The ROC-Boost Design Algorithm for Asymmetric Classification

Guido Cesare
Department of Mathematics
University of Genova (Italy)
guido.cesare@gmail.com

Roberto Manduchi
Department of Computer Engineering
University of California, Santa Cruz
manduchi@soe.ucsc.edu

***Abstract*—In many situations (e.g., cascaded classification), it is desirable to design a classifier with precise constraints on its detection rate or on its false positive rate. We introduce ROC-Boost, a modification of the AdaBoost design algorithm that produces asymmetric classifiers with guaranteed detection rate and low false positive rates. Tested in a visual text detection task, ROC-Boost was shown to perform competitively against other popular algorithms.**

I. Introduction

AdaBoost [6], [7] is a well-known design technique for binary classification, owing its success to a simple and intuitive design rule, proven generalization properties, and the ability to "select" relevant features in a possibly large-dimensional space. A popular implementation of AdaBoost for image classification due to Viola and Jones [11] uses a cascaded approach for improved computational efficiency. The first classifiers in the cascade operate on a set of features that can be computed quickly. More complex and computationally demanding features are left for later stages in the cascade, which are activated only if all previous stages have resulted in positive detection. This approach can be very effective at reducing the average computational cost of classification in applications such as object detection, characterized by low prior probability of occurrence [4], [2].

Assuming statistical independence of the classifiers in the different stages of the cascade, the detection rate and the false positive rate of the cascaded classifier are equal to the product of the individual classifiers' detection rates and false positive rates, respectively. The designer is thus faced with the problem of setting target performances for the individual classifiers in the cascade in order to achieve a desired overall performance level. For example, if the overall system specifications dictate a guaranteed minimum detection rate of $D_{\min}$, the designer may decide to "spread" this value across the N stages of the cascade by imposing a minimum detection rate of $d_{\min} = D_{\min}^{\frac{1}{N}}$ to each classifier, while producing as few false positives as possible. The problem with this approach is that AdaBoost does not provide a simple mechanism to specify bounds on detection or false positive rates – it only gives guaranteed bounds on the (empirical) error rate. Note that cascaded classification is just one example of systems required to support asymmetric specifications. Another example is given by a safety system that can accept a moderate rate of false alarms but cannot tolerate missing potentially critical events.

This contribution introduces a modification of the AdaBoost design algorithm that allows one to specify a minimum guaranteed detection rate, while keeping the false positive rate as low as possible. More precisely, we derive a lower bound for the detection rate and an upper bound for the false positive rate; then, we find parameters that meet the minimum detection rate specification while minimizing the bound on the false positive rate. This strategy generalizes AdaBoost's approach of finding parameters that minimize an upper bound on the empirical error rate. Our design technique allows one to effectively "explore" the ROC curve achievable by AdaBoost with different design parameters; for this reason, we named it "ROC-Boost". In this contribution we introduce the theory underlying ROC-Boost design, and derive closed-form expressions for the classifiers' parameters. We use our algorithm to design asymmetric classifiers trained to detect the presence of text in a given image area, using visual features similar to those proposed by Chen and Yuille [3]. We provide comparative results in terms of the Pareto front of the achievable ROC curves over a range of design targets. The use of Pareto curves enables fair comparison with competing algorithms. Our experimental results show that the ROC-Boost design algorithm compares favorably with AsymBoost, and that it has comparable generalization properties.

II. Related Work

AdaBoost, developed by Freund and Shapire in the second half of the nineties [6], [7], quickly became part of the arsenal of researchers and practitioners in machine learning. In the computer vision community, AdaBoost was popularized by Viola and Jones' work in face and object recognition [11]. In particular, Viola and Jones introduced two important innovations: the use of AdaBoost for feature selection in high-dimensional feature space, and the use of AdaBoost in a cascaded architecture for improved efficiency. The success of cascaded classifiers spurred new research into AdaBoost-like algorithms able to support asymmetric performance. For example, AsymBoost [12] modifies the weight update step in an AdaBoost iteration, giving higher weight to positive examples. A different approach was taken by Wu *et al.* [13]. In their work, the individual hypotheses (weak classifiers) selected

by the standard AdaBoost (or Asymboost) algorithm are used in the final (strong) classifier, but the linear coefficients are different than those selected by AdaBoost (Asymboost). More precisely, the ensemble of outputs from the T weak classifiers is considered as a binary vector in R^T, and a Linear Asymmetric Classifier (LAC) is defined in this space, designed in such a way as to guarantee a false positive rate of 0.5. The resulting algorithm bears a formal resemblance with the classic Fisher Linear Discriminant (FDA), and in some cases even better results were obtained by performing FDA rather then LAC classification on the binary vectors from the individual hypotheses. Shen *et al.* [10] showed that FDA can be viewed as a regularized version of LAC, and extended the work of Wu *et al.* by proposing a criterion for hypothesis selection that relies on the column generation technique [5] while still aiming for a false positive rate of 0.5.

An algorithm for text detection using cascaded AdaBoost classification was proposed by Chen and Yuille [3]. Chen and Yuille [4] and Brubaker *et al.* [2] studied the issue of optimal feature partition across different layers of the cascade.

III. The ROC-Boost Design Algorithm

We begin by introducing some necessary terminology and by briefly recalling the standard AdaBoost design algorithm. We assume that a training set with M binary labeled data $\{x_i, y_i\}$ is available. x_i is a feature vector describing the i-th data point; y_i, the label assigned to it, can only take values of 0 ('negative example') or 1 ('positive example'). A binary classifier is a function $h(x)$ that maps a feature vector x to either 0 or 1. Note that x can have large dimensionality; we will say that the feature vector x is composed by different *features* x^j, where x^j is a proper subvector of x. Examples of features are given in Sec. IV.

Given a weight distribution $w = (w_1, \ldots, w_M)$, we define the empirical detection rate d, false positive rate f, and error rate ϵ of a classifier h with respect to w as follows:

$$\begin{aligned} d(h,w) &= \frac{\sum_{i=1}^M y_i w_i h(x_i)}{\sum_{i=1}^M y_i w_i} \\ f(h,w) &= \frac{\sum_{i=1}^M (1-y_i) w_i h(x_i)}{\sum_{i=1}^M (1-y_i) w_i} \\ \epsilon(h,w) &= \frac{\sum_{i=1}^M w_i |h(x_i) - y_i|}{\sum_{i=1}^M w_i} \end{aligned} \tag{1}$$

In order to simplify notation, the dependence on h or on w will be neglected when such a dependence is clear from the context. In particular, we will denote $d(h_t, w^t)$ by d_t and similarly for the other quantities.

AdaBoost designs a classifier that computes a weighted sum of the outputs of T binary *weak classifiers* (or *hypotheses*):

$$h(x) = \mathbf{1}\left(\sum_{t=1}^T b_t h_t(x) + c\right) \tag{2}$$

where $\mathbf{1}(\cdot)$ is the positive indicator function. The AdaBoost algorithm (summarized in the next table[1]) selects weak classifiers h_t and weights b_t one at a time. A weight distribution w^t is recursively updated based on the output of the current weak classifier. The error rate under this distribution is used to determine the weight b_t of h_t in the final (*strong*) classifier. The threshold c is chosen as follows:

$$c = -\frac{1}{2}\sum_{t=1}^T b_t \tag{3}$$

Freund and Shapire [7] derived an upper bound for the empirical error rate of the resulting strong classifier h given any value of $\beta_t \in (0,1)$:

$$\epsilon(h) \le \prod_{t=1}^T \frac{1-(1-\epsilon_t)(1-\beta_t)}{\sqrt{\beta_t}} \tag{4}$$

where the error rate $\epsilon(h)$ of the strong classifier is computed with respect to the uniform distribution. AdaBoost selects the values of $\{\beta_t\}$ that minimize the upper bound in (4) (line 5: in the algorithm): $\beta_t = \epsilon_t/(1-\epsilon_t)$, resulting in:

$$\epsilon(h) \le 2^T \prod_{t=1}^T \sqrt{\epsilon_t(1-\epsilon_t)} \tag{5}$$

Shapire *et al.* [9] also showed that the AdaBoost strategy leads to margin maximization in function space, which may explain the good generalization properties of the algorithm.

AdaBoost does not dictate the form of the weak classifiers h_t. For example, simple linear classifiers on individual features can be used:

$$h_t(x) = \mathbf{1}\left(\omega_t \cdot x^{j(t)} + \eta_t\right) \tag{6}$$

A weak classifiers (characterized by the feature index j and the parameters ω and η) is selected from a set $\mathcal{H}$. The upper bound in (5) suggests a selection strategy for h_t: at each iteration t, choose the element of $\mathcal{H}$ (i.e., the feature index j and parameters ω, η) that produces the smallest value of ϵ_t, thus

[1]Note that, following [6], we used the convention that negative examples are labeled by '0' rather than '-1'.

AdaBoost design [6]: $h(x) = \mathbf{1}\left(\sum_{t=1}^T b_t h_t(x) + c\right)$

1: Initialize $w_i^1 = 1/M$
2: **for** $t = 1 \to T$ **do**
3: $\quad w_i^t = w_i^t / \sum_{j=1}^M w_j^t$
4: $\quad h_t = \texttt{select}(w^t)$
5: $\quad \beta_t = \epsilon_t/(1-\epsilon_t)$
6: $\quad w_i^{t+1} = w_i^t \beta_t^{1-|h_t(x_i)-y_i|}$
7: $\quad b_t = -\log \beta_t$
8: **end for**
9: **return** $\{h_t\}$; $\{b_t\}$; $c = -\frac{1}{2}\sum_{t=i}^T b_t$

Selection criterion:

$$\arg\min_{h_t \in \mathcal{H}} \epsilon_t$$

minimizing the upper bound in (5). In fact, any hypothesis that performs better than random guess ($\epsilon_t < 0.5$) will contribute to reducing this upper bound. Likewise, it can be seen that any hypothesis with $\epsilon_t < 0.5$ contributes to reducing an upper bound on the fraction of training examples with margin below a small enough constant [9].

As noted earlier, the standard AdaBoost design algorithm is ill-suited for the case of asymmetric target performance. For example, when dealing with a cascaded architecture, we wish to design classifiers with detection rate above a certain threshold $d_{\min}$ (e.g. 0.98) and with "moderate" false positive rate (e.g. 0.5). One simple solution would be to vary the threshold c in (2): for each value of c, the resulting empirical detection and false positive rates can be easily computed, thus building a ROC curve for the strong classifier. The smallest value of c that ensures $d(h) \geq d_{\min}$ can then be selected. An improvement to this simple solution is offered by the AsymBoost algorithm [12]. In this case, the distribution w^t computed at each iteration is artificially modified, by giving more weight to the positive examples. More precisely, AsymBoost modifies line 6 in the AdaBoost design algorithm above as follows:

$$w_i^{t+1} = k^{y_i - \frac{1}{2}} w_i^t \beta_t^{1-|h_t(x_i)-y_i|} \tag{7}$$

where $k > 1$ determines the relative weight assigned to positive examples.

Our proposed ROC-Boost algorithm, summarized in the next table, takes a different approach. ROC-Boost produces a classifier with the same form as (2). One difference with respect to AdaBoost is in the computation of the final threshold c, which takes a different form than in (3) due to the presence of a set of parameters $\{q_t\}$:

$$c = -\sum_{t=1}^{T} b_t q_t \tag{8}$$

More importantly, ROC-Boost defines a different rule to select the weak classifiers h_t as well as the parameters β_t, used in AdaBoost to update the distribution w^t and to produce the coefficients b_t.

The following result, which generalizes Freund and Shapire's upper bound for the error rate (4), is instrumental for our design strategy:

Fact 1: For any choice of $h_t \in \mathcal{H}$, $\beta_t \in (0,1)$, and $q_t \in (0,1)$ in the ROC-Boost design algorithm, the following bounds hold:

$$d(h) \geq 1 - \prod_{t=1}^{T} \bar{D}_t \tag{9}$$

$$f(h) \leq \prod_{t=1}^{T} F_t \tag{10}$$

where

$$\bar{D}_t = \frac{1 - d_t(1-\beta_t)}{\beta_t^{q_t}} \tag{11}$$

$$F_t = \frac{1 - (1-f_t)(1-\beta_t)}{\beta_t^{1-q_t}} \tag{12}$$

ROC-Boost design: $h(x) = \mathbf{1}\left(\sum_{t=1}^{T} b_t h_t(x) + c\right)$

1: Initialize $w_i^1 = 1/M$
2: **for** $t = 1 \rightarrow T$ **do**
3: $\quad w_i^t = w_i^t / \sum_{j=1}^{M} w_j^t$
4: $\quad (h_t, \beta_t, q_t) = \texttt{select}(w^t)$
5: $\quad w_i^{t+1} = w_i^t \beta_t^{1-|h_t(x_i)-y_i|}$
6: $\quad b_t = -\log \beta_t$
7: **end for**
8: **return** $\{h_t\}$; $\{b_t\}$; $c = -\sum_{t=i}^{T} b_t q_t$

Selection criterion:

$$\underset{\substack{h_t \in \mathcal{H},\ \beta_t \in (0,1),\ q_t \in (0,1) \\ \text{such that } \bar{D}_t = (1-d_{\min})^{\frac{1}{T}}}}{\arg\min} F_t$$

The proof of this results follows the same steps as the proof for the upper bound (4) for $\epsilon(h)$ in [6].

Fact 1 suggests that a sufficient condition to ensure that the final classifier h has detection rate larger than or equal to $d_{\min}$ is for $\bar{D}_t$ to be equal to $\bar{D}_{\text{bound}}$, with

$$\bar{D}_{\text{bound}} = (1 - d_{\min})^{\frac{1}{T}} \tag{13}$$

We thus propose a selection criterion for (h_t, β_t, q_t) as follows:

ROC-Boost Selection Criterion: *Find the weak classifier h_t and the parameters (β_t, q_t) that minimize F_t subject to $\beta_t \in (0,1)$, $q_t \in (0,1)$, and $\bar{D}_t = \bar{D}_{\text{bound}}$.*

Note that this criterion is similar to AdaBost criterion of finding h_t and β_t that minimize the upper bound (4) on the detection rate. The following result allows us to compute β_t and q_t in closed form given h_t.

Fact 2: *For a given a weak classifier h_t, the ROC-Boost Selection Criterion produces the following parameters:*

$$\beta_t = \sqrt{\frac{(1-d_t)f_t}{d_t(1-f_t)}} \tag{14}$$

$$q_t = 1 - \log_{\beta_t}\left(\frac{f_t \bar{D}_{\text{bound}}}{d_t[f_t + \beta_t(1-f_t)]}\right) \tag{15}$$

provided that

$$d_t > \frac{f_t}{2} + (1 - \bar{D}_{\text{bound}})(1-f_t) + \sqrt{f_t \bar{D}_{\text{bound}}(1-\bar{D}_{\text{bound}})(1-f_t) + \frac{f_t^2}{4}} \tag{16}$$

Proof (sketch): Define $s = \beta_t^{q_t}$ and $r = \beta_t^{q_t - 1}$. Then $\beta_t = s/r$ and $q_t = \log_{\beta_t} s$. The conditions $\beta_t \in (0,1)$, $q_t \in (0,1)$ are satisfied as long as $s \in (0,1)$ and $r > 1$. Then, $F_t = r f_t + s(1-f_t)$, while the constraint $\bar{D}_t = \bar{D}_{\text{bound}}$ defines an hyperbolic curve:

$$r\ (1-d_t) + s\ d_t - s\ r\ \bar{D}_{\text{bound}} = 0$$

easily leading to (14)–(16). Note that if (16) is not satisfied, then the candidate weak classifier h_t cannot satisfy the constraint $\bar{D}_t = \bar{D}_{\text{bound}}$ for any value of (β_t, q_t), so it cannot be selected.

The expression for β_t in (14) is reminiscent of the AdaBoost expression for the equivalent parameter (see line 5 of the AdaBoost design panel), which was derived by minimizing the upper bound (4) for the error rate of the strong classifier [6]. In a similar way to the AdaBoost algorithm, we can test several weak classifiers h_t; for each tentative weak classifier, we check whether it satisfies (16) and, if so, record the corresponding value of F_t. We then retain the weak classifier and corresponding (β_t, q_t) with the minimum associated F_t.

Combining (14), (15) and (12), one easily derives the following identity:

$$\min F_t = \frac{1}{\bar{D}_{\text{bound}}} \left(1 - \left(\sqrt{d_t(1-f_t)} - \sqrt{f_t(1-d_t)} \right)^2 \right) \tag{17}$$

Thus, one sees that $\min F_t > 1$ as soon as

$$|\sqrt{d_t(1-f_t)} - \sqrt{(1-d_t)f_t}| \leq \sqrt{1-\bar{D}_{\text{bound}}} \tag{18}$$

The fact that $\min F_t$ may be larger than 1 leads to the possibility that, for a given value of $\bar{D}_{\text{bound}}$, the resulting upper bound for the false negative rate $\prod_t F_t$ may also be larger than 1 (and thus meaningless). It would seem that, in these situations, our design strategy could not do much more than enforcing a minimum value for the detection rate (which, as mentioned earlier, could be more easily obtained by simply changing the threshold c of the strong classifier). Yet, our experimental results, presented in the next section, appear to indicate that the ROC-Boost algorithm produces classifiers with better characteristics, in terms of ROC curves, than competing approaches.

IV. Experiments

We used the ROC-Boost design algorithm to design a number of asymmetric text detection classifiers, using features similar to those proposed by Chen and Yuille [3]. We used the ICDAR 2003 image database [8] for the Robust Reading and Text Locating competition. This data set contains 509 labeled images, divided between training (TrialTrain with 258 images) and test (TrialTest with 251 images) sets. Each image contains at least one area with some visible text; a rectangular bounding box encompassing the text area is provided as metadata. Overall, there are 1888 such bounding boxes (with variable size), which form the set of positive examples. We created 6000 negative examples (3000 for the training set and 3000 for the test set) by centering randomly sized boxes (with side length between 10 and 500 pixels) in random positions in the images, ensuring that these negative example boxes do not overlap with any positive example box. Following Chen and Yuille [3], we defined three different classes of features with increasing complexity, which could be used in different layers of a 3-tier cascade. These feature classes are described in detail below, where we denoted by $x^{j,l}$ the j-th feature in the l-th layer.

Layer 1. Each box is subdivided into a set of subboxes in five different patterns [3]. These different decompositions contain between two and five subboxes. For each decomposition, the mean and standard deviation of the gray level data in each subbox are computed, resulting in six variable size vectors (one per decomposition, plus one for the original box) for the mean and six vectors for the standard deviation. To these vectors, we add all of their unordered pairs (each pair combined in a single vector). In total, there are 78 variable size features. (Note that, here as well as in the other layers, at each iteration t only one feature $x^{j,l}$ is selected with replacement, and fed to the t-th weak classifier h_t.)

Layer 2. For each box, 15-bin histograms are computed of the gray level data, of its horizontal and vertical discrete derivatives, and of the magnitude and direction of its discrete gradient. Unordered pairs of histograms, combined into length-30 vectors, are added to this set, resulting in 15 features.

Layer 3. For each box, 8×8-bin two-dimensional histograms are computed of the pairs gray level data – gradient direction and gray level data – gradient magnitude. Each two-dimensional histogram is then represented as a single length-64 vector, resulting in only two features overall.

The weak classifiers are designed using the Fisher Linear Discriminant algorithm [1]. The training data features are multiplied by the normalized weight distribution w^t before computing the mean vectors and covariance matrices, which are then used to derive the classifier's coefficient vector $\omega^{j,l}$. Thus, for each feature $x^{j,l}$, we obtain a family of weak classifiers $h(x^{j,l}) = \mathbf{1}(\omega^{j,l} \cdot x^{j,l} + \eta)$ parameterized by the threshold η. Selection of the hypothesis h_t requires testing all the features j for different values of η. We decided to consider the values $\eta = -\omega^{j,l} \cdot x^{j,l}$ computed only for the positive examples x in the training set. This is because a change in d_t only occurs when η reaches these values. Only those values for η that allow d_t to meet the constraint (16) are retained. Overall, we need to examine less than $M_+ \cdot J(l)$ hypotheses, where $J(l)$ is the number of features in the l-th layer and M_+ is the number of positive examples in the training set. We also designed classifiers using the AsymBoost algorithm for comparison. Different values for η were considered in this case as well, with the aim to minimize the weighted error ϵ_t. Finally, for both the ROC-Boost and the AsymBoost classifier, we considered modified versions in which the weights b_t for the individual hypotheses and the threshold c are recomputed using the LAC and the FDA procedures proposed by Wu *et al.* [13]. Strong classifiers were built from $T = 10$ weak classifiers for all design algorithms considered.

Fig. 1 shows the results, in terms of empirical detection and false positive rate (computed on the training data with Layer 1 features) of the ROC-Boost classifier for three different values of $\bar{D}_{\text{bound}}$, along with the ROC curves obtained by changing the final threshold c. It should be noted that for each value of

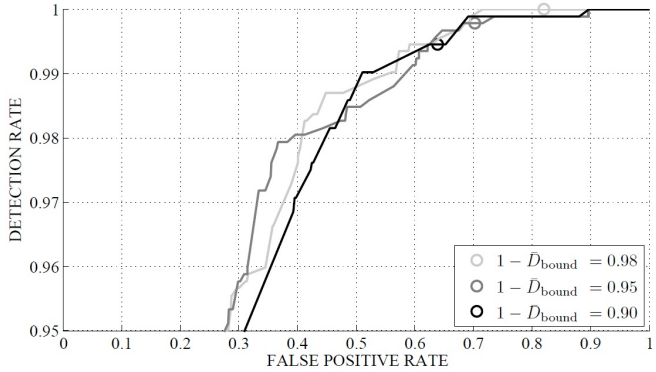

Fig. 1. ROC curves obtained from ROC-Boost classifiers, trained and tested over the whole data set with the features in Layer 1 for three different values of $\bar{D}_{\text{bound}}$. The circles represent the performance of the classifier produced by the ROC-Boost design algorithm; the ROC curve is obtained by varying the threshold c of the strong classifier.

$\bar{D}_{\text{bound}}$, the detection rate using the threshold c selected by ROC-Boost is quite higher than its lower bound $1-\bar{D}_{\text{bound}}$. This is hardly surprising, considering that $\bar{D}_{\text{bound}}$ in (13) is chosen very conservatively. In order to reach a specific target detection rate, the designer needs to test different values for $\bar{D}_{\text{bound}}$ and/or different values for the threshold c. For example, if the target detection rate is 0.99 and the three values of $\bar{D}_{\text{bound}}$ shown in Fig. 1 are tested, the designer may select the classifier with $\bar{D}_{\text{bound}} = 0.10$ with the threshold c that produces $d(h) = 0.99$, as it yields the lowest false positive rate for that value of detection rate. Note that this issue is not specific to ROC-Boost: exactly the same problem occurs with AsymBoost (and with the modified versions using LAC and FDA), when selecting the correct value for k in order to reach a specific detection rate.

In order to account for this variety of results, in the next figures we plotted the Pareto front of all ROC points obtained with all combinations of parameter $(\bar{D}_{\text{bound}}, c)$ (for ROC-Boost) or (k, c) (for AsymBoost) used in our tests. Specifically, we considered seven values for $\bar{D}_{\text{bound}}$ between 0.01 and 0.3 for ROC-Boost, and seven values for k between 2 and 5 for AsymBoost, along with all values of the threshold c that determined a change in the detection rate. For each such choice of parameters we computed the ROC (detection rate vs. false positive rate). The Pareto front contains all ROC points such that no other ROC point has both higher detection rate and false positive rate; it thus identifies the "best" classifiers for varying detection rate. Fig. 2 shows the Pareto curves for all the algorithms considered over the three distinct layers (the classifiers were tested over the training data). It is seen that ROC-Boost consistently outperforms AsymBoost. Modifying the weights b_t produced by ROC-Boost using either the LAC or FDA algorithms yield results that in most cases are worse than with the original algorithm.

In order to explore the generalization properties of ROC-Boost, we compared the performance of the ROC-boost and Asymboost classifiers, trained on the TrialTrain dataset, over the TrialTest dataset. We first defined a set of target

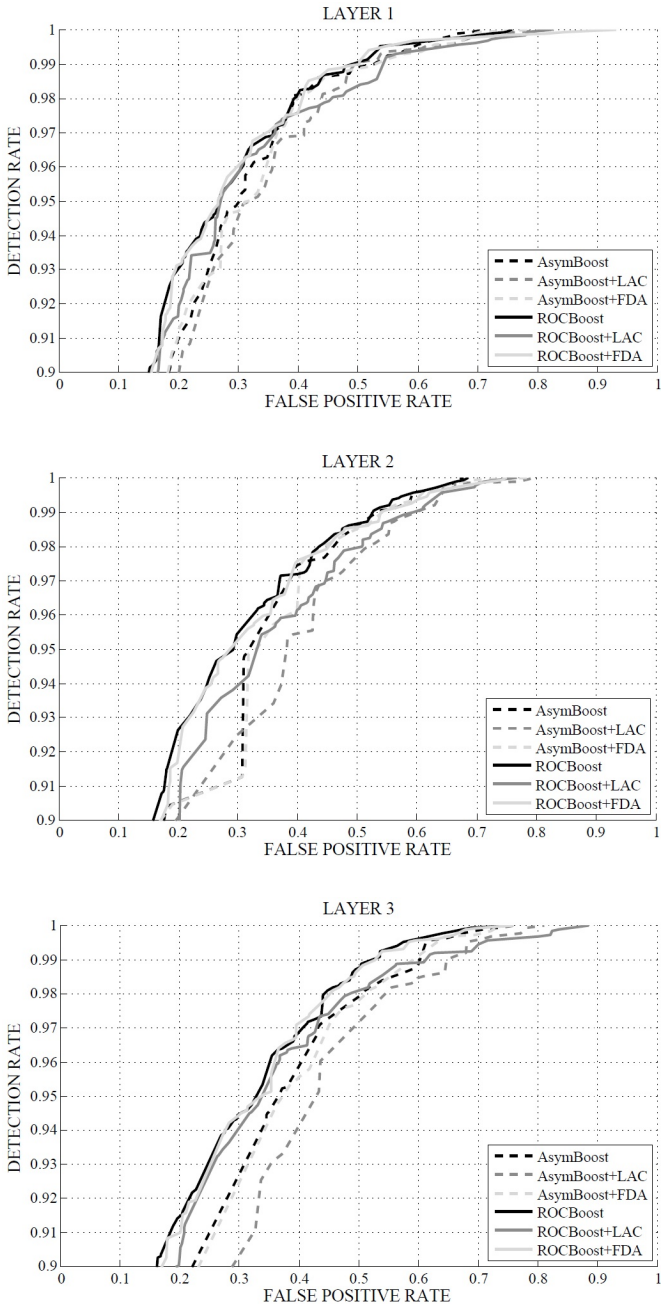

Fig. 2. The Pareto front of the ROC curves for the different classifiers considered. The Pareto fronts were computed over the ROC curves obtained by changing the value of $\bar{D}_{\text{bound}}$ between 0.01 and 0.3 (for ROC-Boost) and of k between 2 and 5 (for Asymboost), and for multiple values of the final threshold c. The classifiers were trained and tested on the same dataset (TrialTrain).

detection rates equally spaced between 0.9 and 1. Then, for each design algorithm, we selected, from the pool of classifiers that generated the Pareto fronts in Fig. 2, the classifier with detection rate equal to or higher than the target rate with the smallest false positive rate. This classifier was then applied to the examples in the test data set, generating a detection rate – false positive rate pair. These values, computed over the chosen target detection rates, are shown in Fig. 3. The two

algorithms are shown to give comparable results.

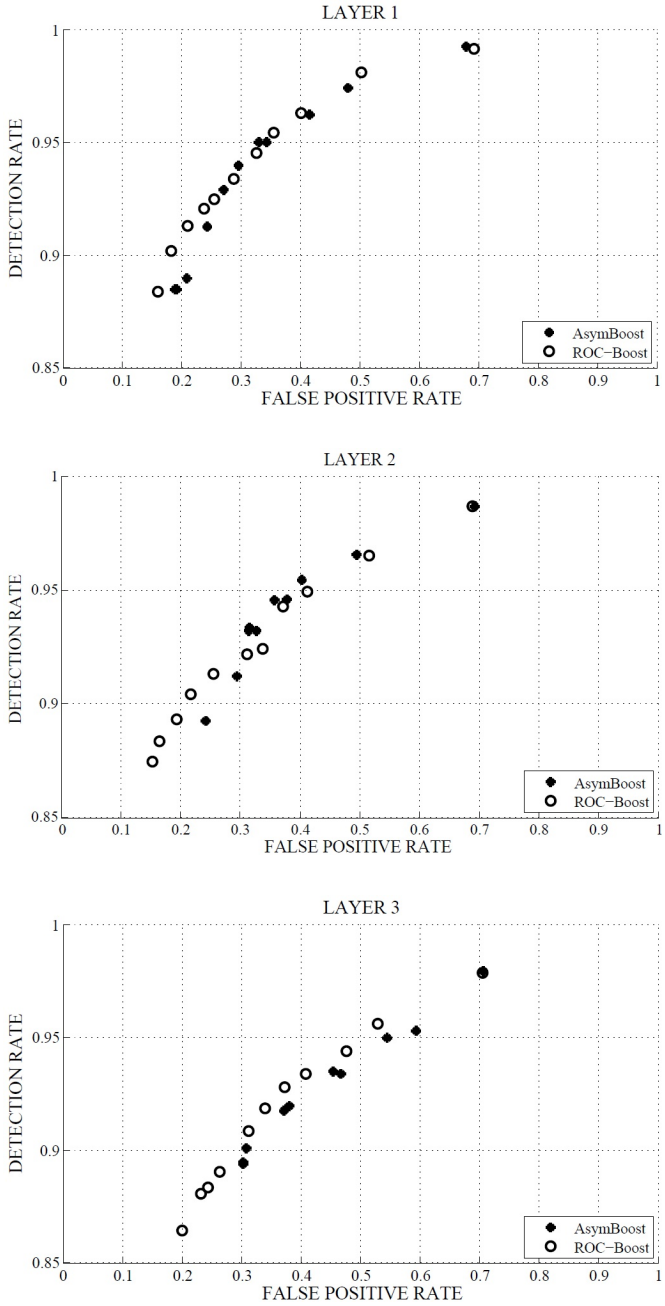

Fig. 3. Detection rate vs. false positive rates for the ROC-Boost and AsymBoost classifiers, trained on TrialTrain and tested on TrialTest (see text for details).

V. Discussion and Conclusions

We have introduced a new design algorithm for binary classification with asymmetric performance. The resulting classifier has the same form as the standard AdaBoost classifier; however, the individual hypotheses and linear combination weights are designed so as to ensure that the empirical detection rate of the strong classifier is higher than a given value while minimizing an upper bound on the false positive rate. Our experimental results have shown that this approach is competitive with other popular algorithms for asymmetric classification.

A complete characterization of the behavior of ROC-Boost classifiers will require addressing a number of open theoretical questions. For example, as noted at the end of Sec. III, the minimum of the upper bound F_t on the false positive rate may end up being larger than 1, making this bound useless. The classifiers obtained by minimizing this upper bound seem to perform very well nonetheless. One may conjecture that F_t may relate to other "useful" quantities – perhaps a stricter bound. In addition, it would be interesting to characterize this design procedure in terms of margin statistics, which would shed light on the generalization properties of the algorithm.

References

[1] C. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2007.

[2] S. Brubaker, J. Wu, J. Sun, M. Mullin, and J. Regh, "On the design of cascades of boosted ensembles for face detection", *International Journal of Computer Vision*, 2008.

[3] X. Chen and A. Youille, "Detecting and reading text in natural scenes", *Proc. CVPR*, 2004.

[4] X. Chen and A. Youille, "A time-efficient cascade for real-time object detection: with applications for the visually impaired", *Proc. IEEE Workshop on Computer Vision Applications for the Visually Impaired*, 2005.

[5] A. Demiriz, K. Bennett, and J. Shawe-Taylor, "Linear programming boosting via column generation", *Machine Learning*, 46:225-54,2002.

[6] Y. Freund and R. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting", *Proc. European Conference on Computational Learning Theory*, 1995.

[7] Y. Freund and R. Schapire, "A short introduction to boosting", *Journal-Japanese Society For Artificial Intelligence*, 1999.

[8] S. Lucas, A. Panaretos, L. Sosa, A. Tang, S. Wong and R. Young, "ICDAR 2003 robust reading competitions", *Seventh International Conference on Document Analysis and Recognition - Volume 2*, 2003.

[9] R.E. Schapire, R. E. Freund, Y. Bartlett, and P. Wee Sun Lee, "Boosting the margin: A new explanation for the effectiveness of voting methods", *Annals of Statistics*, 26(5):1651-86, 1998.

[10] C. Shen, P. Wang, and H. Li, "LACBoost and FisherBoost: optimally building cascade classifiers", *Proc. ECCV*, 2010

[11] P. Viola and M. Jones, "Robust real-time object detection", *International Journal of Computer Vision*, 2001.

[12] P. Viola and M. Jones, "Fast and robust classification using asymmetric Adaboost and a detector cascade", *Proc. NIPS*, 2001.

[13] J. Wu, S. Brubaker, M. Mullin, and J. Regh, "Fast asymmetric learning for cascade face detection", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2008.