Automatic design of cascaded classifiers

Etienne Grossmann*

Center for Visualization and Virtual Environments, University of Kentucky

Abstract. Cascades of boosted classifiers have become increasingly popular in machine vision and have generated a lot of recent research. Most of it has focused on modifying the underlying Adaboost method and far less attention has been given to the problem of dimensioning the cascade, i.e. determining the number and the characteristics of the boosted classifiers. To a large extent, the designer of a cascade must set the parameters in the cascade using ad-hoc methods.
We propose to automatically build a cascade of classifiers, given just a family of weak classifiers a desired performance level and little more. First, a boosted classifier with the desired performance is built using any boosting method. This classifier is then "sliced" using dynamic programming into a cascade of classifiers in a nearly computation-cost-optimal fashion.

1 Introduction

Boosting, [5, 17] combines the output of many weak classifiers - ones that perform slightly better than guessing [11]. By increasing the number of weak classifiers, arbitrarily small error rates can be reached on a training set. For example, one can achieve very good face recognition by combining hundreds of very simple classifiers [21]. However, the computational complexity increases linearly with the number of weak classifiers. If the target application has few true positives, as is the case in face detection, running such a big boosted detector would be an overkill, since, as shown by Viola and Jones [21], most true negatives can be rejected easily, using a small boosted detector. Their approach was thus to build a cascade of increasingly discriminative detectors, resulting in a face detector with performance similar or better to that of previous approaches [16, 18] while being computationally much less demanding.

Since then, cascades of classifiers have become increasingly popular [9, 10, 6]. Moreover, the methodology in [21] is not specific to face detection and was effectively applied to other cases [3]. The designer of a cascade does not need an advanced task-specific classifiers, although doing so can improve the overall performance of the system.

Many variants of [21] of have been developed. In most, the research effort focuses on employing better boosting methods. Viola and Jones [20] adapt Adaboost to the case of different misclassification costs for false positive and false

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

Algorithm 1 Cascaded classifier

```
Given: Weak classifiers $h_1(), \ldots, h_T()$ and weights provided by a boosted classifier
    $H_B(X)$, cascade schedule $1 = T_0 < T_1 < \ldots < T_L = T+1$ and thresholds
    $\theta_1, \ldots, \theta_L$.
Input: Vector $X$
    Initialization: Set $l = 1, \eta = 0$
    While $l \leq L$
        1. Set $\eta = \eta + \alpha_{T_{l-1}} h_{T_{l-1}}(X) + \ldots + \alpha_{T_l - 1} h_{T_l - 1}(X)$
        2. If $\eta < \theta_l$, classify $X$ as a negative, i.e. $H(X) = -1$. Exit.
        3. Set $l = l + 1$.
    Classify $H(X) = \text{sign}(\eta)$.
```

negatives. Li et al [8] modify Adaboost to obtain better performance using less weak classifiers, resulting in better performance on the training set and lower run-time complexity, but increased training time. Wu et al [6] and McCane and Novins [12] reduce drastically the training time[1] at the cost of slightly reduced performance. Little science, in contrast, has been applied to the characteristics of the boosted classifiers in a principled manner : empirical guidelines are given in [21] and by Lienhart et al [10] to dimension the intermediate classifiers of the cascade. McCane and Novins [12] propose a method to do this automatically and we will discuss their method further in this article. The present article proposes another such method.

We argue that choosing the characteristics of the classifiers in the cascade - whether it be number of weak classifiers or error rates - is the fundamental problem. After all, the main difference between a cascade and a single big boosted algorithm is precisely this architecture, which allows to reject true negatives early and thus reduce the computation load.

In the present work, we propose to build a cascade from a single big boosted classifier. The -very simple- idea is to compute a subset of weak classifiers of the big boosted classifier and test whether the input is positive or negative. If it is positive, compute another subset of weak classifiers and test again. We continue this way until the input is rejected or the complete boosted classifier has been computed. The resulting classifier algorithm is shown in Algorithm 1.

Some properties of the proposed method are easily seen. Most importantly, the set of positive inputs of the proposed classifier $H(X)$ is included in that of the boosted classifier $H_B(X)$ on which it is based. The true positive ratio of $H(X)$ is thus smaller than that of $H_B(X)$ and its true negatives is higher. The point in "ROC space" corresponding to $H(X)$ is thus below and on the left of the point corresponding to $H_B(X)$.

This algorithm shows a clear resemblance both with Adaboost, for the choice of weights and with the cascaded classifiers cited above. Figure 1 compares graphically the three architectures considered so far. In Adaboost (a), all weak classi-

[1] Viola and Jones mention weeks of training time.

fiers are computed at once, whereas in the proposed method and in the cascade of [21], only a subset of the weak classifiers is computed between each test. The difference between the proposed method (b) and [21] (c) is that in the former, the output the previous classifiers is kept to contribute to the input of the next. As a result, the output of the last classifier in (b) is exactly the same (assuming the weights and weak classifiers are the same, which is the case) as that of the boosted algorithm (a).

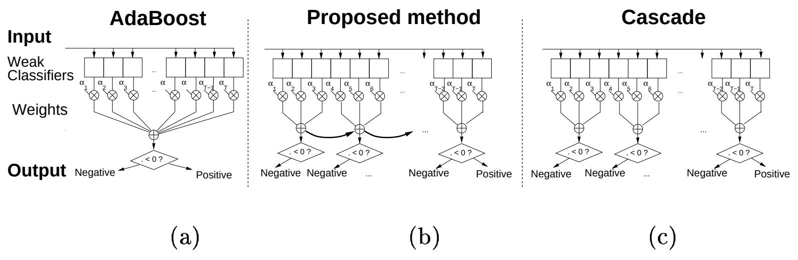

Fig. 1. Comparison of classifier architectures. The difference between adaboost (a) and (b,c) is that in the later, negative classification can be achieved after just a few weak classifier evaluations. The difference between the proposed method (b) and the cascade of [21] (c) is that the output of previous stages of decision is taken into account at each decision (bold arrows in (b)).

Going back to Algorithm 1, one sees that the output of $H(X)$ is defined by the weak classifiers $h_1(X), \ldots, h_T(X)$ and weights $\alpha_1, \ldots, \alpha_T$ that define the boosted classifier $H_B(X)$, but also by the schedule $T_1, \ldots, T_L$ and thresholds $\theta_1, \ldots, \theta_L$ of the tests. The choice of the T_l and θ_l is determinant to obtain a computationally efficient classifier and at first sight, this it may appear that setting these parameters is as hard as setting the parameters for a cascade of classifiers. However we will see that, using dynamic programming, it is possible to set these parameters in a way that :

a) Preserves the output of the boosted classifier on any given data set, e.g. on the training set of the boosted classifier.
b) Nearly optimal in terms of the computational cost of the classifier, amongst all cascades that verify **a)**.

Another benefit of the proposed method is that it is easier to choose the true positive and negative rates of the detector. Indeed, in a boosted classifier, there is a unique threshold that determines the point on the ROC curve on which the classifier lies. It is sufficient to set this threshold prior to building the cascade to guarantee that the cascade will achieve this point of the ROC curve (for the given dataset). Since the computation cost of building the cascade is negligible with respect to that of boosting, it is easy to build classifiers for any point on the ROC curve achievable by the original boosted algorithm. In contrast, previous cascading methods would either require to train a new cascade for each desired

Algorithm 2 Cascade building procedure.

Assume given

- A family of weak classifiers $h(X)$ a dataset $X_1, \ldots, X_N$ and $y_1, \ldots y_N$.
- Target false positive and true positive ratios for the classifier.
- An estimate of the proportion P_0 of positives in the targeted real-world input.
- An estimate of the computational cost of the weak classifiers and of performing a test on the targeted computer architecture.

The cascade is obtained by :

Boost the weak classifiers and build its ROC until the targeted performance or better [14] is attained.

Build a computationally-optimal cascade with the architecture of Alg. 1 that has exactly the same output as the boosted classifier on the training data.

ROC point, or use an ad-hoc method to adjust the thresholds of the boosted classifiers.

Also, the proposed cascading method is not limited to a particular boosting method. It is possible to use a boosted classifier specialized for a given cost metric [4, 20, 13] or aimed at being computationally more efficient [8]. In the present work, we will use the most studied Adaboost method [17, 11].

The proposed procedure for building a cascade of classifiers is described in Algorithm 2.

Note that we assume the proportion of true positives P_0 in the targeted real-world application is approximately known. This proportion is usually not known precisely, but one knows that positives are a small minority - without this assumption, the whole cascade architecture stops making sense. We will see below that the exact value of P_0 does not influence greatly the resulting cascade.

Also, it is assumed that the computational cost of the weak classifiers is known, as well as the computational cost of performing the test in Algorithm 1. These quantities can be determined experimentally by benchmarking the targeted computer system. We do not assume that the cost of a test is negligible, as the cost of the weak classifiers may itself be very low.

We model explicitly the computational complexity of the classifier and determine a nearly optimal cascade amongst all cascades derived from a given boosted classifier. Although there exists work that considers the cost of evaluating a classifying tree [19], we are only aware of McCane and Novins who adopt [12] a similar approach to build a computationally near-optimal cascade of detectors. However, [12] assumes an approximate model for the computational cost of a boosted classifier as a function of its false positive rate. Also, they need to consider every possible (plausible) cascade length separately and determine, by numerical optimization, sub-optimal parameters for each. In contrast, we obtain the nearly-optimal sequence using very few assumptions and with very little computation, by using dynamic programming.

2 Computationally optimal cascade design

We assume having a boosted classifier that achieves desired true positive and true negative rates on a given dataset. This can be done boosting a family of weak classifiers. until the ROC curve of the classifier passes above the desired true positive and negative ratios.

At this point, we have a boosted classifier $H_B(X)$ defined by weak classifiers $h_1(X)$,..., $h_T(X)$ and weights $\alpha_1, \ldots, \alpha_T$:

$$H_B(X) = \begin{cases} 1 & \text{if } \sum_{t=1}^{T} \alpha_t h_t(X) > 0 \\ -1 & \text{otherwise.} \end{cases}$$

Additionally, we are given a dataset $X_1, \ldots, X_N$, with known classes $y_1, \ldots, y_N$ ($y_n \in \{-1, 1\}$). Define I^+ (resp. I^-) the set of indices n such that $H_B(X_n) = 1$ (resp. -1).

2.1 Choice of thresholds $\theta_1, \ldots, \theta_L$

Consider an algorithm as in Alg. 1, characterized by the test schedule $T_1, \ldots, T_L$ and thresholds $\theta_1, \ldots, \theta_L$. Define the intermediate real-valued classifiers :

$$G_t(X) = \sum_{s=1}^{t} \alpha_s h_s(X) \qquad (1)$$

Define thresholds θ'_t for each $1 \leq t \leq T$ so that $G_t(X_n) > \theta'_t$ for all the positive examples $n \in I^+$:

$$\theta'_t > \min \left\{ G_t(X_n) \mid n \in I^+ \right\}. \qquad (2)$$

In practice, we may usually set θ'_t to the midpoint between the value above and the smallest $G_t(X_n)$ that is greater than Eq. (2) and $n \in I^-$.

We will set

$$\theta_l = \theta'_{T_l}. \qquad (3)$$

It is clear that, with this choice, and independently of the $T_1, \ldots, T_L$, the cascade defined in Algorithm 1 has the *same output* as the original boosted algorithm on the training data.

2.2 Computational cost model

The proportion of true positives and false positives that are accepted at level[2] t is estimated by :

$$\begin{aligned} p_t &= \# \{ n \mid y_n = 1 \, \text{and} \, G_t(X_n) > \theta'_t \} / \# \{ n \mid y_n = 1 \} \\ r_t &= \# \{ n \mid y_n = -1 \, \text{and} \, G_t(X_n) > \theta'_t \} / \# \{ n \mid y_n = -1 \}. \end{aligned}$$

[2] We will call levels $1 \leq t \leq T$ the indices of the weak classifiers that constitute the boosted classifier.

where # denotes the cardinal. Assuming that the proportion of positives in the real-world input is P_0, the proportion of real-world inputs that are accepted at level t is approximately :

$$q_t = P_0 p_t + (1 - P_0)\, r_t. \tag{4}$$

We assume in the following that q is decreasing in t. Although this is not necessarily true in the training data, the true ratio can be expected to be monotonously decreasing.

Now, considering any cascade of the type of Algorithm 1, the expected computation cost for a real-world input is approximately :

$$C = (\mathcal{A}_1 + B) + (\mathcal{A}_2 + B)\, q_{T_1} + \ldots + (\mathcal{A}_L + B)\, q_{T_L}, \tag{5}$$

where

$$\mathcal{A}_l = \sum_{T_{l-1} \le t < T_l} A_t,$$

A_t is the computational cost of the t^{th} weak classifier and B is the cost of performing a test on the targeted computer.

2.3 Optimal test schedule $T_1, ..., T_L$

We now show that the optimal computational cost is obtained efficiently using dynamic programming [2], because the cost C_t of the optimal cascade starting at level $t \in \{1, \ldots, T\}$ is can be defined recursively from the costs $C_{s>t}$.

The optimal cascade starting at t is necessarily one of the following : (1) the trivial cascade consisting in computing all the remaining classifiers and testing the result; (2) the cascade consisting in computing classifiers $t, \ldots, T-1$ performing the test at $T-1$ and following the optimal sequence from $T-1$ to T; ... ; $(T-t+1)$ computing the t^{th} weak classifier, testing and following the optimal sequence from $t+1$ to T. The costs of each possibility is :

$$\begin{array}{ll} C_{t,T} & = q_t\,(A_t + \ldots + A_{T-1} + A_T) + 0 \\ C_{t,T-1} & = q_t\,(A_t + \ldots + A_{T-1}) + C_T \\ \vdots & \quad \vdots \\ C_{t,t} & = q_t A_t + C_{t+1} \end{array} \tag{6}$$

The optimal cascade cost and optimal "jump" at t are thus :

$$C_t = \min\{C_{t,s} \mid t < s \le T\},\; S_t = 1 + \arg_s \min\{C_{t,s} \mid t < s \le T\}. \tag{7}$$

From there, the optimal cascade sequence is :

$$T_0 = 1,\, T_1 = S_1,\, T_2 = S_{T_1}, \ldots, T_t = S_{T_{t-1}}, \ldots, T_L = T + 1. \tag{8}$$

Note that it is not necessary to specify the length of the optimal sequence. Also, the resulting cascade does not depend on the proportion of true and false

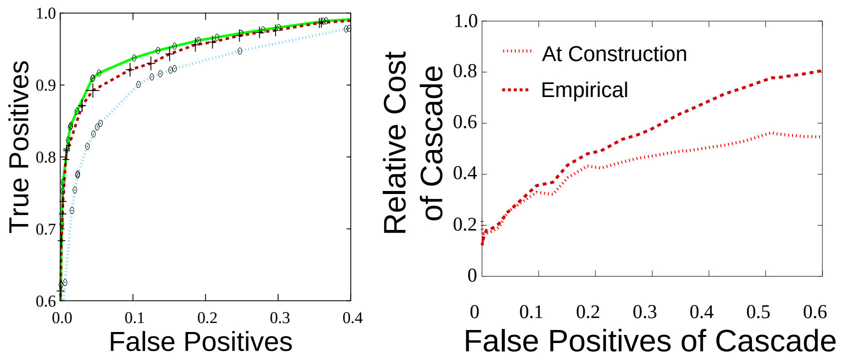

Fig. 2. Left: Detail of ROC of boosted (plain line), cascaded classifiers (dashed) and boosted classifier with same cost as cascades (dotted). **Right:** Relative computational cost of cascade w.r.t. original classifier v.s. false positive rates.

positives in the training data, but on the expected proportion P_0 of positives in the real-world data.

The dependence on P_0 is itself quite mild : the terms in Eq. (6), can be developed as affine expressions of P_0 with coefficients depending on p_t, r_t, A_t and B in which the constant term is proportional to the false positive rate r_t. Near the end of any cascade, the cost is thus, nearly linear in P_0 and the optimal cascade does not depend on P_0. Even earlier in the cascade, when r_t is greater, we have found that the optimal sequence does not vary greatly with P_0.

Finally, on should note that the obtained scheduling is optimal only if the series q_t defined in Eq. (4) is decreasing, which is not necessarily the case for a given dataset, although this is the expected behaviour of the boosted classifier. We found that in practice, these series are not monotonous, and that some smoothing could be applied[3]. However, since the general tendency of the unsmoothed series is monotonous, we consider that the cost of the resulting sequence is near the true minimum.

3 Experimentation

This section, presents experimental validation for the theory developed above. For this purpose, we use a face classifier still in development, which uses the same Haar filters as [21]. The training dataset consists in 2000 face images taken from the BioID database [7], from dataset C of the CMU database [15] and from images gathered on the internet. All face and non-face images were scaled to 18-by-31 pixels and were normalized in variance and range. Images from the BioID database were cropped repeatedly at slightly different positions and scales. A validation dataset of 5000 other images is built in the same way. On our computer system, it was found that the cost A of weak classifiers is approximately 40 times that of performing a test and this value is used in the sequel.

[3] We do not use any in the experimental section, unless otherwise specified.

Comparison of performance of boosted classifier and cascades In this experiment, a boosted classifier[4] consiting of $T = 100$ weak classifiers is built using the training dataset and its ROC is computed using the validation dataset. Only points on the upper convex hull of the ROC curve are kept. For each non-trivial vertex of the ROC, the corresponding cascaded classifier is built, using the training dataset and assuming $P_0 = 0.001$. The cascades have from 29 to 37 levels (average: 34.1).

Figure 2 (left) shows part of the ROC curve of the boosted classifier (topmost curve) the curve linking linking the performances of the cascaded classifiers obtained from the boosted classifier with the thresholds of the ROC vertices. The lower dotted ROC is that of the boosted classifier truncated at the 34th weak classifier, having thus similar computation cost as the cascades. This experiment shows the performance of the cascade is very close to that of the underlying boosted classifier, while decreasing the average computation cost very much.

Figure 2 (right) shows the computational cost of the cascade, assuming that $P_0 = 0.001$ divided by that of the original boosted classifier. The dotted curve is the expected value determined while constructing the cascade, while the the dashed curve above is the value observed on the validation dataset.

Since P_0 is very low, it is natural to set the classifier with a very low false positive rate, to avoid being overwhelmed by false positives. Figure 2 (right) shows that in this condition, the empirical and theoretical compuation costs are very similar and that they are at their lowest.

Architecture of cascade with varying P_0 In this experiment, the cascade is built for $P_0 \in \left\{1/10^{1+i/2} \mid 0 \leq i \leq 6\right\}$ with a fixed boosted classifier ($T = 100$) and input data. This experiment showed that the schedule is the same for nearly all cascades, with those with higher levels of P_0 skipping some tests. This confirms our claim that the value of P_0 is not determinant in the resulting cascade.

4 Discussion and conclusions

We have proposed a method to automatically design a cascade classifier with a desired performance, targeted for a given type of input and a given computer architecture. This methodology applies to any underlying boosting method, whether the best-known algorithm of Schapire and Singer [17] or, as was the case in the experimental section, a boosting method targeted at a given region of the ROC space. Since the resulting cascade is so closely related to its underlying boosted classifier, we can expect that the theoretical properties of this cascade will be easier to study than that of ad-hoc cascades and that this would be a step towards being able to set the generalization bounds of the cascade, rather than its performance on a given dataset.

Future research is required to find ways of setting the thresholds of the cascade levels so as to follow the ROC of the original classifier better still than was

[4] We use an unpublished boosting method inspired of [1] and [20] adapted to unequal importances of false positives and false negatives.

shown in the experimental section. Another direction of research is to better estimate the proportion q_l of examples that reach a level l of the cascade.

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. R. Bellman. *Dynamic programming*. Princeton University Press, 1957.
3. O. Carmichael and M. Hebert. Object recognition by a cascade of edge probes. In *BMVC*, pages 103–112, 2002.
4. 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.
5. Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *proc. International Conference on Machine Learning*, pages 148–156, 1996.
6. M. D. Mullin J. Wu, J. M. Rehg. Learning a rare event detection cascade by direct feature selection. In *NIPS*, 2003.
7. 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.
8. S. Li, Z. Zhang, L. Zhu, H.-Y. Shum, and H. Zhang. Floatboost learning for classification. In *NIPS*, 2003.
9. S.Z. Li, L. Zhu, Z.Q. Zhang, A. Blake, H.J. Zhang, and H. Shum. Statistical learning of multi-view face detection. In *ECCV*, 2002.
10. 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.
11. S. Mannor and R. Meir. Geometric bounds for generalization in boosting. In *Proc. Conference on Computational Learning Theory*, 2001.
12. B. McCane and K. Novins. On training cascade face detectors. In *Image and Vision Computing New Zealand*, 2003.
13. S. Merler, C. Furlanello, B. Larcher, and A. Sboner. Automatic model selection in cost-sensitive boosting. *Information Fusion*, 4(1):3–10, 2003.
14. F. J. Provost and T. Fawcett. Analysis and visualization of classifier performance: comparison under imprecise class and cost distributions. In *International Conference on Knowledge Discovery and Data Mining*, 1997.
15. H. Rowley, S. Baluja, and T. Kanade. Rotation invariant neural network-based face detection. Technical Report CMU-CS-97-201, CMU, 1997.
16. H. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *IEEE Trans. PAMI*, 20:22–38, 1998.
17. R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.
18. H. Schneiderman and T. Kanade. A statistical method for 3D object detection applied to faces and cars. In *Proc. ICCV*, 2000.
19. P. D. Turney. Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm. *Journal of Artificial Intelligence Research*, 2:369–409, 1995.
20. P. Viola and M. Jones. Fast and robust classification using asymmetric adaboost and a detector cascade. In *NIPS*, 2001.
21. P. Viola and M. Jones. Robust real-time object detection. In *Proc. ICCV workshop on statistical and computational theories of vision*, 2001.