EFFICIENT FACE ORIENTATION DISCRIMINATION

Shumeet Baluja[1,2] *Mehran Sahami*[1,3] *Henry A. Rowley*[1]
[1]Google, Inc.
[2]Robotics Institute, Carnegie Mellon University
[3]Computer Science Department, Stanford University

ABSTRACT

This paper presents efficient methods to address the problem of discriminating between five facial orientations. We present the most efficient methods for this task to date, which can accurately discriminate between five facial orientations with approximately 92% accuracy using fewer than 30 pixel comparisons and greater than 99% accuracy using 150 pixel comparisons. We achieve these rates by using a boosting method to select from a large set of extremely simple features. Comparisons to other methods are given.

1. INTRODUCTION

The interest in face orientation discrimination arises from two areas. First, the rapid increase in the availability of inexpensive cameras makes it practical to create systems that automatically monitor a person while using a computer. By using motion, color, and size cues, it is possible to quickly find and segment a person's face when he/she is sitting in front of a computer monitor. By determining whether the person is looking directly at the computer, or is staring away from the computer, we can provide feedback to any user interface that could benefit from knowing whether a user is paying attention or is distracted (such as tutoring systems, computer games, or even car-mounted cameras that monitor drivers).

Second, to perform accurate *face detection* for use in video indexing or content-based image retrieval systems, one approach is to design detectors specific to each face orientation, as in [4][5][8][10]. Rather than applying all detectors to every location and scale, a face-orientation system can be applied to each candidate face location to "route" the candidate to the appropriate detector, thereby reducing the potential for false positives, and also reducing the computational cost of applying every detector. Note that efficiency in orientation detection is paramount in these systems since every 20x20 sub-image is first passed through an orientation determination system before being passed to a face detection system. In typical face detection systems, on the order of 10^6 sub-images (to account for scaling by steps of 1.2) must be examined in a single frame of 640x480 pixels [4].

1.1. The Dataset

The primary dataset that is used in this study is composed of 20x20 pixel intensity images of faces. In each image, the face is in one of five orientations: frontal, right half profile, left half profile, right full profile or left full profile, as shown in Figure 1. The right/left images are mirrors of each other.

Figure 1: Examples of five faces in each orientation. First row: frontal. Second row: right half profile. Third: left half profile. Fourth: Right profile. Fifth: Left profile.

There are a total of 6,000 images for each orientation (30,000 total images). 3000 randomly selected images are used for training, and the remaining 27,000 are used for testing. This is a fairly clean dataset derived from the FERET database [11] with controlled lighting and the faces tightly cropped in the image. The set includes images that were synthesized from originals to add variability through slight in-plane rotations and translations. For each image, the pixel intensities were scaled linearly to span between 0 and 255. This set was originally used to train a face detection system [4]; for that study, it was necessary to segment the face from the background and replace the background pixels with random noise, see [4] for details. This is important because if the background is not replaced with noise, the pose of the face in some of the images could be simply determined by detecting the background (which is fairly uniform in many of the original images). With noise in the background, this task becomes both more realistic and more difficult.

0-7803-8554-3/04/$20.00 ©2004 IEEE.

In the next section, we describe the algorithm and the features used in detail. In Section 3, the results on the face orientation discrimination problem are presented. We also provide comparisons to other approaches and show visual examples of the classifiers. Finally, in Section 4, we close the paper with conclusions and suggestions for future work.

2. ALGORITHMS AND FEATURES

The goal of this work is to make the process of orientation discrimination as efficient as possible. Therefore, any of the features that are used must be easy to compute. Our approach is motivated by the work of [9], in which they use an over-complete set of features based on rectangular subregions which are efficiently computed from the integral image for face detection.

For this study, we use the simplest possible features, relationships between pixels. For each pixel pair (i,j) we effectively compute four binary features: ($pixel_i = pixel_j$), ($pixel_i \neq pixel_j$), ($pixel_i \leq pixel_j$), ($pixel_i > pixel_j$). The images that we use for this study are 20x20; but we need not compare a pixel to itself. Therefore, this process generates (20x20)*[(20x20)-1]*4=638,400 binary features that can be computed for each of the training images. We can easily reduce this set of features by a factor of 2 by restricting ourselves to the upper-diagonal of the pair-wise feature matrix. Further, due to symmetry, the classifier will select either the feature F or ~F, so we can reduce the feature set by another factor of 2. The final number of features is 159,600. Nonetheless, this is still an extremely large number of features to examine – far larger than the number of pixels in each 20x20 sub-image. The goal, given this large set of features, is to minimize the number of features that need to be computed when given a new image, while still achieving high discrimination rates.

In many studies, the task of feature selection is used as a pre-processing step to creating a classifier; a summary of the results with these approaches will be given in the next section. The approach that we take here is to use the AdaBoost learning algorithm in one of its simplest forms [6][9]; AdaBoost can be used to combine the feature selection and classifier training steps. The key idea behind AdaBoost is that a strong classifier can be created by combining many weak classifiers (classifiers that may only classify the training samples correctly just over 50% of the time). At each step, after the weak classifier is selected and applied, the misclassified samples are re-weighted so that the next classifier puts more emphasis on solving the incorrectly labeled samples. The final strong classifier is a weighted combination of the weak classifiers.

There are two challenges to using this approach. First, we only want to use a small set of features – computing all 159,600 binary features to classify a new image would be computationally prohibitive. Second, the combination of these features (the learner) must be cheap to compute, since potentially many learners will be used. As in [9], we limit the weak classifiers that AdaBoost can use in creating a strong classifier to single features; however, here the features are restricted to represent the relationship between two pixels. The classifier can only choose to use the feature F or its complement, ~F. No transformation of the features can be used. Enforcing this limitation has two benefits; first, it is cheap to compute, and second, at each step in the boosting procedure only a single new feature needs to be computed.

Given example images (x_1,y_1) .. (x_n,y_n) where y_i = 0,1 for negative and positive examples.

Initialize weights $w_{1,i}$ = 1/2N, 1/2P for y_i = 0,1 respectively, where N & P are the number of negative and positive samples.

For t = 1,…,T (maximum # of features to use):

1. Normalize weights $w_{t,i}$ such that $\Sigma_i w_{t,i} = 1.0$
2. For each feature, F_j, see how well it (or ~F_j) predicts the classification. Measure the error with respect to the weights w_t: $error_t = \Sigma_i w_{t,i} | F_j - y_i |$ or $error_t = \Sigma_i w_{t,i} | \sim F_j - y_i |$
3. Choose the feature (F_j (or ~F_j)) with the lowest $error_t$.
4. Update the weights:

if example is classified incorrectly:

$$w_{t+1,i} = w_{t,i}$$

else

$$w_{t+1,i} = w_{t,i} B_t$$

where

$$B_t = \frac{error_t}{1 - error_t}$$

For binary classification, the output of the strong classifier is:

$$S(x): \begin{cases} 1: \sum_{t=1}^{T} \log(\frac{1}{B_t}) * F_t(x) \geq 0.5 \sum_{t=1}^{T} \log(\frac{1}{B_t}) \\ 0: \text{ otherwise} \end{cases}$$

where $F_t(x)$ is the value of feature F_t computed for instance x.

Figure 2: A boosting algorithm (taken from [9]). Note that the "classifier" used here is simply the feature (or ~feature) itself. The final classifier outputs a weighted combination of the classifiers (in this case just feature F or ~F).

An instance of the AdaBoost algorithm is shown in Figure 2; it is shown for a binary classification problem. To compare the outputs of multiple classifiers, instead of using a simple 0,1 output, the final classification was taken to be the maximum of the normalized outputs for each of the classifiers: $\frac{\sum_{t=1}^{T} \log(\frac{1}{B_t}) * F_t}{\sum_{t=1}^{T} \log(\frac{1}{B_t})}$

3. RESULTS

Using the AdaBoost algorithm described in the previous section, five separate strong classifiers were built. Each classifier was trained to discriminate images in a given class from images in every other class. AdaBoost was run

until 500 features were selected for each classifier (T=500). A test image is then assigned to the class corresponding to the strong classifier with maximal output. The results are shown in Figure 3. Note that with just 6 features per strong classifier, we achieve over 90% classification accuracy on the test set. By using 30 features per strong classifier, 99% accuracy is achieved[1].

We also consider a variant of the AdaBoost algorithm to speed training. With the original AdaBoost algorithm, at each iteration, approximately 159,600 features must be examined. However, in an image, many pixels may contain redundant information. It may be possible to only examine a small number of these features and achieve the same results. To test this hypothesis, in each round of AdaBoost, instead of looking at all the possible features, P% were randomly selected to be examined; the weak classifier could only be selected from this restricted set. In subsequent rounds, the set was again randomly selected. Several settings of P were tried: 1%, 5%, 10%, 50% and 100%. Setting P=100% corresponds to the original AdaBoost algorithm. Setting P=1% means that at each iteration, on average, only 1% of the total features are candidates for selection. As can be seen in Figure 3, the results are close for the settings of P tried. Note that P has a linear relation to the training time (P=1% is approximately 100 times faster than P=100%).

Because the features that are used are simple pixel comparisons, it is easy to visually examine the features that are used. A detailed look at the top 5 features that were selected for use by the frontal face pose classifier is shown in Figure 4. The first feature selected assumes that the space between the eyes will be brighter than directly below the nose. The feature works well for frontal faces as this is often the case. However, in all of the other orientations, it is often the case that the pixel location that should be bright falls on an eye, and is therefore relatively dark. Therefore, it is a good discriminatory feature.

3.1. Alternate Approaches

We compare the results obtained with AdaBoost with those obtained from a number of standard machine learning and feature reduction approaches. To build efficient classifiers, feature reduction is essential when employing classification schemes that will use all of the features given. A short summary of the alternate approaches compared is given here.

[1] Since there are 5 strong classifiers (one for each class) we conservatively assume that each classifier uses different pixels. Therefore, for achieving an accuracy of 90%, 30 total features are used, for an accuracy of 99%, 150 total features are used.

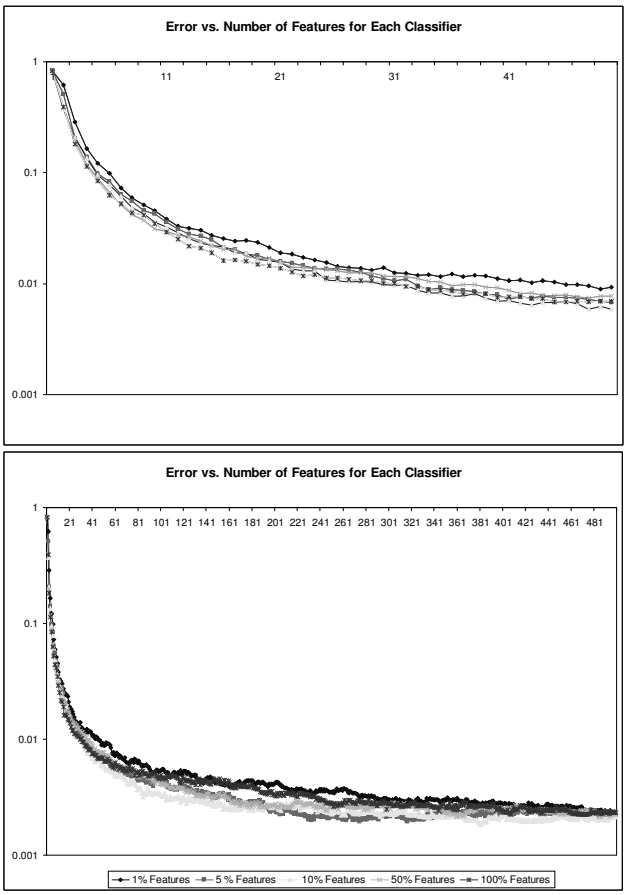

Figure 3: AdaBoost error rate vs. # of features used in each classifier. Note that five settings of P were examined. Top: First 50 features selected. Bottom: Full view of 500 features selected. This is the average of 5 trials per setting.

We perform feature selection by computing the mutual information [1] between each of the 159,600 original features and the class label *(C)*, selecting the top scoring features to build an efficient classifier. The mutual information score $MI(F_i; C)$ is given by:

$$MI(F_i;C) = \sum_{F_i}\sum_{C} P(C,F_i)\log\frac{P(C,F_i)}{P(C)P(F_i)}$$

In the reduced feature space, we employ the Naïve Bayesian classifier (NB) [2], limited-dependence Bayesian classifiers [7] (using both 1 (KDB-1) and 2 (KDB-2) dependence models) and Support Vector Machines [3] with both linear (SVM-l) and quadratic (SVM-q) kernels.

Naïve Bayes defines a linear separator yielding a predicted class C_{pred} as:

$$C_{pred} = \arg\max_C \sum_{F_i} \log P(F_i \mid C) + \log P(C)$$

Limited-dependence classifiers allow for a non-linear classifier where the predicted class C_{pred} is given by:

$$C_{pred} = \arg\max_C \sum_{F_i} \log P(F_i \mid C, \mathbf{F}_{ij}) + \log P(C)$$

where $\mathbf{F}_{ij}$ is a set of at most 1 or 2 features that have greatest class conditional mutual information with feature F_i. Such dependency modeling helps to overcome the limitations of Naïve Bayes.

Since SVMs are binary classifiers we use the same one-vs-rest classification approach we use with AdaBoost. Features are selected and a binary classifier built for each class separately. Each test instance is classified into the category for which it gets the maximal score over all binary classifiers; this approach parallels the one used for the boosting classifier.

We ran each of the classifiers described above on the same training/testing data split used with AdaBoost and the results are given in Table 1. For each number of features, AdaBoost outperforms all other classifiers. For AdaBoost (with P=100%), at 2500 features (500 features per class) the accuracy was 99.7%, at 150 features (30 features per class) the accuracy was 99.0%, and at 30 features (6 features per class) the accuracy was 92%.

# Feat	NB	KDB-1	KDB-2	Feat/class	SVM-l	SVM-q
2500	97.53	94.74	98.27	**500**	99.04	99.14
150	94.78	95.51	92.51	**30**	94.45	94.96
30	77.62	80.99	82.64	**6**	81.73	79.41

Table 1: Classification accuracy (classifier vs. # of features).

More features could potentially lead to slight gains in accuracy for the compared methods. However, the resulting classifiers would be much more expensive to apply. Since the primary motivation for our work is building a classifier that is efficient to apply in practice, using a classifier that utilized many thousands of features during classification would be cost prohibitive. Furthermore, just comparing the number of features is a bit deceptive; it should be noted that applying the best performing of the compared methods, SVMs, requires time that is linear in both the number of features as well as the number of support vectors (which we often found to be on the order of 100s)[2]. Consequently, such a method can be hundreds of times more expensive to apply than AdaBoost even when using the same number of features.

4. CONCLUSIONS

This paper makes several novel contributions, including demonstrating the power of even very simple pixel comparisons. The final system discriminates between 5 face orientations with accuracies over 99% with only 150 pixel comparisons. This approach yields the simplest and most efficient system to date for this task.

We are extending this work first by further exploring the tradeoffs between feature expansion, efficient classification, and classifier complexity. Second, we are extending this work to more complex datasets, such as those in gender determination, within the face domain as well as other domains.

5. REFERENCES

[1] T. Cover and J. Thomas (1991) *Elements of Information Theory*.

[2] I. Good (1965) *The Estimation of Probabilities: An Essay on Modern Bayesian Methods.*

[3] T. Joachims (1999) Making large-Scale SVM Learning Practical. *Advances in Kernel Methods - Support Vector Learning*, B. Schölkopf and C. Burges and A. Smola (ed).

[4] H. Rowley (1999) *Neural Network-Based Face Detection*, Ph.D. Thesis Carnegie Mellon University, CMU-CS-99-117.

[5] H. Rowley, S. Baluja and T. Kanade (1998) "Neural Network-Based Face Detection". *IEEE-PAMI* v20, pp 22-38

[6] R. Schapire, Y. Freund, P. Bartlett, W. Lee (1997), "Boosting the Margin: A New Explanation for the effectiveness of voting Methods", *ICML-1997*.

[7] M. Sahami (1996) "Learning Limited Dependence Classifiers", *KDD 1996*.

[8] K. Sung & T.Poggio (1998), "Example-based learning for view-based face Detection", *IEEE-PAMI*, v20, pp. 39-51.

[9] P. Viola and M. Jones (2001), "Robust Real-Time Object Detection", *2nd Int. Wkshp on Stat. & Comp. Theory of Vision*.

[10] M. Jones and P. Viola (2003), "Fast Multi-View Face Detection", Mitsubishi Electric Research Lab TR-20003-96.

[11] P. Phillips, H. Wechsler, J. Huang, P. Rauss (1998) "The FERET database and evaluation procedure for face-recognition algorithms", *Image and Vision Computing*, 16 (5):295-306.

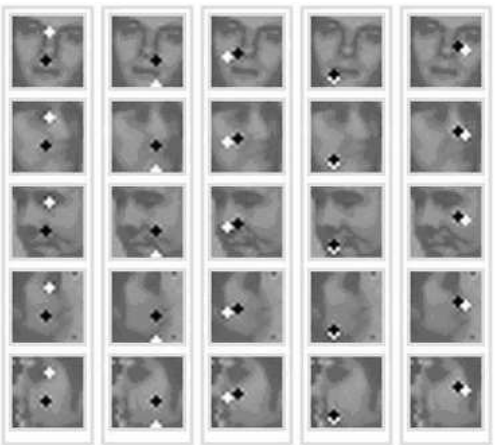

Figure 4: The first 5 features selected by the frontal face classifier. The pixel under the white cross has to be brighter than the pixel under the black cross. Each feature is shown where it would fall on a face in each orientation. Top Row: First five features overlaid on frontal face image. Second Row: First five features overlaid on right half profile, etc.

[2] For completeness, we also tried using SVMs with RBF Kernel Functions directly on the pixel images. This yielded accuracy comparable to the best Adaboost procedure. However, the SVM chose 350 support vectors per pose; which would yield a classifier slower by several orders of magnitude.