AdaTree : boosting a weak classifier into a decision tree

Etienne Grossmann*<etienne@cs.uky.edu>
Center for Visualization and Virtual Environments,
University of Kentucky
1, Quality Street, Suite 857
Lexington, KY, 40507, USA.

Abstract

We present a boosting method that results in a decision tree rather than a fixed linear sequence of classifiers. An equally correct statement is that we present a tree-growing method whose performance can be analysed in the framework of Adaboost.

We argue that Adaboost can be improved by presenting the input to a sequence of weak classifiers, each one tuned to the conditional probability determined by the output of previous weak classifiers. As a result, the final classifier has a tree structure, rather than being linear, thus the name "Adatree". One of the consequences of the tree structure is that different input data may have different processing time. Early experimentation shows a reduced computation cost with respect to Adaboost.

One of our intended applications is real-time detection, where cascades of boosted detectors have recently become successful. The reduced computation cost of the proposed method shows some potential for being used directly in detection problems, without need of a cascade.

1 Introduction

Boosting [18] refers to methods that successively train a weak classifier to improve its performance on a training dataset. One important property is that the final error is upper-bounded by a product of terms bounding the errors of the weak classifiers. As a consequence, the training error decreases exponentially with the number of boosting rounds, as long as the weak classifier performs slightly better than trivial guessing.

1.1 Connection between boosting and tree classifiers

It has been noted [11] that decision tree growing can be viewed as a boosting mechanism and that the train-

*This work was funded by the Kentucky Office of New Economy

AdaBoost

Run-time:
- Sequence of classifiers is fixed.
- Output is linear combination of a fixed set of classifiers.

Training:
- Each weak classifier is given all training data, with variable weights.

Unsmoothed Decision Tree

Run-time
- Output of current weak classifier determines choice of next classifier.
- Output depends only on that of last classifier.

Training:
- Each weak classifier sees only the training examples that reach the corresponding node.

AdaTree

Run-time:
- Output of current weak classifier determines choice of next classifier.
- Output is linear combination of previous classifiers; weights increase near leaves.

Training:
- Each weak classifier sees only the training examples that reach the corresponding node.

Table 1: Comparison of the run-time behavior and of the usage of training examples, in Adaboost, a decision tree and Adatree.

ing error can also be made to decrease exponentially. However, algorithmically, boosted classifiers and decision trees appear totally unrelated. The proposed method bridges this gap, by defining a boosting algorithm that results in a decision tree. An alternative viewpoint is that we propose a tree-growing algorithm that is amenable to the analysis methods of Adaboost, as developed in [20].

Table 1 shows a comparison of Adaboost, decision tree and the proposed method. In Adaboost, the input data always follows the same path, independently of the result of previous classification. In contrast, in a decision tree, the input is presented to classifiers that are adapted to the conditional probability determined by the result of the previous test. This "good" property is preserved in Adatree.

In an unsmoothed decision tree, the final decision belongs to the last classifier, independently of the output of the previous classifiers. In Adaboost, it is possible to use confidence-rated weak classifiers that return a real number and the final output is a linear combination of the output of all the classifiers. This "voting" property of boosting is preserved in Adatree.

It is well-known that decision trees are prone to over-fitting [16, Ch. 3.7], whereas Adaboost appears less subject to it [19], at least if the space of weak learners is sufficiently "small". Because, in Adatree, each classifier is presented with a subset of the training data, one can expect that the proposed method suffers from the same over-fitting problem as decision trees. We will see in Section 3 that this is the case to a limited extent when the input data has a moderate amount of noise.

As noted by an anonymous reviewer, Adatree is similar to a "smoothed" [2, 3] or "shrinked" [2, cited by [5]] decision tree in which the final decision is a combination of the output at inner node of the tree. The combination can be obtained using e.g. statistical [3]or empirical [5] arguments. Due to time constraints, we do not provide here comparison with these methods and focus on our comparison with Adaboost.

1.2 Boosting in real-time detection problems

Adaboost has been used in systems that solve hard classification problems such as face detection [21, 12, 13].

Viola and Jones [21] showed a reasonably good face detector obtained by boosting a thousand times a weak classifier. Because the computation time of the boosted classifier is proportional to the number of boosting rounds and because the intended application was real-time detection, the resulting computation cost was considered prohibitive. To improve the situation, [21] propose to use a cascade of boosted classifiers rather than a single huge boosted classifier.

This approach presents some advantages : the resulting system discards negative inputs as evidence gathers that they are not positives, rather than computing all classifiers before taking a decision. As a result, the average computation cost is much smaller. Moreover, each boosted classifier in the cascade is trained in accordance to the proportion of positive and negatives it will meet and may be tuned for this situation.

The association of a cascade architecture and of boosted classifier thus allows good detection performance while keeping computation cost low. This explains the success of this approach, which has generated many variants [12, 13] and has inspired related work [4, 6].

We argue that the cascade architecture -a special case of a decision tree- is not necessarily optimal. Indeed *all* positives must be considered as such at each level of the cascade. Each level must thus have very low false negative rates for the system to be successful. In contrast, in a general decision tree, each node can produce a split with high error rates, as further splits will improve the estimate. What is important is that each split increases the information content [11, 16].

Moreover, in a cascade, the characteristics of the boosted classifiers at each level have to be set "by hand". Despite some recent research to set these characteristics automatically [14, 9], the issue remains. In contrast, Adatree is a single system which can be trained and studied with the same ease as a single boosted classifier.

1.3 AdaTree

Algorithm 1 describes the proposed method. Many versions of boosting exist. The study that we propose here follows the lines of the most studied Adaboost algorithm [8, 20]. The main difference in our notation is that weak classifiers cannot be indexed simply by their depth. We represent each node s by a sequence of $'+'$ and $'-'$ corresponding to the signs of the output of the weak classifiers that lead to this node. The root node is written $s = \emptyset$. The second node, assuming the first weak classifier $h_{\emptyset}(X)$ has positive output is $s = +$. The third node, assuming a positive outcome of the second weak classifier $h_{+}(X)$, is $s = ++$. Etc.

The following section studies the properties of Adatree. Section 3 compares experimentally the behavior of Adatree and Adaboost. Finally, Section 4 presents some conclusions and directions of ongoing work.

Algorithm 1 AdaTree

Input Training examples $(X_1, y_1), \ldots, (X_N, y_N)$, original weights $D_\emptyset(1), \ldots, D_\emptyset(N)$, target error rate and a family of weak classifiers.

Initialization Initialize the set of leaves to $\{\emptyset\}$.

While error above target

- Choose the leaf s with maximum error.
- Train a weak learner $h_s(X)$ using the p.d.f $D_s(n)$ on the training set.
- Choose α_{s+} and α_{s-}.
- Define

$$\begin{aligned} D_{s+}(n) &= D_s(n)\, e^{-y_n \alpha_{s+} h_s(X_n)}/Z_{s+} \\ &\quad \text{if } h_s(X_n) > 0,\ 0 \text{ otherwise,} \\ D_{s-}(n) &= D_s(n)\, e^{-y_n \alpha_{s-} h_s(X_n)}/Z_{s-} \\ &\quad \text{if } h_s(X_n) \leq 0,\ 0 \text{ otherwise,} \end{aligned}$$

  where $Z_{s\pm}$ normalizes the $D_{s\pm}(n)$ so that they sum up to one.

- Split the leaf s into $s+$ and $s-$.

Final classifier

Define

$$h(X) = \sum_{t=1}^{T(X)} \alpha(t, X)\, h(t, X),$$

where $\alpha(t, X) = \alpha_{s(t+1,X)}$, $h(t, X) = h_{s(t,X)}(X)$ and the $s(t, X)$ are defined recursively by

$$\begin{aligned} s(1, X) &= \emptyset \\ s(t+1, X) &= \begin{cases} s(t, X)+ & \text{if } h_{s(t,X)}(X) > 0, \\ s(t, X)- & \text{if } h_{s(t,X)}(X) \leq 0, \\ undefined & \text{if } h_{s(t,X)} \text{ does not exist.} \end{cases} \end{aligned}$$

$T(X)$ is the largest number for which $s(t+1, X)$ is defined.

2 Properties of AdaTree

We now show that Adatree has properties similar to that of Adaboost. In particular, we show that the training error can be bounded in the framework of [20].

2.1 Bounds on the loss

We first bound the loss of $h(X)$ on the training data. The notation is abbreviated to improve readability : a sum over all indices n such that $h(X_n) > 0$ and $y_n = 1$ is noted $\sum_{h>0, y_n=1}$. Also, $T(X_n)$ is shortened to $T(n)$, $s(t, X_n)$ to $s(t, n)$ and so on.

$$\begin{aligned} L(h) &= \sum_{h>0, y_n=-1} D_\emptyset(n) + \sum_{h\leq 0, y_n=1} D_\emptyset(n) \\ &\leq \sum_{n=1}^{N} D_\emptyset(n)\, e^{-y_n h(X_n)} \\ &= \sum_{n=1}^{N} D_\emptyset(n) \prod_{t=1}^{T(n)} e^{-y_n \alpha(t,n) h(t,n)} \\ &= \sum_{n=1}^{N} D_\emptyset(n) \prod_{t=1}^{T(n)} \frac{D_{s(t+1,n)}(n)\, Z_{s(t+1,n)}}{D_{s(t,n)}(n)} \\ &= \sum_{n=1}^{N} D_{T(n)}(n) \prod_{t=1}^{T(n)} Z_{s(t+1,n)} \\ &= \sum_{s \text{ leaf}} \prod_{s' \leq s} Z_s. \end{aligned} \tag{1}$$

The second line results from a simple majoration. The fourth line is the consequence of the definition of $D_{s(t+1,n)}(n)$. The last equality, where $s' \leq s$ indicates s and all its parent nodes s', results from reordering the sum over the examples into a sum over the leaves

$$\begin{aligned} &\sum_{n=1}^{N} D_{T(n)}(n) \prod_{t=1}^{T(n)} Z_{s(t+1,n)} \\ &= \sum_{s \text{ leaf}} \underbrace{\sum_{n \text{ reaches } s} D_{T(n)}(n)}_{1, \text{ by construction}} \prod_{t=1}^{T(n)} Z_{s(t+1,n)} \end{aligned}$$

2.2 Choice of parameters

We now show how to determine the optimal parameters $\alpha_{s\pm}$ in the general case of real-valued weak classifiers and in the cases of classifiers limited to $[-1, 1]$ and $\{-1, 1\}$.

Given the bound in Eq. (1), we are justified to minimize the loss by greedily minimizing each term $Z_{s\pm}$ of the product with respect to α_{s+} and α_{s-}. One first notes that Z_{s+} (resp. Z_{s-}) does not depend on α_{s-} (resp. α_{s+}). Writing Z_{s+} as

$$\begin{aligned} Z_{s+} &= \sum_{\substack{h>0 \\ y_n=1}} D_s(n)\, e^{-h_s(X_n)\alpha_{s+}} \\ &\quad + \sum_{\substack{h>0 \\ y_n=-1}} D_s(n)\, e^{h_s(X_n)\alpha_{s+}}. \end{aligned}$$

it is clear that there is a unique minimum, provided that there is at least one term $y_n = -1$ and one term $y_n = 1$ with non-zero weight $D_s(n)$: in that case, Z_{s+} tends to $+\infty$ when α_s tends to $\pm\infty$ and is a sum of monotonous convex terms. The derivative of Z_{s+} with respect to α_{s+} is

$$\begin{aligned}\frac{\partial}{\partial\alpha_{s+}}Z_{s+} &= -\sum_{\substack{h>0\\ y_n=1}} D_s(n)\,h_s(X_n)\,e^{-h_s(X_n)\alpha_{s+}} \\ &\quad + \sum_{\substack{h>0\\ y_n=-1}} D_s(n)\,h_s(X_n)\,e^{h_s(X_n)\alpha_{s+}},\end{aligned}$$

which has a unique zero. For the general case of real-valued weak classifiers, the unique zero can be found numerically. If the weak classifier is bounded, an analytic approach can be taken: following [20, 1], we assume that $h_s(X) \in [-1,1]$. The majoration

$$a \in [-1,1] \Rightarrow e^{ab} \leq \frac{1-a}{2}e^{-b} + \frac{1+a}{2}e^{b}$$

yields the bound

$$Z_{s+} \leq \sum_{h>0} D_s(n)\left(\tfrac{1+y_n h_s(X_n)}{2}e^{-\alpha_{s+}} + \tfrac{1-y_n h_s(X_n)}{2}e^{\alpha_{s+}}\right). \quad (2)$$

This time, it is easy to see that the majorant is minimized by

$$\begin{aligned}e^{2\alpha_{s+}} &= \frac{\sum_{h>0} D_s(n)\,(1+y_n h_s(X_n))}{\sum_{h>0} D_s(n)\,(1-y_n h_s(X_n))} \\ &= \frac{1+r_s^+}{1-r_s^+}, \end{aligned} \quad (3)$$

where $r_s^+ = \left(\sum_{h>0} D_s(n)\,y_n h_s(X_n)\right) / \left(\sum_{h>0} D_s(n)\right)$.

In the more specific, but common, case that $h_s(X) \in \{-1,1\}$, Z_{s+} is simply minimized by

$$e^{2\alpha_{s+}} = \frac{W_s^{++}}{W_s^{+-}} \quad (4)$$

where $W_s^{++} = \sum_{h>0,y_n=1} D_s(n)$ and $W_s^{+-} = \sum_{h>0,y_n=-1} D_s(n)$.

Minimizing and bounding Z_{s-} similarly yields the choices

$$\begin{aligned}e^{2\alpha_{s-}} &= \frac{\sum_{h\leq 0} D_s(n)\,(1+y_n h_s(X_n))}{\sum_{h\leq 0} D_s(n)\,(1-y_n h_s(X_n))} \\ &= \frac{1+r_s^-}{1-r_s^-}, \end{aligned} \quad (5)$$

where $r_s^- = \left(\sum_{h\leq 0} D_s(n)\,y_n h_s(X_n)\right) / \left(\sum_{h\leq 0} D_s(n)\right)$. If $h_s(X) \in \{-1,1\}$, Z_{s-} is minimized for

$$e^{2\alpha_{s-}} = \frac{W_s^{--}}{W_s^{-+}}, \quad (6)$$

where $W_s^{-+} = \sum_{h\leq 0,y_n=1} D_s(n)$ and $W_s^{--} = \sum_{h\leq 0,y_n=-1} D_s(n)$.

For $\alpha_{s\pm}$, given in Eqs. (3,5), the bound for $Z_{s\pm}$ is Eq. (2) is :

$$Z_{s\pm} \leq \left(\sum_{h\lesseqgtr 0} D_s(n)\right)\sqrt{1-\left(r_s^\pm\right)^2}.$$

In the case of $h_s(X) \in \{-1,1\}$, Eqs. (4,6) yield :

$$Z_{s\pm} = 2\sqrt{W_s^{\pm+}W_s^{\pm-}}.$$

Notice that our choice of $\alpha_{s\pm}$ coincides with that of Aslam in InfoBoost [1] and to one of the weighing schemes proposed in [20].

We are currently progressing towards a useful comparison of Eq. (1) and the bounds obtained in [20].

2.3 Considerations about special cases

Particular attention should be given when the weak classifier takes only one value on all examples to which it is presented, or when it has no false positives or no false negatives or no true positives or no true negatives. When this occurs, $Z_{s\pm}$ is minimized for $\alpha_{s\pm}$ at infinity. Although these cases can be considered as marginal in Adaboost, they are common in Adatree, where the number of training samples dwindles as one moves up the tree.

First, r_s^+ (resp. r_s^-) is not defined if $h_s(X_n)$ is non-positive (resp. negative) for all n s.t. $D_s(n) > 0$. In that case, the decision tree cannot grow a branch $s+$ (resp. $s-$) from lack of examples and the quality of the weak classifier, when it returns a positive value cannot be assessed from the training data. When this happen, we define $\alpha_{s+} = 1$ if s is the root node and $\alpha_{s+} = 0$ otherwise. In the former case, if $h_\emptyset(X) \leq 0$, then X will be classified as negative. In the latter case, a negative value of $h_s(X)$ will be discarded and the X will be classified only based on the parent nodes of s.

Second, r_s^+ is 1 (resp. -1) when the weak classifier, when it is positive, is equal (resp. opposite) to y_n[1]. In this case, α_{s+} is $+\infty$ (resp. $-\infty$), i.e. the weak

[1] That is, when one has : $D_s(n) > 0$ and $h_s(X_n) > 0 \Rightarrow y_n = 1$ (resp. $D_s(n) > 0$ and $h_s(X_n) > 0 \Rightarrow y_n = -1$). In this paragraph, we only consider r_s^+; it is clear that r_s^- should be treated likewise.

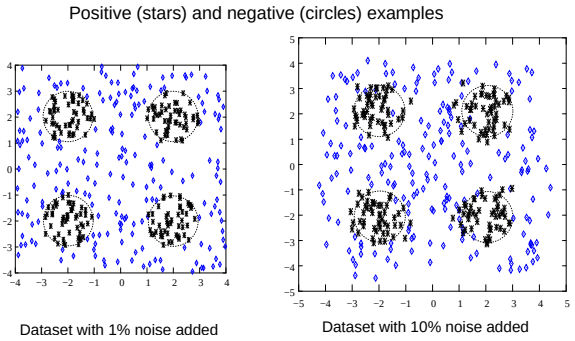

Figure 1: Example datasets, left with 1% noise and right with 10% noise.

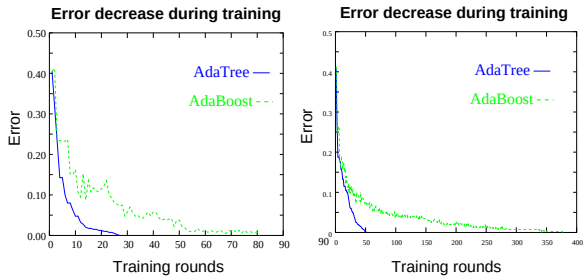

Figure 2: Error decrease during training. **Left**: with 1% noise in the data. **Right**: with 10% noise in the data.

classifier will be weighed infinitely more than its parent nodes. Since the error at s is zero, this node will not be further grown and the final classification of any X that reaches s and has $h_s(X) > 0$ will be positive. In order to avoid over-weighing these leaf classifiers, it is possible, following Schapire and Singer [20, Sec. 4.2], to add a small "smoothing" term to the terms $W_s^{\pm\pm}$.

3 Experimental results

This section presents some experimental results, in a simple 2D classification and a face detection problem.

3.1 2D classification

We now show how Adatree performs in relation to the Adaboost algorithm of [20], in the case of $h_s(X) \in [-1, 1]$. We consider datasets as shown in Figure (1), where, in the absence of noise, positive examples are in one of four circles and negative examples are in the surrounding space. Independently, identically distributed Gaussian noise at a level of 40dB (1%) or 20dB (10%) is added to these, so that the separation of positive and negative regions becomes more complicated and, more importantly, so that the true distributions be not exactly separable. The datasets consist in 400 points.

The weak classifier is a 2D linear classifier composed with a sigmoid so that it ranges in $[-1, 1]$. Adaboost [20] and Adatree were trained on the same data until the error level reached 0.001. Following [20], a smoothing term of 0.01 was added to the top and bottom terms of Eqs. (3,5) for computing the weights of Adatree.

Training Adaboost required 80 (resp. 377) boosting steps on the 1% (resp. 10%) noise datasets. Training Adatree required 28 (resp. 51) steps, resulting in a tree of depth 8 (resp. 10).

Figure 2 shows the evolution of the error during training. This figure shows that Adatree decreases the training error much faster than Adaboost.

The main performance test is of course that on the validation data. As can be expected by the nature of Adatree, its generalisation error is less good than that of Adaboost. Figure 3 shows the error on a validation dataset obtained in the same conditions as the training sets, for increasing numbers of training rounds. On the right, with 10% noise in the input, one clearly sees the effect of over-fitting in Adatree, which manifests itself as the joint increase of the error and of the number of training rounds. This effect is much smaller with 1% noise. In that case, while remaining inferior, Adatree compares better with Adaboost.

The bottom curves of Figure 3 plot the error against the mean computation cost. These curves are parameterized by the number of training rounds. In the case of Adaboost, the computation cost coincides with the number of training rounds, while it corresponds to the mean depth reached by input given to the Adatree classifier. These curves show dramatically the computation cost decrease incurred by Adatree and suggests that it has a big potential for real-time applications.

3.2 Face detection

In this section, we present experiments with face classifiers based on Haar filters combined with Adaboost and Adatree. Figure 4 (left) shows the five types of filters used. The weak classifiers are computed by thresholding the output of one such filter. By setting constraints on the minimum and maximum dimensions, surface and ratio of height to width of the filters, the number of possible filters is reduced to 286, much less than the tens of thousands of [21]. The training dataset consists in 1500 face and as many non-face images. The faces are taken from the BioID database [10], from dataset C

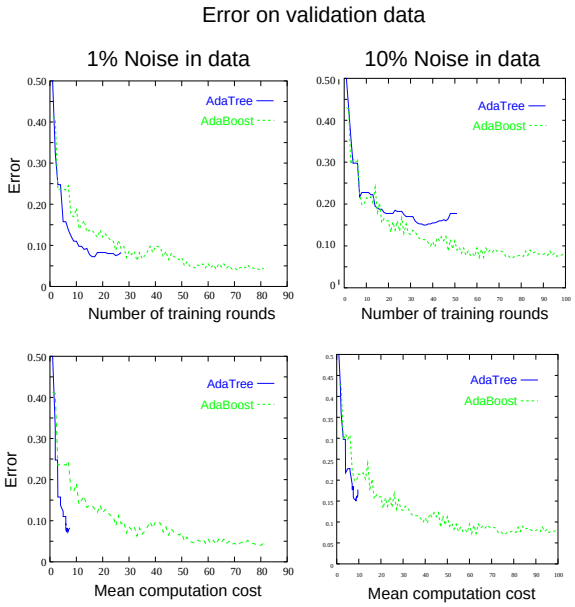

Figure 3: **Top**: Error on validation data vs. number of training rounds. **Bottom**: Error on validation data vs. mean number of weak classifiers used by the classifier. **Left**: with 1% noise in input. **Right**: with 10% noise.

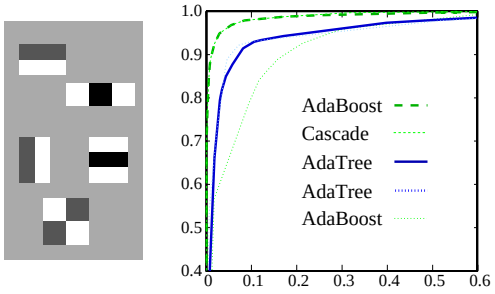

Figure 4: **Left**: Haar filters used in face detection. **Right**: ROC of Adaboost (thick dashed; cost=50), cascade (nearly superposed light dashed; cost=19) Adatree (plain thick curve; cost=6) and Adatree with a minimum of five examples at each leaf (nearly superposed light dotted curve; cost=7). And that of Adaboost (light dotted; cost=7), with a computation cost equivalent to that of Adatree.

of the CMU database [17] and from images gathered on the internet. The nonfaces are cropped from the Weber´s Caltech background dataset [22] and false positives obtained during previous classifier development. All face and non-face images are scaled to 18-by-31 pixels and normalized in variance and range. Images from the BioID database are cropped repeatedly at slightly different positions and scales. A validation dataset of 3000 different images is built in the same way.

The reference boosted classifier is built using 50 boosting rounds of the method of Schapire and Singer [20, Sec. 4.1] (equivalent to [1]), where the weak classifier is chosen according to the criterion of Kearns and Mansour [11] $2\sqrt{W^{++}W^{+-}}+2\sqrt{W^{-+}W^{--}}$. This reference classifier was also transformed into a cascade of 15 classifiers using the technique of [9], so as to reduce the average computation cost. A third reference classifier is defined by truncating the boosted classifier (retaining only the first few weak classifiers - 7 will be kept) so that its computation cost matches that of Adatree. Performance of the Adatree classifiers will be compared below to these three reference classifiers.

Two Adatree classifiers are built from the same weak classifiers using the same criterion. The first follows exactly the procedure described above, the second differs only by never splitting a leaf that has been reached by less than 5 examples. A smoothing factor of $1/N$ is used in the computation of weights. The resulting classifiers have 82 and 80 classifiers, respectively and depths 10 and 11, respectively.

The average number of weak classifiers evaluated by Adatree is 6.2 and 6.6 for the first and second classifiers and is 50 for Adaboost. The cascade of classifiers, whose ROC nearly coincides with that of Adaboost, has a mean computation cost of 19. The third reference classifier uses 7 weak classifiers.

Figure 4 (right) shows that ROC of Adaboost and of the cascade are well above that of the Adatree variants, which themselves are well above that of a boosted classifier with computational cost equivalent to that of Adatree.

4 Conclusions and ongoing work

We have presented a boosting method that bridges a gap between decision tree and boosting methods. This study was centered around the basic Adaboost [20], but it seems that results concerning variants of Adaboost, e.g. those that address the question of variable error costs [7, 15], are tractable to Adatree.

Experimentation showed that Adatree has a reduced training cost and a very much reduced computation cost with respect to Adaboost. When the input data has little noise, the error rates of Adatree and Adaboost are comparable.

The fact that Adatree presents a much reduced computation cost with respect to Adaboost makes it a strong candidate for real-time applications where cascades of detectors have been used to reduce the com-

putation load [21, 4, 13, 12]. If further testing confirms the trends shown above, Adatree -possibly adapted to avoid the over-fitting problem- could be used directly in real-time detectors. Because this would dispense of building a cascade of classifiers, the study of the whole system would be much simplified in comparison to that of cascaded systems.

However, using Adatree would only be practical if the problem of overfitting is solved. Indeed Adatree shows an increased generalization error with respect to Adaboost. We are addressing this issue in our present work. Rather than resorting to pruning or smoothing method, we are looking at better schemes to weigh the examples during the training that allow to stay within the framework of Adaboost and perhaps keep its good property of moderately over-fitting.

References

[1] J. A. Aslam. Improving algorithms for boosting. In *Annual Conf. on Computational Learning Theory*, pages 200–207. Morgan Kaufmann, San Francisco, 2000.

[2] L. R. Bahl, P. F. Brown, P. V. deSouza, and R. L Mercer. A tree-based statistical language model for natural language speech recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(7):1001–1008, 1989.

[3] W. Buntine. Learning classification trees. *Statistics and Computing*, 2(63-73), 1992.

[4] O. Carmichael and M. Hebert. Object recognition by a cascade of edge probes. In *BMVC*, pages 103–112, 2002.

[5] J. M. Chambers and T. J. Hastie. *Statistical models in S.* Chapman & Hall, 1993.

[6] D. Cristinacce and T. Cootes. Facial feature detection using adaboost with shape constraints. In *BMVC*, volume 1, pages 231–240, 2003.

[7] W. Fan, S. J. Stolfo, J. Zhang, and P. K. Chan. AdaCost: misclassification cost-sensitive boosting. In *Intl. Conf. on Machine Learning*, pages 97–105, 1999.

[8] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *proc. International Conference on Machine Learning*, pages 148–156, 1996.

[9] E. Grossmann. Automatic design of cascaded classifiers. In *Statistical Pattern Recognition Workshop, ICPR*, 2004.

[10] O. Jesorsky, K. J. Kirchberg, and R. W. Frischholz. Robust face detection using the hausdorff distance. In *Proc. Intl. Conf. on Audio- and Video-based Biometric Person Authentication*, pages 90–95, 2001.

[11] M. Kearns and Y. Mansour. On the boosting ability of top-down decision tree learning algorithms. In *ACM Symp. on the Th. of Computing*, pages 459–468, 1996.

[12] S. Li, Z. Zhang, L. Zhu, H.-Y. Shum, and H. Zhang. Floatboost learning for classification. In *NIPS*, 2003.

[13] R. Lienhart, A. Kuranov, and V. Pisarevsky. Empirical analysis of detection cascades of boosted classifiers for rapid object detection. In *DAGM Pattern Recognition Symposium*, 2003.

[14] B. McCane and K. Novins. On training cascade face detectors. In *Image and Vision Computing New Zealand*, 2003.

[15] S. Merler, C. Furlanello, B. Larcher, and A. Sboner. Automatic model selection in cost-sensitive boosting. *Information Fusion*, 4(1):3–10, 2003.

[16] T. M. Mitchell. *Machine Learning.* McGraw-Hill, 1997.

[17] H. Rowley, S. Baluja, and T. Kanade. Rotation invariant neural network-based face detection. Technical Report CMU-CS-97-201, CMU, 1997.

[18] R. E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990.

[19] R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. In *ICML*, 1997.

[20] R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.

[21] P. Viola and M. Jones. Robust real-time object detection. In *Proc. ICCV workshop on statistical and computational theories of vision*, 2001.

[22] M. Weber. Background image dataset. www.vision.caltech.edu/html-files/archive.html.