m-1$ iterations. The weights are

$$w_i^{(m)} = p^{(m)}(\mathbf{x}_i)(1-p^{(m)}(\mathbf{x}_i)). \tag{30}$$

3.4 Limitations for cost-sensitive learning

While both the minimization of the exponential and binomial losses are sufficient to obtain the optimal cost-insensitive predictor of (29), we have already seen that everywhere convergence to this predictor is not necessary to produce the optimal cost-insensitive detector. For this, it suffices that the ensemble predictor of (17) converges to any function $\hat{f}(\mathbf{x})$ that satisfies (11) and (12). From a purely cost-insensitive perspective it is, thus, sensible to require a greater accuracy of the approximation inside a neighborhood of the optimal cost-insensitive boundary $\mathcal{C}$ than outside of it. This is exactly what boosting does, through the example re-weighting step of (26), (28), or (30). For both Ada and RealBoost, a simple recursion shows that, after M iterations,

$$\frac{w_i^{(M)}}{w_i^{(0)}} = e^{-y_i \sum_{m=1}^{M} G_m(\mathbf{x}_i)} = e^{-y_i f(\mathbf{x}_i)},$$

where we have also used (17). Assuming that $f(\mathbf{x})$ satisfies the necessary condition for cost-insensitive optimality of (11), this ratio is one along $\mathcal{C}$, exponentially increasing (with the distance to this boundary) for incorrectly classified points, and exponentially decreasing for correctly classified points. Hence, with the exception of a (hopefully) small number of misclassified points, the weight is concentrated on a neighborhood $\mathcal{N}(\mathcal{C})$ of the cost-insensitive boundary $\mathcal{C}$. For LogitBoost, the weight $w_i^{(M)}$ is a symmetric function of $p^{(M)}(\mathbf{x}_i)$, with maximum at $p^{(M)}(\mathbf{x}_i) = 1/2$ or, from (20), at $f(\mathbf{x}_i) = 0$. In fact,

$$w_i^{(M)}(\mathbf{x}_i) = \left(e^{f(\mathbf{x}_i)} + e^{-f(\mathbf{x}_i)}\right)^{-2} \approx e^{-2sgn[f(\mathbf{x}_i)]f(\mathbf{x}_i)} = e^{-2|f(\mathbf{x}_i)|}$$

and the weight decays exponentially with the distance from the boundary, independently of whether the points are correctly classified or not.

In summary, boosting assigns exponentially decaying weight to points that have been well classified during previous iterations, in the *cost-insensitive* sense. These points, which are far from the cost-insensitive boundary, are exponentially discounted as the optimization progresses. The resulting emphasis on $\mathcal{N}(\mathcal{C})$ is a definite advantage for the design of the cost-insensitive detector, by guaranteing a large margin and an ensemble predictor $f(\mathbf{x})$ whose zero-level set very closely approximates $\mathcal{C}$. This is illustrated in Figure 1, where we depict the level sets of a hypothetic optimal cost-insensitive predictor $f_0^*(\mathbf{x})$ and a hypothetic ensemble predictor $f(\mathbf{x})$. Because $f_0^*(\mathbf{x})$ is monotonically increasing to the left of $\mathcal{C}$ (and monotonically decreasing to its right), any ensemble predictor which 1) has $\mathcal{C}$ as a zero-level set, and 2) exhibits the same monotonicity, will both 1) satisfy (11)-(12), and 2) have great generalization ability for cost-insensitive classification.

However, this effort to maximize the margin does not guarantee that, outside $\mathcal{N}(\mathcal{C})$, the level sets of $f(\mathbf{x})$ are *identical* to those of $f^*(\mathbf{x})$. In particular, the level set $f(\mathbf{x}) = T$ is significantly different from the level set $f_0^*(\mathbf{x}) = T$, the optimal cost-sensitive boundary $\mathcal{C}_T$ under the cost-structure correspondent to a threshold of T in (5). It follows that threshold manipulation on the ensemble predictor $f(\mathbf{x})$ does not lead to the optimal cost-sensitive decision rule of (5). The inability of boosting to produce

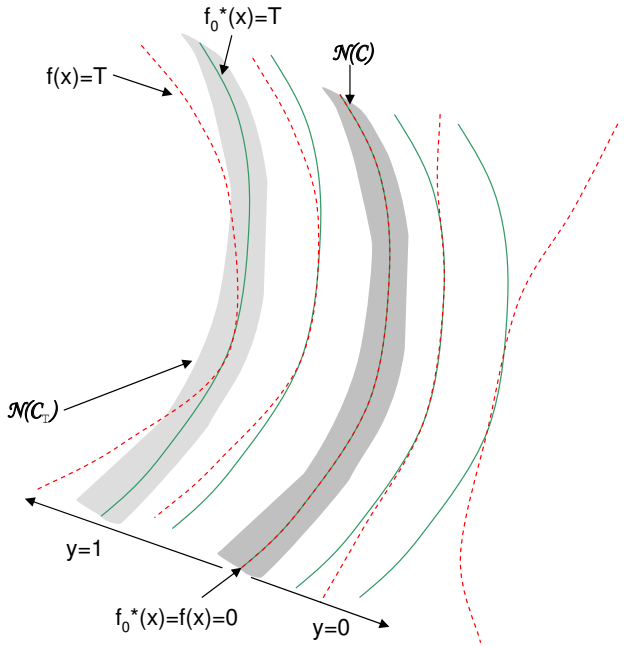


Figure 1: Example of a detection problem where boosting produces the optimal cost-insensitive detector but threshold manipulation does not lead to optimal cost-sensitive detectors. The figure presents level-sets of both the optimal predictor $f_0^*(\mathbf{x})$ (solid line) and the boosted predictor $f(\mathbf{x})$ (dashed line). As iterations progress boosting emphasizes the minimization inside $\mathcal{N}(\mathcal{C})$. In result, while the zero level-set is optimal, the same does not hold for other level-sets. This implies that the decision boundaries produced by threshold manipulation will be sub-optimal. Optimal cost-sensitive rules can, however, be obtained by emphasizing the optimization in other regions, e.g. $\mathcal{N}(\mathcal{C}_T)$.

accurate estimates of the posterior probabilities $P_{Y|\mathbf{X}}(y|\mathbf{x})$, sometimes referred to as calibrated probabilities, has been noted by various authors [26, 27, 30]. In [27] and [26], this is attributed to the fact that the empirical estimate of either the exponential or binomial losses is minimized by letting $y_i f(\mathbf{x}_i)$ grow to infinity for all training points. When the span of the space of weak learners is rich enough to separate the training set into the two classes, this is always possible and, if run for enough iterations, all boosting algorithms produce a distribution of posterior probabilities $P_{Y|\mathbf{X}}(y|\mathbf{x})$ which is highly concentrated in the neighborhoods of 0 and 1, independently of the true distribution. Note that this does not compromise cost-insensitive optimality, but rather reinforces it, since $f(\mathbf{x}_i)$ grows to ∞ for positive, and to $-\infty$ for negative examples. In summary, boosting does not produce calibrated probabilities and will, in fact, converge to a binary posterior distribution (of values 0 and 1) if run sufficiently long. Independent of the number of iterations, the probability estimates are usually not accurate enough to guarantee acceptable cost-sensitive performance by threshold manipulation.

3.5 Prior work on cost-sensitive boosting

This limitation is well known in the boosting literature, where a number of cost-sensitive boosting extensions have been proposed [7,37,38,42]. Since, for cost-sensitive learning, the main problem is that boosting's reweighing mechanism emphasizes $\mathcal{N}(\mathcal{C})$, instead of the optimal cost-sensitive boundary $\mathcal{N}(\mathcal{C}_T)$, it has long been noted that good cost-sensitive performance requires a modification of this mechanism. This is also supported by the intuition that, in cost-sensitive detection, examples from different classes should be weighted differently.

A naive implementation of this intuition would be to modify the initial boosting weights, so as to represent the asymmetry of the costs. However, because boosting re-updates all weights at each iteration, it quickly destroys the initial asymmetry, and the predictor obtained after convergence is usually not different from that produced with symmetric initial conditions. A second natural strategy is to somehow change the weight update equation. For example, one could make the updated weight equal to a mixture of the result of (26), (28), or (30), and the initial cost-sensitive weights. We refer to heuristics of this type as "weight manipulation". Previously proposed cost-sensitive boosting algorithms, such as AdaCost [7], CSB0, CSB1, CSB2 [38], Asymmetric-AdaBoost [42], AdaC1, AdaC2, or AdaC3 [37], fall in this class. For example, CSB2 [38] modifies the weight update rule of AdaBoost to

$$w_i^{(m+1)} = C_i \cdot w_i^{(m)} e^{-y_i g_m^{Ada}(\mathbf{x}_i)}, \tag{31}$$

relying on (24) for the computation of α_m.

While various justifications are available for the different proposals for direct manipulation of boosting equations, these manipulations are essentially heuristic, and provide no guarantees of convergence to a good cost-sensitive decision rule. Furthermore, none of the cost-sensitive extensions can be easily applied to algorithms other than Adaboost. We next introduce a framework for cost-sensitive boosting that addresses these two limitations.

4 Cost-sensitive boosting

The new framework is inspired by two observations. First, the unifying principle behind the different boosting algorithms is that they perform gradient descent [13, 25, 50] with respect to losses whose minimum is the optimal cost-insensitive predictor of (29). Second, their main limitation for cost-sensitive learning is the emphasis on the neighborhood of the cost-insensitive boundary $\mathcal{N}(\mathcal{C})$, as shown in Figure 1. We have already noted that these two properties are interconnected. While the limitation is due to the weight-update mechanism, simply modifying this mechanism (as discussed in the previous section) is usually not sufficient to achieve acceptable cost-sensitive performance. Instead, boosting involves a balance between weight updates and gradient steps which must be components of the minimization of the *common* loss. For cost-sensitive optimality, this balance requires that the loss function satisfies two conditions, which we denote as the necessary conditions for cost-sensitive optimality.

1. It is minimized by the optimal cost-sensitive predictor.
2. It leads to a weight-updating mechanism that emphasizes a neighborhood of the cost-sensitive boundary $\mathcal{N}(\mathcal{C}_T)$.

This suggests an alternative strategy to design cost-sensitive boosting algorithms: *to modify the loss functions so that these two conditions are met*. In what follows, we show how this can be accomplished for Ada, Real and LogitBoost. The framework could be used to derive cost-sensitive extensions of any algorithm that performs gradient descent on the space of combination of weak learners, e.g. GentleBoost [12] or AnyBoost [25]. We limit our attention to the algorithms above for reasons of brevity, and their popularity.

4.1 Cost-sensitive losses

We start by noting that the optimal cost-sensitive detector of (5) can be re-written as $h_T^* = sgn[\log f_T^*(\mathbf{x})]$ with

$$f_T^*(\mathbf{x}) = \frac{P_{Y|\mathbf{X}}(1|\mathbf{x})C_1}{P_{Y|\mathbf{X}}(-1|\mathbf{x})C_2}. \tag{32}$$

Noting that the zero level-set of this predictor is the cost-sensitive boundary $\mathcal{C}_T$, suggests that boosting-style of gradient descent on any loss function minimized, up to a scaling factor, by $f_T^*(\mathbf{x})$ should satisfy the two necessary conditions for cost-sensitive optimality. The following extensions of the expected exponential and binomial losses guarantee that the first is indeed met.

Lemma 2. *The losses*

$$E_{\mathbf{X},Y}\left[I(y=1)e^{-y.C_1 f(\mathbf{x})} + I(y=-1)e^{-y.C_2 f(\mathbf{x})}\right], \tag{33}$$

where $I(\cdot)$ is the indicator function of (23), and

$$-E_{\mathbf{X},Y}[y' \log(p_c(\mathbf{x})) + (1-y')\log(1-p_c(\mathbf{x}))] \tag{34}$$

Algorithm 1 Cost-sensitive AdaBoost

Input: Training set $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, where $y \in \{1, -1\}$ is the class label of example $\mathbf{x}$, costs C_1, C_2, set of weak learners $\{g_k(\mathbf{x})\}_{k=1}^K$, and number M of weak learners in the final decision rule.

Initialization: Select uniformly distributed weights

$$w_i = \frac{1}{2|\mathcal{I}_+|}, \forall i \in \mathcal{I}_+, \qquad w_i = \frac{1}{2|\mathcal{I}_-|}, \forall i \in \mathcal{I}_-.$$

for $m = \{1, \ldots, M\}$ **do**
  for $k = \{1, \ldots, K\}$ **do**
    train a weak learner/step-size pair $(g_k(\mathbf{x}); \alpha_k)$, by considering various thresholds for $g_k(\mathbf{x})$. For each threshold compute α with (41) and the the resulting loss with (40).
  end for
  select $(g_m(\mathbf{x}), \alpha_m)$ as the weak learner/step-size pair of smallest loss.
  update weights w_i according to (39).
end for

Output: decision rule $h(x) = sgn[\sum_{m=1}^M \alpha_m g_m(x)]$.

where

$$p_c(\mathbf{x}) = \frac{e^{\gamma f(\mathbf{x}) + \eta}}{e^{\gamma f(\mathbf{x}) + \eta} + e^{-\gamma f(\mathbf{x}) - \eta}}. \tag{35}$$

with

$$\gamma = \frac{C_1 + C_2}{2} \qquad \eta = \frac{1}{2} \log \frac{C_2}{C_1},$$

are minimized by the asymmetric logistic transform of $P_{Y|\mathbf{X}}(1|\mathbf{x})$,

$$f(x) = \frac{1}{C_1 + C_2} \log \frac{P(y = 1|\mathbf{x})C_1}{P(y = y''|\mathbf{x})C_2}, \tag{36}$$

where $y'' = -1$ *for (33) and* $y'' = 0$ *for (34).*

Proof. See appendix A ∎

We next derive cost-sensitive extensions of the boosting algorithms, by performing gradient descent on these losses, and will later show that these extensions shift the emphasis of the boosting weights from $\mathcal{N}(\mathcal{C})$ to $\mathcal{N}(\mathcal{C}_T)$.

4.2 Cost-sensitive AdaBoost

We start by extending Adaboost.

Theorem 3. (Cost-sensitive AdaBoost) *Consider the minimization of the empirical estimate of the asymmetric loss of (33), based on a training sample* $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, *by*

gradient descent on the space, $\mathcal{S}$, of functions of the form of (17) and (21), and define two sets

$$\mathcal{I}_+ = \{i|y_i = 1\} \qquad \mathcal{I}_- = \{i|y_i = -1\}. \tag{37}$$

The gradient direction and optimal step, at iteration m, are the solution of

$$\begin{aligned}(\alpha_m, g_m) \quad &= \arg\min_{\alpha,g} \quad \sum_{i\in\mathcal{I}_+} w_i^{(m)} \exp(-C_1\alpha g(\mathbf{x}_i)) \\ &\qquad + \sum_{i\in\mathcal{I}_-} w_i^{(m)} \exp(C_2\alpha g(\mathbf{x}_i))\end{aligned} \tag{38}$$

with

$$w_i^{(m+1)} = \begin{cases} w_i^{(m)} e^{-C_1\alpha_m g_m(\mathbf{x}_i)}, & i \in \mathcal{I}_+ \\ w_i^{(m)} e^{C_2\alpha_m g_m(\mathbf{x}_i)}, & i \in \mathcal{I}_-. \end{cases} \tag{39}$$

Given the step size α, the gradient direction is

$$\begin{aligned}g_m = \quad \arg\min_{g} \quad &\Big[(e^{C_1\alpha} - e^{-C_1\alpha}) \cdot b + e^{-C_1\alpha}\mathcal{T}_+ \\ &+ (e^{C_2\alpha} - e^{-C_2\alpha}) \cdot d + e^{-C_2\alpha}\mathcal{T}_-\Big]\end{aligned} \tag{40}$$

and the optimal step size is the solution of

$$\begin{aligned}2C_1 \cdot b \cdot \cosh(C_1\alpha) + 2C_2 \cdot d \cdot \cosh(C_2\alpha) = \\ C_1 \cdot \mathcal{T}_+ \cdot e^{-C_1\alpha} + C_2 \cdot \mathcal{T}_- \cdot e^{-C_2\alpha}\end{aligned} \tag{41}$$

with

$$\mathcal{T}_+ = \sum_{i\in\mathcal{I}_+} w_i^{(m)} \tag{42}$$

$$\mathcal{T}_- = \sum_{i\in\mathcal{I}_-} w_i^{(m)} \tag{43}$$

$$b = \sum_{i\in\mathcal{I}_+} w_i^{(m)}[1 - I(y_i = g(\mathbf{x}_i))] \tag{44}$$

$$d = \sum_{i\in\mathcal{I}_-} w_i^{(m)}[1 - I(y_i = g(\mathbf{x}_i))] \tag{45}$$

Proof. See appendix B ∎

The gradient descent iteration cycles through the weak learners, for each, solving (41). This can be done efficiently with standard scalar search procedures. In the experiments reported in this paper, the optimal α was found in an average of 6 iterations of bisection search. Given α, the loss associated with the weak learner can be computed, and the optimal learner selected with (40). A summary of the cost-sensitive boosting algorithm is presented in Algorithm 1. It is worth mentioning that the algorithm is fully compatible with Adaboost, in the sense that it reduces to the latter when $C_1 = C_2 = 1$.

4.3 Cost-sensitive RealBoost

We next consider the cost-sensitive extension of RealBoost.

Theorem 4. (Cost-sensitive RealBoost) *Consider the minimization of the asymmetric loss of (33), based on a training sample $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, by gradient descent on the space, $\mathcal{S}^r$, of predictors of the form of (17) where the weak learners $G_m(\mathbf{x})$ are real functions. Given a dictionary of features $\{\phi_1(\mathbf{x}), \ldots, \phi_K(\mathbf{x})\}$, the gradient at iteration m has the form*

$$G_m^{real}(\mathbf{x}) = G_{\phi_{k^*}}(\mathbf{x}) \tag{46}$$

where the optimal feature is determined by

$$k^* = \arg\min_k \sum_{i \in \mathcal{I}_+} w_i^{(m)} \exp(-C_1 G_{\phi_k}(\mathbf{x}_i)) + \sum_{i \in \mathcal{I}_-} w_i^{(m)} \exp(C_2 G_{\phi_k}(\mathbf{x}_i))] \tag{47}$$

with weights given by

$$w_i^{(m+1)} = \begin{cases} w_i^{(m)} e^{-C_1 G_m^{real}(\mathbf{x}_i)}, & i \in \mathcal{I}_+ \\ w_i^{(m)} e^{C_2 G_m^{real}(\mathbf{x}_i)}, & i \in \mathcal{I}_-, \end{cases} \tag{48}$$

and where

$$G_\phi(\mathbf{x}) = \left\{ \frac{1}{C_1 + C_2} \log \frac{P_{Y|\mathbf{X}}^{(w)}(1|\phi(\mathbf{x}))C_1}{P_{Y|\mathbf{X}}^{(w)}(-1|\phi(\mathbf{x}))C_2} \right\}. \tag{49}$$

$P_{Y|\mathbf{X}}^{(w)}(y|\phi(\mathbf{x})), y \in \{1, -1\}$ are estimates of the posterior probabilities for the two classes, after the application of the feature transformation $\phi(\mathbf{x})$ to a sample re-weighted according to the weights $w_i^{(m)}$.

Proof. See appendix C ∎

The posterior probabilities $P_{Y|\mathbf{X}}^{(w)}(y|\phi_m(\mathbf{x})), y \in \{1, -1\}$ of (49) can be estimated with standard techniques [5]. For example, if the $\phi_k(\mathbf{x})$ are scalar features, they can be obtained with weighted histograms of feature responses. Standard histogram regularization procedures should be used to avoid empty histogram bins. A summary of the cost-sensitive RealBoost algorithm is presented in Algorithm 2. The algorithm is fully compatible with RealBoost, in the sense that it reduces to the latter when $C_1 = C_2 = 1$.

Algorithm 2 Cost-sensitive RealBoost

Input: Training set $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, where $y \in \{1, -1\}$ is the class label of example $\mathbf{x}$, costs C_1, C_2, and number M of weak learners in the final decision rule.

Initialization: Select uniformly distributed weights

$$w_i = \frac{1}{2|\mathcal{I}_+|}, \forall i \in \mathcal{I}_+, \qquad\qquad w_i = \frac{1}{2|\mathcal{I}_-|}, \forall i \in \mathcal{I}_-.$$

for $m = \{1, \ldots, M\}$ **do**
  for $k = \{1, \ldots, K\}$ **do**
    compute the gradient step $G_{\phi_k}(\mathbf{x})$ with (49).
  end for
  select the optimal direction according to (47) and set the weak learner $G_m^{real}(\mathbf{x})$ according to (46).
  update weights w_i according to (48).
end for
Output: decision rule $h(\mathbf{x}) = sgn[\sum_{m=1}^{M} G_m^{real}(\mathbf{x})]$.

4.4 Cost-sensitive LogitBoost

Finally, we consider LogitBoost.

Theorem 5. (Cost-sensitive LogitBoost) *Consider the minimization of the expected binomial loss of (34), based on a training sample $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, on the space $\mathcal{S}^r$ of predictors of the form of (17) where the weak learners $G_m(\mathbf{x})$ are real functions. Given a dictionary of features $\{\phi_1(\mathbf{x}), \ldots, \phi_K(\mathbf{x})\}$, and a predictor $f^{(m)}(\mathbf{x})$, the Newton step at iteration m has the form*

$$G_m^{logit}(\mathbf{x}) = \frac{1}{2\gamma} G_{\phi_{k^*}}(\mathbf{x}) \tag{52}$$

where $G_\phi(\mathbf{x}) = a_\phi \phi(\mathbf{x}) + b_\phi$ is the result of the weighted regression

$$(a_\phi, b_\phi) = \arg\min_{a_\phi, b_\phi} \sum_i w_i^{(m)} (z_i - a_\phi \phi(\mathbf{x}_i) - b_\phi)^2 \tag{53}$$

with

$$z_i = \frac{y_i' - p_c^{(m)}(\mathbf{x}_i)}{p_c^{(m)}(\mathbf{x}_i)(1 - p_c^{(m)}(\mathbf{x}_i))} \tag{54}$$

$$w_i^{(m)} = p^{(m)}(\mathbf{x}_i)(1 - p^{(m)}(\mathbf{x}_i)), \tag{55}$$

where $p_c^{(m)}(\mathbf{x})$ is the link function of (35), and $p^{(m)}(\mathbf{x})$ that of (20), with $f(\mathbf{x}) = f^{(m)}(\mathbf{x})$. The optimal feature is determined by

$$k^* = \arg\min_k \sum_i w_i^{(m)} (z_i - a_{\phi_k} \phi_k(\mathbf{x}_i) - b_{\phi_k})^2. \tag{56}$$

Algorithm 3 Cost-sensitive LogitBoost

Input: Training set $\mathcal{D} = \{(\mathbf{x}_1, y'_1), \ldots, (\mathbf{x}_n, y'_n)\}$, where $y' \in \{0, 1\}$ is the class label of example $\mathbf{x}$, costs C_1, C_2, $\gamma = \frac{C_1+C_2}{2}$, $\eta = \frac{1}{2}\log\frac{C_2}{C_1}$, $\mathcal{I}_+$ the set of examples with label 1, $\mathcal{I}_-$ the set of examples with label 0, and number M of weak learners in the final decision rule.

Initialization: Set uniformly distributed probabilities $p_c^{(1)}(\mathbf{x}_i) = p^{(1)}(\mathbf{x}_i) = \frac{1}{2} \;\; \forall \mathbf{x}_i$ and $f^{(1)}(\mathbf{x}) = 0$.

for $m = \{1, \ldots, M\}$ **do**

compute the working responses $z_i^{(m)}$ as in (54) and weights $w_i^{(m)}$ as in (55).

for $k = \{1, \ldots, K\}$ **do**

compute the solution to the least squares problem of (53),

$$a_{\phi_k} = \frac{\langle 1 \rangle_w \cdot \langle \phi_k(\mathbf{x}_i) z_i \rangle_w - \langle \phi_k(\mathbf{x}_i) \rangle_w \cdot \langle z_i \rangle_w}{\langle 1 \rangle_w \cdot \langle \phi_k^2(\mathbf{x}_i) \rangle_w - \langle \phi_k(\mathbf{x}_i) \rangle_w^2} \tag{50}$$

$$b_{\phi_k} = \frac{\langle \phi_k(\mathbf{x}_i)^2 \rangle_w \cdot \langle z_i \rangle_w - \langle \phi_k(\mathbf{x}_i) \rangle_w \cdot \langle \phi_k(\mathbf{x}_i) z_i \rangle_w}{\langle 1 \rangle_w \cdot \langle \phi_k^2(\mathbf{x}_i) \rangle_w - \langle \phi_k(\mathbf{x}_i) \rangle_w^2} \tag{51}$$

where we have defined

$$\langle q(\mathbf{x}_i) \rangle_w \doteq \sum_i w_i^{(m)} q(\mathbf{x}_i).$$

end for

select the optimal direction according to (56) and set the weak learner $G_m^{logit}(\mathbf{x})$ according to (52).

set $f^{(m+1)}(\mathbf{x}) = f^{(m)}(\mathbf{x}) + G_m^{logit}(\mathbf{x})$.

end for

Output: decision rule $h(\mathbf{x}) = sgn[\sum_{m=1}^{M} G_m^{logit}(\mathbf{x})]$.

Proof. See appendix D ∎

A summary of the cost-sensitive LogitBoost algorithm is presented in Algorithm 3. It is instructive to compare this to a procedure commonly used to calibrate the probabilities produced by large-margin classifiers, known as Platt calibration [22, 30, 32, 47]. This procedure attempts to map the prediction $f(\mathbf{x}) \in [-\infty, +\infty]$ to a posterior probability $p(\mathbf{x}) \in [0, 1]$, using the link function of (35). The γ and η parameters are determined by gradient descent with respect to the binomial loss of (34), also used in cost-sensitive LogitBoost. The difference is that, in Platt's method, cost-insensitive boosting is first used to learn the predictor $f(\mathbf{x})$ and maximum likelihood is then used to determine the parameters γ and η that best fit a cross-validation data set. On the other hand, cost-sensitive LogitBoost uses the calibrated link function throughout all boosting iterations. Note that, besides requiring an additional validation set, Platt's method does not solve the problem of Figure 1, since the emphasis of the boosting

component remains on $\mathcal{N}(\mathcal{C})$, not on $\mathcal{N}(\mathcal{C}_T)$. We next show that the cost-sensitive boosting algorithms introduced above do provide a solution to this problem.

4.5 Cost-sensitive margins

We have seen, in Section 4.1, that cost-sensitive boosting algorithms should satisfy two conditions:

- convergence to the optimal predictor of (32),
- emphasis on a neighborhood of the cost-sensitive boundary $\mathcal{N}(\mathcal{C}_T)$.

The first condition is guaranteed by the use of the losses of (33) and (34). To investigate the second we consider the weighting mechanisms of the three algorithms.

For both cost-sensitive Ada and RealBoost, a simple recursion shows that, after M iterations,

$$\frac{w_i^{(M)}}{w_i^{(0)}} = e^{-y_i Q_i f(\mathbf{x}_i)},$$

where $Q_i = C_1$ if $i \in \mathcal{I}_+$ and $Q_i = C_2$ otherwise. Assuming that $f(\mathbf{x})$ converges to the optimum of (36), this ratio is one along $\mathcal{C}_T$, exponentially increasing (with the distance to this boundary) for $\mathbf{x}_i$ such that $f(\mathbf{x}_i)y_i < 0$, and exponentially decreasing for $\mathbf{x}_i$ such that $f(\mathbf{x}_i)y_i > 0$. Hence, with respect to the cost-insensitive AdaBoost algorithm, the only difference is whether the points are on the correct side of the cost sensitive boundary $\mathcal{C}_T$. With the exception of the points which lie on the incorrect side, all weight is concentrated on the neighborhood $\mathcal{N}(\mathcal{C}_T)$ of the cost-sensitive boundary. For LogitBoost, the weight $w_i^{(M)}$ is a symmetric function of $p^{(M)}(\mathbf{x}_i)$, with maximum at $p^{(M)}(\mathbf{x}_i) = 1/2$ or, from (20), at $f(\mathbf{x}_i) = 0$. As in the cost-insensitive case,

$$w_i^{(M)}(\mathbf{x}) = \left(e^{f(\mathbf{x}_i)} + e^{-f(\mathbf{x}_i)}\right)^{-2} \approx e^{-2|f(\mathbf{x}_i)|}$$

and the weight decays exponentially with the distance from the zero-level set of $f(\mathbf{x})$, independently of whether the points are correctly classified or not. The only difference is that, as $f(\mathbf{x})$ converges to (36), this zero-level set is the cost-sensitive boundary $\mathcal{C}_T$. This shows that all cost-sensitive boosting algorithms shift the margin emphasis from $\mathcal{N}(\mathcal{C})$ to $\mathcal{N}(\mathcal{C}_T)$.

5 Experimental evaluation

Two sets of experiments were designed to evaluate the cost-sensitive boosting algorithms. The first was based on a simple synthetic problem, for which the BDR is known, allowing explicit comparison to the optimal cost-sensitive detector. These experiments aimed for insight on various properties of the proposed algorithms. The second set was based on standard datasets, and targeted a comparison between the new algorithms and previously proposed methods.

5.1 Synthetic datasets

We start with a synthetic binary scalar classification problem, involving Gaussian classes of equal variance $\sigma^2 = 1$ and means $\mu_- = -1$ ($y = -1$) and $\mu_+ = 1$ ($y = 1$). $10K$ examples were sampled per class, simulating the scenario where the class probabilities are uniform.

5.1.1 Accuracy

To test the accuracy of the classifiers produced by cost-sensitive boosting we relied on the following observations. First, given a cost structure (C_1, C_2), the boosted detector is optimal if and only if the asymmetric logistic transform of (36) holds along the cost-sensitive boundary, i.e. if and only if $x^* = f^{-1}(0)$ where $f(x)$ is the optimal predictor of (36) and x^* the zero-crossing of the boosted predictor. Second, from (36), this is equivalent to

$$P_{Y|X}(1|x^*) = \frac{C_2}{C_1 + C_2}, \tag{57}$$

and it follows that, given cost structure and location x^*, it is possible to infer the true class posterior probabilities at the latter. This is equally valid for multivariate problems, in which case the location x^* becomes a level set. Hence, if the boosting algorithm produces truly optimal cost-sensitive detectors, the plot of $\frac{C_2}{C_1+C_2}$ as a function of x^* should be identical to the plot of the class posterior probability $P_{Y|X}(1|x^*)$. For the Gaussian problem considered, it is straightforward to show that

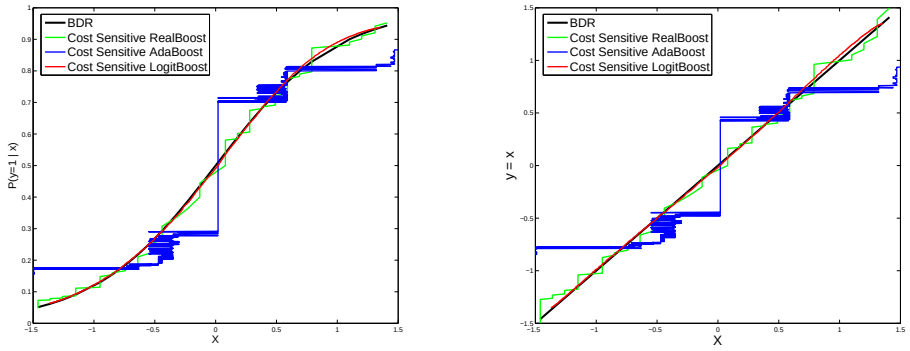
$$P_{Y|X}(1|x) = \frac{1}{1 + e^{-2x}}, \tag{58}$$

and (57) implies that $x^* = -T/2$, with T given by (7). It is therefore possible to evaluate the accuracy of the boosted cost-sensitive detectors, for the entire range of (C_1, C_2) by either measuring the similarity between the plots $(x^*, \frac{C_2}{C_1+C_2})$ and $(x^*, \frac{1}{1+e^{-2x^*}})$ or the plots $(x^*, -\frac{T}{2})$ and (x^*, x^*).

These comparisons are shown on Figure 2 (a) and (b) for the detectors produced by cost-sensitive Ada, Real, and LogitBoost. In all cases $C_2 = 1$ and C_1 was varied over a wide range of values. For each value of C_1, boosting was run for five iterations. It is clear that both Real and LogitBoost produce accurate cost-sensitive detectors. The difficulties of AdaBoost are due to the restriction of the predictor to a combination of binary functions.

5.1.2 Comparison to previous algorithms

We next considered two cases in greater detail, namely the problems with cost structures $C_2 = 1$ and $C_1 \in \{5, 20\}$, and compared the performance of the novel cost-sensitive boosting algorithms to those of the algorithms discussed in Section 3.5. For these cost structures, the perfect detector has $x^* = -.8047$ (when $C_1 = 5$) and $x^* = -1.4979$ (when $C_1 = 20$). The goal was to determine if the different algorithms could generate predictors with these zero-crossings, by manipulating their cost parameters (e.g. the parameter C_1 of cost-sensitive AdaBoost, which we denote by $\hat{C}_1$



(a) (b)

Figure 2: a) Posterior probability $P_{Y|X}(y = 1|x)$ used in the BDR, and estimates produced by cost-sensitive Ada, Logit and RealBoost. b) Comparison of the plots $(x^*, -\frac{T}{2})$ and (x^*, x^*).

to avoid confusions with the true value of the cost). Figure 3 compares how x^* evolved (as a function of boosting iteration) for cost-sensitive AdaBoost and the cost sensitive boosting algorithms previously available in the literature. For brevity, we limit the presentation to cost-sensitive AdaBoost since, as discussed above, this is the weakest of the new cost-sensitive boosting algorithms on this problem. In all cases, a (rather extensive) search over values of the cost parameters of each algorithm was performed, so as to guarantee the best possible performance after 50 iterations.

Despite the simplicity of the problem, this search did not produce a good solution for most of the algorithms. As illustrated by Figure 3, four classes of behavior were observed. Algorithms in the first class (AdaC1, AdaCost) never produced any solutions other than the cost-insensitive optimal $x^* = 0$. The second class consisted of algorithms (CSB0, AdaC2, AdaC3) that never converged to any meaningful solution. Algorithms in the third class (CSB1, CSB2) 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. Only cost-sensitive AdaBoost was able to consistently converge to a good solution in the allotted number of iterations. In particular, the latter produced $x^* = -1.4993$ when $C_1 = 20$ in two iterations, and $x^* = -0.7352$ when $C_1 = 5$ in four iterations. The value of the $\hat{C}_1$ estimate that led to the best solution was, however, not always the true C_1. While when $C_1 = 5$ cost-sensitive AdaBoost was nearly optimal with $\hat{C}_1 = 4.5$, near optimal performance in the case where $C_1 = 20$ required a cost estimate of $\hat{C}_1 = 4.7$. This mismatch is compliant with Figure 2, which shows some inability of cost-sensitive AdaBoost to replicate the posterior class probabilities required for optimal performance with highly unbalanced cost structures (x^* of very large magnitude). This was not observed for cost-sensitive Logit or RealBoost. These results show that 1) the new algorithms are far superior than those previously available, and 2) the optimal solution can be found for most cost-sensitive

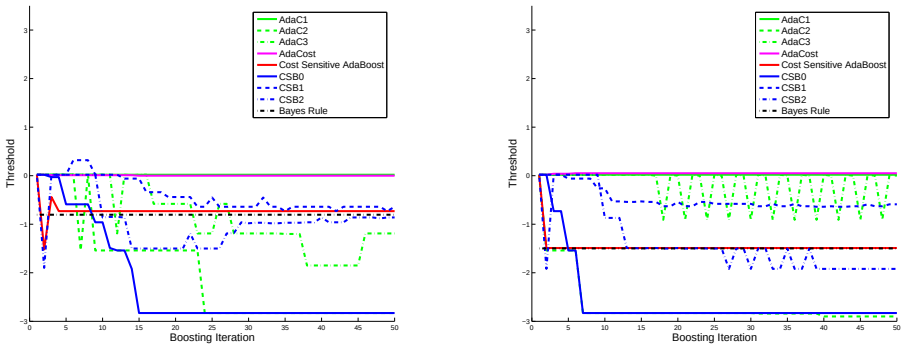


Figure 3: Decision boundaries produced by the different boosting algorithms for various cost factors. Left: $C_1 = 5$, right: $C_1 = 20$.

problems, but may require (in particular for cost-sensitive AdaBoost) cross-validation of cost-parameters.

The most plausible explanation for the poor performance of all other algorithms appears to be the inappropriate choice of the α parameter: while the weight update rules seemed to produce cost-sensitive weak learners, the incorrect choice of α frequently gave disproportionate weight to weak learners with poor decision boundaries. For example, in the case of AdaC1, the first two weak learners had threshold of 0.0152 and -0.9186 but the corresponding values of α were 0.9056 and 0.2404. Although the second threshold is close to optimal ($x^* = -0.8047$), the poor choice of α gave it little weight, much smaller than that of the first. This made the overall decision boundary close to zero. Of all algorithms tested, only CSB1 and CSB2 achieved performance comparable to that of cost-sensitive AdaBoost, even though their slow convergence in this simple problem appears problematic.

5.2 Real datasets

Two sets of experiments were performed with real data. The first involved a number of datasets from the UCI repository, while the second addressed the computer vision problem of face detection. To simplify the comparison of results, the quality of cost-sensitive classification is frequently measured by a scalar metric that weighs errors of one type more than others. A common metric [38], which we have adopted, is

$$\epsilon = p_{false} + f_{cost} \times m_{miss} \tag{59}$$

where p_{false} is the number of false-positives of the detector, m_{miss} the number of misses and $f_{cost} > 1$ a cost factor that weighs misses more heavily than false positives. A number of cost factors were considered, and ϵ computed for each combination of 1) cost sensitive boosting method, 2) training cost structure estimates, and 3) true cost factor f_{cost} used to evaluate performance. By training cost structure estimates we refer

Table 1: Minimum ϵ and corresponding $\hat{C}_1$ for cost-sensitive RealBoost on Cleveland heart.

	$f_{cost} = 2$	$f_{cost} = 5$	$f_{cost} = 10$	Average
ϵ	11.6	14	15	**13.53**
$\hat{C}_1$	1.6	4.2	5.1	

to the parameters used during training, e.g. the parameters $\hat{C}_1$ and $\hat{C}_2$ of cost-sensitive boosting.

5.2.1 UCI datasets

Ten binary UCI [28] data sets were used: Pima-diabetes, breast cancer diagnostic, breast cancer prognostic, original Wisconsin breast cancer, liver disorder, sonar, echocardiogram, Cleveland heart disease, tic-tac-toe and Haberman's survival. In all cases, five fold validation was used to find the best cost estimate by searching over $\hat{C}_1 \in [1, 10]$ ($\hat{C}_2 = 1$). Three cost factors $f_{cost} \in \{2, 5, 10\}$ were considered, and the minimum ϵ was found for each. Table 1 gives an example (cost-sensitive RealBoost and the Cleveland heart disease dataset) of the relationship between the minimum ϵ, f_{cost}, and the best value of the training cost parameter $\hat{C}_1$. Performance across the various values of f was further summarized by computing the average value of the minimum ϵ achieved by each algorithm.

This average is shown in Table 2 for each of the algorithms considered. The table also shows the median value of the average across all datasets, and the number of times that each of the proposed cost-sensitive boosting algorithms outperformed *all* of the previously available methods (# of wins). All new algorithms have a median loss smaller than those in the literature, and outperform all of them in 60 to 90% of the datasets. Overall, cost-sensitive RealBoost has the lowest median loss, and is the top performer in $7/10$ datasets. Cost-sensitive LogitBoost achieves the best performance in the remaining three. This is strong evidence for the superiority of the new cost-sensitive boosting algorithms over those previously available.

5.2.2 Face detection

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 detector cascades [43]. 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 [43] to compare the new cost-sensitive boosting to those previously available. Given the computational complexity of these experiments we, once again, restricted the comparison to (the worst-case performer) cost-sensitive AdaBoost. 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, as described in [43]. 6000 examples were used, per class, for training, the remaining 3832

Table 2: Average minimum ϵ for the UCI datasets considered.

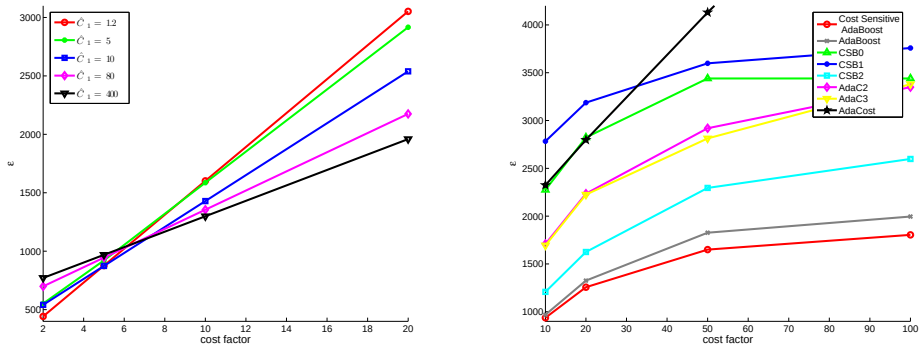
	pima	liver	wdbc	sonar	wpbc	Wisc	echo	heart
CS-Ada	60.2	30.73	6.66	9.73	21.6	4.33	7	**13.53**
CS-Log	59.26	31.73	**5.26**	13.2	19.33	**3.6**	**6.53**	13.93
CS-Real	**56.66**	**30.4**	6.53	**9**	**18.46**	5.6	8.53	**13.53**
CSB0	65.6	33.8	13.33	16	24.13	12.4	10	21.4
CSB1	85.93	33.53	8.53	10.86	21	46.2	12.33	25.06
CSB2	65.93	31.53	8.13	9.46	20.2	20.33	9.93	18.4
AdaC2	73.33	33.26	7.4	10.6	18.6	9.33	10	20.73
AdaC3	70.33	33.4	6.8	9.86	20.73	5.53	9.73	19.53
ADaCost	263.33	135.06	13.2	69.53	39.73	9.46	21.4	113.86

	tic	survival	median	# wins
CS-Ada	94.93	31.2	17.57	7
CS-Log	107.8	32.53	16.63	6
CS-Real	**31.6**	**29.93**	**16.0**	**9**
CSB0	104.66	36.06	22.77	
CSB1	330.73	38.46	29.3	
CSB2	101.6	33.26	20.27	
AdaC2	42.06	35.6	19.67	
AdaC3	86.33	34	20.13	
ADaCost	330.8	72.86	71.2	

being left for testing, and all boosting algorithms were ran for 100 iterations. Four cost factors ($f_{cost} \in \{10, 20, 50, 100\}$) and a number of training cost structures were considered. This is illustrated in Figure 4 a) for cost-sensitive AdaBoost. The figure presents plots of ϵ as a function of f_{cost} for various training cost structures with $\hat{C}_2 = 1$ and $\hat{C}_1 \in [1.2, 1000]$. Note that detectors trained with larger $\hat{C}_1$ perform better when f_{cost} is larger, while smaller $\hat{C}_1$ lead to best performance when ϵ weighs the two errors more equally.

Figure 4 b) presents a comparison of the best performances achieved with cost-sensitive AdaBoost and each of the previously available cost-sensitive boosting methods. The plots were produced by considering four values of f_{cost} and searching for the cost structure and threshold that achieved the minimum ϵ for each of these values. The search across cost-structures produces the cost-sensitive detector with classification boundary $\mathbf{x}^* = f^{-1}(0)$ closest to the optimal boundary for the particular value of f_{cost} under consideration, and threshold manipulation then enables slight adjustments of this boundary. The inclusion of threshold manipulation also permits a fair comparison to the combination of a cost-insensitive detector (learned with the standard AdaBoost algorithm) and threshold manipulation. In fact, because standard AdaBoost is identical to cost-sensitive AdaBoost with $\hat{C}_1 = 1$, this is equivalent to disabling the search over training cost structures.

It is clear that cost-sensitive AdaBoost consistently outperforms all other methods,



(a) (b)

Figure 4: (a) Misclassification cost for cost-sensitive boosting under different training cost structures. (b) Minimum misclassification cost of various cost-sensitive boosting methods on a face detection problem.

for all values of f_{cost}. These results illustrate the importance of choosing the confidence α optimally, at each iteration. Methods that do not use α in the weight update rule (CSB0 and CSB1) have extremely poor performance. Methods that update α but are not provably optimal (AdaC2, AdaC3, and AdaCost) perform worse than standard AdaBoost (or CSB2, which relies on the same α updates). Finally, the combination of standard AdaBoost and threshold manipulation is not sufficient to match the performance of the optimal cost-sensitive version of AdaBoost, except when the costs of the two types of errors are approximately equal (small f_{cost}).

6 Conclusion

We have presented a novel framework for the design of cost-sensitive boosting algorithms. The framework is based on the statistical 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 cost-sensitive boosting losses which, similarly to the original AdaBoost algorithm, can then be minimized by gradient descent in the functional space of convex combinations of weak learners. The general requirements for optimal cost-sensitive classification were identified, laying the groundwork for the cost sensitive extension of many large margin classification algorithms. Specifically, the cost-sensitive extensions of AdaBoost, RealBoost and LogitBoost were derived and shown to satisfy these requirements.

Experimental evidence, derived from a synthetic problem, standard data sets and the (timely) problem of face detection, was presented in support of the cost-sensitive optimality of the new algorithms. 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). Cost-sensitive boosting was shown to consistently outperform all other methods, achieving the smallest misclassification cost at all cost factors considered.

A Proof of Lemma 2

To find the minimum of the cost-sensitive extension of the exponential loss of (33) it suffices to search for the the function $f(\mathbf{x})$ of minimum expected loss conditioned on $\mathbf{x}$

$$\begin{aligned} l_e(\mathbf{x}) &= E_{Y|\mathbf{X}}\left[I(y=1)e^{-y.C_1 f(\mathbf{x})} + I(y=-1)e^{-y.C_2 f(\mathbf{x})}|\mathbf{x}\right] \\ &= P_{Y|\mathbf{X}}(1|\mathbf{x})e^{-C_1 f(\mathbf{x})} + P_{Y|\mathbf{X}}(-1|\mathbf{x})e^{C_2 f(\mathbf{x})}. \end{aligned}$$

Setting derivatives to zero

$$\frac{\partial l_e(\mathbf{x})}{\partial f(\mathbf{x})} = -C_1 P_{Y|\mathbf{X}}(1|\mathbf{x})e^{-C_1 f(\mathbf{x})} + C_2 P_{Y|\mathbf{X}}(-1|\mathbf{x})e^{C_2 f(\mathbf{x})} = 0 \quad (60)$$

it follows that

$$\frac{C_1 P_{Y|\mathbf{X}}(1|\mathbf{x})}{C_2 P_{Y|\mathbf{X}}(-1|\mathbf{x})} = e^{(C_1+C_2)f(\mathbf{x})} \quad (61)$$

and

$$f(\mathbf{x}) = \frac{1}{C_1+C_2}\log\frac{P_{Y|\mathbf{X}}(1|\mathbf{x})C_1}{P_{Y|\mathbf{X}}(-1|\mathbf{x})C_2}. \quad (62)$$

It is straightforward to show that the second derivative is non-negative, from which the loss is minimized by $f(\mathbf{x})$.

To find the minimum of the cost sensitive extension of the binomial loss of (34) it suffices to search for the the function $f(\mathbf{x})$ of minimum expected loss conditioned on $\mathbf{x}$

$$\begin{aligned} l_b(\mathbf{x}) &= -E_{Y|\mathbf{X}}[y'\log(p_c(\mathbf{x})) + (1-y')\log(1-p_c(\mathbf{x}))|\mathbf{x}] \\ &= -P_{Y|\mathbf{X}}(1|\mathbf{x})\log(p_c(\mathbf{x})) - P_{Y|\mathbf{X}}(0|\mathbf{x})\log(1-p_c(\mathbf{x})) \end{aligned}$$

with $p_c(\mathbf{x})$ given by (35). For this, we first compute the minimum with respect to $p_c(\mathbf{x})$, which is given by

$$\frac{\partial l_b(\mathbf{x})}{\partial p_c(\mathbf{x})} = -P_{Y|\mathbf{X}}(1|\mathbf{x})\frac{1}{p_c(\mathbf{x})} + P_{Y|\mathbf{X}}(0|\mathbf{x})\frac{1}{1-p_c(\mathbf{x})} = 0 \quad (63)$$

or

$$\log\frac{p_c(\mathbf{x})}{1-p_c(\mathbf{x})} = \log\frac{P_{Y|\mathbf{X}}(1|\mathbf{x})}{P_{Y|\mathbf{X}}(0|\mathbf{x})}.$$

Using (35), this is equivalent to

$$2(\gamma f(\mathbf{x}) + \eta) = \log \frac{P_{Y|\mathbf{X}}(1|\mathbf{x})}{P_{Y|\mathbf{X}}(0|\mathbf{x})},$$

or

$$f(\mathbf{x}) = \frac{1}{C_1 + C_2} \log \frac{P_{Y|\mathbf{X}}(1|\mathbf{x})C_1}{P_{Y|\mathbf{X}}(0|\mathbf{x})C_2}.$$

Since $\frac{\partial^2 l_b(\mathbf{x})}{\partial p_c(\mathbf{x})^2} \geq 0$ and $p_c(\mathbf{x})$ is monotonically increasing on $f(\mathbf{x})$ this is a minimum.

B Proof of Theorem 3

From (33) the cost function can be written as

$$J[f] = \quad E_{\mathbf{X},Y}[I(y=1)\exp(-C_1 f(\mathbf{x})) + I(y=-1)\exp(C_2 f(\mathbf{x}))]$$

and the addition of the weak learner $G(\mathbf{x}) = \alpha g(\mathbf{x})$ to the predictor $f(\mathbf{x})$ results in

$$\begin{aligned} J[f + \alpha g] \quad = \quad & E_{\mathbf{X},Y}[I(y=1)w(\mathbf{x},1)\exp(-C_1\alpha g(\mathbf{x})) \\ & +I(y=-1)w(\mathbf{x},-1)\exp(C_2\alpha g(\mathbf{x}))] \end{aligned}$$

with

$$w(\mathbf{x},1) = \exp(-C_1 f(\mathbf{x}))$$

and

$$w(\mathbf{x},-1) = \exp(C_2 f(\mathbf{x})).$$

Since $J[f + \alpha g]$ is minimized if and only if the argument of the expectation is minimized for all $\mathbf{x}$, the gradient direction and optimal step size are the solution of

$$\begin{aligned} (\alpha_m, g_m(\mathbf{x})) = \quad \arg\min_{\alpha, g(\mathbf{x})} \quad & E_{Y|\mathbf{X}}\Big[I(y=1)w(\mathbf{x},1)e^{-C_1\alpha g(\mathbf{x})} \\ & +I(y=-1)w(\mathbf{x},-1)e^{C_2\alpha g(\mathbf{x})}|\mathbf{x}\Big]. \end{aligned}$$

Noting that

$$E_{Y|\mathbf{X}}\left[I(y=1)w(\mathbf{x},1)e^{-C_1\alpha g(\mathbf{x})}+I(y=-1)w(\mathbf{x},-1)e^{C_2\alpha g(\mathbf{x})}|\mathbf{x}\right]=$$

$$\begin{aligned}
&= E_{Y|\mathbf{X}}\left[I(y=1)I(g(\mathbf{x})=1)w(\mathbf{x},1)e^{-C_1\alpha}+\right.\\
&\qquad I(y=1)I(g(\mathbf{x})=-1)w(\mathbf{x},1)e^{C_1\alpha}+\\
&\qquad I(y=-1)I(g(\mathbf{x})=1)w(\mathbf{x},-1)e^{C_2\alpha}+\\
&\qquad \left.I(y=-1)I(g(\mathbf{x})=-1)w(\mathbf{x},-1)e^{-C_2\alpha}|\mathbf{x}\right]\\
&= E_{Y|\mathbf{X}}\left[I(y=1)I(g(\mathbf{x})=-1)w(\mathbf{x},1)(e^{C_1\alpha}-e^{-C_1\alpha})+\right.\\
&\qquad I(y=1)w(\mathbf{x},1)e^{-C_1\alpha}+\\
&\qquad I(y=-1)I(g(\mathbf{x})=1)w(\mathbf{x},-1)(e^{C_2\alpha}-e^{-C_2\alpha})+\\
&\qquad \left.I(y=-1)w(\mathbf{x},-1)e^{-C_2\alpha}|\mathbf{x}\right]\\
&= P_{Y|\mathbf{X}}(1|\mathbf{x})w(\mathbf{x},1)I(g(\mathbf{x})=-1)(e^{C_1\alpha}-e^{-C_1\alpha})+\\
&\quad P_{Y|\mathbf{X}}(1|\mathbf{x})w(\mathbf{x},1)e^{-C_1\alpha}+\\
&\quad P_{Y|\mathbf{X}}(-1|\mathbf{x})w(\mathbf{x},-1)I(g(\mathbf{x})=1)(e^{C_2\alpha}-e^{-C_2\alpha})+\\
&\quad P_{Y|\mathbf{X}}(-1|\mathbf{x})w(\mathbf{x},-1)e^{-C_2\alpha}
\end{aligned}$$

it follows that

$$\begin{aligned}
(\alpha_m, g_m(\mathbf{x})) &= \arg\min_{\alpha,g(\mathbf{x})}\left\{P^{(w)}_{Y|\mathbf{X}}(1|\mathbf{x})I(g(\mathbf{x})=-1)(e^{C_1\alpha}-e^{-C_1\alpha})+\right.\\
&\qquad P^{(w)}_{Y|\mathbf{X}}(1|\mathbf{x})e^{-C_1\alpha}+\\
&\qquad P^{(w)}_{Y|\mathbf{X}}(-1|\mathbf{x})I(g(\mathbf{x})=1)(e^{C_2\alpha}-e^{-C_2\alpha})+\\
&\qquad \left.P^{(w)}_{Y|\mathbf{X}}(-1|\mathbf{x})e^{-C_2\alpha}\right\}
\end{aligned}$$

where

$$P^{(w)}_{Y|\mathbf{X}}(y|\mathbf{x})=\frac{P_{Y|\mathbf{X}}(y|\mathbf{x})w(\mathbf{x},y)}{\sum_{y\in\{1,-1\}}P_{Y|\mathbf{X}}(y|\mathbf{x})w(\mathbf{x},y)}$$

are the posterior estimates associated with a sample reweighed according to $w(\mathbf{x}, y)$. Hence, the weak learner of minimum cost is

$$\begin{aligned}(\alpha_m, g_m) &= \arg\min_{\alpha,g} E_{\mathbf{X}} \Big\{ P^{(w)}_{Y|\mathbf{X}}(1|\mathbf{x}) I(g(\mathbf{x}) = -1)(e^{C_1\alpha} - e^{-C_1\alpha}) + \\ & \qquad P^{(w)}_{Y|\mathbf{X}}(1|\mathbf{x}) e^{-C_1\alpha} + \\ & \qquad P^{(w)}_{Y|\mathbf{X}}(-1|\mathbf{x}) I(g(\mathbf{x}) = 1)(e^{C_2\alpha} - e^{-C_2\alpha}) + \\ & \qquad P^{(w)}_{Y|\mathbf{X}}(-1|\mathbf{x}) e^{-C_2\alpha} \Big\}\end{aligned}$$

and, replacing expectations by sample averages,

$$\begin{aligned}(\alpha_m, g_m) &= \arg\min_{\alpha,g} \left[(e^{C_1\alpha} - e^{-C_1\alpha}) \cdot b + e^{-C_1\alpha} \cdot \mathcal{T}_+ + \right. \\ & \qquad \left. (e^{C_2\alpha} - e^{-C_2\alpha}) \cdot d + e^{-C_2\alpha} \cdot \mathcal{T}_- \right],\end{aligned}$$

with the empirical estimates $\mathcal{T}_+$, $\mathcal{T}_-$, b and d of (42) - (45). Given $g(\mathbf{x})$, and setting the derivative with respect to α to zero

$$\begin{aligned}\frac{\partial}{\partial\alpha} &= C_1(e^{C_1\alpha} + e^{-C_1\alpha}) \cdot b - C_1 e^{-C_1\alpha} \cdot \mathcal{T}_+ + \\ & \qquad C_2(e^{C_2\alpha} + e^{-C_2\alpha}) \cdot d - C_2 e^{-C_2\alpha} \cdot \mathcal{T}_- = 0\end{aligned}$$

the optimal step size α is the solution of

$$2C_1 \cdot b \cdot \cosh(C_1\alpha) + 2C_2 \cdot d \cdot \cosh(C_2\alpha) = C_1 \cdot \mathcal{T}_+ \cdot e^{-C_1\alpha} + C_2 \cdot \mathcal{T}_- \cdot e^{-C_2\alpha}.$$

C Proof of Theorem 4

From (33) the cost function can be written as

$$J[f] = E_{\mathbf{X},Y}[I(y = 1) \exp(-C_1 f(\mathbf{x})) + I(y = -1) \exp(C_2 f(\mathbf{x}))]$$

and the addition of the weak learner $G(\mathbf{x})$ to the predictor $f(\mathbf{x})$ results in

$$\begin{aligned}J[f + G] &= E_{\mathbf{X},Y}[I(y = 1) w(\mathbf{x}, 1) \exp(-C_1 G(\mathbf{x})) + \\ & \qquad I(y = -1) w(\mathbf{x}, -1) \exp(C_2 G(\mathbf{x}))]\end{aligned}$$

with

$$w(\mathbf{x}, 1) = \exp(-C_1 f(\mathbf{x})) \tag{64}$$

and

$$w(\mathbf{x}, -1) = \exp(C_2 f(\mathbf{x})). \tag{65}$$

Since $J[f+G]$ is minimized if and only if the argument of the expectation is minimized for all $\mathbf{x}$, and assuming that the weak learners depend on $\mathbf{x}$ only through some feature $\phi(\mathbf{x})$, the optimal weak learner is the solution of

$$\begin{aligned} G_\phi(\mathbf{x}) &= \arg\min_G \quad E_{Y|\mathbf{X}}[I(y=1)w(\mathbf{x},1)\exp(-C_1 G(\mathbf{x})) \\ &\qquad +I(y=-1)w(\mathbf{x},-1)\exp(C_2 G(\mathbf{x}))|\mathbf{x}] \\ &= \arg\min_G \quad P_{Y|\mathbf{X}}(1|\phi(\mathbf{x}))w(\mathbf{x},1)\exp(-C_1 G(\mathbf{x})) \\ &\qquad +P_{Y|\mathbf{X}}(-1|\phi(\mathbf{x}))w(\mathbf{x},-1)\exp(C_2 G(\mathbf{x})) \\ &= \arg\min_G \quad P^{(w)}_{Y|\mathbf{X}}(1|\phi(\mathbf{x}))\exp(-C_1 G(\mathbf{x})) \\ &\qquad +P^{(w)}_{Y|\mathbf{X}}(-1|\phi(\mathbf{x}))\exp(C_2 G(\mathbf{x})) \end{aligned}$$

where

$$P^{(w)}_{Y|\mathbf{X}}(y|\phi(\mathbf{x})) = \frac{P_{Y|\mathbf{X}}(y|\phi(\mathbf{x}))w(\mathbf{x},y)}{\sum_{y\in\{1,-1\}} P_{Y|\mathbf{X}}(y|\phi(\mathbf{x}))w(\mathbf{x},y)}$$

are the posterior estimates associated with a sample reweighed according to $w(\mathbf{x}, y)$. Setting the derivatives of the cost to zero it follows that

$$P^{(w)}_{Y|\mathbf{X}}(1|\phi(\mathbf{x}))C_1\exp(-C_1 G(\mathbf{x})) = P^{(w)}_{Y|\mathbf{X}}(-1|\phi(\mathbf{x}))C_2\exp(C_2 G(\mathbf{x}))$$

and

$$G_\phi(\mathbf{x}) = \frac{1}{C_1+C_2}\log\frac{P^{(w)}_{Y|\mathbf{X}}(1|\phi(\mathbf{x}))C_1}{P^{(w)}_{Y|\mathbf{X}}(-1|\phi(\mathbf{x}))C_2}.$$

The optimal feature ϕ^* is the one of smallest minimum cost

$$\begin{aligned} \phi^* &= \arg\min_\phi J[f+G_\phi] \\ &= \arg\min_\phi E_{\mathbf{X},Y}[I(y=1)w(\mathbf{x},1)\exp(-C_1 G_\phi(\mathbf{x})) + \\ &\qquad I(y=-1)w(\mathbf{x},-1)\exp(C_2 G_\phi(\mathbf{x}))] \\ &= \arg\min_\phi \left[\sum_{i\in\mathcal{I}_+} w(\mathbf{x}_i,1)\exp(-C_1 G_\phi(\mathbf{x}_i)) + \right. \\ &\qquad \left. \sum_{i\in\mathcal{I}_-} w(\mathbf{x}_i,-1)\exp(C_2 G_\phi(\mathbf{x}_i))\right]. \end{aligned}$$

Once $G_m^{real}(\mathbf{x})$ is found, the weights are updated so as to comply with (64) and (65), i.e.

$$w(\mathbf{x}, 1) \leftarrow w(\mathbf{x}, 1) \exp(-C_1 G_{\phi^*}(\mathbf{x}))$$

and

$$w(\mathbf{x}, -1) \leftarrow w(\mathbf{x}, -1) \exp(C_2 G_{\phi^*}(\mathbf{x})).$$

D Proof of Theorem 5

Rewriting the negative loglikelihood as

$$l_b[y', f(\mathbf{x})] = -E_{\mathbf{X},Y}\left[y' \log \frac{p_c(\mathbf{x})}{1 - p_c(\mathbf{x})} + \log(1 - p_c(\mathbf{x}))\right]$$

and using (35), it follows that

$$l_b[y', f(\mathbf{x})] = -E_{\mathbf{X},Y}\left[2y'(\gamma f(\mathbf{x}) + \eta) - \log\left[1 + e^{2(\gamma f(\mathbf{x}) + \eta)}\right]\right].$$

This loss is minimized by maximizing the conditional expectation

$$\begin{aligned} -l_b[y', f(\mathbf{x})|\mathbf{x}] &= E_{Y|\mathbf{X}}\left[2y'(\gamma f(\mathbf{x}) + \eta) - \log\left[1 + e^{2(\gamma f(\mathbf{x}) + \eta)}\right]\right] \\ &= 2E_{Y|\mathbf{X}}[y'|\mathbf{x}](\gamma f(\mathbf{x}) + \eta) - \log\left[1 + e^{2(\gamma f(\mathbf{x}) + \eta)}\right] \end{aligned}$$

for all $\mathbf{x}$, i.e. by searching for the weak learner $G(\mathbf{x})$ that maximizes the cost

$$J[f(\mathbf{x}) + G(\mathbf{x})] = -l_b[y', f(\mathbf{x}) + G(\mathbf{x})|\mathbf{x}].$$

The maximization is done by Newton's method, which requires the computation of the gradient

$$\left.\frac{\partial J[f(\mathbf{x}) + G(\mathbf{x})]}{\partial G(\mathbf{x})}\right|_{G(\mathbf{x})=0} = 2\gamma(E_{Y|\mathbf{X}}[y'|\mathbf{x}] - p_c(\mathbf{x}))$$

and Hessian

$$\left.\frac{\partial^2 J[f(\mathbf{x}) + G(\mathbf{x})]}{\partial G(\mathbf{x})^2}\right|_{G(\mathbf{x})=0} = -4\gamma^2 p_c(\mathbf{x})(1 - p_c(\mathbf{x}))$$

leading to a Newton update

$$G(\mathbf{x}) = \frac{1}{2\gamma} E_{Y|\mathbf{X}}\left[\frac{y' - p_c(\mathbf{x})}{p_c(\mathbf{x})(1 - p_c(\mathbf{x}))}\right].$$

This is equivalent to solving the least squares problem

$$\min_{G(\mathbf{x})} E_{Y,\mathbf{X}}\left[\left(\frac{1}{2\gamma}\frac{y' - p_c(\mathbf{x})}{p_c(\mathbf{x})(1 - p_c(\mathbf{x}))} - G(\mathbf{x})\right)^2\right],$$

and the optimal weak learner can, therefore, be computed with

$$
\begin{aligned}
G^* &= \min_G \int P_{\mathbf{X}}(\mathbf{x}) \sum_{y'=0}^{1} P_{Y|\mathbf{X}}(y'|\mathbf{x}) \left(\frac{1}{2\gamma} \frac{y' - p_c(\mathbf{x})}{p_c(\mathbf{x})(1-p_c(\mathbf{x}))} - G(\mathbf{x}) \right)^2 d\mathbf{x} \\
&= \min_G \int P_{\mathbf{X}}(\mathbf{x}) \sum_{y'=0}^{1} \frac{P_{Y|\mathbf{X}}(y'|\mathbf{x}) w(\mathbf{x})}{\sum_{j=0}^{1} P_{Y|\mathbf{X}}(j|\mathbf{x}) w(\mathbf{x})} \\
&\qquad \left(\frac{1}{2\gamma} \frac{y' - p_c(\mathbf{x})}{p_c(\mathbf{x})(1-p_c(\mathbf{x}))} - G(\mathbf{x}) \right)^2 d\mathbf{x} \\
&= \min_G \int P_{\mathbf{X}}(\mathbf{x}) \sum_{y'=0}^{1} P^{(w)}_{Y|\mathbf{X}}(y'|\mathbf{x}) \left(\frac{1}{2\gamma} \frac{y' - p_c(\mathbf{x})}{p_c(\mathbf{x})(1-p_c(\mathbf{x}))} - G(\mathbf{x}) \right)^2 d\mathbf{x} \\
&= \min_G E^{(w)}_{Y,\mathbf{X}} \left[\left(\frac{1}{2\gamma} \frac{y' - p_c(\mathbf{x})}{p_c(\mathbf{x})(1-p_c(\mathbf{x}))} - G(\mathbf{x}) \right)^2 \right]
\end{aligned}
$$

which is the weighted least squares regression of z_i to $\mathbf{x}_i$ using weights w_i, as given by (54) and (55). The optimal feature is the one of smallest regression error.

References

[1] AGARWAL, S., GRAEPEL, T., HERBRICH, R., HAR-PELED, S., AND ROTH, D. (2005). Generalization bounds for the area under the roc curve. *The Journal of Machine Learning Research 6*, 393–425.

[2] BREIMAN, L. (1998). Arcing classifiers. *The Annals of Statistics* **26**, 3, 801–849.

[3] COLLINS, M., SCHAPIRE, R. E., AND SINGER, Y. (2002). Logistic regression, adaboost and bregman distances. In *Journal of Machine Leaning*. Vol. **48**. 253–285.

[4] DOMINGOS, P. (1999). Metacost: a general method for making classifiers cost-sensitive. In *Knowledge Discovery and Data Mining*. 155–164.

[5] DUDA, R. O., HART, P. E., AND STORK, D. G. (2001). *Pattern Classification*. John Wiley Sons Inc, New York.

[6] ELKAN, C. (2001). The foundations of cost-sensitive learning. In *Seventeenth Intrnl. Joint Conference on Artificial Intelligence*. 973–978.

[7] FAN, W., STOLFO, S., ZHANG, J., AND CHAN, P. (1999). Adacost: Misclassification cost-sensitive boosting. In *Proc. of 6th International Conf. on Machine Learning*. 97–105.

[8] FREUND, Y. AND SCHAPIRE, R. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences* **55**, 1, part 2, 119–139.

[9] FREUND, Y. AND SCHAPIRE, R. (2004). A discussion of "Process consistency for AdaBoost" by Wenxin Jiang, "On the Bayes-risk consistency of regularized boosting methods" by Gbor Lugosi and Nicolas Vayatis, "Statistical behavior and consistency of classification methods based on convex risk minimization" by Tong Zhang. *The Annals of Statistics* **32**, 1.

[10] FREUND, Y. AND SCHAPIRE, R. E. (1996a). Experiments with a new boosting algorithm. In *International Conference on Machine Learning*. 148–156.

[11] FREUND, Y. AND SCHAPIRE, R. E. (1996b). Game theory, on-line prediction and boosting. In *Computational Learing Theory*. 325–332.

[12] FRIEDMAN, J., HASTIE, T., AND TIBSHIRANI, R. (2000). Additive logistic regression: A statistical view of boosting. *The Annals of Statistics 38*, 337–374.

[13] FRIEDMAN, J. H. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics* **29**, 5, 1189–1232.

[14] GREEN, D. AND SWETS, J. (1966). *Signal detection theory and psychophysics*. John Wiley and Sons Inc., New York.

[15] GREINER, R., GROVE, A. J., AND ROTH, D. (2002). Learning cost-sensitive active classifiers. *Artificial Intelligence* **139**, 2, 137–174.

[16] HANLEY, J. A. AND MCNEIL, B. J. (1982). The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology* **143**, 1, 29–36.

[17] HASTIE, TIBSHIRANI, AND FRIEDMAN. (2001). *The Elements of Statistical Learning*. Springer-Verlag Inc, New York.

[18] HSIEH, F. AND TURNBULL, B. W. (1996). Nonparametric and semiparametric estimation of the receiver operating characteristic curve. *The Annals of Statistics* **24**, 1, 25–40.

[19] JIANG, W. (2004). Process consistency for adaboost. *The Annals of Statistics 32*, 13–29.

[20] KONONENKO, I. (2001). Machine learning for medical diagnosis: history, state of the art and perspective. *Artificial Intelligence in Medicine* **23**, 1, 89–109.

[21] LANE, T. AND BRODLEY, C. E. (2003). An empirical study of two approaches to sequence learning for anomaly detection. *Machine Learning* **51**, 1, 73–107.

[22] LIN, H.-T., LIN, C.-J., AND WENG, R. C. (2007). A note on platt's probabilistic outputs for support vector machines. *Machine Learning* **68**, 3, 267–276.

[23] MA, J., NGUYEN, N., AND RAJAPAKSE, J. (2007). Gene classification using codon usage analysis and support vector machines. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*. to be published.

[24] MARGINEANTU, D. D. AND DIETTERICH, T. G. (2000). Bootstrap methods for the cost-sensitive evaluation of classifiers. In *Proc. 17th International Conf. on Machine Learning*. 583–590.

[25] MASON, L., BAXTER, J., BARTLETT, P., AND FREAN, M. (2000). Boosting Algorithms as Gradient Descent. In *Advances in Neural Information Processing Systems*. 512–518.

[26] MEASE, D. AND WYNER, A. J. (2007). Evidence contrary to the statistical view of boosting. *Journal of Machine Learning Research*. to be published.

[27] MEASE, D., WYNER, A. J., AND BUJA, A. (2007). Boosted classification trees and class probability/quantile estimation. *The Journal of Machine Learning Research 8*, 409–439.

[28] NEWMAN, D., HETTICH, S., BLAKE, C., AND MERZ, C. (1998). UCI repository of machine learning databases. http://www.ics.uci.edu/∼mlearn/MLRepository.html.

[29] NEYMAN, J. AND PEARSON, E. S. (1933). On the problem of the most efficient tests of statistical hypotheses. *Philosophical Transactions of the Royal Society of London 231*, 289–337.

[30] NICULESCU-MIZIL, A. AND CARUANA, R. (2005). Obtaining calibrated probabilities from boosting. In *Proc. 21st Conference on Uncertainty in Artificial Intelligence (UAI '05)*. AUAI Press, 413–420.

[31] PARK, S.-B., HWANG, S., AND ZHANG, B.-T. (2003). Mining the risk types of human papillomavirus (hpv) by adacost. In *International Conference on Database and expert Systems Applications*. 403–412.

[32] PLATT, J. (2000). Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. In *Adv. in Large Margin Classifiers*. 61–74.

[33] REYZIN, L. AND SCHAPIRE, R. E. (2006). How boosting the margin can also boost classifier complexity. In *International Conference on Machine Learning*. 753–760.

[34] SCHAPIRE, R. E. (1990). The strength of weak learnability. *Machine Learning 5*, 197–227.

[35] SCHAPIRE, R. E., FREUND, Y., BARTLETT, P., AND LEE, W. S. (1998). Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics* **26**, 5, 1651–1686.

[36] SCHAPIRE, R. E. AND SINGER, Y. (1999). Improved boosting using confidence-rated predictions. *Machine Learning* **37**, 3, 297–336.

[37] SUN, Y., WONG, A. K. C., AND WANG, Y. (2005). Parameter inference of cost-sensitive boosting algorithms. In *Machine Learning and Data Mining in Pattern Recognition,4th International Conference*. 21–30.

[38] TING, K. M. (2000). A comparative study of cost-sensitive boosting algorithms. In *Proc. 17th International Conf. on Machine Learning*. 983–990.

[39] TREE, H. L. V. (1968). *Detection, Estimation and Modulation Theory*. John Wiley and Sons Inc, New York.

[40] VAPNIK, V. N. (1998). *Statistical Learning Theory*. John Wiley Sons Inc, New York.

[41] VIAENE, S., DERRIG, R. A., AND DEDENE, G. (2004). Cost-sensitive learning and decision making for massachusetts pip claim fraud data. *International Journal of Intelligent Systems 19*, 1197–1215.

[42] VIOLA, P. AND JONES, M. (2002). Fast and robust classification using asymmetric adaboost and a detector cascade. In *Advances in Neural Information Processing System*. Vol. **2**. 1311–1318.

[43] VIOLA, P. A. AND JONES, M. J. (2004). Robust real-time face detection. *International Journal of Computer Vision* **57**, 2, 137–154.

[44] VLAHOU, A., SCHORGE, J. O., GREGORY, B. W., AND COLEMAN, R. L. (2003). Diagnosis of ovarian cancer using decision tree classification of mass spectral data. *Journal of Biomedicine and Biotechnology* **2003**, 5, 308314.

[45] WALD, A. (1939). Contributions to the theory of statistical estimation and testing hypotheses. *The Annals of Mathematical Statistics 10*, 299–326.

[46] WANG, L., CHU, F., AND XIE, W. (2007). Accurate cancer classification using expressions of very few genes. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* **4**, 1, 40–53.

[47] WU, T.-F., LIN, C.-J., AND WENG, R. C. (2004). Probability estimates for multi-class classification by pairwise coupling. *The Journal of Machine Learning Research 5*, 975–1005.

[48] ZADROZNY, B. AND ELKAN, C. (2001). Learning and making decisions when costs and probabilities are both unknown. In *7th International Conference on Knowledge Discovery and Data Mining*. 203–213.

[49] ZADROZNY, B., LANGFORD, J., AND ABE, N. (2003). Cost-sensitive learning by cost-proportionate example weighting. In *Third IEEE International Conference on Data Mining*. 435–442.

[50] ZEMEL, R. S. AND PITASSI, T. (2000). A gradient-based boosting algorithm for regression problems. In *Advances in Neural Information Processing Systems*. 696–702.

SVCL-TR 2007/06

June 2007

Cost-Sensitive Boosting

Hamed Masnadi-Shirazi