BAdaCost: Multi-class Boosting with Costs

Antonio Fernández-Baldera[1], José M. Buenaposada[b,*], Luis Baumela[1]

[a] *Universidad Politécnica de Madrid, ETSI Informáticos.*
Campus Montegancedo s/n, 28660 Boadilla del Monte, Spain
[b] *Universidad Rey Juan Carlos, ETSII. C/ Tulipán, s/n, 28933 Móstoles, Spain*

Abstract

We present BAdaCost, a multi-class cost-sensitive classification algorithm. It combines a set of cost-sensitive multi-class weak learners to obtain a strong classification rule within the Boosting framework. To derive the algorithm we introduce CMEL, a *Cost-sensitive Multi-class Exponential Loss* that generalizes the losses optimized in various classification algorithms such as AdaBoost, SAMME, Cost-sensitive AdaBoost and PIBoost. Hence unifying them under a common theoretical framework. In the experiments performed we prove that BAdaCost achieves significant gains in performance when compared to previous multi-class cost-sensitive approaches. The advantages of the proposed algorithm in asymmetric multi-class classification are also evaluated in practical multi-view face and car detection problems.

Keywords: Boosting, Multi-class classification, Cost-sensitive classification,
Multi-view object detection

1. Introduction

Boosting algorithms have been extensively used in many computer vision problems, such as facial expression [1] and gait [2] recognition, but particularly in object and face detection [3, 4, 5, 6]. They learn a "strong" classifier by iteratively combining simple or "weak" predictors. Their popularity is based on their simplicity, good generalization, feature selection capability and fast performance when used in a cascade framework [3, 4, 7, 8]. Boosting algorithms are conceived to minimize the number of incorrect predictions. However, it is well known that this is a bad strategy when solving asymmetric problems such as those with large data imbalances [9, 10, 11, 12], or those with very different class priors or costs [13, 14, 15].

Various cost-sensitive Boosting approaches have been proposed with the aim of addressing asymmetric problems. Most of them consider the standard two-class case [15, 16, 3, 17, 18]. They introduce the asymmetry in the Boosting algorithm by tuning the

[*] Corresponding author

Email addresses: antonio.fbaldera@upm.es (Antonio Fernández-Baldera), josemiguel.buenaposada@urjc.es (José M. Buenaposada), lbaumela@fi.upm.es (Luis Baumela)

classifier decision threshold [3] or manipulating the weight update process [16, 17]. These modifications, however, are not guaranteed to converge to a good classification rule. Masnadi and Vasconcelos introduce a binary cost-sensitive algorithm with asymptotic convergence to the optimal classification rule [15]. This is the first theoretically sound cost-sensitive Boosting algorithm in the binary classification context. However, it is not applicable in asymmetric multi-class situations such as, for example, those arising when detecting in images several objects or a single object in different configurations.

Very few works in the Boosting literature address the multi-class cost-sensitive situation. The first attempt was AdaC2.M1 [9] that combined the re-weighting process developed for AdaC2 [19] with the AdaBoost.M1 schedule [20]. This simple heuristic algorithm has the same problem than its multi-class predecessor when a negative predictor confidence emerges because it is "too weak." Another proposal resorted to the minimization of a p-norm based cost functional [21]. However any L_p-CSB variant requires a relational hypothesis (weak learners that multiply the size of data by the number of labels) which is an important drawback with large datasets. The MultiBoost algorithm [22] is based on the minimization of a new cost-sensitive multi-class loss function. However, it does not generalize any previous approaches and requires an imprecise pool of multi-class weak learners to work.

In this paper we introduce a well founded multi-class cost-sensitive Boosting algorithm, BAdaCost, that stands for *Boosting Adapted for Cost matrix*. To this end we introduce a multi-class cost-sensitive margin that relates multi-class margins with the costs of the decisions and derive the algorithm as a stage-wise minimization of the expected exponential loss. We prove that this algorithm is a generalization of previous multi-class [23, 24] and binary cost-sensitive [15] Boosting algorithms.

We validate the algorithm by comparing it with previous multi-class cost sensitive approaches on several data sets from the UCI repository. The experiments show that BAdaCost achieves a significant improvement in performance. Moreover, we also evaluate our algorithm in multi-view detection settings, that pose a highly asymmetrical multi-class classification problem. In our experiments we show that BAdaCost has several advantages in this context when compared to the usual one-vs-background approach that uses one detector per class:

- We slide a K-classes multi-class detector across the image instead of K binary detectors. Since the multi-class detector shares features between classes [25] it can work with considerably less decision trees nodes, see section 5.2.4.

- Costs may be used to modify pair-wise class boundaries. We can reduce the number of errors between positive classes (e.g. different orientations) and improve detection rates when object classes have different aspect ratios, see section 5.2.3.

- To speed up detection our algorithm is amenable for cascade calibration [7].

A preliminary version of BAdaCost appeared in [26] for solving imbalanced classification problems. Here we present it in the more general context of cost-sensitive classification, provide the proofs that support the algorithm and relate it with other approaches in the literature, and apply it to address the multi-view object detection problem.

The rest of the paper is organized as follows. Sections 2 and 3 review the literature on multi-class and cost-sensitive Boosting. In Section 4 we introduce the new multi-class cost-sensitive margin and the BAdaCost algorithm together with various theoretical results relating it with previous approaches in the literature. Finally, in Sections 5 and 6 we experimentally validate the algorithm and draw conclusions. We provide proofs and additional information as supplementary material.

2. Boosting

In this section we briefly review some Boosting results directly related to our proposal. First we introduce some notation. We use the terms *label* and *class* interchangeably. The set of labels is $L = \{1, 2, \ldots, K\}$. The domain of the classification problem is X. Instances are $(\mathbf{x}, l)$, with $\mathbf{x} \in X$ and $l \in L$. We consider N training data instances $\{(\mathbf{x}_n, l_n)\}$. Besides, $I(\cdot)$ is the indicator function (1 when argument is true, 0 when false). M is the number of Boosting iterations and $\mathbf{w} = \{\omega(n) | n = 1, \ldots, N\}$ is a weighting vector. We use capital letters, e.g. T or H, for denoting classifiers whose co-domain is a finite set of integer numbers, like L. Classifiers having a set of vectors as co-domain are represented with small bold letters, e.g. $\mathbf{f}$ or $\mathbf{g}$.

2.1. Binary Boosting

AdaBoost is the most well-known and first successful Boosting algorithm for the problem of binary classification [20]. Given a training data set, the goal of AdaBoost is learning a classifier $H(\mathbf{x})$ based on a linear combination of weak classifiers, $G_m : X \to L \in \{+1, -1\}$, to produce a powerful "committee"

$$h(\mathbf{x}) = \sum_{m=1}^{M} \beta_m G_m(\mathbf{x}) \ , \tag{1}$$

whose prediction is $H(\mathbf{x}) = \operatorname{sign}(h(\mathbf{x}))$. It can also be interpreted as a stage-wise algorithm fitting an additive model [27]. This interpretation provides, at each round *m*, a *direction* for classification, $G_m(\mathbf{x}) = \pm 1$, and a *step size*, β_m, the former understood as a direction on a line and the latter as a measure of confidence in the predictions of G_m. Both elements are estimated in such a way that they minimize the *Binary Exponential Loss function* (BEL) [27, 28]

$$\mathcal{L}\left(l, G_m(\mathbf{x})\right) = \exp(-l\, G_m(\mathbf{x})), \tag{2}$$

defined on the value of $z = l\, G_m(\mathbf{x})$, usually known as the *margin* [29, 30]. To achieve this, a weight distribution on the training set assigns to each training sample $\mathbf{x}_n$ a weight $w(n)$. At iteration m, the algorithm adds to the ensemble the best weak-learner according to the weight distribution. The training data updates its weights taking into account $\mathcal{L}(l, \beta_m G_m(\mathbf{x}))$. Miss-classified samples increase their weights while those of correctly classified are decreased. In this way, new weak learners concentrate on difficult un-learnt parts of the data-set. Since there are two margin values $z = \pm 1$, only two weight updates, $\omega(n) = \omega(n) e^{\pm \beta_m}$, may be achieved at each iteration. In Section 4 we introduce a vectorial encoding that provides a margin allowing various weight updates depending on the cost associated to the classifier response.

2.2. Multi-class Boosting with multi-class responses

With the emergence of AdaBoost, extensions to multi-class problems also appeared. There are a large number of Boosting algorithms dealing with this type of classification. For ease of cataloging, we divide them into two groups: algorithms that *decompose the problem into binary sub-problems* (thus, using binary weak learners) and algorithms that *work simultaneously with all labels* (using multi-class weak learners or computing a posteriori probabilities at the same iteration). This section describes in detail the second approach, that we adopt for our proposal.

The AdaBoost.M1 algorithm proposed by Freund and Schapire uses multi-class weak learners while maintaining the same structure of the original AdaBoost [31]. The main drawback of this approach is the need for "strong learners", i.e. hypotheses that can achieve an accuracy of at least 50%. This requirement may be too strong when the number of labels is high. A second approach is the multi-class version of LogitBoost [27]. As its binary counterpart, it estimates separately the probability of membership to each label based on a multi-logit parametrization.

The contributions most directly related to our proposal are those grounded on a vectorial insight. We want to generalize the symmetry of class-label representation in the binary case to the multi-class case. A successful way of achieving this goal is using a set of vector-valued class codes that represent the correspondence between the label set $L = \{1, \dots, K\}$ and a collection of vectors $Y = \{\mathbf{y}_1, \dots, \mathbf{y}_K\}$. Vector $\mathbf{y}_l$ has a value 1 in the l-th coordinate and $\frac{-1}{K-1}$ elsewhere. So, if $l = 1$, the code vector representing class 1 is $\mathbf{y}_1 = \left(1, \frac{-1}{K-1}, \dots, \frac{-1}{K-1}\right)^\top$. It is immediate to see the equivalence between classifiers $H(\mathbf{x})$ defined over L and classifiers $\mathbf{f}(\mathbf{x})$ defined over Y:

$$H(\mathbf{x}) = l \in L \ \Leftrightarrow \ \mathbf{f}(\mathbf{x}) = \mathbf{y}_l \in Y \, . \tag{3}$$

This codification was first introduced by Lee, Lin and Wahba [32] for extending the binary Support Vector Machine to the multi-class case. Later Zou, Zhu and Hastie generalized the concept of binary margin to the multi-class case using a related vectorial codification [30]. A K-dimensional vector $\mathbf{y}$ is said to be a *margin vector* if it satisfies the *sum-to-zero* condition $\sum_{k=1}^{K} y(k) = 0$. In other words, $\mathbf{y}^\top \mathbf{1} = 0$, where $\mathbf{1}$ denotes a K-dimensional vector of ones. Margin vectors are useful for multi-class classification problems for many reasons. One of them comes directly from the sum-to-zero property. It is known that, in general, every vectorial classifier $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \dots, f_K(\mathbf{x}))^\top$ has a direct translation into a posteriori probabilities $P(l = k \mid \mathbf{x}), \forall k \in L$, via the Multi-class Logistic Regression Function,

$$P(l = k \mid \mathbf{x}) = \frac{\exp(f_k(\mathbf{x}))}{\sum_{i=1}^{K} \exp(f_i(\mathbf{x}))} \, . \tag{4}$$

However, $\mathbf{f}(\mathbf{x})$ produces the same posterior probabilities as $\mathbf{g}(\mathbf{x}) = \mathbf{f}(\mathbf{x}) + \alpha(\mathbf{x}) \cdot \mathbf{1}$, where $\alpha(\mathbf{x})$ is a real-valued function. Such is the case, for example, when $\alpha(\mathbf{x}) = -f_K(\mathbf{x})$. If $\mathbf{f}(\mathbf{x})$ is a margin vector, then we can define the equivalence relation $\mathbf{f}(\mathbf{x}) \sim \mathbf{g}(\mathbf{x}) \Leftrightarrow \exists \alpha : \mathbf{X} \mapsto \Re \mid \mathbf{g}(\mathbf{x}) = \mathbf{f}(\mathbf{x}) + \alpha(\mathbf{x})\mathbf{1}$ on the set of functions $\mathcal{F} = \left\{\mathbf{f} : \mathbf{X} \mapsto \Re^K\right\}$. Hence, margin functions are representatives of equivalence classes.

Using this codification, Zhu, Zou, Rosset and Hastie generalized the original AdaBoost to multi-class problems under an statistical point of view [23] . Since this work is a key ingredient for subsequent derivations, we describe the main elements upon which it is grounded. First, the binary margin in AdaBoost, $z = yf(\mathbf{x})$, is replaced with the *Multi-class Vectorial Margin*, defined as the scalar product

$$z := \mathbf{y}^\top \mathbf{f}(\mathbf{x}). \tag{5}$$

The essence of the margin approach resides in maintaining negative/positive values of the margin when a classifier has respectively a failure/success. That is, if $\mathbf{y}, \mathbf{f}(\mathbf{x}) \in Y$; the margin satisfies: $z > 0 \Leftrightarrow \mathbf{y} = \mathbf{f}(\mathbf{x})$, and $z < 0 \Leftrightarrow \mathbf{y} \neq \mathbf{f}(\mathbf{x})$. The only two values for the margin are

$$z = \mathbf{y}^\top \mathbf{f}(\mathbf{x}) = \begin{cases} \frac{K}{(K-1)} & \text{if } \ \mathbf{f}(\mathbf{x}) = \mathbf{y} \ , \\ \frac{-K}{(K-1)^2} & \text{if } \ \mathbf{f}(\mathbf{x}) \neq \mathbf{y} \ . \end{cases} \tag{6}$$

The *Multi-class Exponential Loss function* (MEL) is

$$\mathcal{L}\left(\mathbf{y}, \mathbf{f}(\mathbf{x})\right) := \exp\left(-\frac{1}{K}\mathbf{y}^\top \mathbf{f}(\mathbf{x})\right) = \exp\left(-\frac{1}{K}z\right) \ . \tag{7}$$

The presence of the constant $1/K$ is important but not determinant for the proper behavior of the loss function. An interesting property of this function that explains the addition of the constant $1/K$ comes from the following equality:

$$\exp\left(-\frac{1}{K}\sum_{k=1}^{K} y(k) f_k(\mathbf{x})\right) = \left(\prod_{k=1}^{K} \exp\left(-y(k) f_k(\mathbf{x})\right)\right)^{1/K} \ . \tag{8}$$

Hence this multi-class loss function is a geometric mean of the binary exponential loss applied to each pair $(\mathbf{y}, \mathbf{f}(\mathbf{x}))$ (i.e. component-wise margins). In spite of the critique in [33], Zhu, Zou, Rosset, and Hastie [23] proved that this loss function is *Fisher-consistent* [30], which means that the population minimizer

$$\arg\min_{\mathbf{f}} E_{Y|X=\mathbf{x}}\left[\mathcal{L}\left(Y, \mathbf{f}(\mathbf{x})\right)\right], \tag{9}$$

has unique solution in the hyperplane of margin vectors and corresponds to the multi-class Bayes optimal rule [23],

$$\arg\max_{k} f_k(\mathbf{x}) = \arg\max_{k} P\left(Y = k|\mathbf{x}\right). \tag{10}$$

So, with enough samples we may recover the exact Bayes rule by minimizing the MEL (7). These results give formal guarantees for learning multi-class classifiers. We generalize this loss in our proposal (see Section 4). Other loss functions, such as the *logit* or L_2, share this property and may also be used for building Boosting algorithms. Similarly, Saberian and Vaconcelos justified that other margin vectors could have been used for representing labels [34, 30], and therefore develop alternative algorithms.

SAMME [23] (Stage-wise Additive Modeling using a Multi-class Exponential loss function) resorts to the MEL (7) for evaluating classifications encoded with margin vectors. The expected loss is then minimized using a stage-wise additive gradient descent approach. The resulting algorithm only differs from AdaBoost in weak learner weight computation $\alpha_m = \log\left((1 - Err_m)/Err_m\right) + \log(K-1)$ (SAMME adds $\log(K-1)$). It is immediate to prove that the classification rule of SAMME, $H(\mathbf{x}) = \arg\max_k \sum_{m=1}^{M} \alpha_m I\left(T_m(\mathbf{x}) = k\right)$, is equivalent to assigning the maximum margin (5), $H(\mathbf{x}) = \arg\max_k \mathbf{y}_k^\top \mathbf{f}(\mathbf{x})$, that is also directly related to the perspective defined in [34] and our proposal (see Section 4). In the same way it is straightforward to verify that AdaBoost becomes a special case when $K = 2$, what makes SAMME the most natural generalization of AdaBoost using multi-class weak-learners.

3. Cost-sensitive Boosting

Classifiers that weigh certain types of errors more heavily than others are called cost-sensitive. They are used in asymmetric classification situations, for example in medical diagnosis and object detection problems. For this type of problems it is usual to consider a ($K\times K$)-matrix $\mathbf{C}$, where each entry $C(i,j) \geq 0$ measures the cost of misclassifying an instance with *real* label i when the *prediction* is j [13]. We expect of this matrix to have costs for correct assignments lower than any wrong classification, i.e. $C(i,i) < C(i,j), \forall i \neq j$. Hereafter, $\mathbf{M}(j,-)$ and $\mathbf{M}(-,j)$ will be used for referring to, respectively, the j-th row and column vector of a matrix $\mathbf{M}$.

For cost-sensitive problems a cost-dependent classification criterion is applied. If $\mathbf{P}(\mathbf{x}) = (P(1|\mathbf{x}), \ldots, P(K|\mathbf{x}))^\top$ is the vector of a posteriori probabilities for a given $\mathbf{x} \in X$, then the *Minimum Cost Decision Rule* is [13]

$$F(\mathbf{x}) = arg \min_{j \in L} \mathbf{P}(\mathbf{x})^\top \mathbf{C}(-,j)\,, \tag{11}$$

that is the minimizer of the risk function $\mathcal{R}(\mathbf{P}(\mathbf{x}), \mathbf{C}(-,j)) := \mathbf{P}(\mathbf{x})^\top \mathbf{C}(-,j)$ with respect to $j \in L$. When dealing with multi-class problems it is important to understand how the consideration of a cost matrix influences the decision boundaries. O'Brien's et al. [35] discloses a concise glossary of linear algebra operations on a cost matrix and their effects on decision boundaries. Let

$$\mathbf{P}(\mathbf{x})^\top \left[\mathbf{C}(-,i) - \mathbf{C}(-,j)\right] = 0 \tag{12}$$

be the decision boundary between classes i and j, with $i \neq j$:

1. Decision boundaries are not affected when $\mathbf{C}$ is replaced by $\alpha\mathbf{C}$, for any $\alpha > 0$.
2. Adding a constant to row $\mathbf{C}(k,-)$ maintains the result unaffected.

Taking into account the last property we will assume without loss of generality that $C(i,i) = 0, \forall i \in L$, i.e. the cost of correct classifications is zero. We will denote $0|1$-*matrix* to that with zeros in its diagonal and ones elsewhere, i.e., a matrix representing a cost-insensitive multi-class problem.

In the following we consider essentially asymmetric matrices. This regular case is the appropriate for situations where some errors are more important than others. In other words, if $C(i,j) > C(j,i)$, $i \neq j$, we want to push the boundary between classes i and j towards j. In graph theory this case would represent a directed complete graph with paths of different module even between pairs of nodes (labels). Other interesting case comes when considering a symmetric cost matrix. Since symmetrical values are equal, the actual information lies in comparing the costs associated to different decision boundaries. Hence this structure is recommended for problems where some class boundaries are more important than others. In graph theory this case is related to an undirected complete graph with different distances between nodes.

3.1. Binary Cost-sensitive Boosting

Let us assume that a binary classification problem has a (2×2)-cost matrix with non negative real values. Let us also assume zero costs for correct classifications. For ease of notation we will use C_1 and C_2 to denote the constants $C(1,2)$ and $C(2,1)$, respectively. Initial attempts to generalize AdaBoost in this way came essentially from heuristic changes on specific parts of the algorithm, which are essentially different reweighting schemes [10, 16, 17, 19].

Later Masnadi-Shirazi and Vasconcelos formally addressed this problem and introduced the *Cost-Sensitive AdaBoost* (CS-AdaBoost) [15]. The core idea behind the algorithm is replacing the original exponential loss function by the *Cost-sensitive Binary Exponential Loss function* (CBEL):

$$\mathcal{L}_{CS-Ada}(l, F(\mathbf{x})) = I(l=1)\exp\left(-lC_1F(\mathbf{x})\right) + I(l=-1)\exp\left(-lC_2F(\mathbf{x})\right). \quad (13)$$

With a a $0|1$-cost matrix it is clear that it becomes AdaBoost. CS-Adaboost is then derived by fitting an additive model whose objective is to minimize the expected loss. Like previous approaches, the algorithm needs to have a pool of weak learners from which to select the optimal one in each iteration, jointly with the optimal step β. For a candidate weak learner, $g(\mathbf{x})$, they compute two constants summing up the weighted errors associated to instances with the same label, b for label 1 and d for label -1. Then β becomes the only real solution to

$$2C_1 b\cosh\left(\beta C_1\right) + 2C_2 d\cosh\left(\beta C_2\right) = T_1C_1\mathrm{e}^{-\beta C_1} + T_2C_2\mathrm{e}^{-\beta C_2}. \quad (14)$$

The algorithm adds the pair $(g(\mathbf{x}), \beta)$ that minimizes $\mathcal{L}_{CS-Ada}(l, F(\mathbf{x}) + \beta g(\mathbf{x}))$ to the model.

On the other hand, Landesa-Vazquez and Alba-Castro's work [18] discuss the effect of an initial non-uniform weighing of instances to endow AdaBoost with a cost-sensitive behavior. The resulting method, Cost-Generalized AdaBoost, takes advantage of the AdaBoost's original structure. See N. Nikolaou et al.'s paper [36] for an excellent summary of cost-sensitive binary Boosting algorithms.

3.2. Multi-class Cost-sensitive Boosting

There are several works in the literature that address the cost-sensitiveness of a problem in a paradigm-independent framework [13, 37, 38, 39]. We will not consider these cases since we are interested in introducing costs in the multi-class Boosting context. In the following we review the contributions conceived for this purpose.

The *AdaC2.M1* algorithm [9] is probably the first including costs when using multi-class weak learners. The idea behind it is combining the multi-class structure of AdaBoost.M1 [31], with the weighting rule of AdaC2 [19], hence its name. As its multi-class counterpart, it fails in computing α-values only available for "not so weak" learners. Moreover, they code the costs of label l into a single value, $C_l = \sum_{j=1}^{K} C(l,j)$, hence loosing the structure of the cost matrix.

The L_p*-CSB* algorithm [21] was originally conceived to solve instance-dependent cost-sensitive problems. It resorts to relational hypothesis, $h : X \times L \to [0,1]$, satisfying the stochastic condition, $\sum_{l \in L} h(l|\mathbf{x}) = 1$, to solve the minimization

$$\arg\min_{h} \frac{1}{N} \sum_{n=1}^{N} C(l_n, \arg\max_{k} h(k|\mathbf{x}_n)). \tag{15}$$

Expression (15) can be approximated by the following convexification:

$$\arg\min_{h} \frac{1}{N} \sum_{n=1}^{N} \sum_{k=1}^{K} h(k|\mathbf{x}_n)^p C(l_n, k) \ , \tag{16}$$

which becomes the aim of the Boosting algorithm. They follow an extended-data approach, just like AdaBoost.MH [28], for stochastic hypothesis. A drawback when applying L_p-CSB comes from the selection of the optimal value p for the norm. There is no clear foundation for selecting it.

Finally, the *MultiBoost* algorithm [22] resorts to margin vectors, $\mathbf{y}, \mathbf{g}(\mathbf{x}) \in Y$ and the loss

$$\mathcal{L}(l, \mathbf{f}(\mathbf{x})) := \sum_{k=1}^{K} C(l,k) \exp(f_k(\mathbf{x})) \ , \tag{17}$$

to carry out a gradient descent search. The loss (17) does not generalize any previous binary problem.

4. BAdaCost

In this section we introduce a new multi-class cost-sensitive margin, based on which we derive BAdaCost. We also relate it with previous algorithms and prove that it generalizes SAMME [23] and PIBoost [24] multi-class approaches and CS-AdaBoost [15] binary cost-sensitive scheme.

4.1. Multi-class cost-sensitive margin

We assume known the $K \times K$ cost matrix $\mathbf{C}$. We define $\mathbf{C}^*$ as

$$C^*(i,j) = \left\{ \begin{array}{ll} C(i,j) & \text{if } i \neq j \\ -\sum_{h=1}^{K} C(j,h) & \text{if } i = j \end{array} \right. , \ \forall i,j \in L. \tag{18}$$

That is, we obtain $\mathbf{C}^*$ from $\mathbf{C}$ by replacing the j-th zero in the diagonal with the sum of the elements in the j-th row with negative sign. $C^*(j,j)$ represents a "negative cost" associated to a correct classification. In other words, elements in the diagonal should be understood as rewards for successes.

The j-th row in $\mathbf{C}^*$, denoted as $\mathbf{C}^*(j,-)$, is a margin vector that encodes the cost structure associated to the j-th label. We define the *Multi-class Cost-sensitive Margin* for sample $(\mathbf{x},l)$ with respect to the multi-class vectorial classifier $\mathbf{g}$ as $z_C := \mathbf{C}^*(l,-)\cdot\mathbf{g}(\mathbf{x})$. It is easy to verify that if $\mathbf{g}(\mathbf{x}) = \mathbf{y}_i \in Y$, for a certain $i \in L$, then $\mathbf{C}^*(l,-)\cdot\mathbf{g}(\mathbf{x}) = \frac{K}{K-1}\mathbf{C}^*(l,i)$. Hence, multi-class cost-sensitive margins obtained from a classifier $\mathbf{g} : \mathbf{X} \to Y$ can be computed using the label-valued analogous of $\mathbf{g}$, $G : \mathbf{X} \to L$,

$$z_C = \mathbf{C}^*(l,-)\cdot\mathbf{g}(\mathbf{x}) = \frac{K}{K-1}\mathbf{C}^*(l,G(\mathbf{x})). \tag{19}$$

So, when considering a lineal combination of discrete classifiers, $H = \sum_{m=1}^{M}\alpha_m\mathbf{g}_m$, expression

$$z_C = \sum_{m=1}^{M}\alpha_m\mathbf{C}^*(l,-)\cdot\mathbf{g}_m(\mathbf{x}) = \frac{K}{K-1}\sum_{m=1}^{M}\alpha_m\mathbf{C}^*(l,G_m(\mathbf{x})), \tag{20}$$

provides a multi-class cost-sensitive margin.

We introduce this generalized margin in the MEL (7) to obtain the *Cost-sensitive Multi-Class Exponential Loss* function (*CMEL*),

$$\mathcal{L}_C(l,\mathbf{g}(\mathbf{x})) := \exp(z_C) = \exp\left(\mathbf{C}^*(l,-)\cdot\mathbf{g}(\mathbf{x})\right). \tag{21}$$

that we optimize in our algorithm. The new margin, z_C, yields negative values when classifications are correct under the cost-sensitive point of view, and positive values for from costly (wrong) assignments. Hence, $\mathcal{L}_C$ does not need a negative sign in the exponent. Moreover, the range of margin values of z_C in (19) is broader than the $z = \pm 1$ values of AdaBoost and related to the costs incurred by the classifier decisions.

The CMEL is a generalization of the MEL (7) and CBEL (13). Let $\mathbf{C}_{0|1}$ be the cost matrix for a cost-insensitive multi-class problem. Since any matrix $\lambda\mathbf{C}_{0|1}$, with $\lambda > 0$, represents the same problem [35], then, given (6) it is straightforward that $\frac{1}{K(K-1)}\mathbf{C}_{0|1}$ will lead to the same values of (7) when applied on the CMEL. In other words, the MEL is a special case of the CMEL in a cost-insensitive problem. On the other hand, it is also immediate to see that for a binary classification problem the values of $\mathbf{C}^*$ lead to the CBEL (13). Hence, it is a special case of CMEL as well.

Vectorial classifiers, $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}),\dots,f_K(\mathbf{x}))^\top$, fitted using additive models collect information in coordinate $f_k(\mathbf{x})$ that can be understood as a *degree of confidence* for classifying sample $\mathbf{x}$ into class k and, hence, they use the max rule, $\arg\max_k f_k(\mathbf{x})$, for label assignment [23, 24, 34]. It is immediate to prove that this criterion is equivalent to assigning the label that maximizes the multi-class margin, $\arg\max_k \mathbf{y}_k^\top\mathbf{f}(\mathbf{x})$, that in turn is equivalent to $\arg\min_k -\mathbf{y}_k^\top\mathbf{f}(\mathbf{x})$. Since $-\mathbf{y}_k^\top\mathbf{f}(\mathbf{x})$ is proportional to $\mathbf{C}^*_{0|1}(k,-)^\top\mathbf{f}(\mathbf{x})$, we can make the decision rule cost-sensitive selecting the label that provides the minimum multi-class cost-sensitive margin. Hence, the classification rule for a classifier optimizing the multi-class cost-sensitive margin is

$$arg\min_k \mathbf{C}^*(k,-)\mathbf{f}(\mathbf{x}).$$

4.2. BAdaCost: Boosting Adapted for Cost matrix

Here we derive BAdaCost, a multi-class cost-sensitive Boosting algorithm, as the minimizer of the empirical expected loss of the CMEL, $\sum_{n=1}^{N} \mathcal{L}_C(l_n, \mathbf{f}(\mathbf{x}_n))$, where $\{(\mathbf{x}_n, l_n)\}$ is our training data.

Following the statistical interpretation of Boosting [27], the minimization is carried out by fitting a stage-wise additive model, $\mathbf{f}(\mathbf{x}) = \sum_{m=1}^{M} \beta_m \mathbf{g}_m(\mathbf{x})$. The weak learner selected at each iteration m consists of an optimal step of size β_m along the direction $\mathbf{g}_m$ of the largest descent of the expected CMEL. In *Lemma* 1 we show how to compute them.

Lemma 1. *Optimal $(\beta_m, \mathbf{g}_m(\mathbf{x}))$ for CMEL*

Let $\mathbf{C}$ be a cost matrix for a multi-class problem. Given the additive model $\mathbf{f}_m(\mathbf{x}) = \mathbf{f}_{m-1}(\mathbf{x}) + \beta_m \mathbf{g}_m(\mathbf{x})$ the solution to

$$(\beta_m, \mathbf{g}_m(\mathbf{x})) = \arg\min_{\beta, \mathbf{g}} \sum_{n=1}^{N} \exp\left(\mathbf{C}^*(l_n, -)\left(\mathbf{f}_{m-1}(\mathbf{x}_n) + \beta \mathbf{g}(\mathbf{x}_n)\right)\right) \tag{22}$$

is the same as the solution to

$$(\beta_m, \mathbf{g}_m(\mathbf{x})) = \arg\min_{\beta, \mathbf{g}} \sum_{j=1}^{K} S_j \exp\left(\beta C^*(j,j)\right) + \sum_{j=1}^{K} \sum_{k \neq j} E_{j,k} \exp\left(\beta C^*(j,k)\right) \ , \tag{23}$$

where $S_j = \sum_{\{n: \mathbf{g}(x_n) = l_n = j\}} w(n)$, $E_{j,k} = \sum_{\{n: l_n = j, \mathbf{g}(x_n) = k\}} w(n)$, and the weight of the n-th training instance is given by:

$$w(n) = \exp\left(\mathbf{C}^*(l_n, -) \sum_{t=1}^{m-1} \beta_m \mathbf{f}_m(\mathbf{x}_n)\right) \ . \tag{24}$$

Given a known direction $\mathbf{g}$, the optimal step β can be obtained as the solution to

$$\sum_{j=1}^{K} \sum_{k \neq j} E_{j,k} C(j,k) A(j,k)^{\beta} = \sum_{j=1}^{K} \sum_{h=1}^{K} S_j C(j,h) A(j,j)^{\beta} \ , \tag{25}$$

being $A(j,k) = \exp(C^*(j,k))$, $\forall j,k \in L$. Finally, given a known β, the optimal descent direction $\mathbf{g}$, equivalently G, is given by

$$\arg\min_{G} \sum_{n=1}^{N} w(n) A(l_n, l_n)^{\beta} I(G(\mathbf{x}_n) = l_n) + \sum_{n=1}^{N} w(n) \sum_{k \neq l_n} A(l_n, k)^{\beta} I(G(\mathbf{x}_n) = k) \ . \tag{26}$$

The proof of this result is in the supplementary material. BAdaCost pseudo-code is shown in Algorithm 1. Just like other Boosting algorithms we initialize weights with a uniform distribution. At each iteration, we add a new multi-class weak learner $\mathbf{g}_m : X \to Y$ to the additive model weighted by β_m, a measure of the confidence in

Algorithm 1 : BAdaCost

```
1- Initialize weights $\mathbf{w}$ with $w(n) = 1/N$; for $n = 1, \dots, N$.
2- Compute matrices $\mathbf{C}^*$ with equation (18) and $\mathbf{A}$ for $\mathbf{C}$.
3- For $m = 1$ to $M$:
    (a) Obtain $G_m$ solving (26) for $\beta = 1$.
    (b) Translate $G_m$ into $\mathbf{g}_m : X \to Y$.
    (c) Compute $E_{j,k}$ and $S_j, \forall j, k$; as described in Lemma 1.
    (d) Compute $\beta_m$ solving equation (25).
    (e) $w(n) \leftarrow w(n) \exp\left(\beta_m \mathbf{C}^*(l_n, -)\mathbf{g}_m(\mathbf{x}_n)\right)$.
    (f) Re-normalize vector $\mathbf{w}$.
4- Output: $H(\mathbf{x}) = \arg\min_k \mathbf{C}^*(k, -) \left(\sum_{m=1}^{M} \beta_m \mathbf{g}_m(\mathbf{x})\right)$.
```

the prediction of $\mathbf{g}_m$. The optimal weak learner that minimizes (26) is a cost-sensitive multi-class classifier trained using the data weights, $w(n)$, and a modified cost matrix, $\mathbf{A}^{\beta} = \exp(\beta\mathbf{C}^*)$.

Unlike other Boosting algorithms [20, 23], here $\mathbf{g}_m$ and β_m, can not be optimized independently. We may solve this in a similar way to e.g. [15, 24] with a local optimization. In practice, however, we have observed that there are no significant differences if we estimate $\mathbf{g}_m$ for a fixed $\beta_m = 1$ and then, given the optimal $\mathbf{g}_m$, estimate β_m. This may be caused by the greedy nature of Boosting. Thus, we proceed as described in Algorithm 1.

4.3. Direct generalizations

BAdaCost is a natural generalization of previous Boosting algorithms. Here we prove that the multi-class classification algorithms SAMME [23] and PIBoost [24] are specializations of BAdaCost for a cost-insensitive situation. Similarly, we also prove that CS-AdaBoost [15] is a special case of BAdaCost for a binary cost-sensitive problem. The proofs are in the supplementary material.

Corollary 1 (SAMME [23] is a special case of BAdaCost). *When $C(i, j) = \frac{1}{K(K-1)}$, $\forall i \neq j$, then the above result is equivalent to SAMME. The update for the additive model $\mathbf{f}_m(\mathbf{x}) = \mathbf{f}_{m-1}(\mathbf{x}) + \beta_m \mathbf{g}_m(\mathbf{x})$ is given by:*

$$(\beta_m, \mathbf{g}_m(\mathbf{x})) = \arg\min_{\beta, \mathbf{g}} \sum_{n=1}^{N} \exp\left(-\mathbf{y}_n^{\top}\left(\mathbf{f}_{m-1}(\mathbf{x}_n) + \beta\mathbf{g}(\mathbf{x}_n)\right)\right)$$

and both optimal parameters can be computed in the following way:

$$\mathbf{g}_m = \arg\min_{\mathbf{g}} \textstyle\sum_{n=1}^{N} w(n) I\left(\mathbf{g}(\mathbf{x}_n) \neq \mathbf{y}_n\right)$$

$$\beta_m = \frac{(K-1)^2}{K}\left(\log\left(\frac{1-E}{E}\right) + \log(K-1)\right),$$

where E is the sum of all weighted errors.

Corollary 2 (CS-AdaBoost [15] is a special case of BAdaCost). *When $K = 2$ the Lemma 1 is equivalent to the Cost-sensitive AdaBoost. If we denote $C(1, 2) = C_1$ and*

$C(2,1) = C_2$, the update (23) for the additive model $F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \beta_m G_m(\mathbf{x})$ becomes:

$$(\beta_m, G_m(\mathbf{x})) = \arg\min_{\beta,G} \sum_{\{l_n=1\}} w(n)\exp\left(-C_1\beta G(\mathbf{x}_n)\right) + \sum_{\{l_n=2\}} w(n)\exp\left(C_2\beta G(\mathbf{x}_n)\right).$$

For a certain value β the optimal direction $G_m(\mathbf{x})$ is given by

$$\arg\min_{G} \left(\mathrm{e}^{\beta C_1} - \mathrm{e}^{-\beta C_1}\right) b + \mathrm{e}^{-\beta C_1} T_1 + \left(\mathrm{e}^{\beta C_2} - \mathrm{e}^{-\beta C_2}\right) d + \mathrm{e}^{-\beta C_2} T_2 \;,$$

being[1] : $T_1 = \sum_{\{n:l_n=1\}} w(n)$, $T_2 = \sum_{\{n:l_n=2\}} w(n)$, $b = \sum_{\{n:G(\mathbf{x}_n)\neq l_n=1\}} w(n)$ and $d = \sum_{\{n:G(\mathbf{x}_n)\neq l_n=2\}} w(n)$. Given a known direction, $G(\mathbf{x})$, the optimal step β_m can be calculated as the solution to

$$2C_1 b\cosh\left(\beta C_1\right) + 2C_2 d\cosh\left(\beta C_2\right) = T_1 C_1 \mathrm{e}^{-\beta C_1} + T_2 C_2 \mathrm{e}^{-\beta C_2} \;.$$

Corollary 3 (PIBoost [24] is a special case of BAdaCost). *When using margin vectors to separate a group of s-labels, $S \in \mathcal{P}(L)$, from the rest, the result of the Lemma 1 is equivalent to PIBoost. The update for each additive model built in this fashion, $\mathbf{f}_m(\mathbf{x}) = \mathbf{f}_{m-1}(\mathbf{x}) + \beta_m \mathbf{g}_m(\mathbf{x})$, becomes:*

$$(\beta_m, \mathbf{g}_m(\mathbf{x})) = \arg\min_{\beta,\mathbf{g}} \sum_{n=1}^{N} w(n)\exp\left(\frac{-\beta}{K}\mathbf{y}_n^\top \mathbf{g}(\mathbf{x}_n)\right) \;.$$

For a certain value β the optimal direction $\mathbf{g}_m(\mathbf{x})$ is given by

$$\begin{aligned} \arg\min_{\mathbf{g}} & \left(\mathrm{e}^{\frac{\beta}{s(K-1)}} - \mathrm{e}^{\frac{-\beta}{s(K-1)}}\right) E_1 + \mathrm{e}^{\frac{-\beta}{s(K-1)}} A_1 \\ + & \left(\mathrm{e}^{\frac{\beta}{(K-s)(K-1)}} - \mathrm{e}^{\frac{-\beta}{(K-s)(K-1)}}\right) E_2 + \mathrm{e}^{\frac{-\beta}{(K-s)(K-1)}} A_2 \;, \end{aligned}$$

being:

$$\begin{aligned} A_1 &= \sum_{\{n:l_n\in S\}} w(n), \qquad & A_2 &= \sum_{\{n:l_n\notin S\}} w(n), \\ E_1 &= \sum_{\{n:G(x_n)\neq l_n\in S\}} w(n), \qquad & E_2 &= \sum_{\{n:G(x_n)\neq l_n\notin S\}} w(n). \end{aligned}$$

Besides, known a direction $\mathbf{g}(\mathbf{x})$, the optimal step β_m can be calculated as $\beta_m = s(K-s)(K-1)\log R$, where R is the only real positive root of the polynomial

$$P_m(x) = E_1(K-s)x^{2(K-s)} + E_2 s x^K - s(A_2 - E_2)x^{(K-2s)} - (K-s)(A_1 - E_1) \;.$$

[1] Here we adopt the notation used in [15].

	Ada.C2M1	**MultiBoost**	**Lp-CSB**	**BAdaCost**
CarEval	26 (±9)	232 (±36)	38 (±15)	**24** (±15)
Chess	29 (±5)	262 (±34)	**4** (±3)	160 (±9)
Isolet	289 (±48)	140 (±15)	149 (±18)	**66** (±14)
SatImage	478 (±62)	187 (±23)	170 (±26)	**132** (±11)
Letter	491 (±78)	319 (±53)	161 (±23)	**66** (±7)
Shuttle	**2.1** (±0.07)	8.9 (±0.08)	3.5 (±0.13)	3.9 (±0.3)
Cont.Meth.	980 (±129)	1058 (±214)	938 (±359)	**928** (±253)
CNAE9	397 (±103)	**171** (±57)	241 (±108)	191 (±51)
OptDigits	366 (±120)	134 (±27)	170 (±34)	**30** (±8)
PenDigits	326 (±29)	193 (±34)	162 (±72)	**18** (±5)
Segmenta.	242 (±123)	154 (±48)	94 (±18)	**50** (±30)
Waveform	905 (±113)	515 (±128)	632 (±201)	**367** (±96)

Table 1: Average costs and standard deviations (in parentheses) of Ada.C2M1, MultiBoost, L_p-CSB, and BAdaCost for each data set (100 iterations) in 10^{-4} scale. Bold values represent the best result achieved for each data base.

5. Experiments

In this section first we compare BAdaCost with other multi-class cost-sensitive Boosting algorithms in the minimization of the expected cost. We also evaluate our approach in two asymmetric multi-class problems such as the detection of cars and faces in arbitrary orientation. In the experiments we use cost-sensitive tree weak-learners and regularize our Boosting algorithm using shrinkage and feature sampling.

5.1. Minimizing costs: UCI repository

In this test we are interested in evaluating the cost minimization capability of the algorithm. To this end we select 12 data sets from the UCI repository[2] that cover a broad range of multi-class classification problems with regard to the number of variables, labels, and data instances.

We compare BAdaCost with the algorithms presented in section 3.2: AdaC2.M1 [9] , L_p-CSB [21] and MultiBoost [22]. For each data set we proceed in the following way. First, we unify train and test data into a single set. Then we carry out a 5-fold cross validation process taking care of maintaining the original proportion of labels for each fold. When training, we compute a cost matrix for unbalanced problems as done in [26][3]. Then we run 100 iterations of each algorithm. We resort to classification trees as base learners. As discussed in section 3.2, AdaC2.M1 and BAdaCost allow the use of multi-class weak learners. MultiBoost also uses multi-class weak learners but it requires a pool of them to work properly. For this reason we create a pool of 6000 weak learners. We build weak-learners over sample data from 30%, 45%, and 60% of the training data-set (2000 weak learners for each ratio). In third place, L_p-CSB translates the multi-class problem into a binary one, thus allowing the use of binary trees.

For comparison we evaluate the average misclassification cost, $\frac{1}{N}\sum_{n=1}^{N} C(l_n, H(\mathbf{x}_n))$, at the end of each test. Note that, after re-scaling the cost matrix, final costs may sum up to a very small quantity. We show the results in Table 1.

[2] See Additional Material.

[3] See Additional Material

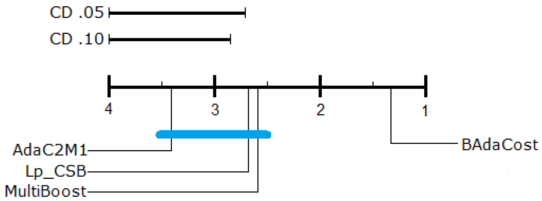


Figure 1: Comparison of ranks through the Bonferroni-Dunn test. BAdaCost's average rank is taken as reference. Algorithms significantly worse than our method for a significance level of 0.10 are unified with a blue line.

BAdaCost outperforms the rest of algorithms in most of the data sets. To assess the statistical significance of the performance differences among the four methods we use the Friedman test of average ranks. The statistic supports clearly the alternative hypothesis, i.e. algorithms do not achieve equivalent results. Then a post-hoc analysis complements our arguments. We carry out the Bonferroni-Dunn test for both significance levels $\alpha = 0.05$ and $\alpha = 0.10$. The confidence distances[4] for these tests are $CD_{0.05} = 1.2617$ and $CD_{0.10} = 1.1216$, respectively. Fig. 1 shows the final result. We can conclude that BAdaCost is significantly better than the AdaC2.M1 and L_p-CSB algorithms for the above levels of significance. In the case of MultiBoost we can state the same conclusion for $\alpha = 0.10$, but not for $\alpha = 0.05$ (the difference between ranks is 1.25).

5.2. Multi-class object detection

The Boosting approach has received much attention because it achieves state-of-the-art performance in various object detection problems such as pedestrians [41], multi-view faces [4] or multi-view cars [5]. The key for the success is the use of the feature selection capabilities of Boosting together with some robust image descriptions such as the *channel features* [8], pooling methods [41], or even using ConvNets channels [42]. The usual framework uses binary classification: AdaBoost, GentleBoost, RealBoost, etc. In this regard, essentially multi-class detection problems, such as face detection [4] or car detection [5], are usually solved with a binary Boosting classifier per positive class (i.e. One-vs-Background strategy). Our goal here is to highlight the advantages of using a multi-class cost-sensitive Boosting algorithm such as BAdaCost.

In this section we adopt a sliding window object detection approach. There are a number of advantages in using BAdaCost for this task:

- **Learn the boundaries between pairs of positive classes**. In multi-view detection problems it is important to avoid errors between extreme views (e.g. a frontal and a size view car). With BAdaCost we can adjust errors between views (positive classes) using the cost matrix. This could not be achieved under the traditional binary perspective.

[4] These values depend on the number of classifiers being compared jointly with the number of data sets over which the comparison is carried out. See [40].

- **A K-classes classifier is faster than K binary ones.** For example, Mathias et al. [4] used 22 face binary detectors in *Headhunter* whereas Ohn-Bar et al. [5] used 75 binary detectors for car detection. In the experiments we will show that BAdaCost can reach comparable or better performance than K binary detectors with much less computational effort.

- **BAdaCost allows cascade calibration.** This is a technique used to speed-up detection algorithms by stopping the evaluation of the strong classifier whenever the weighted sum of weak-learner responses goes below a given threshold [7]. In Section 5.2.1 we introduce a score that can be used for calibrating BAdaCost cascade.

The performance of object detection algorithms depends on a number of details such as image normalization [43], feature channels [8, 44, 42] and pooling method [41] used, occlusion handling [45], multi-scale detection policy [8], etc. The goal of our experiments is not to compete with state-of-the-art algorithms for object detection but to show that BAdaCost provides a number of advantages that are complementary to many other types of improvements in cascade design, feature computation, etc.

For our experiments, we have modified Piotr Dollar's Matlab Toolbox [5] with BAdaCost. Our modified implementation will be available online [6].

5.2.1. BAdaCost positive class score computation

It is not obvious how to compute the score in the multi-class cost-sensitive Boosting. The classification score should be positive whenever the target belongs to the object class and negative when the assigned class is the background. In our case, given M trained weak learners, $\{\mathbf{g}_m(\mathbf{x})\}_{m=1}^M$, and their weights, $\{\beta_m\}_{m=1}^M$, BAdaCost produces a margin vector, $\mathbf{f}(\mathbf{x}) = \sum_{m=1}^M \beta_m \mathbf{g}_m(\mathbf{x})$. So, the predicted costs incurred when classifying sample $\mathbf{x}$ in one of the K classes are $\mathbf{c} = \mathbf{C}^*\mathbf{f}(\mathbf{x}) = (c_1, \dots, c_K)^\top$, and assuming that the negative class has label "1", the score of sample $\mathbf{x}$ is

$$s(\mathbf{x}) = (c_1 - min(c_2, \dots, c_K)). \tag{27}$$

In the following we use (27) to compute the required score to calibrate the BAdaCost cascade [7].

5.2.2. Multi-view face detection

In this group of experiments we consider the problem of detecting faces in images. We follow the experimental methodology of Mathias *et al.* [4], training in the AFLW data set [46], finding the classifier parameters in PASCAL Faces, and testing in AFW [47] and FDDB [48]. In our case we first run HeadHunter[7] [4] to retrieve 23073 face rectangles from AFLW. We train our face detectors with a base size of 40 pixels. This procedure allows us to use the consistent face labeling of HeadHunter to train our

[5] https://github.com/pdollar/toolbox

[6] http://www.dia.fi.upm.es/~pcr/badacost.html

[7] https://bitbucket.org/rodrigob/doppia

BAdaCost-based detector. Second, we make K=5 face classes (see Fig. 2) using the AFLW data set annotations: full right profile (yaw angle less than -60), half right profile (yaw angle from -60 to -20), frontal face (yaw angle from -20 to 20), half left profile (yaw angle from 20 to 60) and full left profile (yaw angle greater than -60). We include only images within -35 to 35 degrees in roll. Any pitch angle is also allowed. We finally use 6136, 14128, 33684, 13483 and 5848 face images respectively in each of the 5 face orientation classes, those that are at least 40 pixel wide. These face images include the actual image in AFLW and its flipped version from the opposite profile view, when applicable. We also flip the frontal view images. As negative examples we use 5772 images without the "Person" label from the PASCAL VOC 2007 [49] and run four rounds of hard negatives mining.

In the experiments we use the *Locally Decorrelated Channel Features* (LDCF) [44]. We have chosen to make the pyramid from one octave up to the actual size of the input image to search for faces greater or equal to 20 pixels.

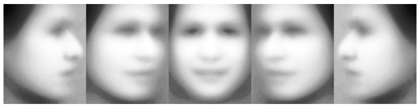


Figure 2: Mean of the AFLW training images in each face view. From left to right: full right profile (view 1), half right profile (view 2), frontal face (view 3), half left profile (view 4) and full left profile (view 5).

In the following, we assign label 1 to the background class and labels 2 to 6 to the five face orientations in Fig. 2. To train our BAdaCost based detector we depart with a 0|1-cost matrix (i.e SAMME). We initially set the number of cost-sensitive trees to T=1024 (4 rounds with 64, 256, 512 and T weak learners, respectively) and tree depth D=6. We look for the number of negatives to add (parameter $nNeg$, or N for sort in figures) per hard negative mining round and the total amount of negatives (parameter $nAccNeg$, or *NA* for sort). In Fig. 3a we show the results with different $nAccNeg$ values training in AFLW and testing in PASCAL Faces (AFLW/PASCAL). The best result in terms of Average Precision (AP) is 85.93 obtained with $nNeg = 10000$ and $nAccNeg = 40000$. After we have found the hard negative parameters, we evaluate different tree depths (D). In Fig. 3b we show the results for various Ds in the AFLW/PASCAL experiment configuration. The best tree depth in this case is $D = 6$ with AP 85.93.

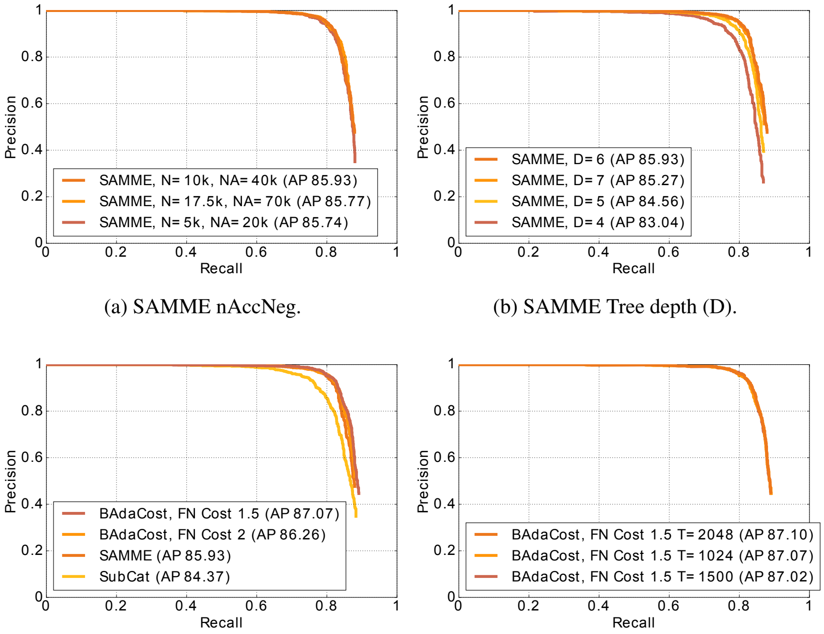
Once selected the parameters for the 0|1-cost matrix, we can vary the costs to adjust the boundaries between the positive classes and background [8]. In order to do so we define the following cost matrix:

$$C_\beta = \left(\begin{array}{cc} 0 & \mathtt{1}_{1\times 5} \\ \beta \cdot \mathtt{1}_{5\times 1} & \mathbf{C}_{0|1,5\times 5} \end{array} \right), \tag{28}$$

where $\mathtt{1}_{m\times n}$ is a $(m \times n)$ matrix full of ones and $\mathbf{C}_{0|1,n\times m}$ is the $(m \times n)$ 0|1-cost matrix. C_β assigns cost 1 to all errors between positives classes and β to False Nega-

[8] Note that errors between face classes in detection do not change AP since all detections have the same window size.

tives (FN). In Fig. 3c we show the results obtained in the AFLW/PASCAL experiment for values of $\beta = 1, 1.5, 2$. We choose $\beta = 1.5$ as a good compromise giving 50% higher costs to FNs, i.e. moving the boundaries between all positive classes towards the background class. Finally, we test the effect of the number of trees (T) when using $\beta = 1.5$, $D = 6$, $nNeg = 10000$ and $nAccNeg = 40000$ (see Fig. 3d). We find that $T = 1024$ gives the optimal number of trees and AP, 87.07 in the AFLW/PASCAL experiments (see qualitative result in first image of Fig. 6).



(a) SAMME nAccNeg.

(b) SAMME Tree depth (D).

(c) SAMME, SubCat, BAdaCost.

(d) BAdaCost #Trees (T).

Figure 3: Training with AFLW and validating with PASCAL (AFLW/PASCAL).

We compare our face detection algorithm with *SquaresChnFtrs-5* and *HeadHunter* [4]. Both approaches are based on Aggregated Channel Features (ACF) [8]. *SquaresChnFtrs-5* uses 5 face classes as we do, but the pooling is given by summing the ACF in squares of different sizes around the feature location. *HeadHunter* uses 21 face classes that accounts also for in plane face rotations. Since in our experiments we use LDCF features, to have comparable results we also train a *One-vs-Background* detector per class with LDCF. Again, we use the same face labels and images from AFLW. We term this approach *SubCat* as in [5]. We first search for the best parameters for this approach: $nNegs = 5000$ and $nAccNeg = 20000$, $D = 2$ and $T = 4096$. Then we compare with the SAMME and BAdaCost approaches in Fig. 3c.

Following the experimentation in [4] we now test with AFW and FDDB the best BAdaCost classifier obtained in the validation experiments. For AFW, since all faces are wider than 40 pixels we do not scale up the input image. As shown in Fig. 4 by using sensible costs (see (28)) BAdaCost can improve (i.e. learn the boundaries between background and positive classes) the results of SAMME. Also, the AP of BAdaCost is 97.00, very close to *HeadHunter*, 97.14, and better than *SquaresChnFtrs-5*, 95.24. On the other hand, *SubCat*, the approach using the same features as BAdaCost, only achieves 93.09 AP. See qualitative results of this experiment in the second image of Fig. 6.

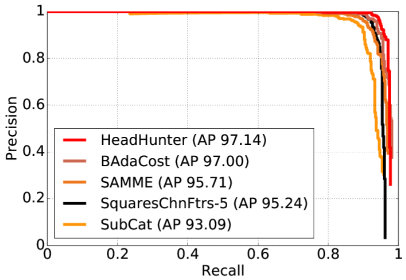
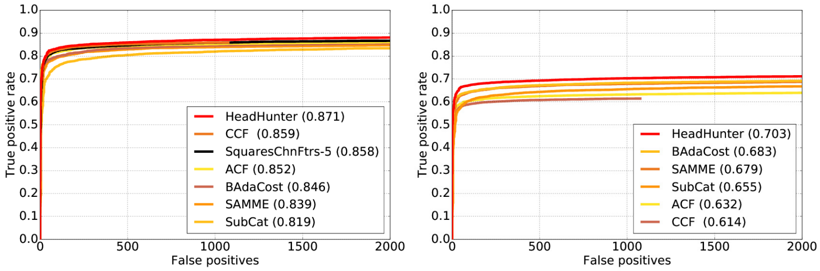


Figure 4: Training with AFLW and testing with AFW (AFLW/AFW experiment).

In the FDDB experiments (results shown in Figs. 5a and 5b) we have transformed face detections into ellipses [4]. This change enables us to achieve a better overlap with the FDDB annotation. We show our FDDB output in the last image of Fig. 6. In the testing protocol of this data set there are two ROC curves: the discrete one (windows with IoU greater than 0.5 add 1 in the True Positive Rate, TPR) and the continuous one (windows add its IoU value in the TPR). Unfortunately, in the latter there are no published results for *SquaresChnFtrs-5*. So we also compare ourselves with Yang et *al.*'s [50] ACF based detector (*ACF* label)) that uses 6 binary detectors tuned to the yaw angle view and Convolutional Channel Features (CCF) [42]. Both approaches achieve equivalent performance to *SquaresChnFtrs-5* in the discrete score, but have also published results for the continuous one. In the discrete ROC experiment we again get better results than *SubCat* and *SAMME*, comparable results to *SquaresChnFtrs-5*, *ACF* and *CCF*, and worse than *Headhunter*. In the continuous setting, *Headhunter* is the best detector, marginally better than BAdaCost.

Our results for face detection using an out-of-the-box rigid approach like BAdaCost are within the state-of-the-art. *HeadHunter* is the best approach with ours marginally behind both in AFW and continuous FDDB. This is possibly due to feature differences, the global color equalization or the large number of face templates tuned to different facial orientations in *HeadHunter*. However, as we discuss in section 5.2.4, BAdaCost is computationally much more efficient.

On the other hand, we also get a performance improvement when Boosting multi-class weak-learners, i.e. *SAMME*, instead of binary ones, i.e. *SubCat*. This could also be expected given that features trained to discriminate among multiple classes are more general than binary ones [25]. Moreover, since in face detection all positive classes have roughly the same aspect ratio, errors between them should not affect recall and, hence, costs would seem useless. However, the usual way to fine tune the negative class boundary in Boosting is by performing hard negative mining, i.e. increasing the negative class *prior*. This can be very time consuming, especially when the classifier has a low negative recall. Our an alternative approach is to use costs. The improvement in performance achieved by BAdaCost vs SAMME is due to this cost-based modification in the FP to FN ratio. Therefore, BAdaCost allows a practical way of learning the negative class boundary at a fixed number of hard negatives to mine.



(a) AFLW/FDDB Discrete Score results. (b) AFLW/FDDB Continuous Score results.

Figure 5: Training with AFLW and testing with FDDB.

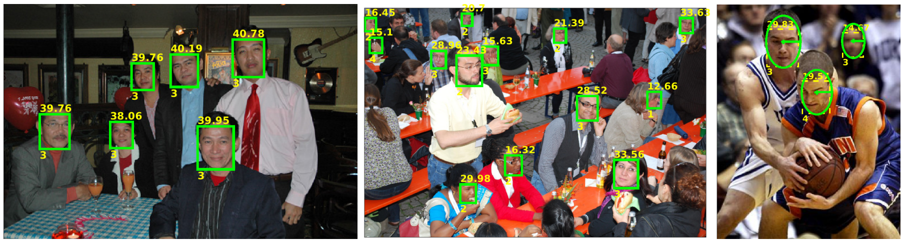


Figure 6: Qualitative detection results for the AFLW/PASCAL experiment (first image), AFLW/AFW experiment (second image), AFLW/FDDB experiment (last image). We display faces with score$\geq$10. Over each detection and under the face we show in yellow, respectively, the score and the estimated face class. In the FDDB results, we show in green the estimated ellipse from the yellow rectangle detection.

5.2.3. *Multi-view car detection*

In this section we consider the problem of detecting cars in images taken "in the wild." Since car detections change their aspect ratio depending on the viewing angle, costs play here a key role to penalize error between positive classes.

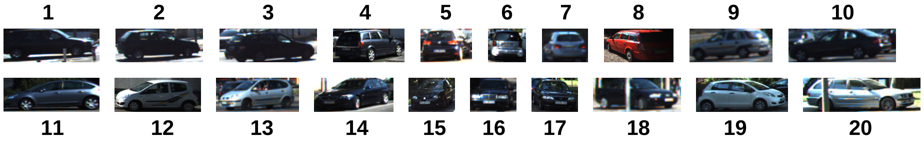


Figure 7: KITTI cars classes we use in our experiments.

We use the object detection benchmark in the KITTI data set [51]. It has three levels of difficulty: easy, moderate and hard (easy $\subset$ moderate $\subset$ hard). We carry out the evaluation in each level separately and use the results in the moderate one to rank the algorithms. In total there are 7481 images for training and 7518 for testing. Since the testing images have no ground truth, we split the train set in training and validation subsets: cars in the first 6733 images (90%) to train (KITTI-train90) and the last 748 images (10%) as validation (KITTI-train10).

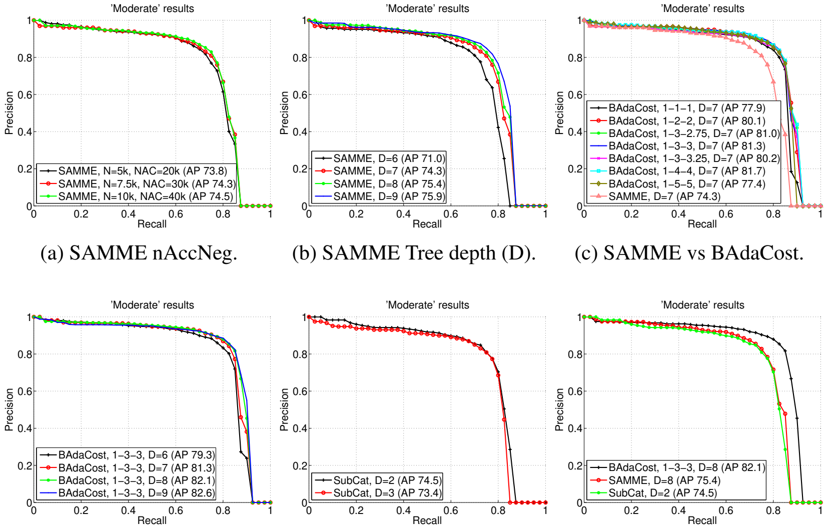
Following Ohn-Bar *et al.*'s *SubCat* [5], we divide the images into $K = 20$ view classes (see Fig. 7) depending on the viewing angle. *SubCat* uses AdaBoost with depth-2 decision trees as weak learners. For BAdaCost we use cost-sensitive decision tree weak learners and LDCF features [44] on the standard ACF channels. In all the experiments with BAdaCost we train a car model of size 48×84 pixels with a one octave-up pyramid to detect cars 24 pixel high. Ohn-Bar et al.'s approach learns a standard Dollar AdaBoost binary detector [8] for each car view (ignoring all other views to extract negatives) and for three heights (25, 32 and 48 pixels). This approach has the advantage of estimating the correct bounding box aspect ratio. The main disadvantages are: 1) it sweeps 20 detectors (at least in one scale) on each image and 2) it cannot address errors between positive (car views) classes. With our multi-class detector we only go through the image once. Moreover, we can use costs to model borders between positive classes. Although in our approach we have to select a fixed window shape for learning and detecting, since the multi-class sliding window detector outcome is the view class, we can correct the fixed size window to the mean training aspect ratio of the predicted view class.

In all experiments with BAdaCost in this section we perform 4 rounds of hard negatives mining with the KITTI training image subset (KITTI-train90). We set the number of cost-sensitive trees to T=1024 (4 rounds with 32, 128, 256 and T weak learners, respectively), tree depth to D=7, and we look for the number of negatives per round to add (parameter N for sort) and the total amount of negatives (parameter NA). In Fig. 8a we show the results with different $nAccNeg$ values in the KITTI-train90/KITTI-10 experiments for the $0|1$-cost matrix (i.e SAMME costs). We choose $nNeg = 7500$ and $nAccNeg = 30000$ since increasing it does not improve AP. Then we evaluate different tree depths (D). In Fig.8b we can see that D=9, AP=75.9, is almost equal to D=8, AP=75.4. We prefer depth $D = 8$ to prevent over-fitting.

Let K_p be the number of positive classes and let us assume that view class labels are arranged in a circular fashion like in Fig. 7. We divide the cost matrix into 3 groups: FP errors (background predicted as any positive class), FN errors (positive class classified as background) and errors in the predicted positive class. We give weights to each of the 3 cost groups: α to the FP error, β to the FN errors and finally γ to the errors between positive classes. Thus, our cost matrix for the multi-view car detection problem is

$$C_{\alpha,\beta,\gamma} = \begin{pmatrix} 0 & \alpha \cdot \mathtt{1}_{1\times 20} \\ \beta \cdot \mathtt{1}_{20\times 1} & \gamma \cdot \mathtt{P} \end{pmatrix}, \tag{29}$$

where $\mathtt{P}$ is a $(K_p \times K_p)$ cost matrix (i.e. main diagonal with zeros) that assigns cost $\mathtt{P}_{i,j}$, when predicting a car of class i as being of class j, as follows



(a) SAMME nAccNeg. (b) SAMME Tree depth (D). (c) SAMME vs BAdaCost.

(d) BAdaCost T. depth (D). (e) SubCat Tree depth (D). (f) Final comparison results.

Figure 8: Training with KITTI-train90 and validating with KITTI-train10.

$$\mathtt{P}(i,j) = 1 - \left|\frac{2|i-j| - K_p}{K_p}\right|. \tag{30}$$

Equation (30) assigns the lowest cost, $2/K_p$, to close view errors ($|i-j| = K_p - 1$ or 1) and 1 to opposite view errors ($|i-j| = K_p/2$, e.g. rear to frontal class errors). Weights α, β and γ balance the relative importance between error groups. We have tested with various combinations of these weights. In Fig. 8c we show different experiments labeled with α-β-γ values. In all of them we use the optimal value of $T = 1024$, $nNeg = 7500$ and $nAccNeg = 30000$. From the results in Fig. 8c and Fig. 8e we can conclude that the cost-less multi-class approach (i.e. SAMME) and the multi-class approach with binary detectors (i.e. One-vs-Background, SubCat) are equivalent in terms of AP. On the other hand, when including a cost matrix to reduce errors in the predicted view class we get noticeable improvements in AP (e.g. from SAMME to BAdaCost 1-1-1). Moreover, since the background has higher prior than any positive class, we need to find the right balance between FN and P costs in (30) w.r.t. the FP. From the BAdaCost experiments in Fig. 8c, we can see that by increasing β and γ to 3, BAdaCost 1-3-3, we get the best results.

In Fig. 8d we can see an improvement in the AP with deeper trees for BAdaCost 1-3-3. There is no improvement in the recall from D=7 to D=9. However, in terms of precision, there is a big improvement from a recall value of 0.6 and above. On the other hand, there is no reason to prefer depth $D = 9$ to $D = 8$ since both plots are almost equal. Thus we can conclude that the best BAdaCost classifier is the one with $\alpha - \beta - \gamma$=1-3-3, $T = 1024$, $D = 8$ with an AP of 82.1 in the KITTI-train90/KITTI-train10 experiments (see qualitative result in Fig. 9).

Data sets	Algorithm	Easy	Moderate	Hard
train90/train10	SubCat	80.00 %	74.50 %	58.30 %
	SAMME	81.00 %	75.40 %	58.80 %
	BAdaCost	**84.8 %**	**82.10 %**	**66.90 %**
train90/testing	SubCat	68.71 %	61.79 %	47.46 %
	BAdaCost	**77.37 %**	**66.66 %**	**55.51 %**

Table 2: AP for the experiments on the KITTI data set.

To make a fair comparison, using the same kind of features and search window size, we have trained a *SubCat* detector with 20 car views (car images 48 pixels heigh). We use AdaBoost with 2048 maximum number of decision trees each. In Fig. 8e we can see the results in the validation subset with depth D=2 and D=3. Finally, we show the results of the best SubCat, SAMME and BAdaCost configurations in Fig. 8f for the moderate samples and in Table 2 for all KITTI experiments. The `train90/train10` data set represents the results of the experiments here described, while `train90/testing` the results produced by the evaluation server [9].

We can conclude that the detector built with a cost-less multi-class AdaBoost, SAMME, is not better than the one built with a set of binary detectors, SubCat. However, the introduction of costs with BAdaCost produces a improvement of about 7 points in AP. This experiment talks for itself about the benefits of using multi-class cost-sensitive Boosting algorithms in asymmetric computer vision problems.

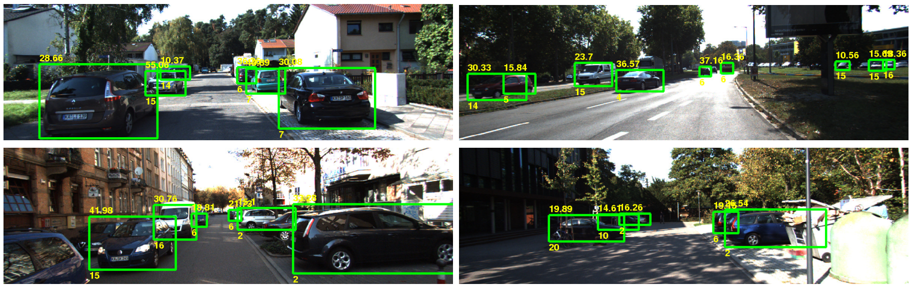


Figure 9: Qualitative detection results in KITTI-train90/KITTI-train10 experiment (shown detections with score≥10). For each detection, we show the score (above) and the estimated view class (below).

5.2.4. Classification efficiency

Computational efficiency is a key issue in classification algorithms, specially those used for object detection in a sliding window approach. The computational cost of Boosting algorithm is directly related to the number of decision tree nodes that have to be evaluated to make a decision.

In Table 3 we show the number of detectors, `#Detect`, number of tree weak-learners in each detector `#T`, tree depths `D` and number of node evaluations, `#Nodes` (number of weak-learners times tree depth times number of detectors).

[9] `http://www.cvlibs.net/datasets/kitti/eval\_object.php` (*SubCat48LDCF* and *BdCost48LDCF*).

Data sets	**Algorithm**	#Detect	#T	D	#Nodes
AFLW	SqChnnFtrs-5	5	2000	2	20000
	SubCat	5	4096	2	40960
	HeadHunter	22	2000	2	88000
	BAdaCost	1	1024	6	**6144**
KITTI train90	SubCat	20	$\leq$2048	2	38984
	BAdaCost	1	1024	8	**8192**

Table 3: Number of decision tree nodes executed to classify.

As shown in Table 3, in the worst case, BAdaCost evaluates fewer decision tree nodes than any of the algorithms discussed in this section. In the face detection problem, it is 3.25 more efficient than SquaresChnFtrs-5, 6.66 times more efficient than SubCat and 14.32 times more efficient than HeadHunter. In the car detection problem, it is 4.75 times more efficient than our trained SubCat detector with LDCF features.

The multi-class cost-sensitive approach presented in this paper is not only more flexible than its binary competitors learning class boundaries, but it also is far more efficient. We can train BAdaCost in roughly one fourth of the time required for SubCat.

6. Conclusions

In this paper we have addressed the problem of extending the notion of multi-class margin to the cost-sensitive margin. We have introduced the BAdaCost algorithm and we have explored its theoretical connections proving that it generalizes SAMME [23], Cost-sensitive AdaBoost [15] and PIBoost [24].

The cost-sensitive multi-class approach introduced in this paper fills an existing gap in the literature. It provides a sound Boosting algorithm for solving multi-class cost-sensitive problems such as the multi-view object detection. The importance of a cost-sensitive multi-class approaches comes from the fact that the cost matrix can be used as a tool to learn class boundaries. There are relevant multi-class problems for which this is important: class imbalance problems [26], asymmetric classification like in detection and segmentation or any problem in which the objective is not the global error but a metric like Jaccard or F1-score [52].

We have shown experimentally that BAdaCost outperforms other relevant multi-class Boosting algorithms in the literature: Ada.C2M1 [9], MultiBoost [22] and L_p-CSB [21]. We have tested our cost-sensitive approach in face and car detection problems. The usual approach to multi-view object detection is to use a binary detector per view. In our experiments we have shown that for face and car detection, the multi-class cost-sensitive Boosting improves by a large margin the usual multi-view approach. This is a relevant outcome since one limitation of current cascade architectures is the difficulty of implementing multi-class detectors [53]. Furthermore, our algorithm uses much fewer weak-learners than the usual approaches to multi-view object detection.

Acknowledgment

The authors thank the anonymous reviewers for their positive comments. This research was funded by the spanish *Ministerio de Economía y Competitividad*, projects

TIN2013-47630-C2-2-R and TIN2016-75982-C2-2-R. José M. Buenaposada acknowledges the support of Computer Vision and Image Processing research group (CVIP) from Universidad Rey Juan Carlos.

References

References

[1] G. Ali, M. A. Iqbal, T.-S. Choi, Boosted NNE collections for multicultural facial expression recognition, Pattern Recognition 55 (2016) 14 – 27.

[2] G. Ma, L. Wu, Y. Wang, A general subspace ensemble learning framework via totally-corrective boosting and tensor-based and local patch-based extensions for gait recognition, Pattern Recognition 66 (2017) 280 – 294.

[3] P. Viola, M. J. Jones, Robust real-time face detection, International Journal of Computer Vision 57 (2) (2004) 137–154.

[4] M. Mathias, R. Benenson, M. Pedersoli, L. Van Gool, Face detection without bells and whistles, in: Proc. European Conf. Computer Vision, 2014.

[5] E. Ohn-Bar, M. Trivedi, Learning to detect vehicles by clustering appearance patterns, IEEE Transactions on Intelligent Transportation Systems 16 (5) (2015) 2511–2521.

[6] H. Ren, Z.-N. Li, Object detection using boosted local binaries, Pattern Recognition 60 (2016) 793 – 801.

[7] C. Zhang, P. A. Viola, Multiple-instance pruning for learning efficient cascade detectors, in: Conf. Neural Information Processing Systems, 2007, pp. 1681–1688.

[8] P. Dollar, R. Appel, S. Belongie, P. Perona, Fast feature pyramids for object detection, IEEE Trans. Pattern Analysis and Machine Intelligence 36 (8) (2014) 1532–1545.

[9] Y. Sun, M. S. Kamel, Y. Wang, Boosting for learning multiple classes with imbalanced class distribution, in: Proc. Int'l Conference on Data Mining, 2006, pp. 592–602.

[10] Y. Sun, M. S. Kamel, A. K. C. Wong, Y. Wang, Cost-sensitive boosting for classification of imbalanced data, Pattern Recognition 40 (12) (2007) 3358–3378.

[11] B. Tang, H. He, GIR-based ensemble sampling approaches for imbalanced learning, Pattern Recognition 71 (2017) 306 – 319.

[12] K. Sheng, W. Dong, W. Li, J. Razik, F. Huang, B. Hu, Centroid-aware local discriminative metric learning in speaker verification, Pattern Recognition 72 (2017) 176 – 185.

[13] C. Elkan, The foundations of cost-sensitive learning, in: Proc. Int'l Joint Conf. on Artificial Intelligence, 2001, pp. 973–978.

[14] H. Masnadi-Shirazi, N. Vasconcelos, Asymmetric boosting, in: Proc. Int'l Conf. on Machine Learning, 2007, pp. 609–619.

[15] H. Masnadi-Shirazi, N. Vasconcelos, Cost-sensitive boosting, IEEE Trans. Pattern Analysis and Machine Intelligence 33 (2) (2011) 294–309.

[16] K. M. Ting, A comparative study of cost-sensitive boosting algorithms, in: Proc. Int'l Conf. on Machine Learning, 2000, pp. 983–990.

[17] P. A. Viola, M. J. Jones, Fast and robust classification using asymmetric AdaBoost and a detector cascade, in: Conf. Neural Information Processing Systems, 2001, pp. 1311–1318.

[18] I. Landesa-Vázquez, J. L. Alba-Castro, Shedding light on the asymmetric learning capability of AdaBoost, Pattern Recognition Letters 33 (3) (2012) 247–255.

[19] Y. Sun, A. K. C. Wong, Y. Wang, Parameter inference of cost-sensitive boosting algorithms, in: Proc. Int'l Conf. on Machine Learning and Data Mining, 2005, pp. 21–30.

[20] Y. Freund, R. E. Schapire, A decision theoretic generalization of on-line learning and an application to boosting, J. of Computer and System Sciences 55 (1997) 199–139.

[21] A. C. Lozano, N. Abe, Multi-class cost-sensitive boosting with p-norm loss functions, in: Proc. Int'l Conf. on Knowledge Discovery and Data Mining, 2008, pp. 506–514.

[22] J. Wang, Boosting the generalized margin in cost-sensitive multiclass classification, J. Computational and Graphical Statistics 22 (1) (2013) 178–192.

[23] J. Zhu, H. Zou, S. Rosset, T. Hastie, Multi-class AdaBoost, Statistics and Its Interface 2 (2009) 349–360.

[24] A. Fernández-Baldera, L. Baumela, Multi-class boosting with asymmetric weak-learners, Pattern Recognition 47 (5) (2014) 2080–2090.

[25] A. Torralba, K. P. Murphy, W. T. Freeman, Sharing features: Efficient boosting procedures for multiclass object detection, in: Proc. IEEE Conf. on Computer Vision and Pattern Recognition, 2004, pp. 762–769.

[26] A. Fernández-Baldera, J. M. Buenaposada, L. Baumela, Multi-class boosting for imbalanced data, in: Proc. of Iberian Conf. Pattern Recognition and Image Analysis, 2015, pp. 57–64.

[27] J. Friedman, T. Hastie, R. Tibshirani, Additive logistic regression: a statistical view of boosting, The Annals of Statistics 28 (2) (2000) 337–407.

[28] R. E. Schapire, Y. Singer, Improved boosting algorithms using confidence-rated predictions, Machine Learning 37 (1999) 297–336.

[29] E. L. Allwein, R. E. Schapire, Y. Singer, Reducing multiclass to binary: A unifying approach for margin classifiers, J. of Machine Learning Research 1 (2000) 113–141.

[30] H. Zou, J. Zhu, T. Hastie, New multicategory boosting algorithms based on multicategory fisher-consistent losses, Annals of Applied Statistics 2 (2008) 1290–1306.

[31] Y. Freund, R. E. Schapire, Experiments with a new boosting algorithm, in: Proc. Int'l Conf. on Machine Learning, 1996, pp. 148–156.

[32] Y. Lee, Y. Lin, G. Wahba, Multicategory support vector machines: theory and application to the classification of microarray data and satellite radiance data, J. American Statistical Association 99 (2004) 67–81.

[33] I. Mukherjee, R. E. Schapire, A theory of multiclass boosting, in: NIPS, 2010, pp. 1714–1722.

[34] M. J. Saberian, N. Vasconcelos, Multiclass boosting: Theory and algorithms, in: Conf. Neural Information Processing Systems, 2011.

[35] D. B. O'Brien, M. R. Gupta, R. M. Gray, Cost-sensitive multi-class classification from probability estimates, in: Proc. Int'l Conf. on Machine Learning, 2008, pp. 712–719.

[36] N. Nikolaou, N. U. Edakunni, M. Kull, P. A. Flach, G. Brown, Cost-sensitive boosting algorithms: Do we really need them?, Machine Learning 104 (2-3) (2016) 359–384.

[37] P. Domingos, Metacost: A general method for making classifiers cost-sensitive, in: Proc. Int'l Conf. on Knowledge Discovery and Data Mining, 1999, pp. 155–164.

[38] Z. Zhou, X. Liu, On multi-class cost-sensitive learning, Computational Intelligence 26 (3) (2010) 232–257.

[39] F. Xia, Y. Yang, L. Zhou, F. Li, M. Cai, D. D. Zeng, A closed-form reduction of multi-class cost-sensitive learning to weighted multi-class learning, Pattern Recognition 42 (7) (2009) 1572–1581.

[40] J. Demsar, Statistical comparisons of classifiers over multiple data sets, J. of Machine Learning Research 7 (2006) 1–30.

[41] S. Zhang, R. Benenson, B. Schiele, Filtered channel features for pedestrian detection, in: Proc. IEEE Conf. on Computer Vision and Pattern Recognition, 2015.

[42] B. Yang, J. Yan, Z. Lei, S. Z. Li, Convolutional channel features, in: Proc. Int'l Conf. Computer Vision, 2015, pp. 82–90.

[43] R. Benenson, M. Mathias, T. Tuytelaars, L. Van Gool, Seeking the strongest rigid detector, in: Proc. IEEE Conf. on Computer Vision and Pattern Recognition, 2013.

[44] J. H. H. W. Nam, P. Dollar, Local decorrelation for improved pedestrian detection, in: Conf. Neural Information Processing Systems, 2014.

[45] M. Mathias, R. Benenson, R. Timofte, L. Van Gool, Handling occlusions with franken-classifiers, in: Proc. Int'l Conf. Computer Vision, 2013.

[46] M. K. östinger, P. Wohlhart, P. M. Roth, H. Bischof, Annotated facial landmarks in the wild: A large-scale, real-world database for facial landmark localization, in: Proc. Int'l Conf. Computer Vision Workshops, 2011, pp. 2144–2151.

[47] X. Zhu, D. Ramanan, Face detection, pose estimation, and landmark localization in the wild, in: Proc. IEEE Conf. on Computer Vision and Pattern Recognition, 2012, pp. 2879–2886.

[48] V. Jain, E. Learned-Miller, FDDB: A benchmark for face detection in unconstrained settings, Tech. Rep. UM-CS-2010-009, University of Massachusetts, Amherst (2010).

[49] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, A. Zisserman, The pascal visual object classes (VOC) challenge, International Journal of Computer Vision 88 (2) (2010) 303–338.

[50] B. Yang, J. Yan, Z. Lei, S. Z. Li, Aggregate channel features for multi-view face detection, in: IEEE Int'l J. Conference on Biometrics, 2014, pp. 1–8.

[51] A. Geiger, P. Lenz, R. Urtasun, Are we ready for autonomous driving? the kitti vision benchmark suite, in: Proc. IEEE Conf. on Computer Vision and Pattern Recognition, 2012.

[52] S. A. P. Parambath, N. Usunier, Y. Grandvalet, Optimizing f-measures by cost-sensitive classification, in: Conf. Neural Information Processing Systems, 2014, pp. 2123–2131.

[53] Z. Cai, Q. Fan, R. S. Feris, N. Vasconcelos, A Unified Multi-scale Deep Convolutional Neural Network for Fast Object Detection, Springer, 2016, pp. 354–370.

Supplementary Material for “BAdaCost: Multi-class Boosting with Costs”

Antonio Fernández-Baldera[1], José M. Buenaposada[b,*], Luis Baumela[1]

[a]Universidad Politécnica de Madrid, ETSI Informáticos.
Campus Montegancedo s/n, 28660 Boadilla del Monte, Spain
[b]Universidad Rey Juan Carlos, ETSII. C/ Tulipán, s/n, 28933 Móstoles, Spain

Abstract

We present BAdaCost, a multi-class cost-sensitive classification algorithm. It combines a set of cost-sensitive multi-class weak learners to obtain a strong classification rule within the Boosting framework. To derive the algorithm we introduce CMEL, a *Cost-sensitive Multi-class Exponential Loss* that generalizes the losses optimized in various classification algorithms such as AdaBoost, SAMME, Cost-sensitive AdaBoost and PIBoost. Hence unifying them under a common theoretical framework. In the experiments performed we prove that BAdaCost achieves significant gains in performance when compared to previous multi-class cost-sensitive approaches. The advantages of the proposed algorithm in asymmetric multi-class classification are also evaluated in practical multi-view face and car detection problems.

Keywords: Boosting, Multi-class classification, Cost-sensitive classification,
Multi-view object detection

[*] Corresponding author
Email addresses: antonio.fbaldera@upm.es (Antonio Fernández-Baldera), josemiguel.buenaposada@urjc.es (José M. Buenaposada), lbaumela@fi.upm.es (Luis Baumela)

1. Proof of BAdaCost Lemma

Let us assume known $\mathbf{f}_m(\mathbf{x})$, the stage-wise additive model up to iteration m. We get $\mathbf{f}_{m+1}(\mathbf{x}) = \mathbf{f}_m(\mathbf{x}) + \beta\mathbf{g}(\mathbf{x})$, where $(\beta, \mathbf{g})$ minimize the expression

$$\begin{aligned}
&\sum_{n=1}^{N} \exp\left(\mathbf{C}^*(l_n, -)\left(\mathbf{f}_m(\mathbf{x}_n) + \beta\mathbf{g}(\mathbf{x}_n)\right)\right) \\
&= \sum_{n=1}^{N} w(n) \exp\left(\beta\mathbf{C}^*(l_n, -)\mathbf{g}(\mathbf{x}_n)\right) \\
&= \sum_{j=1}^{K} \sum_{\{n:l_n=j\}} w(n) \exp\left(\beta\mathbf{C}^*(j, -)\mathbf{g}(\mathbf{x}_n)\right) \\
&= \sum_{j=1}^{K} \sum_{\{n:l_n=j\}} w(n) \exp\left(\frac{\beta K}{K-1} C^*(j, G(\mathbf{x}_n))\right),
\end{aligned}$$

where $w(n) = \exp(\mathbf{C}^*(l_n, -)\mathbf{f}_m(\mathbf{x}_n))$.

In the last step of the previous expression we take into account the equivalence between vector-valued functions, $\mathbf{g}$, and label-valued functions, G. Let $S(j) = \{n : l_n = G(\mathbf{x}_n) = j\}$ be the set of indexes of well classified instances with $l_n = j$ and let $F(j,k) = \{n : l_n = j, G(\mathbf{x}_n) = k\}$ be the indexes where G outputs k when the real label is j. Therefore, we can rewrite the above expression as

$$\begin{aligned}
&\sum_{j=1}^{K} \sum_{n\in S(j)} w(n) \exp\left(\frac{\beta K C^*(j,j)}{K-1}\right) \\
&+ \sum_{j=1}^{K} \sum_{n\in F(j,k)} w(n) \exp\left(\frac{\beta K C^*(j,k)}{K-1}\right) \\
&= \sum_{j=1}^{K} S_j \exp\left(\frac{\beta K C^*(j,j)}{K-1}\right) \\
&+ \sum_{j=1}^{K} \sum_{k\neq j} E_{j,k} \exp\left(\frac{\beta K C^*(j,k)}{K-1}\right).
\end{aligned}$$

Where $S_j = \sum_{n\in S(j)} w(n)$ and $E_{j,k} = \sum_{n\in F(j,k)} w(n)$. Taking into account that these constants are positive values and also $\exp(\beta C^*(i,j)) > 0$ ($\forall i, j \in L$), we can omit the term $K/(K-1)$ in the exponents to solve the minimization. Subsequently, the objective function can be written:

$$\sum_{j=1}^{K} \left(S_j \exp\left(\beta C^*(j,j)\right) + \sum_{k\neq j} E_{j,k} \exp\left(\beta C^*(j,k)\right) \right). \tag{1}$$

Now, fixed a value $\beta > 0$, the optimal step, $\mathbf{g}$, can be found minimizing

$$\begin{aligned}
&\sum_{j=1}^{K}\left(S_j \underbrace{\exp\left(\beta C^*(j,j)\right)}_{A(j,j)^\beta} + \sum_{k\neq j} E_{j,k} \underbrace{\exp\left(\beta C^*(j,k)\right)}_{A(j,k)^\beta}\right)\\
&= \sum_{j=1}^{K}\left(\left(\sum_{n\in S(j)} w(n)\right) A(j,j)^\beta + \sum_{k\neq j}\left(\sum_{n\in F(j,k)} w(n)\right) A(j,k)^\beta\right)\\
&= \sum_{n=1}^{N} w(n)\left(A(l_n,l_n)^\beta I\left(G(\mathbf{x}_n)=l_n\right) + \sum_{k\neq l_n} A(l_n,k)^\beta I\left(G(\mathbf{x}_n)=k\right)\right).
\end{aligned}$$

Finally if we assume known a direction, $\mathbf{g}$, then its weighted errors, $E_{j,k}$, and successes, S_j, will be computable. So deriving (1) with respect to β (note that is a convex function) and equating to zero we get

$$\begin{aligned}
&\sum_{j=1}^{K}\left(\sum_{k\neq j} E_{j,k} C^*(j,k)\exp\left(\beta C^*(j,k)\right) + S_j C^*(j,j)\exp\left(\beta C^*(j,j)\right)\right) = 0\\
&\sum_{j=1}^{K}\sum_{k\neq j} E_{j,k} C^*(j,k)\exp\left(\beta C^*(j,k)\right) = -\sum_{j=1}^{K} S_j C^*(j,j)\exp\left(\beta C^*(j,j)\right)\\
&\sum_{j=1}^{K}\sum_{k\neq j} E_{j,k} C^*(j,k) A(j,k)^\beta = -\sum_{j=1}^{K} S_j C^*(j,j) A(j,j)^\beta\\
&\sum_{j=1}^{K}\sum_{k\neq j} E_{j,k} C(j,k) A(j,k)^\beta = \sum_{j=1}^{K}\sum_{h=1}^{K} S_j C(j,h) A(j,j)^\beta,
\end{aligned}$$

where $A(j,k) = \exp C^*(j,k)$.

2. Proof of Corollary 1 (SAMME is a special case of BAdaCost)

It is easy to see that when $\mathbf{C}$ is defined in the following way:

$$C(i,j) := \begin{cases} 0 & \text{for } i = j \\ \frac{1}{K(K-1)} & \text{for } i \neq j \end{cases} \quad \forall i,j \in L, \tag{2}$$

then a discrete vectorial weak learner, $\mathbf{f}$, yields $\mathbf{C}^*(l,-)\mathbf{f}(\mathbf{x}) = -1/(K-1)$ for correct classifications and $\mathbf{C}^*(l,-)\mathbf{f}(\mathbf{x}) = 1/(K-1)^2$ for errors. Both quantities are, in fact, the possible values of $\frac{-1}{K}\mathbf{y}^\top\mathbf{f}(\mathbf{x})$ in the exponent of the loss function in SAMME. Thus expression (1) can be written

$$
\begin{aligned}
&\sum_{j=1}^{K}\left(S_j \exp\left(\frac{-\beta}{K-1}\right) + \sum_{k\neq j} E_{j,k}\exp\left(\frac{\beta}{(K-1)^2}\right)\right)\\
&= \underbrace{\left(\sum_{j=1}^{K} S_j\right)}_{S}\exp\left(\frac{-\beta}{K-1}\right) + \underbrace{\left(\sum_{j=1}^{K}\sum_{k\neq j} E_{j,k}\right)}_{E}\exp\left(\frac{\beta}{(K-1)^2}\right)\\
&= S\exp\left(\frac{-\beta}{K-1}\right) + E\exp\left(\frac{\beta}{(K-1)^2}\right)\\
&= (1-E)\exp\left(\frac{-\beta}{K-1}\right) + E\exp\left(\frac{\beta}{(K-1)^2}\right)\\
&= \exp\left(\frac{-\beta}{K-1}\right) + E\left(\exp\left(\frac{\beta}{(K-1)^2}\right) - \exp\left(\frac{-\beta}{K-1}\right)\right).
\end{aligned}
$$

So, the above expression is minimized when $E = \sum_{n=1}^{N} w(n) I\left(G(\mathbf{x}_n) \neq l_n\right)$ is minimum. For the second point of the corollary we just need to consider the above expression as a function of β. Computing the derivative and setting equal to zero we get

$$
\begin{aligned}
\frac{E}{K-1}\exp\left(\frac{\beta}{(K-1)^2}\right) &= (1-E)\exp\left(\frac{-\beta}{K-1}\right)\\
\exp\left(\frac{K\beta}{(K-1)^2}\right) &= \frac{(K-1)(1-E)}{E},
\end{aligned}
$$

and taking logarithms

$$
\begin{aligned}
\frac{K\beta}{(K-1)^2} &= \log\left(\frac{1-E}{E}\right) + \log(K-1)\\
\beta &= \frac{(K-1)^2}{K}\left(\log\left(\frac{1-E}{E}\right) + \log(K-1)\right).
\end{aligned}
$$

Hence the second point of the corollary follows.

3. Proof of Corollary 2 (CS-AdaBoost is a special case of BAdaCost)

Given the (2×2)-cost-matrix, let $C_1 = C(1,2)$ and $C_2 = C(2,1)$ denote the non diagonal values. The expression (1) becomes:

$$
\sum_{j=1}^{2}\left(S_j\exp(-\beta C_j) + \underbrace{E_{j,k}}_{k\neq j}\exp(\beta C_j)\right) =
$$

$$
S_1\exp(-\beta C_1) + E_{1,2}\exp(\beta C_1) + S_2\exp(-\beta C_2) + E_{2,1}\exp(\beta C_2).
$$

Let us assume now that $\beta > 0$ is known. Using Badacost Lemma, the optimal discrete weak learner minimizing the expected loss is

$$
\arg\min_{\mathbf{g}}\left[\mathrm{e}^{\beta C_1}E_{1,2} + \mathrm{e}^{-\beta C_1}S_1 + \mathrm{e}^{\beta C_2}E_{2,1} + \mathrm{e}^{-\beta C_2}S_2\right],
$$

changing the notation $T_1 = \sum_{\{n:l_n=1\}} w(n)$, $T_2 = \sum_{\{n:l_n=2\}} w(n)$, $E_{1,2} = b = T_1 - S_1$ and $E_{2,1} = d = T_2 - S_2$,

$$\arg\min_{\mathbf{g}} \left[\mathrm{e}^{\beta C_1} b + \mathrm{e}^{-\beta C_1}(T_1 - b) + \mathrm{e}^{\beta C_2} d + \mathrm{e}^{-\beta C_2}(T_2 - d)\right] =$$

$$\begin{aligned} \arg\min_{\mathbf{g}} \ & [\left(\mathrm{e}^{\beta C_1} - \mathrm{e}^{-\beta C_1}\right) b \\ & \quad + \mathrm{e}^{-\beta C_1} T_1 + \left(\mathrm{e}^{\beta C_2} - \mathrm{e}^{-\beta C_2}\right) d + \mathrm{e}^{-\beta C_2} T_2]. \end{aligned} \tag{3}$$

Besides, if we assume known the optimal weak learner, $\mathbf{g}$, then its weighted success/error rates will be computable. We can find the best value β using BAdaCost Lemma. In this binary case the following expression must be solved

$$E_{1,2} C_1 \mathrm{e}^{\beta C_1} + E_{2,1} C_2 \mathrm{e}^{\beta C_2} = S_1 C_1 \mathrm{e}^{-\beta C_1} + S_2 C_2 \mathrm{e}^{-\beta C_2}.$$

Again, using the notation T_1, T_2, b and d we get

$$\begin{aligned} & bC_1 \mathrm{e}^{\beta C_1} + dC_2 \mathrm{e}^{\beta C_2} = (T_1 - b) C_1 \mathrm{e}^{-\beta C_1} + (T_2 - d) C_2 \mathrm{e}^{-\beta C_2} \\ & bC_1 \left(\mathrm{e}^{\beta C_1} + \mathrm{e}^{-\beta C_1}\right) + dC_2 \left(\mathrm{e}^{\beta C_2} + \mathrm{e}^{\beta C_2}\right) = T_1 C_1 \mathrm{e}^{-\beta C_1} + T_2 C_2 \mathrm{e}^{-\beta C_2} \end{aligned}$$

$$2bC_1 \cosh(\beta C_1) + 2dC_2 \cosh(\beta C_2) = T_1 C_1 \mathrm{e}^{-\beta C_1} + T_2 C_2 \mathrm{e}^{-\beta C_2}\ , \tag{4}$$

that proves the equivalence between both algorithms for binary problems.

4. Proof of Corollary 3 (PIBoost is a special case of BAdaCost)

Let S denote a subset of s-labels of the problem. The margin values are $\mathbf{y}^\top \mathbf{f}(\mathbf{x}) = \frac{\pm K}{s(K-1)}$, when $\mathbf{y} \in S$, and $\mathbf{y}^\top \mathbf{f}(\mathbf{x}) = \frac{\pm K}{(K-s)(K-1)}$, when $\mathbf{y} \notin S$. In both cases there is a positive/negative sign in case of correct/wrong classification. In turn the exponential loss function, $\exp(-\mathbf{y}^\top \mathbf{f}(\mathbf{x})/K)$, yields $\exp(\frac{\mp 1}{s(K-1)})$ and $\exp(\frac{\mp 1}{(K-s)(K-1)})$ respectively. Let $\mathbf{C}$ be the (2×2)-cost-matrix with non diagonal values $C(1,2) = \frac{1}{sK}$ and $C(2,1) = \frac{1}{(K-s)K}$. This matrix produces cost-sensitive multi-class margins with the same values on the loss function. Thus we can apply the Lemma to this binary cost-sensitive sub-problem. In particular we can apply Corollary 2 directly. Replacing in expression (3) we get the optimal weak learner, $\mathbf{g}$, solving

$$\begin{aligned} \arg\min_{\mathbf{g}} & \left(\mathrm{e}^{\frac{\beta}{s(K-1)}} - \mathrm{e}^{\frac{-\beta}{s(K-1)}}\right) E_1 + A_1 \mathrm{e}^{\frac{-\beta}{s(K-1)}} + \\ + & \left(\mathrm{e}^{\frac{\beta}{(K-s)(K-1)}} - \mathrm{e}^{\frac{-\beta}{(K-s)(K-1)}}\right) E_2 + A_2 \mathrm{e}^{\frac{-\beta}{(K-s)(K-1)}}\ . \end{aligned} \tag{5}$$

Where $A_1 = \sum_{\{n:l_n=1\}} w(n)$, $A_2 = \sum_{\{n:l_n=2\}} w(n)$, $E_1 = \sum_{\{n:g(x_n)\neq l_n=1\}} w(n)$ and $E_2 = \sum_{\{n:g(x_n)\neq l_n=2\}} w(n)$. If we assume known the optimal direction of classification $\mathbf{g}$, then its weighted errors and successes, we can compute the optimal step β

using (4) as the solution to

$$\frac{2E_1}{s}\cosh\left(\frac{\beta}{s(K-1)}\right)+\frac{2E_2}{(K-s)}\cosh\left(\frac{\beta}{(K-s)(K-1)}\right)=$$
$$\frac{A_1}{s}\exp\left(\frac{-\beta}{s(K-1)}\right)+\frac{A_2}{(K-s)}\exp\left(\frac{-\beta}{(K-s)(K-1)}\right)\ .$$

Denoting $\beta = s(K-s)(K-1)\log x$ we get

$$\frac{E_1}{s}\left(x^{(K-s)}-x^{-(K-s)}\right)+\frac{E_2}{(K-s)}\left(x^{s}-x^{-s}\right)=\frac{A_1}{s}x^{-(K-s)}+\frac{A_2}{(K-s)}x^{-s}\ .$$

Which is equivalent to finding the only real solution (Descartes Theorem of signs) of the following polynomial:

$$P(x)=E_1(K-s)x^{2(K-s)}+E_2sx^{K}-s(A_2-E_2)x^{(K-2s)}-(K-s)(A_1-E_1)\ .$$

Hence the Corollary follows.

5. UCI Data Bases

In Section 5.1 of the paper we we select 12 data sets from the UCI repository that cover a broad range of multi-class classification problems with regard to the number of variables, labels, and data instances. In Table 1 we provide additional information about them.

Data set	**Variables**	**Labels**	**Instances**
CarEvaluation	6	4	1728
Chess	6	18	28056
CNAE9	856	9	1080
ContraMethod	9	3	1473
Isolet	617	26	7797
Letter	16	26	20000
Shuttle	9	7	58000
OptDigits	64	10	5620
PenDigits	16	10	10992
SatImage	36	7	6435
Segmentation	19	7	2310
Waveform	21	3	5000

Table 1: Summary of selected UCI data sets

6. A cost matrix for imbalanced classification problems[1]

A preliminary issue when using a cost-sensitive algorithm for solving an imbalance problem is establishing the cost matrix, $\mathbf{C}$. A straightforward solution would be to set the costs inversely proportional to the class imbalance ratios. However, this solution does not take into account the complexity of the classification problem. i.e. the amount of class overlap, within-class imbalance, etc. Here we introduce an alternative solution that considers the problem complexity. To this end we introduce a cost matrix that weighs more heavily the errors of poorly classified classes, hence the classifier will concentrate on the difficult minority classes.

Let $\mathbf{F}$ be the confusion matrix and $\mathbf{F}^*$ the matrix obtained when dividing each row i, $\mathbf{F}(i,-)$, by $\mathbf{F}(i,\cdot) = \sum_j F(i,j)$, i.e. the number of samples in class i. Then $F^*(i,j)$ is the proportion of data in class i classified as j. In a complex and imbalanced data-set, a $0|1$-loss classifier (e.g. BAdaCost with $0|1$-losses) will tend to over-fit the majority classes. So, off-diagonal elements in rows $\mathbf{F}^*(i,-)$ for majority (alt. minority) classes will have low (high) scores. Hence, the resulting matrix after setting $F^*(i,i) = 0, \forall i = 1 \dots K$ is already a cost matrix. Finally, to improve numerical conditioning, we set $\mathbf{C} = \lambda \mathbf{F}^*$, for a small $\lambda > 0$.

7. Face detector trained with BAdaCost

Classifiers trained with BAdaCost have a number of interesting features to explore. We first show additional information about the face detector learned with the AFLW database and used in the experiments in the paper. Afterwards we also include information about the car detector trained with KITTI-train90, also used in the experiments.

Fig. 1 (top left) shows the spatial distribution of features selected by BAdaCost in the face detection experiment. We use the weights of the weak learner to compute the map. This means that a feature used in a weak learner tree contributes to the map with the weak learner weight. The more reddish is the color of the corresponding spatial location in the map, the more times a feature in this location has been selected by a

[1] From Section 5 of [1]

weak learner with high weight. On the other hand, Fig. 2 shows the spatial distribution of face features selected in each of the original ACF channels: color (LUV), gradient magnitude ($||.||$) and gradient filters at different orientations. In Fig. 2 we can see that the color channels are symmetrical in terms of feature localization, that is reasonable since skin color is independent of face orientation. The magnitude of the gradient in the center of the face is also important for detection: the face has strong edges around the mouth and nose. Finally, the importance of gradient orientation channels depends on the face orientation class (e.g. 30^o edges are important for half profile faces while 0^o edges are important for frontal faces.).

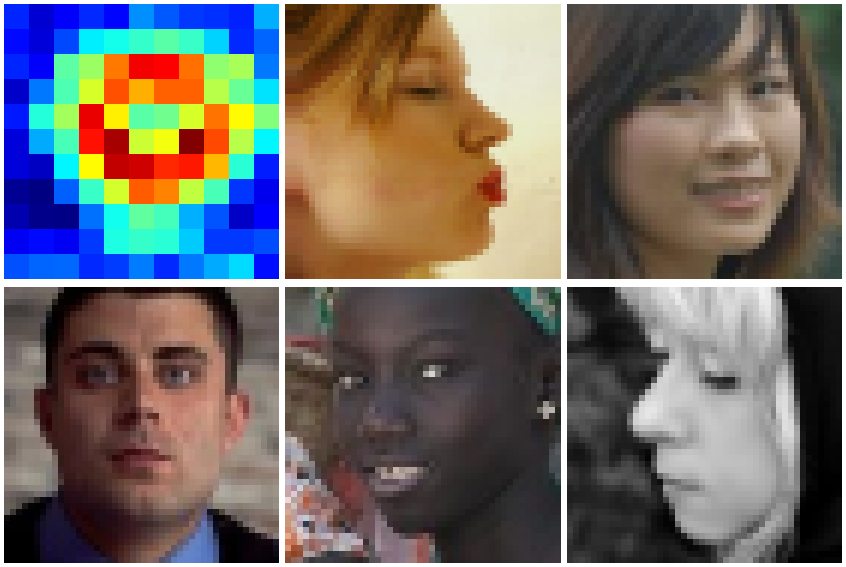


Figure 1: Spatial distribution of features selected by BAdaCost in AFLW (top left). Red color represent the most frequently used features, blue less frequently used ones. We also show several face images to compare them with the feature location map.

Fig. 3 shows the spatial distribution of car features selected by BAdaCost. Fig. 4 shows the spatial distribution of features selected in each of the original ACF channels. In Fig. 4 we can see that color channels have a global role in car detection (see L channel). However, the U channel seems to be used to detect the car rear lights. Also, as in the face detector, gradient orientation channels respond selectively to the car orientation. For example, the 30^o channel responds to view 17 or 18 (half right

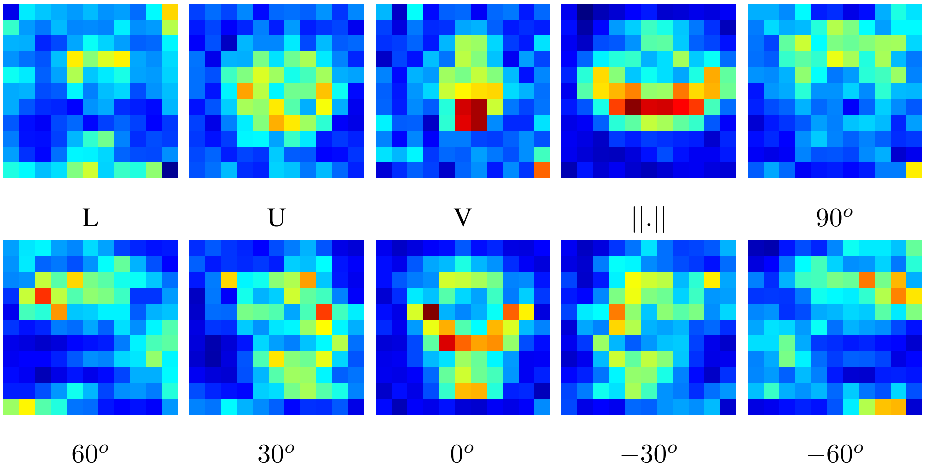


Figure 2: Per channel spatial distribution of features selected by BAdaCost in AFLW. Red color represent the most frequently used features, blue less frequently used ones.

profile).

8. Cascade calibration

Boosting classifiers used in detection can execute all weak learners or stop when we are sure that the image window is negative. In Fig. 5 we show the score in the x axis and the number of executed weak learners in the y axis. In it we plot the largest and lowest score for positive examples (in green) and the lowest and largest score for negatives (in red). It is clear that negative examples scores have lower values than those of the positive ones. So, we can stop evaluating weak learners when the score falls below a calibrated threshold.

We use a slightly modified version of Direct Backward Pruning (DBP) [2]. The DBP algorithm finds the positive training example with the lowest score in the last weak learner (LPSE, Lowest Positive Score Example), see blue plot in Fig. 5. In our case, we set a threshold that corresponds to the lowest LPSE (see purple horizontal line in Fig. 5).

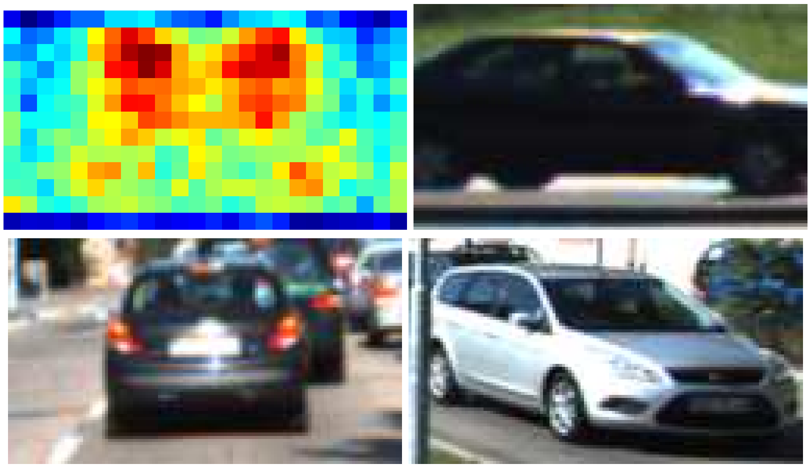


Figure 3: Spatial distribution of selected features by the BAdaCost algorithm in KITTI-train90 (top left). Red color represent the most frequently used features, blue less frequently used ones. We also show training face images to compare them with the feature location map.

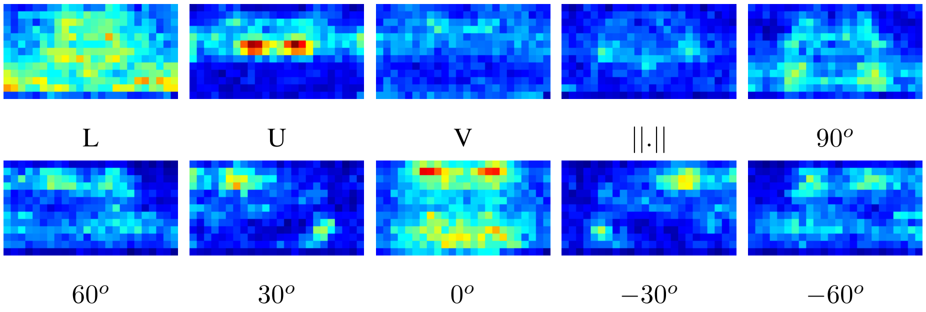


Figure 4: Per channel spatial distribution of selected features by the BAdaCost algorithm in KITTI-train90. Red color represent the most frequently used features, blue less frequently used ones.

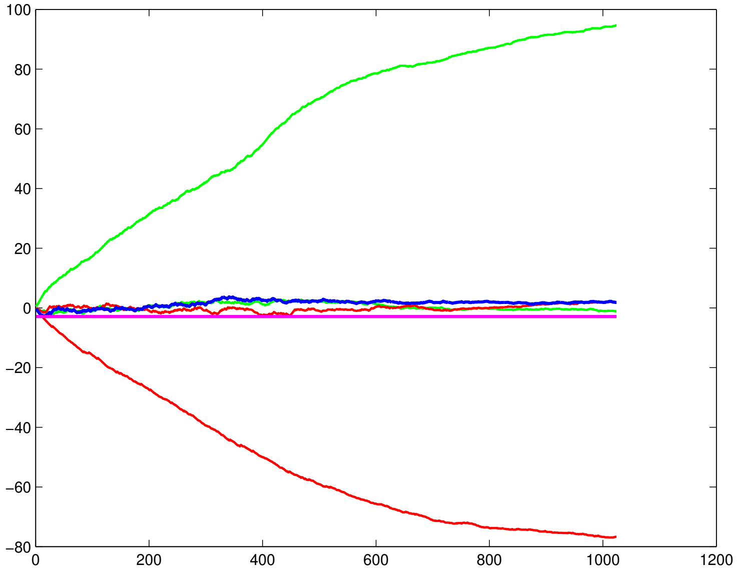


Figure 5: Score as a function of weak learner evaluated (i.e. example trace) for the best BAdaCost classifier trained with the KITTI-train90 database.

References

[1] A. Fernández-Baldera, J. M. Buenaposada, L. Baumela, Multi-class boosting for imbalanced data, in: Proc. of Iberian Conf. Pattern Recognition and Image Analysis, 2015, pp. 57–64.

[2] C. Zhang, P. A. Viola, Multiple-instance pruning for learning efficient cascade detectors, in: Conf. Neural Information Processing Systems, 2007, pp. 1681–1688.