This document is shared under a **CC-BY-NC-ND license**.

You are free to share (that is, copy and redistribute the material in any medium or format) if you follow these licence terms:

Attribution (by)

You must give appropriate credit, provide a link to the license.

Non Commercial (nc)

You can copy, distribute, display, perform, and use this material for any purpose other than commercially (unless you get permission first). Non Commercial means not primarily intended for or directed towards commercial advantage or monetary compensation.

No Derivatives (nd)

If you remix, transform, or build upon the material, you may **not** distribute the modified material. But note that simply changing the format does not create a derivative.

No additional restrictions — You may not apply legal terms or technological measures that legally restrict others from doing anything the licence permits.

Notices
You do not have to comply with the license for elements of the material in the public domain or where your use is permitted by an applicable exception or limitation.

For the full text of this Creative Commons licence go to:

https://creativecommons.org/licenses/by-nc-nd/4.0/legalcode

An effective learning strategy for cascaded object detection

A. Bria[a], C. Marrocco[a], M. Molinara[a], F. Tortorella[a,*]

[a]*Department of Electrical and Information Engineering, University of Cassino and L.M., Via Di Biasio 43, 03043 Cassino (FR), Italy*

Abstract

To distinguish objects from non-objects in images in real-time, a suitable solution is to employ a cascade detector that consists of a sequence of node classifiers with increasing discriminative power. However, among the millions of image patches generated from an input image, only very few contain the searched object. When trained on these highly unbalanced data sets, the node classifiers tend to have poor performance on the minority class. Thus, we propose a learning strategy aimed at maximizing the node classifiers ranking capability rather than their accuracy. We also provide an efficient implementation yielding the same time complexity of the original Viola-Jones cascade training. Experimental results on highly unbalanced real problems show that our approach is both efficient and effective when compared to other node training strategies for skewed classes.

Keywords: pattern recognition, image processing, object detection, cascade, unbalanced data, ranking

1. Introduction

Detecting objects in images and videos is a crucial task in many real-world problems, ranging from face detection in images to pedestrian detection on

*Corresponding author. Address: Department of Electrical and Information Engineering, University of Cassino and L.M., Via Di Biasio 43, 03043 Cassino (FR), Italy. Tel.: +39 07762993605

Email addresses: a.bria@unicas.it (A. Bria), c.marrocco@unicas.it (C. Marrocco), m.molinara@unicas.it (M. Molinara), tortorella@unicas.it (F. Tortorella)

https://doi.org/10.1016/j.ins.2016.01.021

roads, from biomedical image analysis to videosurveillance applications. From the pattern recognition point of view, object detection can be cast as a classification problem in which one has to distinguish between the object class and the "non-object" class. The difficulty is that the "non-object" class actually contains all the patches in the images that are not instances of the searched object and this makes very challenging this apparently simple problem.

First, whereas the object class is quite precisely defined, the non-object class can be very inhomogeneous, since it includes several subclasses of other objects very different each other. In real scenes, the searched objects are embedded in a cluttered background containing various kinds of non-objects, some of which are very different from what we are searching for, whereas others can be very similar. As an example, when detecting faces in an office scene, it would be simple to distinguish between a face and a chair, but it could be more complex to distinguish between a face and a cartoon.

The second issue is that almost every real-world detection task requires processing a huge number of pixels and this could involve a computational load not easy to bear, specially if a complex classifier architecture is used.

The last point is related to the skewed class priors: in a typical detection problem, the vast majority of image locations do not contain the searched object and this makes detection a severely unbalanced classification problem. As a consequence, learning an effective classifier is very difficult since classifiers trained on highly unbalanced data sets tend to have poor performance on the minority class.

In this regard, a commonly adopted solution, originally proposed by Viola and Jones [22], is to employ an ensemble of classifiers structured as a cascade of dichotomizers with increasing complexity. Such an approach allows each dichotomizer in the cascade to deal with only a part of the non-object class, thus parting the complexity of the whole problem among the classifiers. In particular, the first stages of the cascade are built to reject the most distinguishable background regions, while the last stages are specialized to discriminate between actual object and the most confusing background patches (see Fig. 1). This

https://doi.org/10.1016/j.ins.2016.01.021

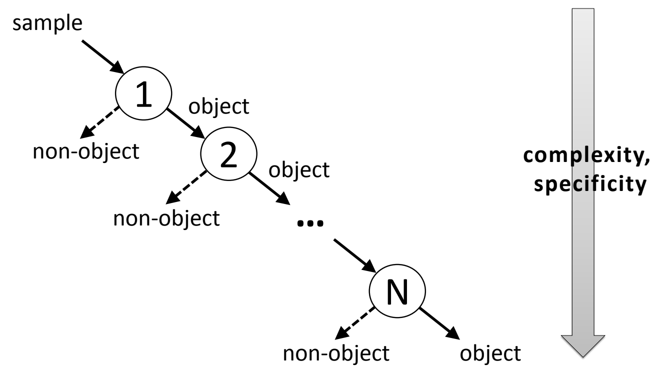

Figure 1: Scheme of the cascade structure with N nodes

aims at reducing the number of false positives produced by the detector and concentrate the computational complexity of the system on the last classifiers of the cascade.

Such approach is a valid solution to the first two issues. Indeed, whereas a monolithic classifier would hardly ensure both a good sensitivity and a good specificity and, in any case, typically requires an excessive computational burden, the cascade of dichotomizers provides a high constant sensitivity and a growing specificity through the stages at a reasonable computational cost. This is obtained by connecting classifiers that provide high sensitivity and sufficient specificity on the subproblem they face and are sufficiently simple to ensure a real-time response. To this end, a commonly used learning algorithm is *AdaBoost* [9] that allows both effective feature selection and proficient classification.

However, the original Viola-Jones approach does not deal well with the class imbalance problem. In fact, even though each dichotomizer in the cascade is trained to discriminate between the object class and a part of the non-object class, the problem is asymmetric anyway since the object class is still much less numerous. In this regard, *AdaBoost* is not able to effectively face the asymmetry between the classes since it minimizes a quantity related to classification error, that is significantly biased by skewed class priors. To address this problem, some new approaches have been developed. In particular, Viola and Jones proposed

https://doi.org/10.1016/j.ins.2016.01.021

a new algorithm, *AsymBoost* [23], that handles the unbalanced classes through an asymmetric weight updating mechanism of the samples in the training set. Starting from this approach, Visentini et al. [24] proposed *AsymBoost** that extended the *AsymBoost* cascade algorithm by introducing a reactive control of the asymmetry at both cascade and node learning level. Both these approaches modify the learning strategy of the dichotomizer without abandoning the beneficial *AdaBoost* scheme that simultaneously selects features and builds the classifier.

Other approaches alter the Viola-Jones standard framework: for example, a *multiexit cascade* is proposed in [13], where the ith node combines the scores from the first $i-1$ dichotomizers, while [28] decouples the feature selection process and the node classifier, which is designed to explicitly address the class asymmetry. In [25] the two previous approaches are combined. All such solutions, however, do not guarantee that the benefits of the original cascade framework are still provided: in fact, the multiexit cascade increases the complexity and the computational load of the single node whereas the separated feature selection process could not make the optimal feature choice for the subsequent classifier learning.

In this paper we address the problem of class imbalance within the Viola-Jones standard cascade framework where we introduce a new learning algorithm for the node classifiers aimed at maximizing their ranking capability rather than their accuracy. Although employed by all the techniques described before, accuracy is not a correct measure for handling rare cases that have less impact than common cases [27]. As a consequence, learning algorithms based on accuracy lead to poor minority-class performance [26]. A common way to alleviate this problem is to adopt a cost-sensitive learning method that assigns a higher cost to the errors made on the minority class (as in *AsymBoost* and *AsymBoost**). The difficulty of this approach lies in correctly defining the cost matrix: a common choice is to set the ratio of the error costs equal to the ratio of the prior probabilities of the two classes, but it is not always possible to reliably estimate such value. Another possible approach is to attempt to balance the class distri-

https://doi.org/10.1016/j.ins.2016.01.021

butions by undersampling the majority class or by oversampling the minority class, but both solutions present serious drawbacks. Whereas undersampling can potentially remove important examples of the majority class, the samples artificially generated and added to the minority class in oversampling can produce an unfaithful training set [12].

Our approach is to use rank metrics instead of accuracy metrics for training the single dichotomizer in the cascade. More specifically, rank metrics are more concerned with the relative ordering of cases than with making absolute predictions for cases and thus place more emphasis on learning to distinguish classes than on learning the internal structure of classes [3]. To this end, the training of the node dichotomizers is based on a reformulation of *RankBoost* for bipartite ranking problems [8], suitably modified to be embedded in a cascade structure. This solution allows the detector to take advantage of the benefits of the standard cascade framework and to effectively face the asymmetry still present in each node classifier training. A partial and preliminary presentation of this approach has been made in [2], whereas in this paper we extend the theoretical framework and consider other significant detection problems in the experiments. In particular, the cascade trained with rank-based node dichotomizers is compared with the original Viola-Jones method, with *AsymBoost* and with *AsymBoost**. Moreover, we have also considered a cascade of dichotomizers trained with *RUSBoost* [18], a boosting-based learning algorithm that faces the class imbalance by undersampling the majority class. As real-world applications, we have examined in our experiments a face detection problem and two medical imaging problems: the detection of microcalcifications on digital mammograms (to extend the preliminary comparison in [2]) and the detection of microaneurysms in digital fundus images. All the examined applications are characterized by a considerable class imbalance and the obtained results show that the proposed approach provides performance similar or better than the other algorithms thus confirming its effectiveness.

The rest of this paper is organized as follows: Section 2 recalls some previously proposed learning strategies for the node classifier, while in Section 3 we

https://doi.org/10.1016/j.ins.2016.01.021

describe the rank-based learning algorithm for the node dichotomizer. Section 4 provides an efficient implementation of the proposed approach. In Section 5 the results of an extensive experimental comparison are reported and discussed. Finally, Section 6 draws some conclusions and outlines directions for future research.

2. Accuracy-based node learning strategies for skewed classes

To point out the drawbacks of accuracy-based node training in the original Viola-Jones framework, let us briefly recall the *AdaBoost* approach. A weak learner $h_\tau(\cdot)$ is selected in each of a series of rounds $\tau = 1, 2, \ldots,$ so as to minimize the *weighted exponential loss* $\sum_j D_\tau(j) \exp\left(-y_j h_\tau\left(\mathbf{x}_j\right)\right)$, where $D_\tau(j)$ is the weight on the jth sample $\mathbf{x}_j$ and $y_j \in \{-1, 1\}$ is the class label of $\mathbf{x}_j$. It is easy to see that, in the case of highly skewed classes, such sum is dominated by the error produced on the majority class and thus the choice of the weak learner is not optimal for predicting the minority class. The approaches proposed so far to alleviate such problem can be attributed to two different groups: *cost-based strategies* and *sampling-based strategies*.

2.1. Cost-based strategies

This category includes the methods that employ the *Adaboost* learning procedure, with a cost assigned to false negatives greater than to false positives. Both *AsymBoost* and *AsymBoost** follow this way. In *AsymBoost*, each sample $\mathbf{x}_j$ is pre-weighted at each round with an asymmetric cost $\exp\left(\frac{1}{T} y_j \log \sqrt{k}\right)$ where T is the total number of rounds of the dichotomizer. The parameter k estimates how much more false negatives cost than false positives and its choice should bias the classifier to perform well on the minority object class. The difference between *AsymBoost* and *AsymBoost** is in the value of k that is fixed in *AsymBoost* whereas it is automatically tuned at each round in *AsymBoost**.

Other cost-sensitive boosting algorithms have been proposed in the literature [11, 7, 21, 19]. The problem common to all these methods is that specific cost information is rarely available or hard to estimate, as well as the ratio between the costs.

https://doi.org/10.1016/j.ins.2016.01.021

2.2. Sampling-based strategies

Another popular solution is to modify the original distribution of the data so as to obtain an artificially balanced training set. This approach can be accomplished by two means: undersampling the majority class by eliminating data points until it reaches the size of the minority class or oversampling the minority class by adding artificially generated data points until the desired balance is achieved. An algorithm introducing data undersampling into the *AdaBoost* procedure is *RUSBoost* [18] that applies a random undersampling of the majority class for building a balanced training set at each iteration of the boosting algorithm. On the other hand, *SMOTEBoost* [5] applies oversampling to *AdaBoost* according to *SMOTE* [4], an algorithm that oversamples the minority class by introducing new, non-replicated minority-class examples. Even though very popular, such approach is hardly applicable when dealing with high dimensional feature spaces, as in the case of the Viola-Jones framework. Moreover, in particular applications (e.g. medical imaging) it could be unsafe to add artificially generated samples that do not correspond to real situations.

3. Ranking-based node learning

In this section we draw a ranking-based learning strategy to train the cascade dichotomizers. The dichotomizer $H_i(\mathbf{x})$ added at the ith stage is based on a reformulation of *RankBoost* for bipartite ranking problems [8], suitably modified to be embedded in a cascade structure. It consists of a linear combination of weak learners $h_{i,\tau}(\mathbf{x}) \in \{0, 1\}$ (0 for the negative class, 1 for the positive one) weighted by $\alpha_{i,\tau} \in \mathbb{R}$ and added in subsequent rounds $\tau = 1, 2, \ldots$ so that after t rounds we obtain

$$H_{i,t}(\mathbf{x}) = \sum_{\tau=1}^{t} \alpha_{i,\tau} h_{i,\tau}(\mathbf{x}). \qquad (1)$$

https://doi.org/10.1016/j.ins.2016.01.021

To simplify the notation, let us omit the subscript i throughout this section. A weak learner $h_\tau(\mathbf{x})$ consists of a simple *decision stump* given by

$$h_\tau(\mathbf{x}) = \begin{cases} 1 & \text{if } \varphi_\tau(\mathbf{x}) > \theta_\tau \\ 0 & \text{if } \varphi_\tau(\mathbf{x}) \leq \theta_\tau \end{cases} \tag{2}$$

where $\varphi_\tau(\mathbf{x}) \in \mathcal{F}$ is the feature selected at round τ from the feature set $\mathcal{F}$ and θ_τ is the threshold selected at round τ in the range $\left(\min_{\mathbf{x}\in\mathcal{T}} \varphi_\tau(\mathbf{x}),\ \max_{\mathbf{x}\in\mathcal{T}} \varphi_\tau(\mathbf{x}) \right)$, with $\mathcal{T}$ being the training set. Differently from the original *RankBoost*, such weak learners do not abstain and are $\{0, 1\}$-valued so as to preserve the ordering information provided by the feature while ignoring its specific scoring information.

To maximize the ranking capability of the dichotomizer, let us consider the *crucial pairs* $(\mathbf{p}, \mathbf{n})$, defined as all the sample pairs made by a positive sample $\mathbf{p}$ and a negative sample $\mathbf{n}$. A crucial pair is correctly ranked when $H_t(\mathbf{n}) < H_t(\mathbf{p})$. Therefore, our goal is to minimize the *ranking loss* R_t, defined as the number of misranked crucial pairs and given by[1]

$$R_t = \sum_{(\mathbf{p},\mathbf{n})} [H_t(\mathbf{n}) \geq H_t(\mathbf{p})] \tag{3}$$

A weight distribution $w_t(\mathbf{p}, \mathbf{n})$ is maintained over the crucial pairs so that the misranked crucial pairs will be more influential in the following rounds. The weight update rule is given by

$$w_{t+1}(\mathbf{p}, \mathbf{n}) = \frac{w_t(\mathbf{p}, \mathbf{n}) \exp\left(\alpha_t \left(h_t(\mathbf{n}) - h_t(\mathbf{p})\right)\right)}{W_t} \tag{4}$$

where W_t is a normalization factor so that $\sum_{\mathbf{p},\mathbf{n}} w_{t+1}(\mathbf{p}, \mathbf{n}) = 1$. Assuming $\alpha_t > 0$, the update rule decreases the weight of crucial pairs in case of correct ranking (i.e., $h_t(\mathbf{p}) = 1$ and $h_t(\mathbf{n}) = 0$) and increases the weight otherwise. Combining Eq. 3 with Eq. 4, the goal becomes to minimize the weighted number

[1] the notation $[pr]$ (Iverson bracket) is defined to be 1 if predicate pr holds and 0 otherwise.

https://doi.org/10.1016/j.ins.2016.01.021

of misranked crucial pairs R_t^w (*weighted ranking loss*) given by

$$R_t^w = \sum_{(\mathbf{p},\mathbf{n})} w_t(\mathbf{p},\mathbf{n})[H_t(\mathbf{n}) \geq H_t(\mathbf{p})] \tag{5}$$

from which $h_t(\cdot)$ (i.e., $\varphi_t(\cdot)$ and θ_t) and α_t can be chosen by

$$(h_t, \alpha_t) = \arg\min_{h,\alpha} \sum_{(\mathbf{p},\mathbf{n})} w_t(\mathbf{p},\mathbf{n})[H_{t-1}(\mathbf{n}) + \alpha h(\mathbf{n}) \geq H_{t-1}(\mathbf{p}) + \alpha h(\mathbf{p})] \tag{6}$$

In fact, α_t can be found directly by minimizing an upperbound of R_t^w as in [8], which yields

$$R_t^w \leq \sqrt{1 - r_t^2} \tag{7}$$

by choosing

$$\alpha_{h_t} = \frac{1}{2} \ln \left(\frac{1 + r_t}{1 - r_t} \right) \tag{8}$$

where

$$r_t = \sum_{\mathbf{p},\mathbf{n}} w_t(\mathbf{p},\mathbf{n})(h_t(\mathbf{p}) - h_t(\mathbf{n})) \tag{9}$$

Then, combining Eq. 6 with Eq. 7, the choice of $h_t(\cdot)$ at round t becomes

$$h_t = \arg\max_h |r_t| \tag{10}$$

In summary, at each round t, the choice of the weak learner $h_t(\cdot)$ and subsequently the selection of the feature $\varphi_t(\cdot)$ is made to maximize the correct pairwise ranking. An example of the training process described so far is shown in Fig. 2.

4. Efficient implementation of ranking-based training

The training procedure presented in the previous section is embedded in a Viola-Jones standard cascade structure which is constructed as follows. Let $\mathcal{P}$ and $\mathcal{N}$ denote the sets of positive and negative samples. The node classifiers $H_i(\mathbf{x})$ are added in subsequent stages $i = 1, 2, \ldots$ until the desired false positive rate F_{target} is reached or $\mathcal{N} = \{\emptyset\}$. To construct the ith node classifier $H_i(\mathbf{x})$, $\mathcal{P}$ and $\mathcal{N}$ (hereafter referred to as *pool*) are used to form a training set $\mathcal{T}_i$ and

https://doi.org/10.1016/j.ins.2016.01.021

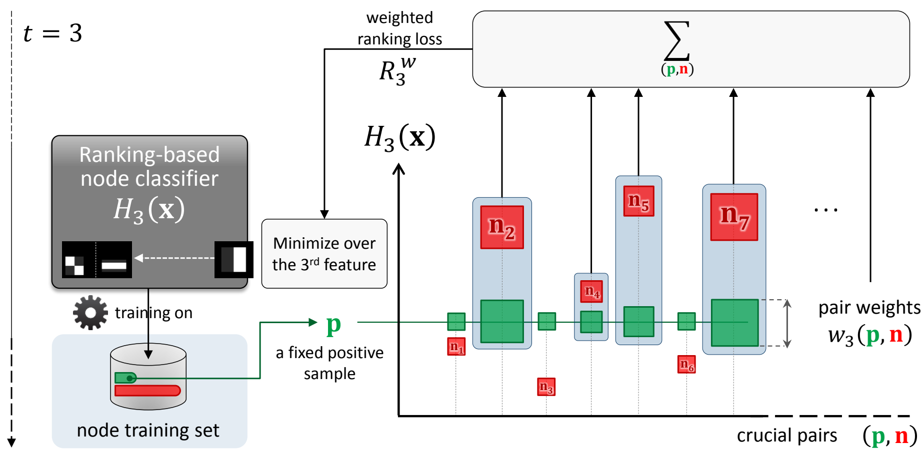

Figure 2: An example of the training process for a node classifier. At each round t ($t = 3$ in the depicted graph), a new weak learner which is restricted to use a single feature is added. The resulting classifier function is then used to rank the samples in the node training set. The sum of weights of misranked crucial pairs (highlighted with a gray box) is evaluated and the feature minimizing this quantity is selected and added to the current node.

a validation set $\mathcal{V}_i$, respectively to train and to set the decision threshold Θ_i of the ith node classifier $H_i(\mathbf{x})$ so as to meet the node learning goals d (detection rate) and f (false positive rate) (see Algorithm 1, lines 13-14).

A naive implementation of this approach may result in a computational workload not easy to sustain in the training phase, especially when dealing with millions of negative samples, as in the case of many object detection problems. Indeed, there are three major issues that make it challenging to efficiently implement the proposed learning strategy: (i) the high computational complexity of weak learner selection in Eq. 10; (ii) the computation of features throughout training; and (iii) the need of a criterion to control the number of training rounds in a node. Each of these problems is discussed in the following subsections. The pseudocode of the overall training procedure is provided in Algorithm 1 along with the list of used symbols.

4.1. Reducing the time complexity of weak learner selection

From Eq. 10, the time-per-round requirements are $O(|\mathcal{T}_i^{\mathbf{p}}| \cdot |\mathcal{T}_i^{\mathbf{n}}| \cdot |\mathcal{H}|)$, where $\mathcal{T}_i^{\mathbf{p}}$ is the subset of $\mathcal{T}_i$ containing only positive samples $\mathbf{p}$, $\mathcal{T}_i^{\mathbf{n}}$ is the subset of $\mathcal{T}_i$ containing only negative samples $\mathbf{n}$ and $\mathcal{H}$ is the set of candidate weak learners

https://doi.org/10.1016/j.ins.2016.01.021

$h(\cdot)$. To improve this, as suggested in [8] we maintain only a one-argument weight distribution $v_{i,t}(\mathbf{x})$ under the constraint

$$w_{i,t}(\mathbf{p}, \mathbf{n}) = v_{i,t}(\mathbf{p}) v_{i,t}(\mathbf{n}) \tag{11}$$

Then, omitting i, t subscripts we obtain from Eq. 9

$$\begin{aligned}
r &= \sum_{\mathbf{p},\mathbf{n}} w(\mathbf{p}, \mathbf{n}) \Big(h(\mathbf{p}) - h(\mathbf{n}) \Big) \\
&= \sum_{\mathbf{p}} \sum_{\mathbf{n}} v(\mathbf{p}) v(\mathbf{n}) \Big(h(\mathbf{p}) s(\mathbf{p}) + h(\mathbf{n}) s(\mathbf{n}) \Big) \\
&= \sum_{\mathbf{p}} \Big(v(\mathbf{p}) \sum_{\mathbf{n}} v(\mathbf{n}) \Big) h(\mathbf{p}) s(\mathbf{p}) + \sum_{\mathbf{n}} \Big(v(\mathbf{n}) \sum_{\mathbf{p}} v(\mathbf{p}) \Big) h(\mathbf{n}) s(\mathbf{n}) \\
&= \sum_{\mathbf{x}} \pi(\mathbf{x}) h(\mathbf{x})
\end{aligned} \tag{12}$$

where

$$s(\mathbf{x}) = \begin{cases} +1, & \text{if } \mathbf{x} \in \mathcal{T}_i^{\mathbf{p}} \\ -1, & \text{if } \mathbf{x} \in \mathcal{T}_i^{\mathbf{n}} \end{cases} \tag{13}$$

and

$$\pi(\mathbf{x}) = s(\mathbf{x}) v(\mathbf{x}) \sum_{\mathbf{x}' : s(\mathbf{x}) \neq s(\mathbf{x}')} v(\mathbf{x}') \tag{14}$$

is referred to as the *potential* $\pi(\mathbf{x})$ and can be precomputed at the beginning of each round in only $O(|\mathcal{T}_i^{\mathbf{p}}| + |\mathcal{T}_i^{\mathbf{n}}|) = O(|\mathcal{T}_i|)$ time. Combining Eqs. 2, 10 and 12 we obtain the final solution

$$h_{i,t} = \arg\max_h \left| \sum_{\mathbf{x} : \varphi(\mathbf{x}) > \theta} \pi_{i,t}(\mathbf{x}) h(\mathbf{x}) \right| \tag{15}$$

which has the reduced time complexity $O(|\mathcal{T}_i| \cdot |\mathcal{H}|)$. Remarkably, this is the same time complexity of the weak learner selection in the Viola-Jones standard cascade framework.

4.2. On-line efficient features computation

From Eq. 2, it follows that each candidate weak learner $h(\mathbf{x}) \in \mathcal{H}$ relies on the evaluation of a feature $\varphi(\mathbf{x}) \in \mathcal{F}$ on a sample $\mathbf{x}$ with respect to a

https://doi.org/10.1016/j.ins.2016.01.021

Algorithm 1 Ranking-based cascade training

F_{target}, F:	desired and actual false positive rate of the cascade classifier
$f, f_{i,t}$:	desired and actual false positive rate of the ith node at round t
d:	desired detection rate of each node
$\mathcal{P}$, $\mathcal{N}$:	sets of positive samples $\mathbf{p}$ and negative samples $\mathbf{n}$
$\mathcal{T}_i$, $\mathcal{V}_i$:	training and validation sets of the ith node
$\mathcal{T}_i^{\mathbf{x}}$, $\mathcal{V}_i^{\mathbf{x}}$:	subsets of $\mathcal{T}_i$, $\mathcal{V}_i$ containing only positive ($\mathbf{x}$=$\mathbf{p}$) or negative ($\mathbf{x}$=$\mathbf{n}$) samples
$v_{i,t}(\mathbf{x})$:	weight of sample $\mathbf{x}$ at round t of the ith node
$[pr]$:	defined to be 1 if pr holds and 0 otherwise (Iverson bracket)
$s(\mathbf{x})$:	function equal to $+1$ if $\mathbf{x} = \mathbf{p}$ or to -1 if $\mathbf{x} = \mathbf{n}$
$\varphi(\mathbf{x})$:	a feature evaluated on a sample $\mathbf{x}$
$H_i(\mathbf{x})$:	ith node classifier
Θ_i:	decision threshold of the ith node classifier
$h_{i,t}(\mathbf{x})$:	weak learner of the ith node at round t evaluated on the sample $\mathbf{x}$
$\alpha_{i,t}$:	the weight of $h_{i,t}(\mathbf{x})$

$i \leftarrow 0; \quad F \leftarrow 1.0;$

while $F > F_{target} \wedge \mathcal{N} \neq \{\emptyset\}$ **do** /* add a new node */

  $i \leftarrow i+1; \quad t \leftarrow 0; \quad f_{i,0} \leftarrow 1.0$

  precompute all candidate weak learners $h(\cdot) \in \mathcal{H}$ on $\mathcal{T}_i$

  $v_{i,0}(\mathbf{p}) = 1/\,|\mathcal{T}_i^{\mathbf{p}}|; \quad v_{i,0}(\mathbf{n}) = 1/\,|\mathcal{T}_i^{\mathbf{n}}|$ /* initialize sample weights */

  while $f_{i,t} > f \wedge \neg varstop$ **do** /* add a new weak learner */

    $t \leftarrow t+1$

    $\pi_{i,t}(\mathbf{x}) = s(\mathbf{x}) v_{i,t}(\mathbf{x}) \sum_{\mathbf{x}':s(\mathbf{x}) \neq s(\mathbf{x}')} v_{i,t}(\mathbf{x}')$ /* precompute potentials */

    $h_{i,t} = \arg\max_h |r_{i,t}|$ /* find best weak learner */

    $\alpha_{i,t} = \frac{1}{2} \ln \left(\frac{1+r_{i,t}}{1-r_{i,t}} \right) \Big|_{h=h_{i,t}}$

    where $r_{i,t} = \sum_{\mathbf{x}:\varphi(\mathbf{x})>\theta} \pi_{i,t}(\mathbf{x}) h(\mathbf{x})$

    $v_{i,t+1}(\mathbf{x})|_{\mathbf{x}=\mathbf{p},\mathbf{n}} = \frac{v_{i,t}(\mathbf{x}) \mathrm{e}^{-s(\mathbf{x})\alpha_{i,t} h_{i,t}(\mathbf{x})}}{\sum_{\mathbf{x}} v_{i,t}(\mathbf{x}) \mathrm{e}^{-s(\mathbf{x})\alpha_{i,t} h_{i,t}(\mathbf{x})}}$ /* sample weights update */

    precompute weak learner $h_{i,t}$ on $\mathcal{V}_i$ and $\mathcal{N}$

    $H_{i,t}(\mathbf{x})|_{\mathbf{x} \in \mathcal{V}_i} = \sum_{\tau=1}^{t} \alpha_{i,\tau} h_{i,\tau}(\mathbf{x})$ /* evaluate $H_{i,t}(\cdot)$ on $\mathcal{V}_i$ */

    $\Theta_i : |\{\mathbf{p} \in \mathcal{V}_i : H_{i,t}(\mathbf{p}) \geq \Theta_i\}| \geq d\,|\mathcal{V}_i^{\mathbf{p}}|$ /* meet node learning goal d */

    $f_{i,t} \leftarrow f_{i,t-1}\,|\{\mathbf{n} \in \mathcal{V}_i : H_{i,t}(\mathbf{n}) \geq \Theta_i\ \}|$ /* update false positive rate $f_{i,t}$ */

    $varstop \leftarrow [Var\,(\{f_{i,\tau}\}_{\tau=t-\Delta,\ldots,t}) \leq \epsilon]$ /* stop criterion */

  end while

  output $H_i(\mathbf{x}) = \left[\sum_{\tau=1}^{t} \alpha_{i,\tau} h_{i,\tau}(\mathbf{x}) \geq \Theta_i \right]$ /* the final ith node classifier */

  $F \leftarrow F \times f_{i,t}$ /* update overall false positive rate */

  if $F > F_{target}$ **then** /* prepare $\mathcal{T}_{i+1}$ and $\mathcal{V}_{i+1}$ for a new node */

    $\mathcal{T}_{i+1} = \mathcal{T}_i \setminus \{\mathbf{n} \in \mathcal{T}_i : H_i(\mathbf{n}) = 0\}$; $\mathcal{V}_{i+1} = \mathcal{V}_i \setminus \{\mathbf{n} \in \mathcal{V}_i : H_i(\mathbf{n}) = 0\}$

    $\mathcal{N} = \{\mathbf{n} \in \mathcal{N} : H_i(\mathbf{n}) = 1\}$

    refill adequately both $\mathcal{T}_{i+1}$ and $\mathcal{V}_{i+1}$ by random sampling from $\mathcal{N}$

  end if

end while

https://doi.org/10.1016/j.ins.2016.01.021

threshold θ. Being $O(1)$ the per-feature evaluation time requirement [22], pre-calculating the features for all the samples used in the training phase would cost $O\big(|\mathcal{F}|\,(|\mathcal{P}|+|\mathcal{N}|)\big)$. Moreover, additional costs have to be considered when there is no available system memory to keep all these feature values at the same time, and a caching strategy is used instead. To cope with these problems, we draw an *on-line* features computation strategy aimed at minimizing the number of needed features calculations and keeping feature values into memory until they are no longer needed. In fact, only three contributions are required to train the ith node: (i) $|\mathcal{T}_i|\,|\mathcal{F}|$ feature values for finding the best weak learner on $\mathcal{T}_i$ (see Algorithm 1, line 9); (ii) $t\,|\mathcal{V}_i|$ feature values for evaluating the node classifier $H_{i,t}(\mathbf{x})$ on $\mathcal{V}_i$ (see Algorithm 1, lines 12 and 20); and (iii) $t\,|\mathcal{N}|$ feature values for evaluating the ith node classifier on $\mathcal{N}$ (see Algorithm 1, line 21). Therefore, we can pre-compute and keep into memory the $|\mathcal{T}_i|\,|\mathcal{F}|$ feature values before training the ith node (see Algorithm 1, line 4) and do the same for each selected feature evaluated on $\mathcal{V}_i$ and $\mathcal{N}$ at each round (see Algorithm 1, line 11). When a node has been trained, we release the memory used for storing the feature values calculated on the samples discarded from $\mathcal{T}_i$, $\mathcal{V}_i$ and $\mathcal{N}$. In this way, the time-per-node requirements for feature computation are only $O\left(|\mathcal{T}_i|\,|\mathcal{F}|\right)$, where we supposed $t\left(|\mathcal{V}_i|+|\mathcal{N}|\right) \ll |\mathcal{T}_i|\,|\mathcal{F}|$ since $|\mathcal{N}|$ decreases exponentially throughout the cascade (see Algorithm 1, lines 21-22) and t can be supposed to be a relatively small number (as it will be discussed in the next paragraph).

4.3. Determining the number of training rounds

In principle, the training rounds for the ith node go on until the node learning goals d and f are met. In fact, the decision threshold Θ_i of the ith node classifier $H_i(\mathbf{x})$ is always chosen at each round so as to meet d, hence the number of rounds is governed only by whether the condition $f_{i,t} \leq f$ is satisfied, being $f_{i,t}$ the actual false positive rate of the ith node achieved at round t. However, when the classification task is getting more and more complex throughout the cascade, it could be too difficult to satisfy such a condition, thus causing several unnecessary features to be added without substantially reducing $f_{i,t}$.

https://doi.org/10.1016/j.ins.2016.01.021

To solve this problem, we add new features until a significant reduction of $f_{i,t}$ can be achieved. Let $\psi_i(\Delta) = \{f_{i,\tau}\}_{\tau=t-\Delta,t-\Delta+1,\dots,t}$ be the latest Δ achieved false positive rates of the ith node. Then, we define a stopping criterion being $varstop = [Var\,(\psi_i(\Delta)) \leq \epsilon]$ so that new features are added until the condition $f_{i,t} > f \wedge \neg varstop$ holds (see Algorithm 1, lines 6,15).

5. Experiments

To evaluate the performance of the proposed approach, we have considered three different real problems with highly unbalanced classes. The first is face detection that represents a well-known topic in scientific literature and a historical problem for the cascade approach as it was used in the seminal paper of Viola and Jones [22]. The other two problems belong to the field of medical image analysis. In particular, we examined the detection of microcalcifications (MC) on digital mammograms for the automated early detection of breast cancer, and the detection of microaneurysms (MA) on digital ocular fundus images, that is an important task in computer aided diagnosis of diabetic retinopathy.

In all the experiments, Haar-like features were used to describe the regions to be detected. These features are simple and computationally efficient and have been successfully used in face detection and classification problems. As in [10], we considered the set of (i) edge features; (ii) line features; (iii) center-surround features; (iv) and special diagonal line features. All features have been scaled and separately translated across all possible combinations on the subwindow, thus obtaining tens of thousands of features.

To verify the effectiveness of our approach (hereafter referred to as *RankingCascade*) we also analyzed the behavior of other methods proposed in the literature. In particular, we implemented and evaluated the performance of different cascades that represent different solutions for facing class imbalance:

- *AsymBoost-based Cascade* (hereafter abbreviated as *Asym*). AsymBoost [23] is a cost-sensitive variant of AdaBoost that uses a parameter k to estimate the balance between the errors on the positive (False Negative

https://doi.org/10.1016/j.ins.2016.01.021

Rate, FNR) and the negative class (False Positive Rate, FPR). The value of k is usually chosen equal to the imbalance level, i.e., the ratio between the cardinality of the negative and the positive sets.

- *AsymBoost** (hereafter abbreviated as *Asym**). This method, proposed in [24], is a variant of *Asym* and introduces two principal changes: a dynamic optimization of the FPR in each stage of the cascade and a control of the FNR in each round so as to avoid an exponential increment of the false positives.

- *RUSBoost-based Cascade* (hereafter abbreviated as *RUS*). This approach is based on the random undersampling of the majority class. To this end, we implemented in each node of the cascade the RUSBoost classifier [18]. The training data in input at each stage are undersampled so as to obtain a ratio of $35:65$ (positive:negative).

For the sake of comparison, we have also considered the original Viola-Jones cascade detector (hereafter abbreviated as *Ada*), where the node stage is built with an AdaBoost dichotomizer.

The detectors have been evaluated in terms of Receiver Operating Characteristics (ROC) curve by plotting True Positive Rate (TPR) against FPR for a series of thresholds on the confidence degree associated to each sample. In this case not all the ROC curve is of interest since only low values of FPR are acceptable. For this reason, our analysis was focused on the initial portion of the curve and, subsequently, we considered as performance measure the *partial Area Under the ROC Curve* (pAUC) defined as $pAUC(x) = \int_0^x TPR\ dFPR$ where $[0, x]$ is the range of interest for FPR [30].

To determine if *RankingCascade* performs significantly different from the other approaches we applied the bootstrap procedure [17]. The test set was sampled with replacement 2,000 times so that each new set of sampled data contained the same number of examples as the original set. We considered two different FPR ranges $[0, 10^{-3}]$ and $[0, 10^{-4}]$ which are close to practical application requirements for all the problems considered. For each range, the

https://doi.org/10.1016/j.ins.2016.01.021

Table 1: Performance differences between the compared detectors for a pAUC evaluated at 10^{-3} on the face detection problem.

Methods	**Mean Differences**	**p-value**
Ada	$2.19 \cdot 10^{-5}$	$<$ **0.001**
Asym	$-4.68 \cdot 10^{-6}$	0.842
*Asym**	$6.42 \cdot 10^{-7}$	0.430
RUS	$2.05 \cdot 10^{-5}$	$<$ **0.001**

Table 2: Performance differences between the compared detectors for a pAUC evaluated at 10^{-4} on the face detection problem.

Methods	**Mean Differences**	**p-value**
Ada	$6.33 \cdot 10^{-6}$	$<$ **0.001**
Asym	$3.29 \cdot 10^{-6}$	**0.001**
*Asym**	$1.07 \cdot 10^{-6}$	0.081
RUS	$1.98 \cdot 10^{-5}$	$<$ **0.001**

differences in pAUC between *RankingCascade* and the other detectors were computed. Resampling 2,000 times resulted in 2,000 values for each performance difference. To compare the performance of our approach against the other 4 methods we evaluated the p-value, defined as the fraction of the corresponding pAUC differences that were negative or zero. The statistical significance level was $\alpha = 0.01$ but, due to the number of comparisons, we applied the Bonferroni correction [6] and thus, performance differences were considered significant if $p < 0.0025$, (i.e., $p < \alpha/4$).

5.1. Face Detection

Face detection is the first analyzed problem. Following a common procedure in the literature [22, 23, 28], two different data sets have been created to train and test the cascades. The positives samples for the training phase were 1,600 subwindows extracted from the frontal faces of the well-known FERET database [14] and scaled in a standard format of 24×24 pixels. The negative samples were 22,906,769 subwindows, all of size 24×24, taken from the 9,028 images of a publicly available non-face dataset [28] at different scales. The positive samples were equally parted between training and validation set. The negative samples were divided in three data sets: training set, validation set and pool respectively of 20,000, 60,000 and 22,826,769 samples. As test set we adopted

https://doi.org/10.1016/j.ins.2016.01.021

the MIT+CMU database [20, 16] that consists of 180 images containing 734 faces. The negative set has been taken from 1,417 images randomly extracted from different categories (abbey, forests, greenhouse, shipyard, skyscraper) of the publicly available SUN database [29] for a total of 21,181,195 samples.

The cascade detectors were built using $d = 0.999$ and $f = 0.300$. The training stage produced 4 nodes for all the cascades except *Asym** that was composed by 6 stages. The total number of features considered at each stage was 116,544.

Results of the comparisons between detectors are reported in Tables 1 and 2 for the two FPRs considered in the pAUC evaluation. In both tables, the second column shows the mean difference between the pAUC of the proposed approach and the compared detectors while for each comparison the p-value is given in the third column. Statistically significant differences are listed in bold.

In Table 1 the performance of the proposed method is compared to the other cascades for an FPR lower than 10^{-3}. *RankingCascade* reveals to be significantly better than *Ada* and *RUS*. The differences, instead, are not statistically significant when compared to *Asym** and *Asym*. When looking at Table 2, i.e., for a pAUC evaluated between 0 and 10^{-4}, results are even better since the performance of our method becomes statistically higher than those of *Asym*. The difference with *Asym**, instead, is again not statistically significant. As a final remark, it is worth noting that *Asym* is a parametric approach whereas *Asym**, using a dynamic optimization of the FPR in each stage, tends to a classification system with a higher number of stages and a higher number of features per stage, so increasing the computational complexity of the cascade.

5.2. Microcalcification Detection

The second experiment deals with the problem of detecting microcalcifications on digital mammograms. Microcalcifications appear as bright small circular spots in the image and represent a subtle sign of breast cancer in women. A private full-field digital mammographic database of 198 images has been exploited to extract 8,000 microcalcifications (positive samples) and 22,760,320

https://doi.org/10.1016/j.ins.2016.01.021

Table 3: Performance differences between the compared detectors for a pAUC evaluated at 10^{-3} on the microcalcifications detection problem.

Methods	Mean Differences	p-value
Ada	$2.82 \cdot 10^{-5}$	$< \mathbf{0.001}$
Asym	$8.81 \cdot 10^{-6}$	$< \mathbf{0.001}$
*Asym**	$2.74 \cdot 10^{-5}$	$< \mathbf{0.001}$
RUS	$9.52 \cdot 10^{-5}$	$< \mathbf{0.001}$

Table 4: Performance differences between the compared detectors for a pAUC evaluated at 10^{-4} on the microcalcifications detection problem.

Methods	Mean Differences	p-value
Ada	$5.61 \cdot 10^{-6}$	$< \mathbf{0.001}$
Asym	$3.51 \cdot 10^{-6}$	$< \mathbf{0.001}$
*Asym**	$4.64 \cdot 10^{-6}$	$< \mathbf{0.001}$
RUS	$1.69 \cdot 10^{-5}$	$< \mathbf{0.001}$

background regions (negative class), all of size 12×12. Ten-fold cross validation has been performed considering 9 folds for training set, validation set and pool and the remaining fold as test set. In each cross validation step we used 7,200 positive samples equally parted between training and validation sets and 20,484,351 negative samples subdivided in 20,000, 60,000 and 20,404,351 respectively for training set, validation set and pool.

The cascade detectors were built using $d = 0.99$ and $f = 0.30$. The training stage consists in ten different cross validation runs and thus, for each method, ten different cascades were obtained. The number of stages for the *RankingCascade* was equal to 5 for all the runs while for the other approaches varied among 4 and 6 according to the cross validation step considered. The total number of features considered at each stage was 14,709.

The results of the comparisons for the two different FPRs used in pAUC evaluation (i.e., 10^{-3} and 10^{-4}) are reported respectively in Tables 3 and 4. In both cases the *RankingCascade* exhibits higher performance than all the other approaches even with a high statistical significance as proved by the very low p-values.

The effectiveness of the approach on this particular problem has been also confirmed by the results obtained in [1], where it has been employed in a

https://doi.org/10.1016/j.ins.2016.01.021

Table 5: Performance differences between the compared detectors for a pAUC evaluated at 10^{-3} on the microaneurysm detection problem.

Methods	**Mean Differences**	**p-value**
Ada	$2.82 \cdot 10^{-5}$	$<$ **0.001**
Asym	$8.81 \cdot 10^{-6}$	$<$ **0.001**
*Asym**	$2.74 \cdot 10^{-5}$	$<$ **0.001**
RUS	$9.52 \cdot 10^{-5}$	$<$ **0.001**

Table 6: Performance differences between the compared detectors for a pAUC evaluated at 10^{-4} on the microaneurysm detection problem.

Methods	**Mean Differences**	**p-value**
Ada	$5.61 \cdot 10^{-6}$	$<$ **0.001**
Asym	$3.51 \cdot 10^{-6}$	$<$ **0.001**
*Asym**	$4.64 \cdot 10^{-6}$	$<$ **0.001**
RUS	$1.69 \cdot 10^{-5}$	$<$ **0.001**

Computer-Aided Detection and Diagnosis (CAD) system that revealed to be competitive with the state-of-the-art commercial CAD systems.

5.3. Microaneurysm Detection

The third experiment deals with the problem of detecting microaneurysms on digital ocular fundus images. Microaneurysms appear as a dark small circular spots in the image and represent a subtle sign of retinopathy. A public database [15] of 50 digital ocular fundus images has been used to extract 2,890,972 patches of size 15×15 pixels, 1,997 containing microaneurysms (positive samples) and 2,888,975 containing background tissue (negative samples). Ten-fold cross validation has been performed considering 9 folds for training set, validation set and pool and the remaining fold as test set. In each cross validation step we used 1,797 positive samples equally parted between training and validation sets and 2,620,077 negative samples subdivided in 10,000, 10,000 and 2,600,077 respectively for training set, validation set and pool.

The cascade detectors were built using $d = 0.95$ and $f = 0.30$. The training stage consists in ten different cross validation runs and thus, for each method, ten different cascades were obtained. The number of stages for all the methods considered varied among 14 and 16 according to the cross validation step considered, except for *RUSBoost* where it varied among 3 and 5. The total number

https://doi.org/10.1016/j.ins.2016.01.021

Table 7: Average execution time (in seconds) for selecting a weak learner

	Face Detection	MC Detection	MA Detection
RankingCascade	**22.6**	**3.5**	**3.7**
Ada	104.7	9.1	12.4
Asym	89.6	11.6	15.8
*Asym**	92.2	10.6	15.4
RUS	19.1	3.1	3.3

Table 8: Average execution time (in seconds) for building a node classifier

	Face Detection	MC Detection	MA Detection
RankingCascade	**360.0**	**49.4**	**103.0**
Ada	830.7	107.0	306.2
Asym	1030.1	135.9	400.3
*Asym**	1625.6	150.9	509.2
RUS	248.4	34.1	67.5

of features considered at each stage was 43,172.

The results of the comparisons for the two different FPRs used in pAUC evaluation (i.e., 10^{-3} and 10^{-4}) are reported respectively in Tables 5 and 6. In both cases the *RankingCascade* exhibits higher performance than all the other approaches even with a high statistical significance as proved by the very low p-values.

5.4. Computational time

To confirm the effectiveness of the implementation proposed in Section 4, for each performed experiment we report in Table 7 the average execution time for selecting a weak learner and in Table 8 the average execution time for building a node classifier. All the considered methods have been implemented in C++ with multi-threading and run on a workstation equipped with two Intel Xeon E5520 and 96.0 GB of RAM. Remarkably, *RankingCascade* performed faster than all the other learning strategies except *RUS*, which takes advantage of undersampling to drastically reduce the number of calculations (at the cost of worse detection performance, as shown before).

https://doi.org/10.1016/j.ins.2016.01.021

6. Conclusions

In this paper we have addressed the problem of class imbalance in the node classifier learning within the standard Viola-Jones cascade framework and proposed a new learning strategy aimed at maximizing the node classifiers ranking capability rather than their accuracy. Such approach revealed to be an effective solution to the class asymmetry problem and provided good performance in experiments when compared with other approaches. In particular, when tested on three real-world severe detection problems, our learning strategy provided similar or better results than methods, such as *AsymBoost* and *AsymBoost**, which rely on a cost-based approach to face class imbalance. In this regard, it is worth mentioning that our method (as well as *AsymBoost**) does not introduce any further parameter to the original cascade whereas *AsymBoost* requires a cost ratio to be specified. In the same experiments, the proposed approach showed significantly better results also when compared with a cascade using *RUSBoost*, so demonstrating that the ranking-based strategy in node classifier learning performs better than undersampling. In summary, the experimental results supported the rationale on which our method is based, i.e., that ranking-based learning is effective in facing class imbalance and allows the construction of proficient cascade detectors. Possible directions for future work include evaluating the effectiveness of the proposed approach also in other cascade architectures which extend the standard Viola-Jones framework.

References

[1] A. Bria, N. Karssemeijer, F. Tortorella, Learning from unbalanced data: A cascade-based approach for detecting clustered microcalcifications, Medical Image Analysis 18 (2) (2014) 241–252.

[2] A. Bria, C. Marrocco, M. Molinara, F. Tortorella, A ranking-based cascade approach for unbalanced data, in: Pattern Recognition (ICPR), 2012 21st International Conference on, 2012, pp. 3439–3442.

https://doi.org/10.1016/j.ins.2016.01.021

[3] R. Caruana, Learning from Imbalanced Data: Rank Metrics and Extra Tasks, in: The AAAI Workshop on Learning from Imbalanced Data Sets, 2000, pp. 51–57.

[4] N. V. Chawla, K. W. Bowyer, L. O. Hall, W. P. Kegelmeyer, SMOTE: Synthetic Minority Over-sampling Technique, Journal of Artificial Intelligence Research 16 (1) (2002) 321–357.

[5] N. V. Chawla, A. Lazarevic, L. O. Hall, K. W. Bowyer, SMOTEBoost: Improving Prediction of the Minority Class in Boosting, in: N. Lavrač, D. Gamberger, L. Todorovski, H. Blockeel (eds.), Knowledge Discovery in Databases: PKDD 2003, vol. 2838 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2003, pp. 107–119.

[6] O. J. Dunn, Multiple Comparisons Among Means, Journal of the American Statistical Association 56 (293) (1961) 52–64.

[7] W. Fan, S. J. Stolfo, J. Zhang, P. K. Chan, AdaCost: Misclassification Cost-Sensitive Boosting, in: Proceedings of the Sixteenth International Conference on MachineLearning, ICML '99, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999, pp. 97–105.

[8] Y. Freund, R. Iyer, R. E. Schapire, Y. Singer, An efficient boosting algorithm for combining preferences, Journal of Machine Learning Research 4 (2003) 933–969.

[9] Y. Freund, R. E. Schapire, A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting, Journal of Computer and System Sciences 55 (1) (1997) 119–139.

[10] R. Lienhart, E. Kuranov, V. Pisarevsky, Empirical Analysis of Detection Cascades of Boosted Classifiers for Rapid Object Detection, in: DAGM 25th Pattern Recognition Symposium, 2003, pp. 297–304.

[11] H. Masnadi-Shirazi, N. Vasconcelos, Cost-Sensitive Boosting, Pattern Analysis and Machine Intelligence, IEEE Transactions on 33 (2) (2011) 294–309.

https://doi.org/10.1016/j.ins.2016.01.021

[12] M. Molinara, M. Ricamato, F. Tortorella, Facing Imbalanced Classes through Aggregation of Classifiers, in: Image Analysis and Processing, 2007. ICIAP 2007. 14th International Conference on, 2007, pp. 43–48.

[13] M.-T. Pham, V.-D. Hoang, T.-J. ChamP, Detection with multi-exit asymmetric boosting, in: Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on, 2008, pp. 1–8.

[14] P. Phillips, H. Moon, S. Rizvi, P. Rauss, The FERET evaluation methodology for face-recognition algorithms, Pattern Analysis and Machine Intelligence, IEEE Transactions on 22 (10) (2000) 1090–1104.

[15] P. Prentasic, S. Loncaric, Z. Vatavuk, G. Bencic, M. Subasic, T. Petkovic, L. Dujmovic, M. Malenica-Ravlic, N. Budimlija, R. Tadic, Diabetic Retinopathy Image Database (DRiDB): a new database for diabetic retinopathy screening programs research, in: Image and Signal Processing and Analysis (ISPA), 8th International Symposium on, IEEE, 2013, pp. 711–716.

[16] H. Rowley, S. Baluja, T. Kanade, Neural network-based face detection, Pattern Analysis and Machine Intelligence, IEEE Transactions on 20 (1) (1998) 23–38.

[17] F. W. Samuelson, N. Petrick, Comparing image detection algorithms using resampling, in: Biomedical Imaging: Nano to Macro, 2006. 3rd IEEE International Symposium on, 2006, pp. 1312–1315.

[18] C. Seiffert, T. Khoshgoftaar, J. Van Hulse, A. Napolitano, RUSBoost: A Hybrid Approach to Alleviating Class Imbalance, Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on 40 (1) (2010) 185–197.

[19] Y. Sun, A. Wong, Y. Wang, Parameter Inference of Cost-Sensitive Boosting Algorithms, in: P. Perner, A. Imiya (eds.), Machine Learning and Data

https://doi.org/10.1016/j.ins.2016.01.021

Mining in Pattern Recognition, vol. 3587 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2005, pp. 21–30.

[20] K.-K. Sung, T. Poggio, Example-based learning for view-based human face detection, Pattern Analysis and Machine Intelligence, IEEE Transactions on 20 (1) (1998) 39–51.

[21] K. M. Ting, A Comparative Study of Cost-Sensitive Boosting Algorithms, in: Proceedings of the Seventeenth International Conference on Machine Learning, ICML '00, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2000, pp. 983–990.

[22] P. Viola, M. Jones, Robust Real-Time Object Detection, Int. J. of Comp. Vis. 57 (2) (2001) 137–154.

[23] P. Viola, M. Jones, Fast and Robust Classification using Asymmetric AdaBoost and a Detector Cascade, Adv. in Neur. Inf. Proc. Syst. 16 (2002) 1311–1318.

[24] I. Visentini, C. Micheloni, G. L. Foresti, Reactive Learning Strategy for AsymBoost Based Face Detectors, 14th International Conference on Image Analysis and Processing (2007) 357–362.

[25] P. Wang, C. Shen, H. Zheng, Z. Ren, Training a multi-exit cascade with linear asymmetric classification for efficient object detection, in: Image Processing (ICIP), 2010 17th IEEE International Conference on, 2010, pp. 61–64.

[26] G. Weiss, F. Provost, Learning When Training Data are Costly: The Effect of Class Distribution on Tree Induction, Journal of Artificial Intelligence Research 19 (2003) 315–354.

[27] G. M. Weiss, Mining with Rarity: A Unifying Framework, SIGKDD Explor. Newsl. 6 (1) (2004) 7–19.

https://doi.org/10.1016/j.ins.2016.01.021

[28] J. Wu, S. Brubaker, M. Mullin, J. Rehg, Fast Asymmetric Learning for Cascade Face Detection, Pattern Analysis and Machine Intelligence, IEEE Transactions on 30 (3) (2008) 369–382.

[29] J. Xiao, J. Hays, K. Ehinger, A. Oliva, A. Torralba, SUN database: Large-scale scene recognition from abbey to zoo, in: Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, 2010, pp. 3485–3492.

[30] W. A. Yousef, Assessing classifiers in terms of the partial area under the ROC curve, Computational Statistics & Data Analysis 64 (2013) 51–70.

https://doi.org/10.1016/j.ins.2016.01.021