Boosting Object Detection in Cyberphysical Systems

José M. Buenaposada[1][0000−0002−4308−9653] and Luis Baumela[2]

[1] Univ. Rey Juan Carlos, Spain. josemiguel.buenaposada@urjc.es
[2] Univ. Politécnica de Madrid, Spain. lbaumela@fi.upm.es

Abstract. The construction of Cyberphysical systems requires providing intelligent behavior to physical agents at the smallest scale and, therefore, the need to develop very efficient and resource-aware algorithms. In this paper we present an object detection algorithm that may endow an agent with perceptual object detection capabilities at a small computational cost. To this end we adapt a recent Multi-class Boosting scheme to create an efficient detector with the capability of regressing the object bounding box. In the experiments we prove that the resulting algorithm shows Average Precision (AP) improvements in a multi-view car detection problem.

Keywords: Object Detection · Multi-class Boosting · Cyberphysical Systems

1 Introduction

The last years have witnessed such evolution in internet technology that almost everything can now be connected transparently and seamlessly through the Internet of Things (IoT). In parallel, advances in Artificial Intelligence enable the construction of autonomous agents that perceive and take actions in the real world. In this context Cyberphysical (CBP) systems emerge as an evolution of the IoT in which physical objects not only have computing and communication abilities, but also sensing and operation capacities, enabling them to co-operate in the construction of distributed and autonomous ecosystems [15, 11].

Perceptual skills, such as for example detecting and recognizing objects of interest, is a requirement for a CBP agent to interact with the environment. Powered by the use of deep neural nets, modern object detection algorithms have achieved remarkable performance [8]. However, these approaches require advanced computational resources such as Graphical Processing Units (GPUs). Although there is an ever-increasing number of devices shipping GPUs, it is also true that such intelligent behavior is required at the smallest scale, such as in micro-scale mobile robots, that are inherently limited in powering and computational capabilities [2]. Hence, the necessity of developing very efficient algorithms, that are aware of the time and energy required for their execution [15].

In this paper we propose a detection algorithm based on Viola and Jones seminal Boosting scheme [12]. Boosting classifiers have been extensively used

for building multi-class object detectors [14]. This approach has received much attention because it is very efficient and achieves very good performance in various object detection problems [1, 9]. The key for its success is the exploitation of the feature selection capabilities of Boosting together with efficient image descriptions such as the *Integral Channel Features* [3]. The usual framework for Boosting-based object detection uses binary classification (e.g. AdaBoost). In this regard, multi-class detection problems are usually solved with K detectors, one per object view or positive class. Here we propose the use of a single multi-class Boosting algorithm.

A key methodological advance in object detection is the bounding box refinement. When dealing with objects that can present different aspect ratios depending on their pose or configuration, the bounding box refinement step is crucial to get better precision. Recent CNN-based detectors already perform bounding box parameters regression, e.g. [10].

In this paper we improve the Boosting-based object detection paradigm [12] in two ways. First using BAdaCost [4], a recent multi-class cost-sensitive Boosting algorithm. With it we get a precise control over class boundaries (e.g. errors between positive classes). Hence improving the performance compared to approaches based on plain binary classifiers, e.g. [7]. Second, we extend BAdaCost so it is able to regress the detected target bounding box. We present our approach in a car detection problem and evaluate it with the KITTI benchmark. In the experiments we show that our approach results in an improvement in AP from previous baseline results.

2 Multi-class boosting algorithm

A Boosting algorithm is a supervised learning scheme that requires N training data instances $\{(\mathbf{x}_i, l_i)\}_{i=1}^{N}$, where $\mathbf{x}_i \in X$ encodes the object to be classified with class label $l_i \in L = \{1, 2, \ldots, K\}$. Each label $l \in L$ has a corresponding margin vector $\mathbf{y}_l \in Y$ where $Y = \{\mathbf{y}_1, \ldots, \mathbf{y}_K\}$ [4]. $\mathbf{y}_l$ has a value 1 in the l-th coordinate and $\frac{-1}{K-1}$ elsewhere. So, if $l = 1$, the margin vector representing class 1 is $\mathbf{y}_1 = \left(1, \frac{-1}{K-1}, \ldots, \frac{-1}{K-1}\right)^\top$. Hence, it is immediate to see the equivalence between classifiers G defined over L and classifiers $\mathbf{g}$ defined over Y, $G(\mathbf{x}) = l \in L \;\Leftrightarrow\; \mathbf{g}(\mathbf{x}) = \mathbf{y}_l \in Y$.

2.1 BAdaCost: Cost-sensitive Multi-class Boosting classification

Cost-sensitive classification endows the traditional Boosting scheme with the capability to to modify pair-wise class boundaries. In this way, we can reduce the number of errors between positive classes (e.g. different target orientations) and improve recall when object classes have different aspect ratios. To this end we use BAdaCost [4] (Boosting Adapted for Cost matrix), a recently introduced multi-class cost-sensitive Boosting classifier. In this section we briefly introduce it.

The costs are encoded in a $K \times K$-matrix $\mathbf{C}$, where each entry $C(i,j)$ represents the cost of miss-classifying an instance with real label i as j. Here it is assumed that $C(i,i) = 0, \forall i \in L$, i.e. the cost of correct classifications is zero.

Let $\mathbf{C}^*$ be a $K \times K$-matrix defined in the following way

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

In a cost-sensitive classification problem each value $C^*(j,j)$ represents a "reward" associated to a correct classification. The j-th row in $\mathbf{C}^*$, denoted as $\mathbf{C}(j,-)$, is a margin vector that encodes the costs associated to the j-th label. The multi-class cost-sensitive margin associated to instance $(\mathbf{x}, l)$ is given by $z_C := \mathbf{C}^*(l,-) \cdot \mathbf{g}(\mathbf{x})$. 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, using this generalized margin, BAdaCost defines a *Cost-sensitive Multi-Class Exponential Loss Function* (*CMELF*):

$$\mathcal{L}_C(l, \mathbf{g}(\mathbf{x})) := \exp(z_C) = \exp\left(\mathbf{C}^*(l,-) \cdot \mathbf{g}(\mathbf{x})\right) = \exp\left(\frac{K}{K-1}\mathbf{C}^*(l, G(\mathbf{x}))\right). \tag{2}$$

The margin, z_C, yields negative values when the classification is correct under the cost-sensitive point of view, and positive values for costly (wrong) outcomes. The CMELF is a generalization of the Multi-class Exponential Loss introduced in [17].

BAdaCost resorts to the CMELF (2) for evaluating classifications encoded with margin vectors. The expected loss is minimized using a stage-wise additive gradient descent approach. The strong classier that arises has the following structure:

$$\mathtt{H}(\mathbf{x}) = \arg\min_k \left(\mathbf{C}^*(k,-) \cdot \sum_{m=1}^{M} \beta_m \mathbf{g}_m(\mathbf{x})\right) = \arg\min_k \left(\mathbf{C}^*(k,-) \cdot \mathbf{f}(\mathbf{x})\right), \tag{3}$$

where $\mathbf{f}(\mathbf{x})$ is a linear combination of M cost-sensitive weak learners, $\{\mathbf{g}_m(\mathbf{x})\}_{m=1}^{M}$, that the algorithm learns incrementally. In this case $\mathbf{f}(\mathbf{x})$ is a vector with the estimated per-class costs from the feature vector $\mathbf{x}$.

2.2 Object detection score for BAdaCost

When building an object detector it is necessary to have a confidence measure or detection score. In BAdaCost the predicted costs incurred when classifying sample $\mathbf{x}$ in one of the K classes are given by the vector:

$$\mathbf{c} = \mathbf{C}^* \cdot \mathbf{f}(\mathbf{x}) = (c_1, \ldots, c_K)^\top. \tag{4}$$

From now on, in multi-class detection problems, we assume that the background (negative) class has label $l = 1$ and the object classes (e.g. different views

of a car) have label $l > 1$. Therefore, we can compute the score of $\mathbf{x}$ as

$$s(\mathbf{x}) = (c_1 - min(c_2, \ldots, c_K)). \tag{5}$$

This score has desirable properties for detection problems: 1) $s(\mathbf{x}) > 0$ when the winner class (i.e. the class with lowest cost) has label $l > 1$; 2) $s(\mathbf{x}) < 0$ when the winner class is $l = 1$. Given that score definition, we can use any cascade calibration algorithm for Boosting, for example [16], and stop execution of weak learners whenever the score falls below a calibrated threshold.

3 Bounding Box Aspect ratio estimation

In our experiments we build a multi-view car detector. One of the challenges in the car detection problem is that the bounding box aspect ratio (AR) changes with the view (i.e. frontal cars have lower AR than side view ones). By posing a classification problem where the labels are the 20 car views (plus no-car label), we can also compute information related to the AR in the Boosted tree leaves. The overall approach to simultaneous detection and AR estimation is shown in Fig. 1.

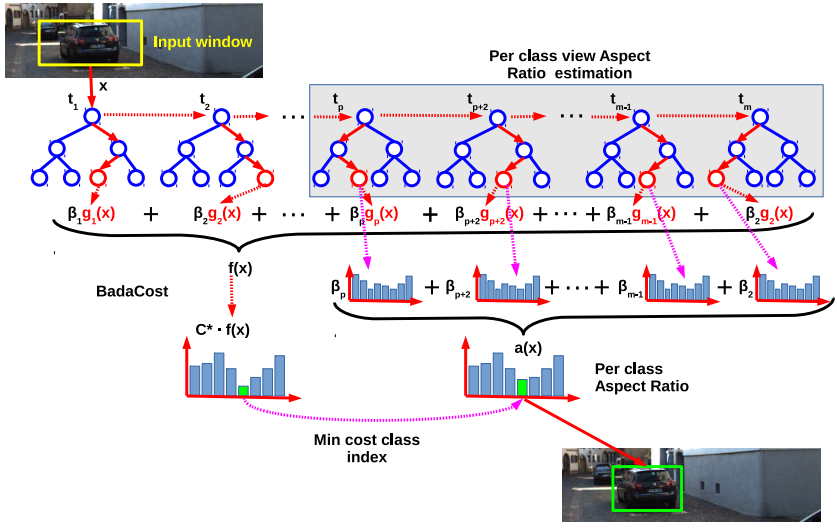

Fig. 1: Algorithmic pipeline. BAdaCost learns an ensemble of multi-class cost-sensitive trees. The estimation of AR distribution is computed using all the trees starting with the t_p one. The final AR is the one of the the minimal cost class.

The classifier learns m weak-learners that are cost-sensitive decision trees. The split measure used in each tree node is the Gini impurity. The modifications of the tree to make it cost-sensitive are two-fold:

1. On each split node S the probability of class l, $p(l)$, is multiplied by the percentage of costs associated to class l (i.e. sum of a row in the cost matrix), $c(l)$:

$$p'(l) = \underbrace{\frac{\sum_{i \in S} w_i I(y_i = l)}{\sum_{i \in S} w_i}}_{p(l)} \underbrace{\frac{\sum_{k=1}^{K} C(l,k)}{\sum_{i=1}^{K}\sum_{j=1}^{K} C(i,j)}}_{c(l)}, \tag{6}$$

   where $I()$ is the indicator function.
2. On each leaf node the minimum cost rule is applied for classification:

$$h = \arg\min_{l} \mathbf{C}(l,-)(p(1),\ldots,p(K))^{\top}. \tag{7}$$

During the training phase, for every decision tree and leaf node S, we store: 1) the minimum cost label, h_S and 2) a $K \times 1$ vector, $\mathbf{a}_S$, with the mean AR of each view class. During the detection phase, the trees are traversed with the feature vector $\mathbf{x}$ corresponding to a candidate window (see Fig.1). As we have seen in section 2, the vector $\mathbf{f}(\mathbf{x})$ is computed as a linear combination of tree labels outputs h, codified as its corresponding margin vector $\mathbf{g}(\mathbf{x}) = \mathbf{y}_h \in Y$. Vector $\mathbf{f}(\mathbf{x})$ is then used in equation (3) to obtain the minimum cost view class estimation.

Our procedure to estimate the aspect ratio follows a similar approach. Let $\mathbf{a}_t(\mathbf{x})$ be the per class view aspect ratios stored in the leaf node of t-th tree in which $\mathbf{x}$ ends. After traversing the weak learners trees, the vector of class aspect ratios is computed as a linear combination: $\mathbf{a}(\mathbf{x}) = \sum_{i=t_p}^{M} \beta_i \mathbf{a}_i(\mathbf{x})$. If h is the class estimated by the BAdaCost strong learner, $H(\mathbf{x})$, then the estimated AR is given by $\mathbf{a}(h)$. Note here that we drop all the trees below the t_p-th one. The rationale is that in the first trees the strong classifier are not accurate enough. In the experimental section we will see that this is in fact true and it is better to use only the final trees in the ensemble to estimate the AR.

4 Experiments

In our experiments we have modified Piotr Dollar's Matlab Toolbox[3] with BAdaCost (e.g adding cost-sensitive decision trees and multi-class detection). Our modified implementation with the BAdaCost detectors is already available[4].

In the experiments we use a detection problem in which the target object changes its Bounding Box AR depending on the view angle. Car detection in the KITTI dataset [6] is a good example of this kind of problem. The database presents three subsets: easy, moderate and hard (easy $\subset$ moderate $\subset$ hard). We carry out the evaluation in each level separately. 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).

[3] https://github.com/pdollar/toolbox
[4] https://github.com/jmbuena/toolbox.badacost.public

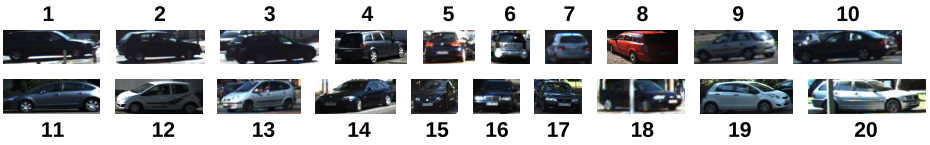

Fig. 2: KITTI cars classes we use in our experiments.

We divide the images into K=20 view classes (see Fig. 2). With the BAdaCost algorithm we use cost-sensitive decision tree weak learners that select features from LDCF channels [13]. In all the experiments we train a car model of size 48×84 pixels, AR= 1.75. We start the pyramid one octave above the actual image size, to detect cars 25 pixel high. This approach produces detection bounding boxes with fixed AR of 1.75 (Fixed-Equal approach). On the other hand, since the multi-class detector outputs the view class, we can correct the fixed size window to the training mean (Fixed-Class-Mean approach) AR of the predicted class view as done in [9, 5].

We train the classifier storing the mean AR of each class in tree leaves as explained in section 3. During training 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=8, the number of negatives per round to add to $N = 7500$ and the total amount of negatives to $NA = 30000$.

The costs matrix is set to weight up gross errors between view classes. This is important because estimating the wrong class will output a Bounding Box with the wrong AR (e.g. frontal car, AR=1.0, to left side car, AR>>1.0). We show the cost matrix we use in Fig. 3. The non-car class has label 1. Positive classes have the labels shown in Fig. 2 plus one.

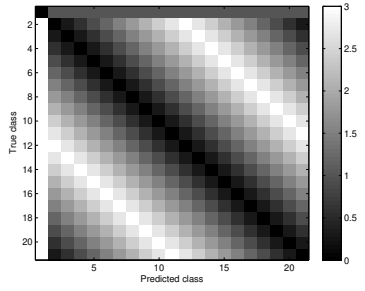

Fig. 3: Costs matrix used in our experiments.

First, we train only one detector with the mean aspect ratio of each view class stored on the tree leaves. The detector uses KITTI-train90 for training

and KITTI-train10 for testing. We set the detection threshold to an entersection over union (IoU) value 0.7, es established in the KITTI benchmark. Then, we use different strategies to estimate the ARs using this detector. First we test the AR estimation algorithm introduced section 3 with different values of t_p (first tree to use in the estimation). In Fig. 4 we show that neither using all the trees nor using the last few trees are the best strategies. We can see that, in the Moderate KITTI car detection problem, using all trees starting with $t_p = 950$ we get the best result.

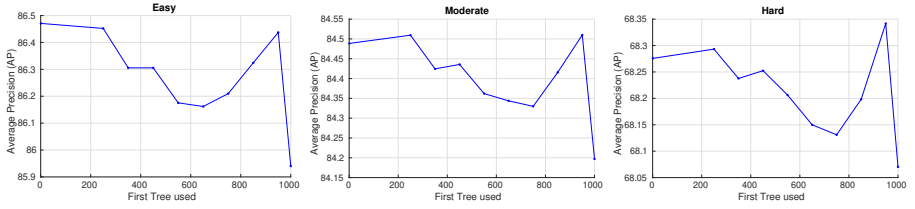

Fig. 4: Using different number of weak learners for estimating the detections AR. Here we train in KITTI-train90 and test in the validation subset, KITTI-train10.

Secondly, once we have found the best t_p value we can compare it with other AR estimation strategies: use the output of the fixed size sliding window detector (Fixed-Equal), modify the fixed aspect ratio window with the estimated class view aspect ratio (Fixed-Class-Mean) and, finally, our proposal (Estimated-AR). In Fig. 5 we confirm that a fixed aspect ratio detector as Fixed-Equal, gets the worst results. We get a much better result in the Moderate KITTI subset (the one used for ranking) with the Fixed-Class-Mean procedure. On the other hand, we can improve even further the AP by using our Estimated-AR strategy. We get an better AP by 1.7%, 1.2% and 1.1%, respectively, in the Easy, Moderate and Hard settings. Given that our procedure is computationally cheap it is a significant improvement. On Fig. 6 we show results of the different methods on images were our method (Estimated-AR) improves over the baseline (Fixed-Class-Mean).

To further analyze the performance of our procedure (Estimated-AR) with respect to the baseline (Fixed-Class-Mean), we have performed an additional experiment varying the IoU threshold (see table 1). The good behavior of our approach is more evident when we look for higher overlapping in the detection. With a threshold of IoU= 0.8, Estimated-AR, in Moderate subset, is better by 11,53% (from 44.2% to 49.3%) and with IoU= 0.9 it is better by 63.15% (from 1.9% to 3.1%).

5 Conclusions

Detection algorithms have evolved over time by changing various components of the pipeline. Some of these improvements, however, have been exploited only in

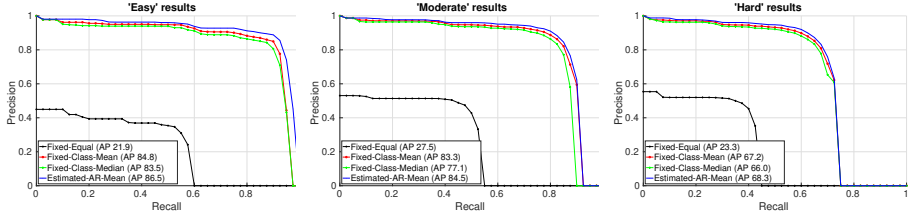

Fig. 5: Comparing different strategies to compute the detection AR. Here we train in KITTI-train90 and test in the validation subset, KITTI-train10.

Fig. 6: Sample car detection improvement with AR estimation on KITTI-train10. In green, red and yellow we show respectively the ground truth, Estimated-AR-Mean and Fixed-Class-Mean true positives for the data base moderate settings.

Table 1: AP for different IoU values on the KITTI train90/train10 experiment

	Algorithm / IoU	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**
Easy	Fixed-Class-Mean	95.6 %	94.9 %	84.8 %	44.2 %	1.5 %
	Estimated-AR	95.6 %	94.8 %	86.4 %	49.3 %	2.8 %
Moderate	Fixed-Class-Mean	90.3 %	89.8 %	83.3 %	44.2 %	1.9 %
	Estimated-AR	90.3 %	89.8 %	84.5 %	47.7 %	3.1 %
Hard	Fixed-Class-Mean	79.5 %	78.2 %	67.2 %	36.9 %	1.7 %
	Estimated-AR	79.5 %	78.1 %	68.3 %	39.1 %	2.9 %

the context of modern deep neural nets. In this paper we improve the performance of Boosting-based detectors by refining the target bounding box using a new cost-sensitive multi-class boosting scheme. This is a relevant result in the construction of Cyberphysical Systems, given the computational efficiency of this family of algorithms.

In the experiments we show that our approach improves the detection AP with respect to the baseline Fixed-Class-Mean regressor. Moreover, it beats the results of its closest Boosting competitor [7]. This Boosting-based detector achieves 52.9% AP in the moderate KITTI testing set, where as our result is 67.23%.

If we analyze the results in the moderate set in Table 1 we can see that for an IoU of 0.5, we achieve a result above 90% AP. However, as the IoU threshold increases, the AP goes down to 3.1% in the most demanding case, IoU=0.9. This means that most of the detections are correct, but the accurate location of the object bounding box is an important source of errors that should be further studied in the future.

The use of proper data augmentation and alternative and more accurate bounding box regression algorithms are future research avenues that will further improve AP with no extra computing cost.

Acknowledgements

The authors gratefully acknowledge funding from the Spanish Ministerio de Economía y Competitividad, project TIN2016-75982-C2-2-R.

References

1. Benenson, R., Mathias, M., Timofte, R., Van Gool, L.: Pedestrian detection at 100 frames per second. In: Proc. IEEE Conf. on Computer Vision and Pattern Recognition (2012)
2. Diller, E., Metin Sitti, M.: Micro-scale mobile robotics. Foundations and Trends in Robotics **2**(3), 143–259 (2013)
3. Dollar, P., Appel, R., Belongie, S., Perona, P.: Fast feature pyramids for object detection. IEEE Trans. Pattern Analysis and Machine Intelligence **36**(8), 1532–1545 (2014)
4. Fernández-Baldera, A., Buenaposada, J.M., Baumela, L.: Multi-class boosting for imbalanced data. In: Proc. of Iberian Conf. Pattern Recognition and Image Analysis. pp. 57–64 (2015)
5. Fernández-Baldera, A., Buenaposada, J.M., Baumela, L.: Badacost: Multi-class boosting with costs. Pattern Recognition **79**, 467 – 479 (2018)
6. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the kitti vision benchmark suite. In: Proc. IEEE Conf. on Computer Vision and Pattern Recognition (2012)
7. Juranek, R., Herout, A., Dubska, M., Zemcik, P.: Real-time pose estimation piggybacked on object detection. In: Proc. Int'l Conf. Computer Vision (December 2015)

8. Liu, L., Ouyang, W., Wang, X., Fieguth, P.W., Chen, J., Liu, X., Pietikäinen, M.: Deep learning for generic object detection: A survey. CoRR **abs/1809.02165** (2018), http://arxiv.org/abs/1809.02165
9. Ohn-Bar, E., Trivedi, M.: Learning to detect vehicles by clustering appearance patterns. IEEE Trans. Intelligent Transportation Systems **16**(5), 2511–2521 (Oct 2015)
10. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: Towards real-time object detection with region proposal networks. In: Conf. Neural Information Processing Systems (2015)
11. Serpanos, D.: The cyber-physical systems revolution. Computer **51**(3), 70–73 (March 2018)
12. Viola, P.A., Jones, M.J.: Fast and robust classification using asymmetric adaboost and a detector cascade. In: Conf. Neural Information Processing Systems. pp. 1311–1318 (2001)
13. W. Nam, P. Dollar, J.H.H.: Local decorrelation for improved pedestrian detection. In: Conf. Neural Information Processing Systems (2014)
14. Wang, X., Yang, M., Zhu, S., Lin, Y.: Regionlets for generic object detection. IEEE Trans. Pattern Analysis and Machine Intelligence **37**(10), 2071–2084 (2015)
15. Wolf, W.: Cyber-physical systems. Computer **42**(3), 88–89 (March 2009)
16. Zhang, C., Viola, P.A.: Multiple-instance pruning for learning efficient cascade detectors. In: Conf. Neural Information Processing Systems. pp. 1681–1688 (2007)
17. Zhu, J., Zou, H., Rosset, S., Hastie, T.: Multi-class AdaBoost. Statistics and Its Interface **2**, 349–360 (2009)