This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

# Cost-Sensitive Boosting

Hamed Masnadi-Shirazi, and Nuno Vasconcelos, *Senior Member, IEEE* Statistical Visual Computing Laboratory,
University of California, San Diego
La Jolla, CA 92039

***Abstract*— A novel framework is proposed for the design of cost sensitive boosting algorithms. The framework is based on the identification of two necessary conditions for optimal cost-sensitive learning: that 1) expected losses must be minimized by optimal cost-sensitive decision rules, and 2) empirical loss minimization must emphasize the neighborhood of the target cost-sensitive boundary. It is shown that these conditions enable the derivation of cost-sensitive losses that can be minimized by gradient descent, in the functional space of convex combinations of weak learners, to produce novel boosting algorithms. The proposed framework is applied to the derivation of cost-sensitive extensions of AdaBoost, RealBoost, and LogitBoost. Experimental evidence, with a synthetic problem, standard data sets, and the computer vision problems of face and car detection, is presented in support of the cost-sensitive optimality of the new algorithms. Their performance is also compared to those of various previous cost-sensitive boosting proposals, as well as the popular combination of large margin classifiers and probability calibration. Cost-sensitive boosting is shown to consistently outperform all other methods.**

***Index Terms*— Boosting, AdaBoost, cost-sensitive learning, asymmetric boosting.**

## I. Introduction

Classification problems such as fraud detection [1], medical diagnosis [2], or object detection in computer vision [3], [4], [5], [6], [7], [8], [9], [10], are naturally cost sensitive [11]. 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 [11]. 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 [12], [13]. 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) [14], [15], 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 [16], [17], [18]. These are algorithms that 1) learn a predictor by combining weak classification rules (weak learners), and 2) use a sample re-weighting mechanism to emphasize points that are difficult to classify.

In this work, we consider the problem of how to extend boosting algorithms so as to achieve optimal *cost-sensitive* decision rules. The starting point is the observation, by Friedman et al. [18], that in the (asymptotic) limit of infinite training data 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. Convergence to this optimal predictor is, however, not guaranteed *everywhere* for finite training samples. It is, in fact, well known that, in this case, boosting does not produce calibrated estimates of class posterior probabilities [19], [20], [21], [18], [22]. This is due to the emphasis of sample reweighing on the classification boundary: while the boosted predictor converges to the optimal predictor in a small neighborhood of this boundary, it does not approximate the latter well away from it. This does not compromise *cost-insensitive* classification performance, which only requires the two predictors to have the same sign, but impairs *cost-sensitive* performance, which requires a good approximation of the optimal predictor throughout the feature space.

Two conditions are identified as necessary for optimal cost-sensitive boosting: 1) that the expected boosting loss is minimized by the optimal cost-sensitive decision rule, and 2) that empirical loss minimization emphasizes a neighborhood of the target cost-sensitive boundary, rather than that optimal in the cost-insensitive sense. We propose that this is best accomplished by modifying boosting's loss function, so that boosting-style gradient descent can satisfy the two necessary conditions above. 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 underlie AdaBoost [16], RealBoost [18], [23], and LogitBoost [18]. Cost-sensitive extensions of the algorithms are derived, and shown to satisfy the necessary conditions for cost-sensitive optimality. The new algorithms are compared with various cost-sensitive extensions of boosting available in the literature, including AdaCost [24], CSB0, CSB1, CSB2 [25] asymmetric-AdaBoost [3] and AdaC1, AdaC2, AdaC3 [26]. All of these extensions are heuristic, achieving cost-sensitivity by manipulation of AdaBoost's weights and confidence parameters. In most cases it is not clear if, or how, these manipulations modify boosting's loss. This is unlike the framework now proposed, which inherits all properties of cost-insensitive boosting, simply shifting 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 also evaluated through experiments on synthetic problems, and datasets from the UCI repository [27] and computer vision face [28] and car [29] detection problems. These experiments show that the proposed algorithms do indeed possess cost sensitive optimality, and can meet target detection rates without (sub-optimal) weight manipulation.

Digital Object Indentifier 10.1109/TPAMI.2010.71

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

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 II we review the main principles of cost-sensitive classification. Section III 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 IV, where the extensions of AdaBoost, RealBoost, and LogitBoost, are also derived. Finally, empirical evaluation is discussed in Section V, and some conclusions are drawn in Section VI.

## II. Cost-sensitive classification

We start with the fundamentals of cost-sensitive classification. Most concepts apply to multi-way classification, but here we only consider the problem of binary classification, or *detection*.

### A. 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

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

where $f : \mathcal{X} \rightarrow \mathbb{R}$ is a predictor, and $\text{sgn}[x] = 1$ if $x \geq 0$, and $\text{sgn}[x] = -1$ otherwise. Feature vectors are samples from a random process $\mathbf{X}$ that is described by a probability distribution $P_{\mathbf{X}}(\mathbf{x})$ on $\mathcal{X}$, and labels are samples from a random variable $Y$ of probability distribution $P_Y(y)$, $y \in \{-1, 1\}$. 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 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 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 $C_2$ and $C_1$. There are, nevertheless, problems in which it is more natural to specify target detection or false-positive rates than costs. The two types of problems can be addressed within a common optimal detection framework.

### B. Optimal detection

When $C_1$ and $C_2$ are specified, the optimal predictor is given by the BDR [14], [15], i.e.

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

with

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

An alternative specification is in terms of error rates, where the goal is to minimize the false-positive rate of the classifier given a target detection rate. The optimal solution can be obtained with recourse to the Neyman-Pearson Lemma [30]: for any detection rate $\xi$, the optimal predictor is still (3). However, for a given $\xi$, the constants $(C_1, C_2)$ must be such that the specified detection rate is met, i.e.

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

with

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

The only difference is that, rather than specifying costs, one has to search for the costs that satisfy (4). This can be done by cross-validation. Since all that matters is $C_1/C_2$, $C_2$ can be set to one and the search is one-dimensional. In any case, the optimal detector can be written as

$$h^*_T(\mathbf{x}) = \text{sgn}\left[f^*_0(\mathbf{x}) - T\right] \tag{5}$$

where

$$f^*_0(\mathbf{x}) = \log \frac{P_{Y|\mathbf{X}}(1|\mathbf{x})}{P_{Y|\mathbf{X}}(-1|\mathbf{x})}, \tag{6}$$

is the optimal cost-insensitive predictor and

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

Hence, for any cost structure $(C_1, C_2)$, cost-sensitive optimality differs from cost-insensitive optimality only through the threshold $T$: all optimal cost-sensitive rules can be obtained from $f^*_0(\mathbf{x})$ by threshold manipulation. Furthermore, from (4), different thresholds correspond to different detection rates, and threshold manipulation can produce the optimal decision rule at any detection (or false-positive) rate. This is the motivation for the widespread use of receiver operating curves (ROCs) [31], [32], [33], and the tuning of error rates by threshold manipulation.

### C. Practical detection

In practice, the probabilities of (6) are unknown, and a learning algorithm is used to estimate the predictor $\hat{f}(\mathbf{x}) \approx f^*_0(\mathbf{x})$, producing an approximately optimal cost-sensitive rule

$$\hat{h}_T(\mathbf{x}) = \text{sgn}[\hat{f}(\mathbf{x}) - T]. \tag{8}$$

This, however, does not guarantee good cost-sensitive performance for the particular cost-structure $(C_1, C_2)$ associated with $T$. In fact, there are no guarantees of the latter *even when the cost-insensitive detector is optimal*, i.e. when $\hat{h}_0(\mathbf{x}) =$

$\text{sgn}[f_0^*(\mathbf{x})]$. While the necessary and sufficient conditions for cost-insensitive optimality are that

$$\hat{f}(\mathbf{x}) = f_0^*(\mathbf{x}) = 0, \ \forall \mathbf{x} \in \mathcal{C} \qquad (9)$$

$$\text{sgn}[\hat{f}(\mathbf{x})] = \text{sgn}[f_0^*(\mathbf{x})], \ \forall \mathbf{x} \notin \mathcal{C}, \qquad (10)$$

where

$$\mathcal{C} = \left\{ \mathbf{x} \left| \log \frac{P_{Y|\mathbf{X}}(1|\mathbf{x})}{P_{Y|\mathbf{X}}(-1|\mathbf{x})} = 0 \right. \right\}$$

is the optimal cost-insensitive classification boundary, the optimality of (8) requires that

$$\hat{f}(\mathbf{x}) = f_0^*(\mathbf{x}) = T, \ \forall \mathbf{x} \in \mathcal{C}_T \qquad (11)$$

$$\text{sgn}[\hat{f}(\mathbf{x}) - T] = \text{sgn}[f_0^*(\mathbf{x}) - T], \ \forall \mathbf{x} \notin \mathcal{C}_T \qquad (12)$$

with

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

Hence, the necessary condition for cost-sensitive optimality of $\hat{f}$ at any point $\mathbf{x}$ in the boundary $\mathcal{C}_T$, $\hat{f}(\mathbf{x}) = f_0^*(\mathbf{x})$, is much tighter than the sufficient condition for cost-insensitive optimality of $\hat{f}$ at that point, $\text{sgn}[\hat{f}(\mathbf{x})] = \text{sgn}[f_0^*(\mathbf{x})]$.

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, (9) and (10), of 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 [34].

## III. Boosting

This work addresses the cost-sensitive extension of boosting algorithms. Such algorithms learn a predictor $f(\mathbf{x})$ by linear combination of simple decision rules, or *weak learners* [35], $G_m(\mathbf{x})$

$$f(\mathbf{x}) = \sum_{m=1}^{M} G_m(\mathbf{x}). \qquad (13)$$

Optimality is defined with respect to some risk, such as the expected exponential loss

$$E_{\mathbf{X},Y}[\exp(-yf(\mathbf{x}))], \qquad (14)$$

or the expected negative binomial log-likelihood

$$-E_{\mathbf{X},Y}[y' \log(p(\mathbf{x})) + (1-y')\log(1-p(\mathbf{x}))] \qquad (15)$$

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

$$p(\mathbf{x}) = \frac{e^{f(\mathbf{x})}}{e^{f(\mathbf{x})} + e^{-f(\mathbf{x})}}. \qquad (16)$$

Learning is based on a finite sample $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, empirical loss estimates, and iterative selection of weak learners. At iteration $m$, a weight $w_i^{(m)}$ is assigned to example $(\mathbf{x}_i, y_i)$, reweighing $\mathcal{D}$ to amplify the importance of points that are poorly classified with the current predictor. We next review some popular algorithms in this family, whose cost-sensitive extensions will be later introduced. All of these can be interpreted as gradient descent on a functional space of linear combinations of weak learners, with respect to one of the losses above[36], [37], [38].

### A. AdaBoost

AdaBoost [16], [39] learns combinations of scaled binary classifiers

$$G_m^{Ada}(\mathbf{x}) = \alpha_m g_m(\mathbf{x}), \qquad (17)$$

where $\{\alpha_m\}_{m=1}^M$ is a weight sequence and $\{g_m(\mathbf{x})\}_{m=1}^M$ a sequence of binary rules, $g_m(\mathbf{x}) : \mathcal{X} \rightarrow \{-1,1\}$, usually implemented with a *decision stump* $g_m(\mathbf{x}) = \text{sgn}[\phi_m(\mathbf{x}) - t_m]$, where $\phi_m(\mathbf{x})$ is a feature response (projection of $\mathbf{x}$ along a basis function $\phi_m$) and $t_m$ a threshold. The ensemble predictor of (13) is learned by gradient descent with respect to the exponential loss. The direction of largest descent at the $m^{th}$ iteration is [40], [36]

$$g_m(\mathbf{x}) = \arg\min_g (err_{(m)}) \qquad (18)$$

where

$$err_{(m)} = \sum_{i=1}^{n} w_i^{(m)} [1 - I(y_i = g_m(\mathbf{x}_i))], \qquad (19)$$

is the total error of $g_m(\mathbf{x})$ and $I(\cdot)$ the indicator function

$$I(y = x) = \begin{cases} 1 & y = x \\ 0 & y \neq x. \end{cases} \qquad (20)$$

The optimal step size in the descent direction has closed-form

$$\alpha_m = \frac{1}{2} \log\left( \frac{1 - err_{(m)}}{err_{(m)}} \right), \qquad (21)$$

and the weights are updated according to

$$w_i^{(m+1)} = w_i^{(m)} e^{-y_i G_m^{Ada}(\mathbf{x}_i)}. \qquad (22)$$

### B. RealBoost

RealBoost [18], [23] is an extension of AdaBoost that produces better estimates of $f_0^*(\mathbf{x})$ by using real-valued weak learners in (13) (in contrast with binary-valued weak learners.) In this case, the direction of greatest descent of the exponential loss is a (re-weighted) log-odds ratio

$$G_m^{real}(\mathbf{x}) = \frac{1}{2} \log \frac{P_{Y|\mathbf{X}}^{(w)}(1|\phi_m(\mathbf{x}))}{P_{Y|\mathbf{X}}^{(w)}(-1|\phi_m(\mathbf{x}))}, \qquad (23)$$

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

$$w_i^{(m+1)} = w_i^{(m)} e^{-y_i G_m^{real}(\mathbf{x}_i)}. \qquad (24)$$

### *C. LogitBoost*

LogitBoost is motivated by the following observation, initially made by Friedman et al. [18].

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

$$f_0^*(x) = \frac{1}{2}\log\frac{P_{Y|\mathbf{X}}(1|\mathbf{x})}{P_{Y|\mathbf{X}}(-1|\mathbf{x})}. \tag{25}$$

*Proof:* See [18]. ■

This implies that both Ada and RealBoost are stage-wise procedures for fitting an additive logistic regression model. Friedman et al. argued that this is more naturally accomplished by stage-wise minimization of (15). At the $m^{th}$ boosting iteration, the optimal step is given by 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 (16) based on the predictor of (13) after $m-1$ iterations. The weights are

$$w_i^{(m)} = p^{(m)}(\mathbf{x}_i)(1-p^{(m)}(\mathbf{x}_i)). \tag{26}$$

### *D. Limitations for cost-sensitive learning*

We have already seen that the optimal cost-insensitive detector does not require the optimal predictor of (25): it suffices that (13) converges to any function satisfying (9) and (10). While Lemma 1 guarantees that the minimization of the exponential or binomial losses are sufficient to obtain (25), these guarantees are asymptotic, and do not necessarily hold for finite samples. In fact, the large-margin classification theory suggests that good out-of-sample generalization requires a greater accuracy of the approximation inside a neighborhood of the optimal cost-insensitive boundary $\mathcal{C}$ than outside of it. For boosting, the emphasis on the boundary is accomplished through the example re-weighting of (22), (24), or (26). This, however, usually implies that (13) does not converge to the optimal predictor *everywhere*, and is not necessarily a good predictor for *cost-sensitive* detection.

To obtain some intuition, we consider a detection problem with a bounded optimal predictor $f_0^*(\mathbf{x})$. Assume a finite training sample $\mathcal{D}$ and that, as is common in the large-margin literature, sample points from the two classes are separable, i.e. the detector $\text{sgn}[f_0^*(\mathbf{x})]$ has zero classification error on $\mathcal{D}$[1]. Define the neighborhood $\mathcal{N}(\mathcal{C}) = \{\mathbf{x}; |f_0^*(\mathbf{x})| < \epsilon\}$, where $\epsilon > 0$ is such that $\mathcal{N}(\mathcal{C})$ contains at least one positive and one negative example. Let $\hat{f}^{(m)}(\mathbf{x})$ be the predictor learned by $m$ iterations of boosting, and assume that

$$\hat{f}^{(m)}(\mathbf{x}) = \begin{cases} f_0^*(\mathbf{x}), & \forall \mathbf{x} \in \mathcal{N}(\mathcal{C}) \\ +\infty, & \text{if } f_0^*(\mathbf{x}) > 0 \text{ and } \mathbf{x} \notin \mathcal{N}(\mathcal{C}) \\ -\infty, & \text{if } f_0^*(\mathbf{x}) < 0 \text{ and } \mathbf{x} \notin \mathcal{N}(\mathcal{C}). \end{cases} \tag{27}$$

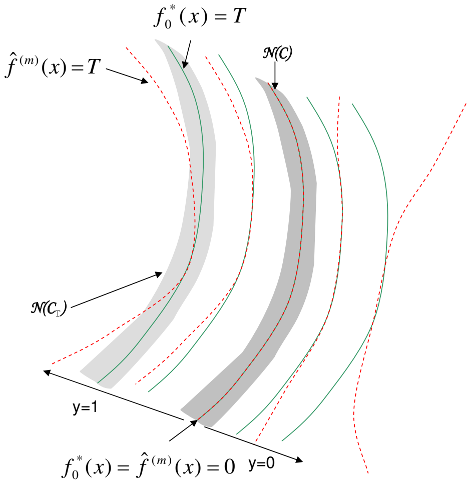

$f_0^*(x) = T$
$\hat{f}^{(m)}(x) = T$
$\mathcal{N}(\mathcal{C})$
$\mathcal{N}(\mathcal{C}_T)$
y=1
$f_0^*(x) = \hat{f}^{(m)}(x) = 0$
y=0

Fig. 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 $\hat{f}^{(m)}(\mathbf{x})$ (dashed line). While boosting emphasizes the approximation of $f_0^*(\mathbf{x})$ in $\mathcal{N}(\mathcal{C})$, optimal cost-sensitive rules require a good approximation in other regions, e.g. $\mathcal{N}(\mathcal{C}_T)$.

For both Ada and RealBoost, a simple recursion shows that

$$\frac{w_i^{(m)}}{w_i^{(0)}} = e^{-y_i\sum_{k=1}^m G_k(\mathbf{x}_i)} = e^{-y_i\hat{f}^{(m)}(\mathbf{x}_i)}, \tag{28}$$

where we also used (13). Let the initial weight distribution be uniform, $w_i^{(0)} = 1/n$, as is customary in boosting practice. Since $y_i\hat{f}^{(m)}(\mathbf{x}_i) \geq 0, \forall i \in \mathcal{D}$, it follows that

$$nw_i^{(m)} = e^{-|\hat{f}^{(m)}(\mathbf{x}_i)|}. \tag{29}$$

Similarly, for LogitBoost,

$$\begin{aligned} w_i^{(m)}(\mathbf{x}_i) &= \left(e^{\hat{f}^{(m)}(\mathbf{x}_i)} + e^{-\hat{f}^{(m)}(\mathbf{x}_i)}\right)^{-2} \\ &\approx e^{-2\text{sgn}[\hat{f}^{(m)}(\mathbf{x}_i)]\hat{f}^{(m)}(\mathbf{x}_i)} = e^{-2|\hat{f}^{(m)}(\mathbf{x}_i)|}. \end{aligned} \tag{30}$$

In either case, $nw_i^{(m)}$ or $w_i^{(m)}$ can be seen as a measure of the importance of training point $i$ (relative to the remainder of $\mathcal{D}$). Inside the neighborhood $\mathcal{N}(\mathcal{C})$ this importance is one for points along the *cost-insensitive* boundary $\mathcal{C}$ (where $\hat{f}^{(m)}(\mathbf{x}) = 0$), and decreases exponentially with the distance to it. Outside $\mathcal{N}(\mathcal{C})$ all points have zero importance (because $|\hat{f}^{(m)}(\mathbf{x})| = \infty$). Hence, despite the facts that 1) the predictor is already perfect in $\mathcal{N}(\mathcal{C})$ but 2) approximates $f_0^*(\mathbf{x})$ very poorly outside this neighborhood, all points outside $\mathcal{N}(\mathcal{C})$ are disregarded by subsequent boosting iterations. This implies that the predictor will not get any better in the sense of cost sensitive classification.

The example above turns out not to be a mathematical curiosity. Extensive empirical studies show that, when the span of the space of weak learners is rich enough to separate the training set into the two classes, and boosting is run for enough iterations, all boosting algorithms produce a distribution of posterior probabilities $P_{Y|\mathbf{X}}(y|\mathbf{x})$ highly concentrated around 0 or 1, independently of the true distribution [19], [20]. Note that this does not compromise cost-insensitive optimality:

[1] Note that the classification error does not have to be zero in general, only for the particular sample $\mathcal{D}$.

$\hat{f}^{(m)}(\mathbf{x}_i)$ simply grows to $\infty$ for positive, and to $-\infty$ for negative examples. But the boosted predictor has very poor *cost-sensitive* performance. This problem cannot be addressed by early stopping. In the iterations before class separation, boosting assigns exponentially decaying weight to points correctly classified by previous iterations, in the *cost-insensitive* sense. Hence, points far from $\mathcal{C}$ are exponentially discounted as boosting progresses, creating a soft neighborhood $\mathcal{N}(\mathcal{C})$ of nearby points that dominate the optimization. In result, boosting does not produce accurate posterior estimates, even in this regime [21], [19], [20]. This is, in fact, the reason for the popularity of post-processing boosting's predictions with probability calibration techniques, such as the method of Platt [41], or isotonic regression [42], when posterior accuracy is important [21].

The lack of everywhere convergence to the optimal predictor is illustrated in Fig. 1, which depicts $f_0^*(\mathbf{x})$ and $\hat{f}^{(m)}(\mathbf{x})$. Because $f_0^*(\mathbf{x})$ increases (decreases) monotonically to the left (right) of $\mathcal{C}$, any $\hat{f}^{(m)}(\mathbf{x})$ with 1) $\mathcal{C}$ as a zero-level set, and 2) the same monotonicity, satisfies (9)-(10). The emphasis on $\mathcal{N}(\mathcal{C})$ guarantees that the zero-level set of $\hat{f}^{(m)}(\mathbf{x})$ closely approximates $\mathcal{C}$, assuring good cost-insensitive generalization. But the level sets of $\hat{f}^{(m)}(\mathbf{x})$ and $f_0^*(\mathbf{x})$ are not *identical* beyond $\mathcal{N}(\mathcal{C})$. In particular, the set $\hat{f}^{(m)}(\mathbf{x}) = T$ can differ significantly from $f_0^*(\mathbf{x}) = T$, the optimal cost-sensitive boundary $\mathcal{C}_T$ for the cost-structure of threshold $T$ in (5). Hence, threshold manipulation on $\hat{f}^{(m)}(\mathbf{x})$ *does not* produce the optimal cost-sensitive rule of (5).

### *E. Prior work on cost-sensitive boosting*

This limitation is well known in the boosting literature, and motivated various cost-sensitive algorithms [24], [25], [3], [26]. Since, for cost-sensitive learning, the main problem is boosting's reweighing emphasis on $\mathcal{N}(\mathcal{C})$, instead of $\mathcal{N}(\mathcal{C}_T)$, it has long been noted that good cost-sensitive performance requires a different reweighing mechanism. This also complies with the intuition that cost-sensitive detection should weigh differently examples from different classes. A naive implementation of this intuition would be to modify the initial boosting weights, so as to represent the cost asymmetry. 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 heuristic is to modify the weight update equation. For example, the updated weight could be a mixture of (22), (24), or (26), and the initial cost-sensitive weights. We refer to such heuristics as “weight manipulation”. Previously proposed cost-sensitive boosting algorithms, such as AdaCost [24], CSB0, CSB1, CSB2 [25], Asymmetric-AdaBoost [3], AdaC1, AdaC2, or AdaC3 [26], fall in this class. For example, CSB2 [25] 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 (21) for the computation of $\alpha_m$. While various justifications are available for the different heuristic manipulations of the boosting equations, these manipulations provide no guarantees of asymptotic 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.

## IV. COST-SENSITIVE BOOSTING

The new framework is inspired by two observations. First, the different boosting algorithms are gradient descent methods [36], [37], [38] for empirical minimization of losses that are asymptotically minimized by the cost-insensitive predictor of (25). Second, the main limitation, for cost-sensitive learning, is the emphasis of the empirical loss minimization on a neighborhood $\mathcal{N}(\mathcal{C})$ of the cost-insensitive boundary, as shown in Figure 1. 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) does not guarantee acceptable cost-sensitive performance. Instead, boosting involves a balance between weight updates and descent 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) The expected loss is minimized by the optimal cost-sensitive predictor $f^*(\mathbf{x})$ of (3).
2) Empirical loss minimization leads to a weight-updating mechanism that emphasizes a neighborhood of $\mathcal{N}(\mathcal{C}_T)$.

This suggests an alternative strategy for cost-sensitive boosting: *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 other boosting algorithms, e.g. GentleBoost [18] or AnyBoost [36]. We limit our attention to the ones referred for reasons of brevity, and their popularity.

### *A. Cost-sensitive losses*

We start by noting that the optimal cost-sensitive detector of (5) can be re-written as $h_T^* = \text{sgn}[f^*(\mathbf{x})]$ with $f^*(\mathbf{x})$ as in (3). Since the zero level-set of this predictor is the cost-sensitive boundary $\mathcal{C}_T$, boosting-style gradient descent on loss functions asymptotically minimized by $f^*(\mathbf{x})$ should satisfy the two necessary conditions for cost-sensitive optimality. The first is indeed met by the following extensions of the exponential and binomial losses.

*Lemma 2:* The expected losses

$$E_{\mathbf{X},Y}\left[I(y=1)e^{-y.C_1 f(\mathbf{x})} + I(y=-1)e^{-y.C_2 f(\mathbf{x})}\right], \tag{32}$$

$$-E_{\mathbf{X},Y}[y' \log(p_c(\mathbf{x})) + (1-y')\log(1-p_c(\mathbf{x}))] \tag{33}$$

where $I(\cdot)$ is the indicator function of (20) and

$$p_c(\mathbf{x}) = \frac{e^{\gamma f(\mathbf{x})+\eta}}{e^{\gamma f(\mathbf{x})+\eta} + e^{-\gamma f(\mathbf{x})-\eta}}, \tag{34}$$

$$\text{with} \quad \gamma = \frac{C_1 + C_2}{2}, \quad \eta = \frac{1}{2}\log\frac{C_2}{C_1},$$

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

$$f(\mathbf{x}) = \frac{1}{C_1 + C_2} \log \frac{P(y=1|\mathbf{x})C_1}{P(y=y''|\mathbf{x})C_2}, \tag{35}$$

where $y'' = -1$ for (32) and $y'' = 0$ for (33).

*Proof:* See appendix I. ■

We next derive cost-sensitive boosting extensions, by gradient descent on empirical loss estimates, and later show that they shift the emphasis of boosting weights from $\mathcal{N}(\mathcal{C})$ to $\mathcal{N}(\mathcal{C}_T)$.

## B. Cost-sensitive AdaBoost

*Result 3: (Cost-sensitive AdaBoost)* Consider the minimization of the empirical estimate of the expected loss of (32), 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 (13) and (17), and define two sets

$$\mathcal{I}_+ = \{i|y_i = 1\} \qquad \mathcal{I}_- = \{i|y_i = -1\}. \tag{36}$$

The weak learner selected at iteration $m$ consists of an optimal step $\alpha_m$ along the direction $g_m$ of largest descent of the expected loss, and is given by

$$(\alpha_m, g_m) = \arg\min_{\alpha, g} \sum_{i \in \mathcal{I}_+} w_i^{(m)} \exp(-C_1 \alpha g(\mathbf{x}_i)) + \sum_{i \in \mathcal{I}_-} w_i^{(m)} \exp(C_2 \alpha g(\mathbf{x}_i)) \tag{37}$$

with

$$w_i^{(m+1)} = \begin{cases} w_i^{(m)} e^{-C_1 \alpha_m g_m(\mathbf{x}_i)}, & i \in \mathcal{I}_+ \\ w_i^{(m)} e^{C_2 \alpha_m g_m(\mathbf{x}_i)}, & i \in \mathcal{I}_-. \end{cases} \tag{38}$$

The optimal step $\alpha(g)$ along a direction $g$ is the solution of

$$2C_1 \cdot b \cdot \cosh(C_1\alpha) + 2C_2 \cdot d \cdot \cosh(C_2\alpha) = C_1 \cdot \mathcal{T}_+ \cdot e^{-C_1\alpha} + C_2 \cdot \mathcal{T}_- \cdot e^{-C_2\alpha} \tag{39}$$

with

$$\mathcal{T}_+ = \sum_{i \in \mathcal{I}_+} w_i^{(m)} \qquad \mathcal{T}_- = \sum_{i \in \mathcal{I}_-} w_i^{(m)} \tag{40}$$

$$b = \sum_{i \in \mathcal{I}_+} w_i^{(m)} [1 - I(y_i = g(\mathbf{x}_i))]$$
$$d = \sum_{i \in \mathcal{I}_-} w_i^{(m)} [1 - I(y_i = g(\mathbf{x}_i))] \tag{41}$$

and the descent direction is given by

$$g_m = \arg\min_g \Big[ (e^{C_1\alpha(g)} - e^{-C_1\alpha(g)}) \cdot b + e^{-C_1\alpha(g)} \mathcal{T}_+ + (e^{C_2\alpha(g)} - e^{-C_2\alpha(g)}) \cdot d + e^{-C_2\alpha(g)} \mathcal{T}_- \Big] \tag{42}$$

*Proof:* See appendix II. ■

For AdaBoost, possible descent directions are defined by a set of binary classifiers $\{g_k(\mathbf{x})\}_{k=1}^K$. The gradient descent iteration cycles through these, for each solving (39). This can be done efficiently with standard scalar search procedures. In our experiments, the optimal $\alpha$ was found in an average of 6 iterations of bisection search. Given $\alpha$, the loss associated with the binary classifier is computed and the best classifier selected by (42). A summary of the cost-sensitive boosting algorithm is presented in Algorithm 1. It is worth mentioning that it is fully compatible with AdaBoost, in the sense that it reduces to the latter when $C_1 = C_2 = 1$.

**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 binary classifiers $\{g_k(\mathbf{x})\}_{k=1}^K$, and number $M$ of weak learners in the final decision rule.
Initialization: Select uniformly distributed weights for each class
$w_i = \frac{1}{2|\mathcal{I}_+|}, \forall i \in \mathcal{I}_+$,    $w_i = \frac{1}{2|\mathcal{I}_-|}, \forall i \in \mathcal{I}_-$.
for $m = \{1, \ldots, M\}$ do
  for $k = \{1, \ldots, K\}$ do
    Compute (40)-(41) with $g(\mathbf{x}) = g_k(\mathbf{x})$ and solve (39) with respect to $\alpha$.
    Use (42) to compute the loss of the weak learner $(g_k(\mathbf{x}); \alpha_k)$
    .
  end for
  select the weak learner $(g_m(\mathbf{x}), \alpha_m)$ of smallest loss.
  update weights $w_i$ according to (38).
end for
Output: decision rule $h(x) = \text{sgn}[\sum_{m=1}^M \alpha_m g_m(x)]$.
```

## C. Cost-sensitive RealBoost

*Result 4: (Cost-sensitive RealBoost)* Consider the minimization of the empirical estimate of the expected loss of (32), 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 (13) 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 direction of largest descent at iteration $m$ has the form

$$G_m^{real}(\mathbf{x}) = G_{\phi_{k^*}}(\mathbf{x}) \tag{43}$$

where the optimal feature is determined by

$$k^* = \arg\min_k \sum_{i \in \mathcal{I}_+} w_i^{(m)} \exp(-C_1 G_{\phi_k}(\mathbf{x}_i)) + \sum_{i \in \mathcal{I}_-} w_i^{(m)} \exp(C_2 G_{\phi_k}(\mathbf{x}_i)) \tag{44}$$

with weights given by

$$w_i^{(m+1)} = \begin{cases} w_i^{(m)} e^{-C_1 G_m^{real}(\mathbf{x}_i)}, & i \in \mathcal{I}_+ \\ w_i^{(m)} e^{C_2 G_m^{real}(\mathbf{x}_i)}, & i \in \mathcal{I}_-, \end{cases} \tag{45}$$

and where

$$G_\phi(\mathbf{x}) = \left\{ \frac{1}{C_1 + C_2} \log \frac{P_{Y|\mathbf{X}}^{(w)}(1|\phi(\mathbf{x}))C_1}{P_{Y|\mathbf{X}}^{(w)}(-1|\phi(\mathbf{x}))C_2} \right\}. \tag{46}$$

$P_{Y|\mathbf{X}}^{(w)}(y|\phi(\mathbf{x})), y \in \{1, -1\}$ 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 $w_i^{(m)}$.

*Proof:* See appendix III. ■

The posterior probabilities $P_{Y|\mathbf{X}}^{(w)}(y|\phi_m(\mathbf{x})), y \in \{1, -1\}$ of (46) can be estimated with standard techniques [15]. For

**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 for each class
    $w_i = \frac{1}{2|\mathcal{I}_+|}, \forall i \in \mathcal{I}_+, \qquad w_i = \frac{1}{2|\mathcal{I}_-|}, \forall i \in \mathcal{I}_-.$
for $m = \{1, \ldots, M\}$ do
    for $k = \{1, \ldots, K\}$ do
        compute the gradient step $G_{\phi_k}(\mathbf{x})$ with (46).
    end for
    select the optimal direction according to (44) and set the weak learner $G_m^{real}(\mathbf{x})$ according to (43).
    update weights $w_i$ according to (45).
end for
Output: decision rule $h(\mathbf{x}) = \text{sgn}[\sum_{m=1}^{M} G_m^{real}(\mathbf{x})]$.
```

example, using weighted histograms of feature responses if the $\phi_k(\mathbf{x})$ are scalar features. Histogram regularization should be used to avoid empty histogram bins. A summary of cost-sensitive RealBoost is presented in Algorithm 2. This is fully compatible with RealBoost, reducing to it when $C_1 = C_2 = 1$, and has identical computational complexity.

### D. *Cost-sensitive LogitBoost*

Finally, we consider LogitBoost.

*Result 5: (Cost-sensitive LogitBoost)* Consider the minimization, by Newton's method, of the empirical estimate of the expected binomial loss of (33), 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 (13) with real-valued weak learners $G_m(\mathbf{x})$. Given a dictionary of features $\{\phi_1(\mathbf{x}), \ldots, \phi_K(\mathbf{x})\}$, and a predictor $\hat{f}^{(m)}(\mathbf{x})$, the Newton step at iteration $m$ has the form

$$G_m^{logit}(\mathbf{x}) = \frac{1}{2\gamma} G_{\phi_{k^*}}(\mathbf{x}) \tag{47}$$

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

$$(a_\phi, b_\phi) = \arg\min_{a_\phi, b_\phi} \sum_i w_i^{(m)} (z_i - a_\phi \phi(\mathbf{x}_i) - b_\phi)^2 \tag{48}$$

with

$$z_i = \frac{y'_i - p_c^{(m)}(\mathbf{x}_i)}{p_c^{(m)}(\mathbf{x}_i)(1 - p_c^{(m)}(\mathbf{x}_i))} \tag{49}$$

$$w_i^{(m)} = p^{(m)}(\mathbf{x}_i)(1 - p^{(m)}(\mathbf{x}_i)), \tag{50}$$

where $p_c^{(m)}(\mathbf{x})$ is the link function of (34), and $p^{(m)}(\mathbf{x})$ that of (16), with $f(\mathbf{x}) = \hat{f}^{(m)}(\mathbf{x})$. The optimal feature is determined by

$$k^* = \arg\min_k \sum_i w_i^{(m)} (z_i - a_{\phi_k} \phi_k(\mathbf{x}_i) - b_{\phi_k})^2. \tag{51}$$

*Proof:* See appendix IV. ■

A summary of cost-sensitive LogitBoost is presented in Algorithm 3. The algorithm is fully compatible with LogitBoost, in the sense that it reduces to the latter when $C_1 = C_2 = 1$ and has identical computational complexity. It is instructive to compare it with Platt's method for posterior probability calibration [41], [21], [43]. 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 (34). The $\gamma$ and $\eta$ parameters are determined by gradient descent with respect to the binomial loss of (33), 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 $\gamma$ and $\eta$ that best fit a cross-validation data set. On the other hand, cost-sensitive LogitBoost uses the calibrated link function throughout the 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 boosting remains on $\mathcal{N}(\mathcal{C})$, not on $\mathcal{N}(\mathcal{C}_T)$. We next show that all proposed cost-sensitive boosting algorithms solve this problem.

**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 $\hat{f}^{(1)}(\mathbf{x}) = 0$.
for $m = \{1, \ldots, M\}$ do
    compute the working responses $z_i^{(m)}$ as in (49) and weights $w_i^{(m)}$ as in (50).
    for $k = \{1, \ldots, K\}$ do
        compute the solution to the least squares problem of (48),
            $a_{\phi_k} = \frac{\langle 1 \rangle_w \cdot \langle \phi_k(\mathbf{x}_i) z_i \rangle_w - \langle \phi_k(\mathbf{x}_i) \rangle_w \cdot \langle z_i \rangle_w}{\langle 1 \rangle_w \cdot \langle \phi_k^2(\mathbf{x}_i) \rangle_w - \langle \phi_k(\mathbf{x}_i) \rangle_w^2}$    (52)
            $b_{\phi_k} = \frac{\langle \phi_k(\mathbf{x}_i)^2 \rangle_w \cdot \langle z_i \rangle_w - \langle \phi_k(\mathbf{x}_i) \rangle_w \cdot \langle \phi_k(\mathbf{x}_i) z_i \rangle_w}{\langle 1 \rangle_w \cdot \langle \phi_k^2(\mathbf{x}_i) \rangle_w - \langle \phi_k(\mathbf{x}_i) \rangle_w^2}$    (53)
        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 (51) and set the weak learner $G_m^{logit}(\mathbf{x})$ according to (47).
    set $\hat{f}^{(m+1)}(\mathbf{x}) = \hat{f}^{(m)}(\mathbf{x}) + G_m^{logit}(\mathbf{x})$.
end for
Output: decision rule $h(\mathbf{x}) = \text{sgn}[\sum_{m=1}^{M} G_m^{logit}(\mathbf{x})]$.
```

### E. *Cost-sensitive weights*

We have mentioned above that cost-sensitive boosting algorithms should

- converge asymptotically to the optimal predictor of (3),
- emphasize a neighborhood of the cost-sensitive boundary $\mathcal{N}(\mathcal{C}_T)$, when learning from finite samples.

The first condition is guaranteed by the losses of (32) and (33). To investigate the second we consider the weight mechanisms of the three algorithms. Let $\hat{f}^{(m)}$ be the boosted predictor after $m$ iterations. For both cost-sensitive Ada and RealBoost, a simple recursion shows that, for correctly classified points,

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

where $Q_i = C_1$ if $i \in \mathcal{I}_+$ and $Q_i = C_2$ otherwise. 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 (16), at $\hat{f}^{(m)}(\mathbf{x}_i) = 0$. As in the cost-insensitive case,

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

These equations are qualitatively identical to (29) and (30). The only difference is that, as $\hat{f}^{(m)}(\mathbf{x})$ converges to (35), its zero-level set is the cost-sensitive boundary $\mathcal{C}_T$. Hence, points along $\mathcal{C}_T$ have unitary importance, while the importance of the remaining points decreases exponentially with their distance to $\mathcal{C}_T$. This implies that all cost-sensitive boosting algorithms shift the boosting emphasis from $\mathcal{N}(\mathcal{C})$ to a soft neighborhood of the cost-sensitive boundary $\mathcal{N}(\mathcal{C}_T)$.

## V. Experimental evaluation

To evaluate the proposed algorithms we started with a synthetic problem, of known BDR, which allows explicit comparison to the optimal cost-sensitive detector. Comparisons against previous methods were then performed with data from the UCI repository and a large face detection dataset. Finally, we compared cost-sensitive boosting and a number of state-of-the-art solutions to the computer vision problem of car detection. Unless otherwise noted, all boosting algorithms used decision stumps as weak learners, and all parameters were selected by cross-validation. The data was divided into train and test sets, and the training set split into five folds, four of which were used for training and one for validation. The latter served to tune parameters (cost parameters and classifier threshold) so as to minimize a classification cost. For car detection, this was the equal error rate (EER), the quantity usually reported for the dataset adopted (UIUC). Elsewhere, it was the number of false positives at a given detection rate. In this case, cross validation was repeated for detection rates between 80% and 95%, with increments of 2.5%. Cross validation was applied to all parameters of all methods. For example, support vector machines (SVMs) required validation of kernel bandwidth, margin/outliers trade-off parameter, and threshold.

### A. *Synthetic datasets*

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

To test the accuracy of the cost-sensitive detectors we relied on the following observations. First, given a cost structure $(C_1, C_2)$, a necessary condition for the optimality of the boosted detector is that the asymmetric logistic transform of (35) holds along the cost-sensitive boundary, i.e. $x^* = f^{-1}(0)$ where $f(x)$ is the optimal predictor of (35) and $x^*$ the zero-crossing of the boosted predictor. Second, from (35), this is equivalent to

$$P_{Y|X}(1|x^*) = \frac{C_2}{C_1 + C_2}. \tag{54}$$

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

$$P_{Y|X}(1|x) = \frac{1}{1 + e^{-2x}}, \tag{55}$$

and (54) implies that $x^* = -T/2$, with $T$ as in (7). It is thus possible to evaluate the accuracy of the 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 are shown on Figure 2 for detectors learned with five iterations of cost-sensitive Ada, Real, and LogitBoost. In all cases $C_2 = 1$ and $C_1$ was varied over a range of values. Both Real and LogitBoost produce near optimal cost-sensitive detectors, but the restriction of the predictor to a combination of binary functions creates difficulties for AdaBoost.

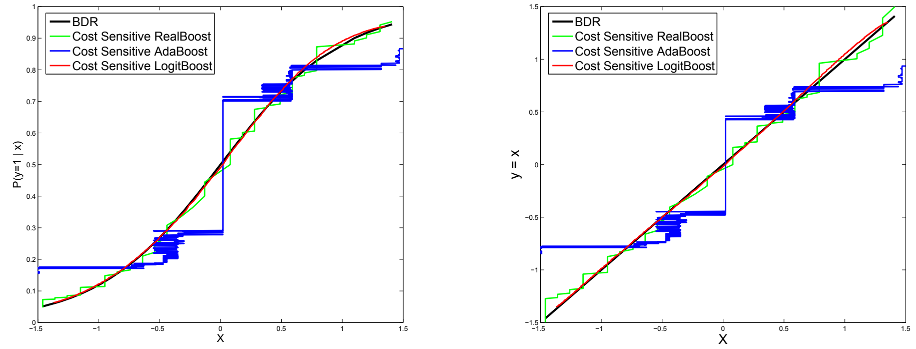
### B. *Real datasets*

To evaluate performance on real data, various algorithms were compared on datasets from the UCI repository [27], and the face detection problem [28].

*1) UCI:* Ten data sets were selected - Pima-diabetes, breast cancer diagnostic, breast cancer prognostic, original Wisconsin breast cancer, liver disorder, sonar, echo-cardiogram, Cleveland heart disease, tic-tac-toe, and Haberman's survival. In all cases, data points with missing values were ignored. The multi-class Cleveland heart disease data was converted to the problem of detecting presence (classes 1, 2, 3, 4) vs. absence (value 0) of disease. We compared the performance of the proposed cost-sensitive boosting algorithms (CS-Ada, CS-Real, and CS-Log), their previously available counterparts[2] (CSB0, CSB1, CSB2, AdaC2, AdaC3, and AdaCost), and the combination of standard AdaBoost, RealBoost, or LogitBoost with Platt calibration [41]. For completeness, we have also tested SVMs with linear and Gaussian kernels, and Platt calibration. In all cases, one point was first removed from the dataset and reserved for testing. The classifier was trained on the remaining data so as to meet a target detection rate (all parameters cross-validated), and used to classify this test point. The process was iterated, each point taking a turn as test set, and the total number of classification errors recorded.

Table I presents the average number of errors for each classifier and dataset, across the five detection rates considered. To simplify the comparison, the table includes two overall statistics. The first is the number of datasets in which each cost-sensitive boosting algorithm achieved lower error than *all*

[2] Note that, because Asymmetric-AdaBoost [3] and CSB2 [25] are identical, we do not report results for the former.



(a)

(b)

Fig. 2. a) True posterior class probability $P_{Y|X}(y=1|x)$, as a function of $x$, and estimates by cost-sensitive Ada, Logit and RealBoost. b) Comparison of the plots $(x^*, -\frac{T}{2})$ and $(x^*, x^*)$.

TABLE I

AVERAGE NUMBER OF ERRORS FOR EACH CLASSIFIER AND UCI DATASET, ACROSS FIVE DETECTION RATES. THE LOWEST AVERAGE ERROR ACHIEVED ON EACH DATASET IS SHOWN IN BOLDFACE. RANK INDICATES THE AVERAGE RANKING OF THE CLASSIFIER ACROSS DATASETS, AND #WINS IS THE NUMBER OF DATASETS ON WHICH A COST SENSITIVE BOOSTING ALGORITHM ACHIEVED LOWER ERROR THAN ALL PREVIOUS BOOSTING METHODS.

| | pima | liver | wdbc | sonar | wpbc | Wisc | echo | heart | tic | survival | Rank | #w |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CS-Ada | **205.6** | **143** | 26.4 | 52.2 | 128.4 | 37.2 | 44 | **61.4** | 433.8 | 172.8 | 4.84 | 6 |
| CS-Log | 248.6 | 146.4 | **25.8** | 67 | **85.6** | 35 | **40** | 74.6 | 463.2 | 178.6 | 5.35 | 5 |
| CS-Real | 256.2 | 144 | 32.4 | 56.8 | 101.2 | 35.4 | 54 | 69.6 | **110.4** | **96.6** | 5.35 | 4 |
| CSB0 | 241.2 | 161 | 43.8 | 66.6 | 140.2 | 40.8 | 46 | 89 | 329.2 | 101.8 | 8.2 | |
| CSB1 | 384 | 175.8 | 30.8 | 65.8 | 121.8 | 89 | 65 | 100.8 | 415 | 188.6 | 10.95 | |
| CSB2 | 223 | 143.5 | 31 | 42.6 | 118.8 | 45.8 | 61 | 88.8 | 317.4 | 145.2 | 6.45 | |
| AdaC2 | 249.4 | 162.2 | 36 | 56 | 111.4 | 42.4 | 53 | 64.2 | 180 | 131.2 | 6.65 | |
| AdaC3 | 250.4 | 169 | 29.6 | 48.2 | 113.8 | 39.6 | 57 | 102.6 | 258.6 | 205.2 | 8.4 | |
| ADaCost | 365 | 170 | 42.2 | 88 | 111 | 43.4 | 65 | 110 | 366 | 189 | 11.35 | |
| SVM-L | 415.2 | 153.2 | 32.2 | 74 | 111.4 | 33 | 43 | 66.8 | 550.2 | 181.4 | 7.75 | |
| SVM-G | 390 | 161.2 | 31 | **35.8** | 122 | **30.6** | 44 | 153.6 | 625 | 153.6 | 8.1 | |
| Ada | 244.2 | 168 | 28.4 | 57.4 | 132.8 | 37.6 | 48 | 73.8 | 465.6 | 174.6 | 8.1 | |
| Real | 263.8 | 154.6 | 32.4 | 67.2 | 104.8 | 35 | 47 | 67.6 | 119 | 152 | 6.4 | |
| Logit | 263 | 154 | 26 | 68 | 120.6 | 33.2 | 41 | 68.2 | 545.8 | 184.6 | 7.1 | |

prior cost-sensitive boosting algorithms. This is referred to as the number of *wins*. The second is the classifier ranking of [44]: the algorithms were first ranked on each dataset (rank one for lowest error) and the average rank of each classifier, across datasets, is reported. The three cost-sensitive boosting algorithms achieve the three smallest average ranks. From this point of view, only CSB2, AdaC2, and RealBoost with Platt calibration can be seen as competitive with CS-Ada, CS-Real, and CS-Logit. But the worse of the latter has an average rank 15% smaller than the best of the former.

The average ranks, across datasets, for the five detection rates considered, are presented in Table II. While the overall conclusions are the same, note that AdaBoost, RealBoost, and LogitBoost tend to rank lower (relative to their cost-sensitive counterparts) as the detection rate increases. This follows from their cost-insensitivity (despite Platt calibration and threshold tuning). On the other hand, the ranks of CS-AdaBoost, CS-LogitBoost and CS-RealBoost improve relatively. For example, while the difference in rank between AdaBoost and CS-AdaBoost is $7.25 - 6.1 = 1.15$ at 85% detection rate, it grows to $9.5 - 5.2 = 4.3$ at 95%. This confirms our previous claim that threshold manipulation produces inferior results as the distance between cost-sensitive and insensitive boundaries increases.

To investigate the impact of the choice of weak learners in these conclusions, we performed the same experiments with decision trees [45] as weak learners. Following [18], we used four terminal node trees. To enable a comparison to the results achieved with decision stump methods, we limited the total number of features to 50. Since each tree contains three features, this implies $50/3 \approx 17$ weak learners per classifier. The implementations of CS-AdaBoost and CS-RealBoost relied on (42) and (44), respectively, as tree splitting criteria. All other aspects were identical to [18]. CS-Logit was not considered since it would require the implementation of regression trees, instead of classification trees that we have used. Tables III and IV compares the results obtained for the various cost sensitive boosting algorithms, datasets, and detection rates. For completeness, we also implemented a detector based on Random Forests [46] of 17 four terminal node trees and Platt calibration, which did not prove competitive with the proposed algorithms. There is no significant qualitative difference between the results of tables I-II, and III-IV, suggesting that the proposed cost-sensitive boosting algorithms

TABLE II

AVERAGE CLASSIFIER RANK, ACROSS TEN UCI DATASETS, FOR FIVE DETECTION RATES.

| Det% | CSAda | Ada | CSLog | Log | CSReal | Real | CSB0 | CSB1 | CSB2 | AdaC2 | AdaC3 | SVML | SVMG |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 85% | 6.1 | 7.25 | 5.6 | 6.65 | **5.3** | 5.75 | 8.85 | 10.35 | 6.7 | 7.8 | 7.85 | 8.15 | 7.45 |
| 87.5% | **5.2** | 7.2 | 5.9 | 6.45 | 5.5 | 6.25 | 8.5 | 10.7 | 6.25 | 6.9 | 8.7 | 8.05 | 7.75 |
| 90% | 5.45 | 7.55 | 5.65 | 7.5 | **4.3** | 6.6 | 7.9 | 12.1 | 6.9 | 6.6 | 8.55 | 7.8 | 7.7 |
| 92.5% | 5.2 | 7.9 | 5.8 | 7.55 | **4.95** | 6.6 | 8.0 | 11.6 | 6.25 | 6.15 | 8.3 | 7.8 | 8.05 |
| 95% | 5.2 | 9.5 | 5.25 | 7.85 | **5.05** | 5.2 | 7.95 | 10.65 | 7.25 | 6.0 | 8.15 | 8.55 | 7.9 |

have superior performance independently of the weak learner adopted. In summary, with either decision stumps or trees, the proposed algorithms outperform the state-of-the-art in cost-sensitive boosting.

*2) Face detection:* UCI datasets are sometimes criticized as too small, or low-dimensional, to allow meaningful conclusions. We repeated the comparisons above on the real, large-scale, large-dimensional problem of face detection. This problem is also becoming an important area of application for cost-sensitive boosting, given the widespread use of boosting for the design of detector cascades [28]. We emphasize, however, that the goal here is not to compete with algorithms for cascade design, but simply compare cost-sensitive boosting algorithms. While cost-sensitive boosting can be used to design cascade nodes, the overall cascade design requires the solution of additional problems, such as determining the optimal cascade architecture (number of nodes and computation per node), whose solution is beyond the scope of this work. Furthermore, cascade (or face detector) design frequently involves steps, such as bootstrapping (automated collection of negative examples) or manual tuning of classifier parameters, that make objective comparisons of algorithms quite difficult. Our goal is simply to exploit the high-dimensionality of the face detection data ($50,000$ features) and the availability of a large dataset to compare cost-sensitive boosting algorithms in a realistic scenario.

These experiments were based on the experimental protocol of [28]: a face database of 9832 positive and 9832 negative examples, and weak learners based on a combination of decision stumps and Haar wavelet features. 6000 examples were used per class for training, and the remaining 3832 for testing, and all boosting algorithms were trained for 100 iterations. Given the computational complexity of these experiments, we restricted the comparison to CS-Ada and the previously proposed cost-sensitive boosting algorithms (CSB0, CSB1, CSB2, AdaC2, AdaC3). All classifier parameters were tunned with the cross validation procedure described at the start of this section. The detection rate and number of false positives of each method are shown in Table V, for each of the cross-validation detection rates. The number above each pair of columns is the target detection rate (used for cross-validation), while the detection rate and number of false positives measured on the test set are shown in the columns themselves. Note that all methods maintain a test detection rate very similar to the target, CS-Ada achieves the best performance, and only that of CSB2 is comparable. These results illustrate the importance of choosing the confidence $\alpha$ optimally, at each iteration. Methods that ignore $\alpha$ in the weight update rule (CSB0 and CSB1) have extremely poor performance. Methods that update $\alpha$ but are not asymptotically optimal (AdaC2, AdaC3) perform worse than CSB2, which relies on the $\alpha$ updates of AdaBoost.

## C. Car detection

We finish by investigating how the simple application of the proposed cost-sensitive boosting algorithms fare against state-of-the-art object detection algorithms in computer vision. For this, we selected the problem of car detection on the popular UIUC Car dataset [29]. This is a dataset that precisely defines all variables of the experimental evaluation, e.g. a rigorous procedure for counting detections and false positives (which is not the case in [28]), and allows rigorous comparisons to a large literature. It is also a challenging data set, in the sense that only 500 positive and 500 negative examples are available for training. Unfortunately, not all results in the literature comply with the original protocol. For example classifiers are sometimes trained with much larger datasets, and significant variations in error rate can be achieved by optimizing the post-processing procedure (non-maximum suppression) to eliminate the false-positives that always occur in the neighborhood of a correct detection. Hence, even for this thoroughly standardized dataset, assessments of detector performance based on comparison of published results have to be taken with caution. We will discuss these problems in detail below.

We compared CS-Ada to both regular AdaBoost and a number of methods previously proposed in the literature. All images were re-scaled to 20x50 pixels, and detection based on a pool of $162,000$ Haar features [28]. CS-Ada was used to learn 300 feature detectors, with the cross-validation procedure described at the start of this section. As is advised for this dataset, the resulting detectors were tested with the neighborhood suppression algorithm proposed in [29] and performance quantified by the EER. For completeness, we also indicate the maximum F-measure and corresponding detection and false-positive rates, although these statistics are not always reported in the literature. The F-measure is the weighted harmonic mean of precision and recall, summarizing the trade-off between these two statistics at each point of the ROC curve. The maximum F-measure, and the reported detection and false-positive rates, are those observed at the point where this trade-off is optimal. We limited the comparison to the single scale test set, with the results of Table VI.

The left side of the table presents results of methods that rigorously follow the experimental set up of [29]. Agarwal and AdaBoost classify rectangular image patches and can be seen as template classifiers. However, because they rely on highly

TABLE III

AVERAGE NUMBER OF ERRORS FOR EACH CLASSIFIER AND UCI DATASET, ACROSS FIVE DETECTION RATES USING DECISION TREES. THE LOWEST AVERAGE ERROR ACHIEVED ON EACH DATASET IS SHOWN IN BOLDFACE. RANK INDICATES THE AVERAGE RANKING OF THE CLASSIFIER ACROSS DATASETS, AND #WINS IS THE NUMBER OF DATASETS ON WHICH A COST SENSITIVE BOOSTING ALGORITHM ACHIEVED LOWER ERROR THAN ALL PREVIOUS BOOSTING METHODS.

| | pima | liver | wdbc | sonar | wpbc | Wisc | echo | heart | tic | survival | Rank | #w |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CS-Ada | **230.6** | **129.4** | **42.2** | 63 | **95** | 37 | **46** | **49.4** | 343.8 | 129.6 | 2.2 | 6 |
| CS-Real | 252.2 | 148 | 47.2 | **62.6** | 95.4 | 33.2 | 51 | 80 | 297.8 | 145 | 3.2 | 3 |
| CSB0 | 252 | 178 | 42.6 | 91.6 | 123.6 | 46.6 | 57 | 74.4 | 238.2 | **109** | 4.4 | |
| CSB1 | 313.4 | 176 | 50.4 | 88.6 | 112.8 | 40 | 62 | 138.4 | 490.2 | 161.6 | 6.4 | |
| CSB2 | 299.8 | 162.8 | 57.8 | 83 | 117 | 32 | 50 | 103 | 342.2 | 131.2 | 4.7 | |
| AdaC2 | 278.4 | 151.4 | 49.4 | 81 | 114.8 | 37.4 | 64 | 85.8 | 185.2 | 111.8 | 4.2 | |
| AdaC3 | 272 | 163 | 43 | 82.6 | 118 | **26.4** | 47 | 82.4 | **169.8** | 121.8 | 3.4 | |
| RForest | 364.2 | 189 | 69.6 | 102.8 | 124.4 | 37.8 | 60 | 117.6 | 546 | 186 | 7.5 | |

TABLE IV

AVERAGE CLASSIFIER RANK, ACROSS TEN UCI DATASETS, FOR FIVE DETECTION RATES USING DECISION TREES.

| | CS-Ada | CS-Real | CSB0 | CSB1 | CSB2 | AdaC2 | AdaC3 | RForest |
|---|---|---|---|---|---|---|---|---|
| 85% | 2.3 | 2.55 | 4.85 | 6.25 | 4.2 | 4.6 | 4.0 | 7.25 |
| 87.5% | 2.4 | 3.05 | 4.7 | 6.35 | 3.95 | 4.3 | 4.05 | 7.2 |
| 90% | 2.65 | 3.6 | 4.05 | 6.5 | 4.9 | 4.4 | 2.7 | 7.2 |
| 92.5% | 1.85 | 3.55 | 4.3 | 6.05 | 5.1 | 4.4 | 3.25 | 7.5 |
| 95% | 2.2 | 4.55 | 4.25 | 6.0 | 4.8 | 3.9 | 3.1 | 7.2 |

TABLE V

FACE DETECTION RATE AND NUMBER OF FALSE POSITIVES AT VARIOUS CROSS-VALIDATION DETECTION RATES.

| | 85% | | 87.5% | | 90% | | 92.5% | | 95% | |
|---|---|---|---|---|---|---|---|---|---|---|
| Method | Det% | #FP | Det% | #FP | Det% | #FP | Det% | #FP | Det% | #FP |
| CS-Ada | 85.2 | **22** | 87.44 | **28** | 90.37 | **34** | 92.64 | **52** | 95.25 | **113** |
| CSB2 | 85.2 | 24 | 87.7 | 33 | 90.29 | 53 | 92.82 | 78 | 95.14 | 152 |
| AdaC2 | 85.54 | 137 | 87.91 | 175 | 90.52 | 239 | 92.77 | 315 | 95.22 | 437 |
| AdaC3 | 85.93 | 202 | 88.39 | 340 | 91.96 | 409 | 93.21 | 412 | 95.25 | 538 |
| CSB0 | 86.01 | 276 | 88.12 | 325 | 90.63 | 418 | 92.95 | 592 | 97.57 | 933 |
| CSB1 | 85.12 | 689 | 87.73 | 803 | 90.29 | 967 | 92.72 | 1142 | 95.12 | 1429 |

TABLE VI

PERFORMANCE ON UIUC CAR DATASET, SINGLE SCALE TEST SET. LEFT SIDE OF THE TABLE PRESENTS METHODS THAT RIGOROUSLY FOLLOW THE EXPERIMENTAL SET UP OF [29] † : USE VARIATIONS OF POST-PROCESSING. ◇ : USE EXTENDED TRAINING SET. N.R: NOT REPORTED.

| Method | EER | F-Measure | Det% | #FP | Method | EER | F-Measure | Det% | #FP |
|---|---|---|---|---|---|---|---|---|---|
| CS-AdaBoost | **93.5%** | **93.50%** | 93.5% | 13 | Mutch$^{\dagger}$ [47] | 99.94% | N.R | N.R | N.R |
| Shotton [48] | 92.8% | N.R | N.R | N.R | Wu$^{\diamond}$ [49] | 97.5% | N.R | N.R | N.R |
| Bar-Hillel [50] | 92.4% | N.R | N.R | N.R | Leibe+MDL$^{\dagger\diamond}$ [51] | 97.5% | N.R | N.R | N.R |
| Leibe[51] | 91% | N.R | N.R | N.R | Schneidermann$^{\diamond}$ [52] | 97% | N.R | N.R | N.R |
| AdaBoost | 90% | 90.27% | 90.5% | 20 | CS-AdaBoost$^{\dagger}$ | 95.5% | 95.26% | 95.5% | 9 |
| Fergus [53] | 88.5% | N.R | N.R | N.R | Grabner$^{\dagger\diamond}$ [54] | 93% | 93.5% | N.R | N.R |
| Agarwal [29] | 79% | 77.08% | 76.5% | 44 | AdaBoost$^{\dagger}$ | 92.5% | 92.23% | 92.5% | 15 |

localized features, they can also be seen as either learning a rough object segmentation (object outline within the patch), or a representation of the object as a spatial configuration of features. Both ideas have been explored in detail in the literature, with classifiers that *explicitly* segment the object to detect [51], [48], [55], [49], [56], learn configurations of its parts [53], [50] or both [48], [49]. Training such representations is manually intensive (e.g. requires precisely segmented examples) and the resulting decision rules have far more computation than those of the AdaBoost/Haar combination. Yet, at least when the protocol of [29] is followed precisely (left half of table), there is little evidence that they have benefits. On the contrary, simply replacing AdaBoost by CS-AdaBoost produces the best overall performance.

There are a number of ways in which performance can be improved by relaxing the experimental protocol. One popular modification is to improve the post-processing of the detector output, so as to eliminate spatially adjacent detections (non-maximum suppression). Methods that use variations of post-processing are identified in the right-side of the table with a †. These variations can lead to a dramatic performance increase. For example, Leibe et al. report an improvement from 91%

to 97% EER by introducing their MDL procedure [51]. For the classifiers that we implemented, the simple extension of the suppression window from 71 to 140 pixels (similar to [47] which used 111 pixels for their detector) led to an improvement from 90% to 92.5% for Adaboost and from 93.5% to 95.5% for CS-Adaboost. We have not attempted to optimize performance any further in this way. Another popular performance enhancement strategy is to rely on an extended training set. Variations range from adopting completely different sets of positive and negative training examples [51], to extended sets of positives and negatives (the dataset of [29] plus additional data) [49], to the same set of positives but an extended set of negatives [54], [52]. Methods that rely on such extensions are identified by a ◇ in the table. Given the reduced size of the UIUC car dataset, any of these extensions is likely to improve performance significantly. Unfortunately, they also make it virtually impossible to compare the underlying classification algorithms in an objective manner.

We emphasize that our claim here is not that the combination of CS-AdaBoost and Haar features is the ultimate solution for object detection. In fact, two of the top performing algorithms in each of the sides of Table VI - Bar-Hillel [50] and Wu [49] - rely on the combination of boosting and other image representations (weak learners). It is likely that they could also benefit from the cost-sensitive extensions proposed in this work. What our results show is that 1) for object detection, CS-AdaBoost can lead to substantial performance improvements over AdaBoost, and 2) the combination of CS-AdaBoost and Haar wavelets is at least competitive with the state-of-the-art methods in the literature. This is not insignificant, since most of these competitors involve special purpose features, segmentation, or other vision operations which cost-sensitive boosting does not have access to, and are expensive. On the other hand, the architecture used with cost-sensitive boosting is completely generic, e.g. identical to that used by [28] for face detection.

## VI. Conclusion

We have presented a novel framework for the design of cost-sensitive boosting algorithms. The framework is based on the identification of two necessary conditions for the design of optimal cost-sensitive learning algorithms: that 1) expected losses must be minimized by optimal cost-sensitive decision rules, and 2) empirical loss minimization must emphasize the neighborhood of the target cost-sensitive boundary. These enable the derivation of cost-sensitive boosting losses which (similarly to the original cost-insensitive ones) can be minimized by gradient descent, in the functional space of convex combinations of weak learners, to produce boosting algorithms. The proposed framework was used to derive cost-sensitive extensions of AdaBoost, RealBoost and LogitBoost. Experimental evidence, derived from a synthetic problem, standard data sets, and the computer vision problems of face and car 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) as well as the popular combination of large margin classifiers and probability calibration. Cost-sensitive boosting was shown to consistently outperform all other methods tested. In the future, we plan to investigate the application of the cost-sensitive boosting algorithms now introduced to the fully automated design of optimal object detection cascades.

## Appendix I
## Proof of Lemma 2

To find the minimum of the cost-sensitive extension of the exponential loss of (32) 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

$$\begin{aligned} \frac{\partial l_e(\mathbf{x})}{\partial f(\mathbf{x})} =& -C_1 P_{Y|\mathbf{x}}(1|\mathbf{x})e^{-C_1 f(\mathbf{x})} + C_2 P_{Y|\mathbf{x}}(-1|\mathbf{x})e^{C_2 f(\mathbf{x})} \\ =& \;\; 0 \end{aligned} \tag{56}$$

it follows that

$$\frac{C_1 P_{Y|\mathbf{x}}(1|\mathbf{x})}{C_2 P_{Y|\mathbf{x}}(-1|\mathbf{x})} = e^{(C_1+C_2)f(\mathbf{x})} \tag{57}$$

and

$$f(\mathbf{x}) = \frac{1}{C_1+C_2}\log\frac{P_{Y|\mathbf{x}}(1|\mathbf{x})C_1}{P_{Y|\mathbf{x}}(-1|\mathbf{x})C_2}. \tag{58}$$

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

$$\frac{\partial l_b(\mathbf{x})}{\partial p_c(\mathbf{x})} = -P_{Y|\mathbf{x}}(1|\mathbf{x})\frac{1}{p_c(\mathbf{x})} + P_{Y|\mathbf{x}}(0|\mathbf{x})\frac{1}{1-p_c(\mathbf{x})} = 0 \tag{59}$$

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 (34), 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.

## Appendix II
## Proof of Result 3

From (32) the cost function can be written as

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

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

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

with

$$
w(\mathbf{x},1) = \exp(-C_1 f(\mathbf{x})) \qquad w(\mathbf{x},-1) = \exp(C_2 f(\mathbf{x})).
$$

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

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

Noting that

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

it follows that

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

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,

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

with the empirical estimates $\mathcal{T}_+$, $\mathcal{T}_-$, $b$ and $d$ of (40) - (41). Given $g(\mathbf{x})$, and setting the derivative with respect to $\alpha$ to zero

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

the optimal step size $\alpha$ is the solution of

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

## Appendix III
## Proof of Result 4

From (32) the cost function can be written as

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

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

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

with

$$
w(\mathbf{x},1) = \exp(-C_1 f(\mathbf{x})) \tag{60}
$$

and

$$
w(\mathbf{x},-1) = \exp(C_2 f(\mathbf{x})). \tag{61}
$$

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

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

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

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

The optimal feature $\phi^*$ is the one of smallest minimum cost

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

Once $G^{real}_m(\mathbf{x})$ is found, the weights are updated so as to comply with (60) and (61), i.e.

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

and

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

## Appendix IV
## Proof of Result 5

Rewriting the negative log-likelihood as

$$\begin{aligned}
& l_b[y', \hat{f}^{(m)}(\mathbf{x})] = \\
& -E_{\mathbf{X},Y}\left[y' \log \frac{p_c(\mathbf{x})}{1 - p_c(\mathbf{x})} + \log(1 - p_c(\mathbf{x}))\right]
\end{aligned}$$

and using (34), it follows that

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

This loss is minimized by maximizing the conditional expectation

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

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

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

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

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

and Hessian

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

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

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

which is the weighted least squares regression of $z_i$ to $\mathbf{x}_i$ using weights $w_i$, as given by (49) and (50). The optimal feature is the one of smallest regression error.

## References

[1] S. Viaene, R. A. Derrig, and G. Dedene, "Cost-sensitive learning and decision making for massachusetts pip claim fraud data," *International Journal of Intelligent Systems*, vol. 19, pp. 1197–1215, 2004.

[2] A. Vlahou, J. O. Schorge, B. W. Gregory, and R. L. Coleman, "Diagnosis of ovarian cancer using decision tree classification of mass spectral data," *Journal of Biomedicine and Biotechnology*, vol. 2003, no. 5, p. 308314, 2003.

[3] P. Viola and M. Jones, "Fast and robust classification using asymmetric adaboost and a detector cascade," in *Advances in Neural Information Processing System*, vol. 2, 2002, pp. 1311–1318.

[4] M. Turk and A. Pentland, "Eigenfaces for Recognition," *Journal of Cognitive Neuroscience*, vol. 3, 1991.

[5] K. Sung and T. Poggio, "Example Based Learning for View-Based Human Face Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 1, pp. 39–51, January 1998.

[6] H. A. Rowley, S. Baluja, and T. Kanade, "Neural network-based face detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, pp. 23–38, 1998.

[7] M. Oren, C. Papageorgiou, P. Sinha, E. Osuna, and T. Poggio, “Pedestrian Detection Using Wavelet Templates,” in *IEEE Conference in Pattern Recognition and Computer Vision*, 1997.

[8] H. Schneiderman and T. Kanade, “Object Detection Using the Statistics of Parts,” *International Journal of Computer Vision*, vol. 56, no. 3, pp. 151–177, 2004.

[9] Y. Amit and D. Geman, “Shape Quantization and Recognition with Randomized Trees,” *Neural Computation*, vol. 9, pp. 1545–1588, 1997.

[10] D. Roth, M. Yang, and N. Ahuja, “Learning to Recognize Three-Dimensional Objects,” *Neural Computation*, vol. 14, pp. 1071–1103, 2002.

[11] C. Elkan, “The foundations of cost-sensitive learning,” in *Seventeenth Intrnl. Joint Conference on Artificial Intelligence*, 2001, pp. 973–978.

[12] B. Zadrozny and C. Elkan, “Learning and making decisions when costs and probabilities are both unknown.” in *7th International Conference on Knowledge Discovery and Data Mining*, 2001, pp. 203–213.

[13] P. Domingos, “Metacost: a general method for making classifiers cost-sensitive,” in *Knowledge Discovery and Data Mining*, 1999, pp. 155–164.

[14] A. Wald, “Contributions to the theory of statistical estimation and testing hypotheses.” *The Annals of Mathematical Statistics*, vol. 10, pp. 299–326, 1939.

[15] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. New York: John Wiley Sons Inc, 2001.

[16] Y. Freund and R. Schapire, “A decision-theoretic generalization of on-line learning and an application to boosting,” *Journal of Computer and System Sciences*, vol. 55, no. 1, part 2, pp. 119–139, 1997.

[17] L. Breiman, “Arcing classifiers,” *The Annals of Statistics*, vol. 26, no. 3, pp. 801–849, 1998.

[18] J. Friedman, T. Hastie, and R. Tibshirani, “Additive logistic regression: A statistical view of boosting,” *The Annals of Statistics*, vol. 38, pp. 337–374, 2000.

[19] D. Mease, A. J. Wyner, and A. Buja, “Boosted classification trees and class probability/quantile estimation,” *The Journal of Machine Learning Research*, vol. 8, pp. 409–439, 2007.

[20] D. Mease and A. J. Wyner, “Evidence contrary to the statistical view of boosting,” *Journal of Machine Learning Research*, vol. 9, pp. 131–156, 2008.

[21] A. Niculescu-Mizil and R. Caruana, “Obtaining calibrated probabilities from boosting,” in *Proc. 21st Conference on Uncertainty in Artificial Intelligence (UAI ’05)*. AUAI Press, 2005, pp. 413–420.

[22] W. Jiang, “Process consistency for adaboost,” *The Annals of Statistics*, vol. 32, pp. 13–29, 2004.

[23] R. E. Schapire and Y. Singer, “Improved boosting using confidence-rated predictions,” *Machine Learning*, vol. 37, no. 3, pp. 297–336, 1999.

[24] W. Fan, S. Stolfo, J. Zhang, and P. Chan, “Adacost: Misclassification cost-sensitive boosting,” in *Proc. of 6th International Conf. on Machine Learning*, 1999, pp. 97–105.

[25] K. M. Ting, “A comparative study of cost-sensitive boosting algorithms,” in *Proc. 17th International Conf. on Machine Learning*, 2000, pp. 983–990.

[26] Y. Sun, A. K. C. Wong, and Y. Wang, “Parameter inference of cost-sensitive boosting algorithms,” in *Machine Learning and Data Mining in Pattern Recognition,4th International Conference*, 2005, pp. 21–30.

[27] D. Newman, S. Hettich, C. Blake, and C. Merz, “UCI repository of machine learning databases,” 1998. [Online]. Available: http://www.ics.uci.edu/∼mlearn/MLRepository.html

[28] P. A. Viola and M. J. Jones, “Robust real-time face detection,” *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2004.

[29] S. Agarwal, A. Awan, and D. Roth, “Learning to detect objects in images via a sparse, part-based representation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 11, pp. 1475–1490, 2004.

[30] J. Neyman and E. S. Pearson, “On the problem of the most efficient tests of statistical hypotheses.” *Philosophical Transactions of the Royal Society of London*, vol. 231, pp. 289–337, 1933.

[31] H. L. V. Tree, *Detection, Estimation and Modulation Theory*. New York: John Wiley and Sons Inc, 1968.

[32] D. Green and J. Swets, *Signal detection theory and psychophysics*. New York: John Wiley and Sons Inc., 1966.

[33] S. Agarwal, T. Graepel, R. Herbrich, S. Har-Peled, and D. Roth, “Generalization bounds for the area under the roc curve,” *The Journal of Machine Learning Research*, vol. 6, pp. 393–425, 2005.

[34] V. N. Vapnik, *Statistical Learning Theory*. New York: John Wiley Sons Inc, 1998.

[35] R. E. Schapire, “The strength of weak learnability,” *Machine Learning*, vol. 5, pp. 197–227, 1990.

[36] L. Mason, J. Baxter, P. Bartlett, and M. Frean, “Boosting Algorithms as Gradient Descent,” in *Advances in Neural Information Processing Systems*, 2000, pp. 512–518.

[37] J. H. Friedman, “Greedy function approximation: A gradient boosting machine,” *The Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001.

[38] R. S. Zemel and T. Pitassi, “A gradient-based boosting algorithm for regression problems,” in *Advances in Neural Information Processing Systems*, 2000, pp. 696–702.

[39] Y. Freund and R. E. Schapire, “Experiments with a new boosting algorithm,” in *International Conference on Machine Learning*, 1996, pp. 148–156.

[40] Hastie, Tibshirani, and Friedman, *The Elements of Statistical Learning*. New York: Springer-Verlag Inc, 2001.

[41] J. Platt, “Probabilistic outputs for support vector machines and comparison to regularized likelihood methods.” in *Adv. in Large Margin Classifiers*, 2000, pp. 61–74.

[42] B. Zadrozny and C. Elkan, “Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers,” in *Proceedings of the Eighteenth International Conference on Machine Learning*, 2001, pp. 609–616.

[43] H.-T. Lin, C.-J. Lin, and R. C. Weng, “A note on platt’s probabilistic outputs for support vector machines,” *Machine Learning*, vol. 68, no. 3, pp. 267–276, 2007.

[44] J. Demšar, “Statistical comparisons of classifiers over multiple data sets,” *The Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

[45] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*. Belmont, CA: Wadsworth, 1984.

[46] L. Breiman, “Random forests,” *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[47] J. Mutch and D. G. Lowe, “Object class recognition and localization using sparse features with limited receptive fields,” *International Journal of Computer Vision*, vol. 80, no. 1, pp. 45–57, 2008.

[48] J. Shotton, A. Blake, and R. Cipolla, “Contour-based learning for object detection,” in *IEEE international Conference on Computer Vision*, vol. 1, 2005, pp. 503–510.

[49] B. Wu and R. Nevatia, “Simultaneous object detection and segmentation by boosting local shape feature based classifier,” 2007, pp. 1–8.

[50] A. Bar-Hillel and D. Weinshall, “Efficient learning of relational object class models,” *Int. J. Comput. Vision*, vol. 77, no. 1-3, pp. 175–198, 2008.

[51] B. Leibe, A. Leonardis, and B. Schiele, “Combined object categorization and segmentation with an implicit shape model,” in *European Conference on Computer Vision, Workshop on Statistical Learning in Computer Vision*, May 2004, pp. 17–32.

[52] H.Schneiderman, “Feature-centric evaluation for efficient cascaded object detection,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2004.

[53] R. Fergus, P. Perona, and A. Zisserman, “Object class recognition by unsupervised scale-invariant learning,” in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, 2003, p. 264.

[54] H. Grabner, C. Beleznai, and H. Bischof, “Improving adaboost detection rate by wobble and mean shift,” in *Proceedings Computer Vision Winter Workshop*, 2005, pp. 23–32.

[55] E. Seemann, B. Leibe, and B. Schiele, “Multi-aspect detection of articulated objects,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2006, pp. 1582–1588.

[56] J. Winn and J. Shotton, “The layout consistent random field for recognizing and segmenting partially occluded objects,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2006, pp. 37–44.

**Hamed Masnadi-Shirazi** received the BS degree in electrical engineering from Shiraz University, Iran and University of Texas at Arlington in 2003. He is currently pursuing the Ph.D. degree at the University of California San Diego, Electrical and Computer Engineering Department in the Statistical Visual Computing Laboratory. He was the recipient of a US National Science Foundation (NSF) IGERT Fellowship from 2007 to 2009. His research interests are in machine learning and computer vision.

**Nuno Vasconcelos** received the licenciatura in electrical engineering and computer science from the Universidade do Porto, Portugal, in 1988, and the MS and PhD degrees from the Massachusetts Institute of Technology in 1993 and 2000, respectively. From 2000 to 2002, he was a member of the research staff at the Compaq Cambridge Research Laboratory, which in 2002 became the HP Cambridge Research Laboratory. In 2003, he joined the Electrical and Computer Engineering Department at the University of California, San Diego, where he heads the Statistical Visual Computing Laboratory. He is the recipient of a US National Science Foundation CAREER award, a Hellman Fellowship, and has authored more than 50 peer-reviewed publications. His work spans various areas, including computer vision, machine learning, signal processing and compression, and multimedia systems. He is a member of the IEEE.