Automatic Cascade Training with Perturbation Bias

Jie Sun James M. Rehg Aaron Bobick
GVU Center / College of Computing
Georgia Institute of Technology
{sun, rehg, afb}@cc.gatech.edu

Abstract

Face detection methods based on a cascade architecture have demonstrated fast and robust performance. Cascade learning is aided by the modularity of the architecture in which nodes are chained together to form a cascade. In this paper we present two new cascade learning results which address the decoupled nature of the cascade learning task. First, we introduce a cascade indifference curve framework which connects the learning objectives for a node to the overall cascade performance. We derive a new cost function for node learning which yields fully-automatic stopping conditions and improved detection performance. Second, we introduce the concept of perturbation bias which leverages the statistical differences between target and non-target classes in a detection problem to obtain improved performance and robustness. We derive necessary and sufficient conditions for the success of the method and present experimental results.

1. Introduction

Cascade classifiers have shown great success in face detection by addressing the rare event nature of the detection task. One strength of the cascade architecture is its modularity, which is achieved by decomposing the classifier into a chain of nodes. In comparison to training a monolithic classifier, the training task for each node is more tractable. The price of this decomposition, however, is the loss of a direct relationship between the decisions taken at each stage and the overall performance of the cascade. What is the relationship between node performance and cascade performance? How do the statistical properties of the target class affect our ability to learn good classifier nodes?

In this paper we present two new cascade learning results which address these important questions. First, we introduce a cascade indifference curve framework to reason about the impact of individual node performance on overall cascade performance, and we propose a novel cost function for relating them. We show that this cost function yields an optimal learning goal for each node (with respect to the overall cascade performance) and adjusts the goal according to the difficulty of the node learning problem. The result is a fully-automatic cascade learning algorithm which eliminates much of the ambiguity in building high-quality nodes.

Second, we introduce the concept of perturbation bias which leverages statistical differences between the target and non-target classes in a detection problem to obtain increased performance. We derive necessary and sufficient conditions for the success of the perturbation method. We demonstrate experimentally that training a face detection cascade with symmetric perturbations significantly improves the detection performance.

2. Related Work

Robust and fast object detection is a difficult problem in computer vision. One popular example is human face detection. A recent survey can be found in [19]. Our approach is based on the cascade framework of [14], which uses a feature selection method based on AdaBoost to form ensembles of features in each node. Recently, several alternative node training methods have been proposed. Examples include Asymmetric AdaBoost [15], Float Boost [7], Kullback-Leibler Boost [9], Gentle Boost [8], and boosting chain [18]. A computationally efficient node training algorithm based on forward feature selection is described in [17].

In addition to the cascade framework, other methods for computationally efficient classification have been proposed. The neural network-based detector of Rowley et. al. [11] incorporated a manually-designed two node cascade. Keren et. al. [5] is based on the sequential application of Antifaces to the input image. Other cascade structures have been constructed for SVM classifiers [10, 4] and likelihood ratio tests [12].

In other related work, Carmichael and Hebert propose a hierarchical strategy for detecting wiry objects at different orientations and scales [3]. Baker and Nayar described a pattern rejection theory for multi-class recognition in [2]. Leung et. al. use local feature detectors to avoid the brute force search over all image windows [6]. Amit et. al. explore techniques for learning tree classifiers in [1].

3. Optimized Cascade Learning

By decomposing a monolithic face detector into a cascade of nodes, the cascade architecture achieves computational efficiency and a decoupled learning problem. With this decoupling, however, comes the danger that decisions taken in isolation for individual nodes will not be optimal with respect to the overall performance of the cascade.

There are two important properties of the cascade architecture. First, there are many different ways to achieve the same overall detection rate h and false positive rate f. For example, if we assume that cascade nodes make independent errors, then a chain of two nodes, each of which having $h = 0.999$ and $f = 0.7$, can achieve the same performance as a single node of $h = 0.998$ and $f = 0.49$. Second, while false rejects are generally nonrecoverable, false positives from early nodes can be rejected at later nodes. Thus it is preferable to postpone uncertain rejection. Ideally, the learning goal for a node should reflect these two basic properties of the cascade architecture.

In [14] it is suggested that each node be designed to achieve a fixed performance goal (h, f). Assuming that the learner is successful and that the nodes make independent errors, a cascade of length n would have performance (h^n, f^n). However it is not practical to specify a fixed objective for every node. Due to the increasing difficulty of the learning problem with the depth of the cascade, it is not possible to learn a node that meets both the h and f goals. An alternative is to specify f for each node and when this goal is met, select the highest possible h. However, in practice, this approach has serious problems, as we show in the following illustrative example.

Figure 1(a) shows the 2-D feature space for a classification problem where the light grey squares belong to the target class and the dark grey/blue squares belong to the non-target class. Further, we suppose that our training algorithm for each node can learn classifiers with a square boundary $S(p, l)$, where p is the position of the center of the square and l is the length of the side, such that the area inside the boundary is rejected. Under this experimental setting, the cascade classifier rejects a square region at each node.

Figure 1(b) illustrates a cascade of 6 nodes that achieves perfect classification. However, if we apply a stopping criterion with a predefined false positive rate $f = 50\%$, the learner will make substantial false rejections even in the first node, as illustrated in figure 1(c). The node learning goal f is too ambitious, and in order to achieve it, the detection rate cannot exceed 79.3%. Choosing a higher fixed false positive rate goal is not necessarily any better, as shown in figure 1(d). The problem is that the learner does not adjust its decision to obtain the best false positive rate, but uses a predefined $f = 80\%$, which is unnecessarily high. As a consequence, the first two nodes leave the third node with a difficult classification task, as evidenced by the zigzag regions.

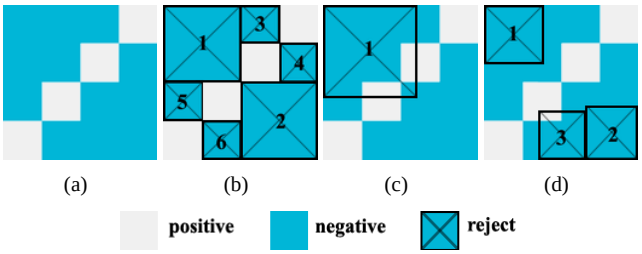

Figure 1: Example of cascade training. (a) The distribution of the target class (light gray) and the non-target class (dark gray/blue) in a 2-D feature space. (b) A cascade that achieves perfect classification. (c) The result of setting the false positive rate goal to 50%. (d) The result of setting a higher false positive rate goal of 80%. As a consequence of the first two nodes, the third node is problematic.

These examples suggest that the learning objectives for each node should be adjustable: while aggressive objectives are most appropriate for early nodes in the cascade, later nodes should have relaxed objectives. A classical method for flexible trade off between false positive and false negative is the Bayes risk criterion:

$$R = \eta(1 - h) + f, \tag{1}$$

where η is the Bayes cost. Several authors have explored this idea in a cascade setting [15, 16, 18]. However, we will show that any single choice of η in this approach will not be optimal in the cascade setting.

In order to address these problems, we introduce a *cascade indifference curve* framework. We design a new cost function such that the minimum cost corresponds to the optimal node configuration with respect to the overall cascade performance. Our cost function enables the training algorithm to reject the non-target class as effectively as possible while maintaining a high detection rate. As a result, it can generate the optimal cascade illustrated in figure 1(b).

3.1. Cascade Indifference Curve

A classifier node can be configured to achieve a desired (h, f) target by adjusting the decision parameters, θ. In an ensemble classifier, for example, θ is the threshold in the decision rule $\sum_i \alpha_i b_i(x) > \theta$. Each value of θ corresponds to a particular operating point (h, f) on a ROC curve. If we assign a cost to each possible (h, f), then the optimal configuration for the node is attained by choosing the θ^* with the lowest cost.

A set of (h, f) points with the same cost value is known as an indifference curve. Specifying a fixed (h, f) target for

a node is equivalent to specifying an indifference region defined by linear inequalities (e.g. $h > 0.99$ and $f < 0.5$). Using the Bayes risk from equation 1 as the cost leads to indifference curves that are lines with slope $1/\eta$, as illustrated in figure 2. Unfortunately, neither of these criteria applied to a node can be optimal for the cascade as a whole, as we will show in the following section.

Now consider two nodes, A and B, with performance (h_1, f_1) and (h_2, f_2), respectively. Suppose that integers n and m can be found such that $(h_1^n, f_1^n) = (h_2^m, f_2^m)$. This equation can be viewed as defining two points on a cascade indifference curve, which reflects the fact that the performance of a chain of n nodes of type A is equivalent to a chain of m nodes of type B. Eliminating n and m from the relation yields $\log(h_1)/\log(f_1) = \log(h_2)/\log(f_2)$. In general the equation $\log(h)/\log(f) = t$ defines a family of indifference curves indexed by t. We will now use this observation to design a cost function for cascade learning.

3.2. A Risk Formulation for Cascade Learning

In order to compare the cost of two node configurations from separate indifference curves, we relate the performance of a single node to the expected hit rate for a cascade which has been designed to achieve a very small overall false positive rate ϵ. For a node with performance (h, f), let n be the number of nodes required to meet the false positive goal, $n = \log(\epsilon)/\log(f)$. This results in an expected hit rate for the cascade

$$H = h^{\log(\epsilon)/\log(f)} = \epsilon^{\log(h)/\log(f)}. \quad (2)$$

We define the basic cost of a node in the cascade with performance (h, f) to be:

$$C_b = -\epsilon^{\log(h)/\log(f)} \quad (3)$$

This cost function gives an optimal trade-off between the false positive and false negative rates, based on the desired overall false positive rate for the cascade.

Figure 2 illustrates a comparison between our cascade risk and the more conventional strategy of applying a standard Bayes risk at each node. Note that although the Bayes risk criterion with cost $\eta = 12$ selects the same point as our method on the ROC curve for node 9, its choice at node 1 is inferior, because the hit rate is unnecessarily low. The difference is that our method adjusts the trade off between the false positive and false negative rates dynamically according to the overall performance of the node. It tends to penalize false negatives more if a very high hit rate can be achieved with a moderate false positive rate. At earlier nodes in the cascade, if a very high hit rate can be achieved, then our algorithm is willing to accept a relatively high false positive rate and proceed to the next node. This favor toward the hit rate is gradually reduced as the training of the

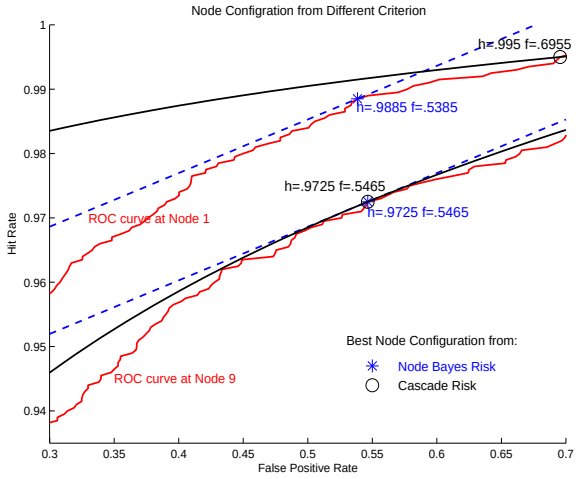

Figure 2: Indifference curves for Cascade Risk (solid) vs. Node Bayes Risk (dashed) with cost $\eta = 12$.

cascade advances to later nodes. It is clear that no single cost η in the conventional Bayes risk framework will provide the same flexibility.

In order to obtain the complete cost function for a node, we augment C_b to include the expected computational cost. In an ensemble classifier, this is proportional to the number of features that need to be evaluated. This allows us to select a single operating point on the cascade indifference curves, corresponding to the least expensive solution. The expected computational cost for adding a node of N features to a cascade is given by

$$C_c = N[P(F)H_{i-1} + (1 - P(F))F_{i-1}] \approx NF_{i-1} \quad (4)$$

where $P(F)$ is the probability that an input pattern is a face and H_{i-1} and F_{i-1} are the hit rate and false positive rate of the cascade evaluated through node $i - 1$. In the rare event case $P(F) << 1$, justifying the approximation.

The final cost function can then be written

$$C = \begin{cases} -\epsilon^{\log(h-\delta)/\log(f)} + \lambda NF_{i-1} & \text{if } f \in [f_1, f_2] \\ \infty & \text{otherwise} \end{cases} \quad (5)$$

where λ is a constant to balance accuracy and computational efficiency, and the small constant δ avoids a constant indifference curve in the case where $h = 1$.[1] Minimizing the cost function in equation 5 automatically determines the number of features in the cascade by providing a stopping criterion for the node learning algorithm. We limit the false positive rate to the range $[f_1, f_2]$ so that the cascade is of reasonable length. The following pseudocode gives the *node learning algorithm* based on the cascade risk cost function:

[1] Note that C is implicitly a function of N, the number of features, and θ, the threshold setting which determines h and f.

1. Set the current best cost, C_{min}, to ∞. Set $N = 1$.

2. Select the N_{th} feature for the ensemble.

3. Apply the ensemble with N features on the validation data set to create a ROC curve. Adjust the threshold to θ_N, which corresponds to the point on the ROC curve with the smallest cost C_N according to equation 5.

4. If $C_N < C_{min}$, then the best point on this ROC curve is better than all points on all previous ROC curves. Therefore set $N_{best} = N$ and $C_{min} = C_N$

5. If $\lambda N F_{i-1} > C_{min}$ or $N = N_{max}$, end with an ensemble of the first N_{best} features with threshold $\theta_{N_{best}}$. Otherwise, let $N = N + 1$ and go to step 2.

At later nodes in the cascade, λF_{i-1} will approach 0 and the cost value will not be sensitive to N. In this case, the algorithm always terminates once N_{max} features have been selected. Early nodes, on the other hand, will terminate before N reaches N_{max}, when the additional computational cost cannot be compensated by a decrease in C_b in equation 3. This agrees with the intuition that later nodes in the cascade require more features to obtain good performance, and this can be achieved without negatively impacting the computational efficiency in the testing phase.

3.3. Experimental Results for Face Detection

We performed two comparative face detection experiments. In each experiment, two cascade classifiers were trained on the same data set, one using the fixed false positive rate target for each node, the other using the cost function in equation 5.

In the first experiment, both cascades were trained using the AdaBoost method and the feature set from [14]. Each node has a maximum number of 100 features. They were trained with 1000 face and 1000 non-face samples, and the validation set contained another 1000 faces and 4000 non-faces. The two cascades were tested on 9000 face and 9,552,529 non-face samples. The ROC curves for the two cascades are shown in Figure 3.[2] The ROC curve trained using the cost function in equation 5 is consistently better than the approach with the predefined fixed false positive rate of 0.5.

In the second experiment, we used the forward feature selection (FFS) algorithm described in [17]. We obtained similar improvement,[3] suggesting that our method is applicable to general cascade learning. In these experiments, we observed that for the difficult, deeper levels in the cascade, our method tended to add more nodes to achieve a given false positive rate, while preserving the high detection rate. Note that practitioners commonly address this problem by making manual tradeoffs in performance for deeper nodes in the cascade. This introduces an unwanted level of non-determinism into the learning process.

[2]Each point in the plot corresponds to iteratively removing the last node from the cascade.

[3]The ROC curve for FFS (without weight setting) in this experiment is not representative of the best performance of the method, due to a greatly reduced training set.

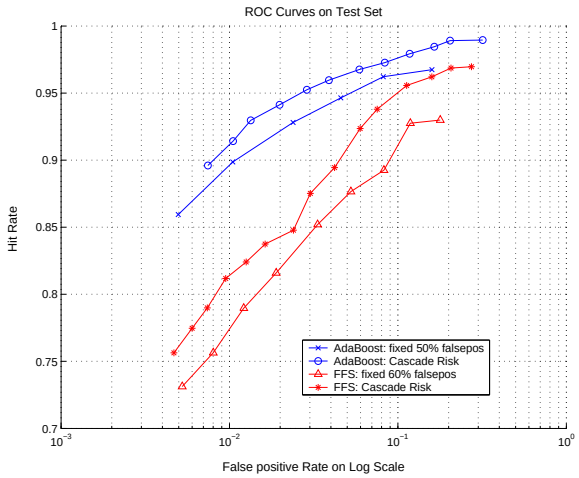

Figure 3: Training with cascade risk in comparison to a fixed false positive rate. Cascade risk yields improved performance using both AdaBoost and FFS.

4. Perturbation Bias for Improved Detection

We have described a relationship between node performance and cascade performance that leads to an improved cost function for cascade learning. We now examine the statistical properties that differentiate detection problems from other pattern recognition problems. In pattern recognition, we often have a multi-class problem in which each class defines a smooth manifold in some feature space. Furthermore, we may require recognition to be invariant to certain classes of transformations such as rotations. In this scenario, methods such as Tangent Distance [13], which is based on calculating the distance between two manifolds, can achieve transformation invariant classification. In contrast, pattern *detection* is the problem of discriminating a target pattern against all others. The non-target "class" frequently does not lie on a smooth manifold in the feature space. Typically, the existence of such a "none of the above" category complicates the pattern detection task requiring not only a likelihood test but also some absolute threshold.

Normally, we have some prior knowledge of perturbation invariance for the target class, i.e. the perturbation does not affect our perception of whether an instance belongs to the target class or not. For example, suppose we have a de-

tector that accepts faces. Noting that symmetrically flipped[4] versions of face images are still valid face images, it should be the case that the symmetrically flipped versions of any face input should likewise be accepted.

Now consider a negative example that is incorrectly accepted by the classifier - a *false positive*. What is the probability that the symmetrically flipped version of this negative example will also be accepted as a face? Clearly this quantity depends upon the particular classifier. For example, if the classifier is designed to be transformation invariant, then flipping the input will have no effect. If this is not the case, however, then it is quite *unlikely* that the flipped version of an arbitrarily chosen negative example will also be accepted as a face. We refer to this statistical difference as the *perturbation bias*.

Let us begin to formalize this insight for some perturbation g of an image. Let $I^{w\times h}$ be the space of all image windows of width w and height h. A perturbation is defined as a transformation of the input, such that the perturbed is still a valid input: $g : I^{w\times h} \rightarrow J^{w\times h}$

Let $D(x)$ represent the event that the detector evaluates positively on an input image x; let $x \in F$ denote that the input is indeed an image of the target class (in this case faces); and let $g(x)$ be the perturbation of the input image x. Then the perturbation bias can be described probabilistically as

$$p(D(g(x))|D(x), x \in F) > p(D(g(x))|D(x), x \notin F). \tag{6}$$

When is this bias likely to be present? If the classifier does not inherently enforce perturbation invariance and both x and $g(x)$ are present in the positive training set, then the perturbation bias should exist. For face detection, for example, existing training algorithms usually do not inherently enforce symmetric invariance.

Let us now consider a new joint detector that classifies an input as a face if and only if both the original and the symmetrically flipped images are assigned a positive label by the base classifier. We can get an estimate of the improved detection performance. Suppose originally, $p(x \in F|D(x)) = \alpha$, then with an additional test on the perturbed input:

$$\frac{p(x \in F|D(x), D(g(x)))}{p(x \notin F|D(x), D(g(x)))}$$

$$= \frac{p(D(g(x))|D(x), x \in F)}{p(D(g(x))|D(x), x \notin F)} \cdot \frac{p(x \in F|D(x))}{p(x \notin F|D(x))} = \frac{k\alpha}{(1-\alpha)} \tag{7}$$

where

$$k = \frac{p(D(g(x))|D(x), x \in F)}{p(D(g(x))|D(x), x \notin F)}.$$

[4]Reflection about the vertical axis of symmetry (i.e. the mirror image).

Thus we have:

$$p(x \in F|D(x), D(g(x))) = \frac{k(1-\alpha)+k\alpha}{1-\alpha+k\alpha} p(x \in F|D(x)) \tag{8}$$

If the condition in equation 6 holds, then $k > 1$ and

$$p(x \in F|D(x), D(g(x))) > p(x \in F|D(x)). \tag{9}$$

This implies that an additional positive test result on the perturbed input can reduce the risk of false positives.

4.1. Necessary and Sufficient Conditions for Improvement

In the Appendix, we derive some necessary and sufficient conditions such that perturbation bias can be exploited. Here we describe the implications of these conditions for constructing a more effective classifier.

To proceed, let us redefine the detection event as $D^\theta(x)$ where θ represents the decision parameters of the detector. One naive approach to apply equation 9 is to make a new classifier $D^\theta(x) \cdot D^\theta(g(x))$, which requires both the original input and its perturbation to pass the original test. We can show however, this does not always improves the overall performance. Let h' and f' be the new hit rate and false positive rate respectively. Each new rate is clearly the product of the original rate and the conditional probabilities of $D^\theta(g(x))$:

$$\begin{aligned} h' &= p(D^\theta(x)|x \in F)) \cdot p(D^\theta(g(x))|D^\theta(x), x \in F) \\ f' &= p(D^\theta(x)|x \notin F)) \cdot p(D^\theta(g(x))|D^\theta(x), x \notin F) \end{aligned} \tag{10}$$

Since the conditional probabilities are usually less than 1, both the hit rate and the false positive rate will be reduced. Therefore, the potential benefit of this approach hinges on the specific trade-off between the hit rate and the false positive rate. In some cases, the performance of the new classifier could be worse than before.

Figure 4 illustrates a case where the straightforward construction of a joint classifier is not effective. In this example we took a trained cascade and make a new detector by duplicating each node in the original cascade to obtain two nodes: one that tests x and another that tests $g(x)$. The input is rejected if it fails either test. As the figure illustrates, this strategy does not result in significant improvement.

The appendix contains the conditions under which $D^\theta(x) \cdot D^\theta(g(x))$ improves the performance of $D^\theta(x)$. We show that in addition to the condition in equation 6, the value of $p(D^\theta(g(x))|D^\theta(x), x \in F)$ should also be above a certain threshold. Note that this analysis assumes that the *same threshold* is used in both $D^\theta(x) \cdot D^\theta(g(x))$ and $D^\theta(x)$.

However, the freedom to adjust the classifier threshold during training would allow us to bias the learner and

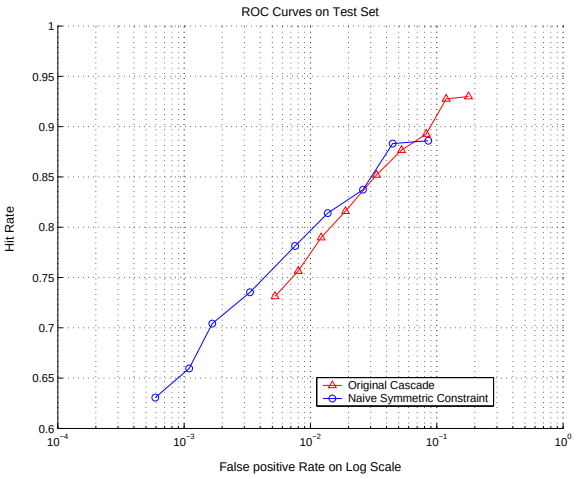

Figure 4: Failure of the naive approach to exploiting symmetry, in which a new cascade classifier $D^{\theta}(x) \cdot D^{\theta}(g(x))$ is formed from a previously-trained $D^{\theta}(x)$.

achieve improved performance. For example, we can adjust the threshold to θ', such that the joint detector $D^{\theta'}(x) \cdot D^{\theta'}(g(x))$ has the same hit rate as the previous $D^{\theta}(x)$, while the false positive rate, under certain reasonable conditions, is reduced. In general, we use a cost function to identify the threshold that results in best trade off between the new hit rate h' and the false positive rate f'.

In particular, for cascade learning, we use the cost function from equation 5. The training of a node proceeds as follows: each super node is a sequence of two identical sub nodes, with one of them testing the input image and the other testing the symmetric transformation. Training a super node consists of adjusting θ' for the two subnodes to minimize the cost.

4.2. Training with Input Perturbation

A general algorithm for adding input perturbation to an existing training algorithm A is as follows:

1. Find a perturbation that satisfies the following conditions:
   (a) It is known a priori that a perturbed instance of the target class still belongs to that class.
   (b) The training algorithm A does not enforce the invariance, i.e. it does not guarantee that a perturbed image has the same classification result as the original image.
2. For every positive training sample x, add its perturbation $g(x)$ to the training set.
3. Use algorithm A to train a classifier D, and adjust the threshold to θ' such that the final classifier $D^{\theta'}(x) \cdot D^{\theta'}(g(x))$ gives the best performance.

With the cascade framework in particular, such training is done for each node individually. Note that standard node training algorithms and feature sets [14] do not enforce symmetric invariance. We therefore choose symmetric transformation as our perturbation, which is defined as: $g : I^{w \times h} \rightarrow J^{w \times h}$ such that $J(x, y) = I(w - x, y)$.

A detailed cascade training algorithm which incorporates both the cascade risk criterion and symmetrical input perturbation can be found in figure 5:

1. Given a training set with M_F faces and M_N non-faces, include the symmetric transformation of the positive samples, giving a total of $2M_F$ faces and M_N non-faces.
2. Use a node training algorithm (such as AdaBoost) to select a maximum number of features, N_{max}.
3. For every $N \leq N_{max}$, identify the threshold θ_N for the ensemble that achieves the minimum cost C_N according to equation 5.
   The cost is measured on a validation set. In this step, an input x is considered positive if and only if both x and its symmetric transform $g(x)$ pass the test.
4. Identify the first N_{best} features, where
   $$N_{best} = \arg\min_{N \leq N_{max}} C_N.$$
   The final ensemble uses the N_{best} features and the threshold $\theta_{N_{best}}$.
5. Bootstrap to get a new non-face training data set, where each new non-face sample and its symmetric transformation pass all previous nodes.
6. Estimate the false positive rate of the current cascade in step 5. If the overall cascade false positive rate goal is not met, go to step 1; otherwise exit.

Figure 5: Cascade training algorithm, incorporating input perturbation and cascade risk.

4.3. Experimental Results and Discussion

We performed two experiments to assess the benefit of training with input perturbation. In the first experiment, we trained a cascade using the algorithm from figure 5 (where step 2 employed the FFS method from [17]) along with the datasets described in Section 3.3. We compared the result to

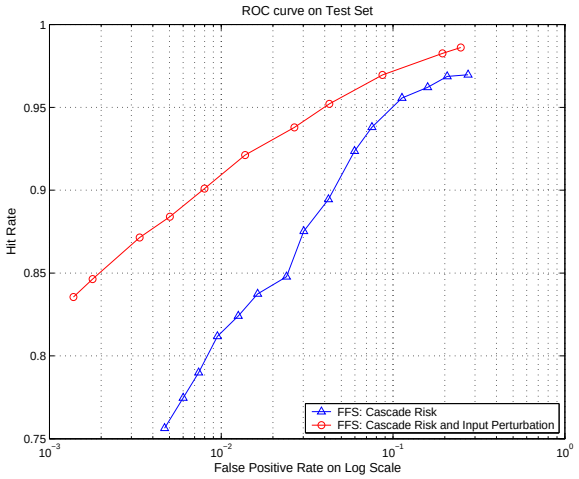

Figure 6: The ROC curve from figure 3 (triangles), using cascade risk only, is significantly improved by incorporating perturbation bias (circles).

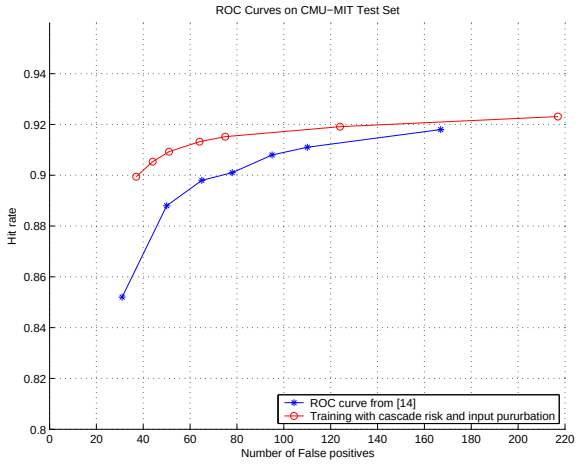

Figure 7: Evaluation using CMU-MIT test set.

the cascade from figure 3, which was trained with the Cascade risk criterion alone. The only difference between the two cascades is the use of input perturbation during training; all other factors are the same. Figure 6 illustrates the substantial improvement that resulted from using the input perturbation method.

In our second experiment, we compared the performance of our method to a baseline classification result on a standard test set. We used the algorithm in figure 5 in conjunction with AdaBoost to train a frontal face detector. The training set at each node consisted of 2000 faces and 2000 non-faces, all of which were 19×19 image patches. While the face training samples were the same for each node, we used bootstrapping to obtain new non-face samples. The validation set contained another 2000 faces and 2000 non-faces. The final 38 node cascade was evaluated on the CMU-MIT test set, following the procedure in [14]. Figure 7 shows a comparison between the ROC curve produced by our method and the ROC curve from [14], which serves as a baseline. This result, in combination with figure 6, establishes the benefit of our method. Another important advantage of our approach is that it is fully automatic.

We note that although the class of face images possesses symmetric invariance, most existing face detectors fail to enforce this property. In fact, our perturbation bias method results in a classifier that is invariant to symmetric perturbations, since $g(g(x)) = x$ in the case of symmetric flipping. In general, our input perturbation approach leverages the absence of perturbation invariance in a detector and yields improved performance with only minor modifications to the training algorithm.

5. Conclusions

We have described two new results for cascade learning which address the consequences of decoupling in the cascade architecture. Our first set of results connect the learning objectives for a node to the overall performance of the cascade. We present a new cascade indifference curve which leads to improved cascade performance and fully-automatic stopping conditions for node learning.

Our second set of results explore the concept of perturbation bias for detection problems in which the target class (such as faces) is highly structured while the non-target class (all non-faces) is not. We show that this statistical difference can be exploited through perturbations of the input patterns. We demonstrate that significant improvements in detection performance can be obtained through perturbation bias. We derive necessary and sufficient conditions for the success of the method.

In future work, we plan to explore other classes of perturbations beyond the symmetric transform and investigate the application of perturbation bias in other problem domains.

Acknowledgements

This work was funded in part by NSF CAREER grant IIS-0133779. The authors would like to thank Charlie Brubaker and Howard Zhou for useful discussions about this work.

References

[1] Y. Amit, D. Geman, and K. Wilder. Joint induction of shape features and tree classifiers. *IEEE Trans. PAMI*, 19(11):1300–1305, 1997.

[2] S. Baker and S.K. Nayar. Pattern rejection. In *Proc. CVPR*, pages 544–549, 1996.

[3] O. Carmichael and M. Hebert. Shape-based recognition of wiry objects. In *Proc. CVPR*, 2003.

[4] B. Heisele, T. Serre, S. Mukherjee, and T. Poggio. Feature reduction and hierarchy of classifiers for fast object detection in video images. In *Proc. CVPR*, volume 2, pages 18–24, 2001.

[5] D. Keren, M. Osadchy, and C. Gotsman. Antifaces: A novel, fast method for image detection. *IEEE Trans. on PAMI*, 23(7):747–761, 2001.

[6] T. Leung, M. Burl, and P. Perona. Finding faces in cluttered scenes using random labeled graph matching. In *Proc. ICCV*, pages 637–644, 1995.

[7] S.Z. Li, Z.Q. Zhang, H. Shum, and H.-J. Zhang. FloatBoost learning for classification. In *NIPS 15*, December 2003.

[8] R. Lienhart, A. Kuranov, and V. Pisarevsky. Empirical analysis of detection cascades of boosted classifiers for rapid object detection. Technical report, MRL, Intel Labs, 2002.

[9] C. Liu and H. Shum. Kullback-leibler boosting. In *Proc. CVPR*, volume I, pages 587–594, 2003.

[10] S. Romdhani, P. Torr, B. Schoelkopf, and A. Blake. Computationally efficient face detection. In *Proc. ICCV*, pages 695–700, 2001.

[11] H. A. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *IEEE Trans. on PAMI*, 20(1):23–38, 1998.

[12] H. Schneiderman and T. Kanade. A statistical method for 3d object detection applied to faces and cars. In *Proc. CVPR*, pages 746–751, 2000.

[13] P. Y. Simard, Y. A. LeCun, J. S. Denker, and B. Victorri. Transformation invariance in pattern recognition-tangent distance and tangent propagation. *International Journal of Imaging System and Technology*, 11(3):181–194, 2001.

[14] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proc. CVPR*, pages 511–518, 2001.

[15] P. Viola and M. Jones. Fast and robust classification using asymmetric AdaBoost and a detector cascade. In *NIPS 14*, 2002.

[16] J. Wu, J. M. Rehg, and M. D. Mullin. Direct feature selection for face detection. In *Proc. of Intl. Workshop on Statistical and Computational Theories of Vision*, 2003.

[17] J. Wu, J. M. Rehg, and M. D. Mullin. Learning a rare event detection cascade by direct feature selection. In *NIPS 16*, 2004.

[18] R. Xiao, L. Zhu, and H.-J. Zhang. Boosting chain learning for object detection. In *Proc. ICCV*, volume I, pages 709–715, 2003.

[19] M.-H. Yang, D. J. Kriegman, and N. Ahujua. Detecting faces in images: a survey. *IEEE Trans. on PAMI*, 24(1):34–58, 2002.

Appendix

We derive the necessary and sufficient conditions under which the joint classifier $D(x) \cdot D(g(x))$ yields improved performance over $D(x)$. Following the notation from equation 10, we can express the hit rate and false positive rate of the joint classifier as

$$\frac{h'}{f'} = \frac{p(D(x), D(g(x))|x \in F)}{p(D(x), D(g(x))|x \notin F)}$$
$$= \frac{p(D(x)|x \in F)}{p(D(x)|x \notin F)} \cdot \frac{p(D(g(x))|D(x), x \in F)}{p(D(g(x))|D(x), x \notin F)} = k\frac{h}{f} \tag{11}$$

We consider two cases. The first case is a single (monolithic) classifier using Bayes risk to measure performance. From equation 1, an improvement in performance requires that

$$\Delta R = \eta(1 - h') + f' - [\eta(1 - h) + f] < 0, \tag{12}$$

resulting in

$$p(D(g(x))|D(x), x \in F) \cdot (\eta h - \frac{f}{k}) > \eta h - f. \tag{13}$$

Since $0 < p(D(g(x))|D(x), x \in F) \leq 1$, equation 13 requires that:

$$\begin{cases} p(D(g(x))|D(x), x \in F) > \frac{k\eta h - kf}{k\eta h - f} \\ k\eta h - f > 0 \end{cases} \tag{14}$$

This is the necessary and sufficient condition for improving performance in the Bayes risk metric. And for a classifier of moderate performance, $k\eta h - f > 0$ is usually true, because $k\eta$ is greater than 1, and h is greater than f.

The second case is a node in a cascade classifier. In this situation the cascade risk from equation 3 (without considering computational efficiency) is used to measure performance. The necessary and sufficient condition for a performance improvement from the joint classifier is:

$$\Delta C < 0 \Leftrightarrow \frac{\log(h')}{\log(f')} < \frac{\log(h)}{\log(f)} \tag{15}$$

which leads to:

$$p(D(g(x))|D(x), x \in F) > \exp(\frac{\log(h)\log(k)}{\log(h) - \log(f)}). \tag{16}$$

The above derivation assumes the same threshold is used for both $D^\theta(x)$ and $D^\theta(x) \cdot D^\theta(g(x))$. Although there is no direct comparison of performance between $D^\theta(x)$ and $D^{\theta'}(x) \cdot D^{\theta'}(g(x))$ without knowing more about the whole ROC curve and $D(x)$, in order for $D^{\theta'}(x) \cdot D^{\theta'}(g(x))$ to have smaller cost than any $D^\theta(x)$, equation 14 or equation 16 is only a necessary condition at θ'.