Boosting Sex Identification Performance

Shumeet Baluja[1, 2] shumeet@google.com
Henry Rowley[1] har@google.com

[1]Google, Inc.
[2]Carnegie Mellon University, Computer Science Department

Abstract

This paper presents a method based on AdaBoost to identify the sex of a person from a low resolution grayscale picture of their face. The method described here is implemented in a system that will process well over 10^9 images. The goal of this work is to create an efficient system that is both simple to implement and maintain; the methods described here are extremely fast and have straightforward implementations. We achieve 80% accuracy in sex identification with less than 10 pixel comparisons and 90% accuracy with less than 50 pixel comparisons. The best classifiers published to date use Support Vector Machines; we match their accuracies with as few as 500 comparison operations on a 20×20 pixel image. The AdaBoost based classifiers presented here achieve over 93% accuracy; these match or surpass the accuracies of the SVM-based classifiers, and yield performance that is 50 times faster.

Introduction

Perhaps the single most requested set of images from search engines are those that contain people. The queries for people range from specific individuals, such as celebrities, actors, musicians, and politicians, to general queries such as adult-content and stock-photography images. Considering the enormous number of images that are indexed in search engines today (commonly well above 10^9 images), it is impossible to manually label all of the content. Because of the strong interest in being able to retrieve images of people, we are attempting to create a variety of filters to better categorize and recognize the people that appear in images. One basic filter is to determine the sex of the person in the image.

Because of the large number of images that must be examined, speed is a key concern when deciding whether an approach can be used in practice. Recent work has shown that the pose of a face can be determined with high accuracy by simply comparing the intensities of a few pixels in grayscale images (Baluja, Sahami, and Rowley, 2004). These pose-classifiers are trained with AdaBoost. AdaBoost works by choosing and combining weak classifiers together to form a more accurate strong classifier. The weak classifiers used to distinguish pose were pixel comparison operations applied to pairs of pixels in a 20×20 image. Two comparison operators (and their inverses) were used: equal to and less than. Classification of faces into one of five pose classes was possible with 92% accuracy using just 30 pixel comparisons; 99% accuracy was possible using 150 comparisons. Because of the efficiency of the AdaBoost approach, we apply it to this task. Sample images for this domain are shown in Figure 1.

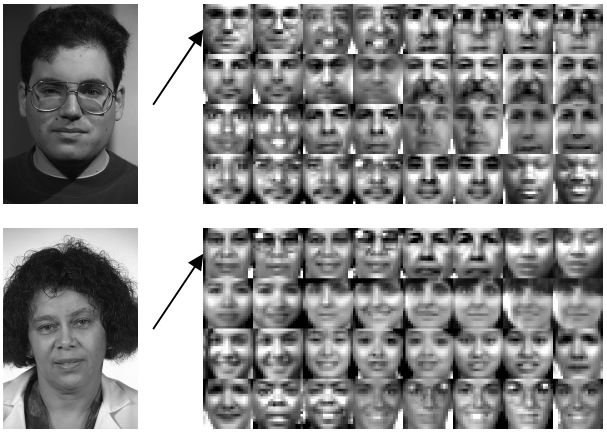

Figure 1: Samples male (top) and female (bottom) aligned 20×20 pixel face images which will be used in this paper, along with a representative sample at the original resolution.

In the next section, we describe a few recent pieces of related work. Section 3 describes AdaBoost and the features used in detail. Section 4 presents the data that is used for training and testing, and the various experiments conducted to explore the differences in performance obtained by altering the pre-processing steps. The experimental results are given in Section 5. Section 6 gives a detailed breakdown of the timing comparisons. Finally, we close this paper with conclusions and suggestions for future work in Section 7.

Previous Work

There have been several pieces of recent work on determining sex from facial images that have been tested on large data sets. Three approaches are described here.

(Shakhnarovich, Viola, and Moghaddam, 2002) applied AdaBoost to the features used by the face detection system created by (Viola and Jones, 2001) on 24×24 pixel images collected by crawling the web. They obtained an accuracy of 79%.

(Gutta, Wechsler, and Phillips, 1998) applied a hybrid system of RBFs and decision trees to FERET images at a resolution of 64×72 pixels, and achieved an accuracy of 96%. The training and testing sets were augmented with artificially generated images (by adding random noise and random rotations). In this paper, for efficiency, we concentrate on lower resolution images. We also do not augment the testing or training sets.

(Moghaddam and Yang, 2002) used SVMs on the FERET database of face images and achieved accuracies as high as 96.6%. These are the highest reported to date. This accuracy is much higher than in Shakhnarovich's work for two reasons: the FERET images are very clean (noise-free, fairly consistent lighting, no background clutter, etc), and because images of the same person may have appeared in both the training and test sets for the FERET experiments. This may have allowed the SVM to recognize individual faces rather than generalizing properly for this domain. In our experiments, we will control for this explicitly – this will be explored further in the experiments section.

Using Ada-Boost with Pixel-Comparisons

It is common in vision tasks to compute a variety of features that represent a large number of pixels. In contrast, we use only extremely simple features: the relationship between two pixels. Five types of pixel comparison operators (and their inverses) are used:[1]

1. $pixel_i > pixel_j$
2. $pixel_i$ intensity within 5 units (out of 255) of $pixel_j$
3. $pixel_i$ intensity within 10 units (out of 255) of $pixel_j$
4. $pixel_i$ intensity within 25 units (out of 255) of $pixel_j$
5. $pixel_i$ intensity within 50 units (out of 255) of $pixel_j$

[1] Note that adding other types of comparisons is easy. Also note that adding more comparison types only increases the training time and not the run time. Assuming that the comparison operations use roughly equal CPU time, only the *number of features* employed will impact the actual classification speed at runtime, not *which features are used* or the number of unique comparison types that are available.

Each comparison yields a binary feature. This feature is used as a *weak-classifier*. A weak classifier is only required to have accuracy slightly better than random chance. For this study, the result of the comparison is trivially considered the output of the classifier: an output corresponds to "male" if the comparison is true, "female" if it is false. Numerically these outputs are represented as 1 and 0 respectively.

There exist weak classifiers for each pair of different pixels in the image for each comparison operator. For 20×20 pixel images, this means there are 2*5*400*399 or 1,596,000 distinct weak classifiers. Even accounting for symmetries, this still yields an extremely large number of classifiers to consider. 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 identification rates.

We use AdaBoost to combine multiple weak classifiers together to form a single strong classifier with better accuracy. The AdaBoost training algorithm is an iterative procedure for picking a classifier to add at each step and also its associated weight. The final strong-classifier is a thresholded linear function of the selected weak-classifiers.

The main steps of the AdaBoost algorithm are shown in Figure 2. Essentially, it is a greedy learner that at each step selects the best weak classifier for the weighted errors of the previous step. The weight changes in Step 4 are such that the weak classifier picked in Step 3 would have an error of 0.5 on the newly weighted samples, so it will not be picked again at the next iteration. Once all the weak classifiers are selected, they are combined to form a strong classifier by a weighted sum, where the weights are related to the reweighting factors that were applied in Step 4 (normalized to sum to one).

One of the time consuming steps in this algorithm is computing the accuracy of all the weak classifiers in each iteration. Although the number of candidate classifiers effects only the training-time and not the run-time, there are a few easy methods to improve the training time. One approach is presented in (Wu, Rehg, and Mullin, 2003): the error rates for each weak classifier are kept fixed, rather than being reevaluated in each iteration. Another approach for reducing training times is to randomly select which weak classifiers will be evaluated at each iteration, and select the new classifier from only those that were evaluated. At each iteration, the set of classifiers to evaluate is randomly chosen again. In the experiments reported here, we explored this approach. In addition to running the experiments that evaluated all of the classifiers in every iteration, we also experimented with evaluating only 10% and 1% of the weak classifiers during each iteration.

As a baseline to compare against, we also applied SVMs to the pixel data; this approach parallels the one taken in (Moghaddam and Yang, 2002). The SVM implementation we used was SVM Light (Joachims, 1999); the parameters used will be discussed with the experiments.

Input: samples (x_1,y_1) .. (x_n,y_n) where x_i are the images and $y_i = 0$ for the female and 1 for male samples.

Initialize weights $w_{1,i} = 0.5/F$, $0.5/M$ for $y_i = 0,1$ respectively, where F and M are the number of female and male samples.

For $t = 1,\ldots,T$ (maximum number of weak classifiers to use):

1. Normalize weights $w_{t,i}$ such that $\Sigma_i w_{t,i} = 1.0$

2. For each weak classifier, C_j, see how well it predicts the classification. Measure the error with respect to the weights w_t: $error_t = \Sigma_i w_{t,i} \mid C_j(x_i) - y_i \mid$

3. Choose the weak classifier (denoted C_t) 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

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

The result of the strong classifier is:

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

Figure 2: A boosting algorithm - adapted from (Viola, Jones, 2001). Note that the weak classifiers used here are simply the comparison operators, and the final classifier output is based on a weighted sum of these weak classifiers.

Training and Test Data

The training data for this algorithm is taken from the Color FERET database (Phillips et al., 2000). This database contains images of 994 people (591 male, 403 female). We use only frontal images labeled "fa" and "fb" in the database that also have their eye coordinates labeled in the database, for a total of 2409 faces images (1495 male, 914 female).

Following previous approaches, we partition the sets of images 5 different ways. Each partition uses 80% of the data for training and 20% for testing, in such a way that each sample is used only once as a test image. For the "unmixed" data sets, we make sure that images of a particular *individual* appear only in the training set or test set for a partition of the data. For the "mixed" data sets, there is no such restriction, and the images are mixed randomly. One would expect that the unmixed case is a harder task than the mixed case, since for the mixed case the classifier has the opportunity to memorize or recognize individual faces rather than using more general features. For our tests, the unmixed data sets are the more important because of their applicability to the expected performance when analyzing billions of images with large numbers of faces unseen during training.

The images are taken with a variety of sizes of faces, lighting conditions, and positions. The following steps are used to normalize each image for input to the classifier:

1. Convert the image to grayscale by averaging the red, green and blue color components.
2. Compute a rigid transform which maps the labeled eye locations to (5,5) and (15,5) for a 20×20 window, and (1.5,8) and (10.5,8) for a 12×21 window as used in (Moghaddam and Yang, 2002).
3. Scale image by averaging blocks of pixels down to the smallest size larger than that specified by the rigid transform.
4. Sample each of the pixels for the target window using bilinear interpolation.
5. (Optional) Normalize the intensities of the image to have a mean of 127, standard deviation 64, clipped at 0 and 254.
6. (Optional) Mask the image by setting pixels corresponding to black pixels in the mask to 127 (see Figure 3). When a mask is used, the normalization in the previous step does not take the masked pixels into account when computing the mean and standard deviation.
7. For the SVM experiments, the range of input values was mapped to -1 to 1.

For the optional steps 5 and 6, we will report results separately for the experiments that include the steps and those that do not.

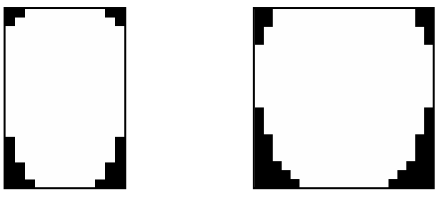

Figure 3: Masks used to normalize face images for (left) 12×21 pixel and (right) 20×20 pixel windows.

Experiments

In this section we summarize a large set of experiments with varying types of preprocessing on the data and various types of classifiers.

We first applied SVMs to this problem to provide a baseline, and for comparison with earlier work showing state of the art accuracies with SVMs. All numbers given are for SVMs with Radial Basis Function (RBF) kernels. In order to find the best setting for the SVM, we tried a variety of settings for the parameters. The settings given to SVM Light for gamma (related to the RBF radius) ranged from 0.001 to 0.100 (in steps of 0.001), and for C (the tradeoff between margin and training error) were the default and 100,000. Based on the preprocessing steps in the previous section, there are eight different ways to generate train and test images: with or without normalization, with or without masking, and at a size of 20×20 or 12×21 pixels. For each of these cases, tests of all parameters values were tried. Since each parameter setting required 5 SVMs to be trained (for the 5-fold cross validation), this resulted in a total of 5*8*200=8,000 SVMs being trained.[2] The preprocessing that gave the best accuracy was with normalization, no masking, and a size of 20×20 pixels; the accuracy for this case as a function of the parameters is shown in Figure 4.

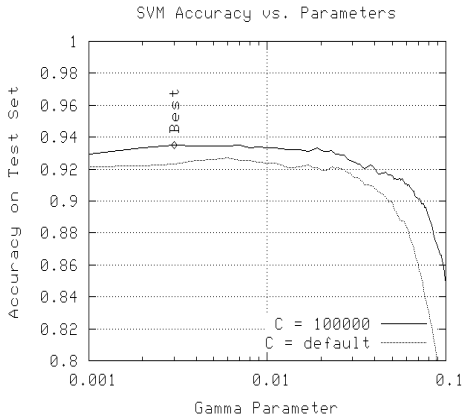

Figure 4: Accuracy of SVM while varying the C and gamma parameters.

For all the results reported below, setting C to 100,000 gave the highest test accuracy (with very little or no error on the training set). The number of support vectors varied quite significantly, ranging from 300 to 600, out of around 1,900 training samples.

[2] These experiments were done to find the best parameter settings for the SVMs, as well as the best preprocessing options. This baseline will be used for the remainder of the experiments to ensure that we compare AdaBoost with the best possible SVMs.

The AdaBoost classifier was trained three times, with a limit of 1000 weak classifiers. The training runs differed in the number of weak classifiers that were randomly selected for evaluation in each iteration. We examined the performance achieved from evaluating 1%, 10% and 100% of all possible weak classifiers per iteration. The results are given in Table 1. The first row is for the preprocessing that gave the best SVM result. The remaining rows each change one preprocessing parameter from that best case.

Table 1: Classification accuracy for a variety of test sets.

Data Processing Steps			Training Algorithm			
Normalized Intensities	Mask Used?	Window Size	Best SVM	AdaBoost (1000 Weak-Classifiers)		
				1%	10%	100%
Yes	No	20×20	93.5%	93.6%	94.4%*	94.0%
Yes	Yes	20×20	92.5%	91.5%	91.7%	91.7%
No	No	20×20	92.3%	94.2%	93.8%	94.3%
Yes	No	12×21	90.7%	91.5%	91.4%	91.0%

As can be seen, in all but the second case, the AdaBoost algorithm gives slightly better accuracy than the SVM result. Overall, the best accuracy for both types of classifiers seems to be the first set of experiments, with normalized but unmasked images of 20×20 pixels.

The differences in the performances are small – we are not interested in claiming that one method is better in accuracy than another. Rather, what is most interesting is the difference in the amount of computation required. An SVM measures the distance from the test sample to every support vector, which for a 20×20 pixel image and 300 support vectors leads to at least 400*300=120,000 pixel comparisons. The AdaBoost classifier with 1000 weak classifiers uses only 1000 pixel comparisons, yielding results that should be orders of magnitude faster.

Figure 4 shows how the accuracy of the classifier varies as the number of weak classifiers it contains is varied. As can be seen, we match the accuracy of the SVM classifier on the normalized, non-masked, 20×20 data at 500 weak classifiers.

The previous best reported results for this domain are in (Moghaddam and Yang, 2002), which use SVMs. For the experiments conducted in this paper, we carefully controlled the separation of individuals (not just images) between the test and train sets. In Moghaddam and Yang's work, unlike the above results, pictures of individual people may appear in both the training and test sets (people have multiple pictures in our image set), which makes the task both easier and much less applicable to the problem that we are interested in – being

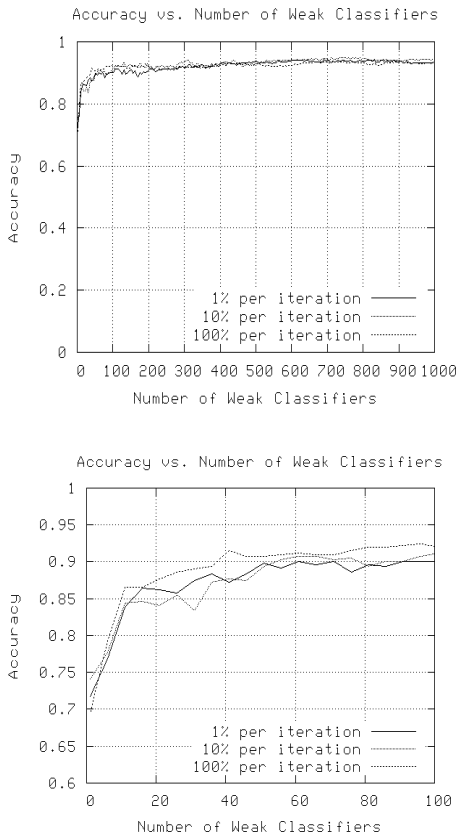

Figure 4: Accuracy as the number of weak classifiers in the AdaBoost classifier is varied. Top: Full graph. Bottom: First 100 weak classifiers enlarged.

able to recognize identify the sex of people for whom we have not trained our classifiers. For completeness, Table 2 shows the results of our algorithms on data where individuals appear in both the training and test sets. For the first test, we used the best preprocessing from Table 1; for the second test we used preprocessing matched as closely as possible to earlier reported work.

Table 2: Classification accuracy for test sets in which the people are mixed across training and test sets.

Data Processing Steps			Training Algorithm			
Normalized Intensities	Mask Used?	Window Size	Best SVM	AdaBoost (1000 Weak-Classifiers)		
				1%	10%	100%
Yes	No	20×20	97.1%	96.3%	96.6%	96.4%
Yes	Yes	12×21	96.9%	94.6%	94.4%	95.6%

As can be seen, when allowed to train on the same people who are in the testing set, the SVM gives better accuracy than the AdaBoost algorithm – perhaps because the SVM has a greater capacity to memorize individual faces and their correct classifications. (Moghaddam and Yang, 2002) gives an accuracy of 96.6% on their data, which is close to that reported in the second line of Table 2 for the SVM. In that work, the SVMs used about 20% of the training vectors, while in our experiments, 650 – 950 support vectors were used out of approximately 1,900 training vectors. The high number of support vectors also suggests the possibility that the SVM is overfitting.

Timing Results

In this section, we give the performance of the algorithms in terms of time required to classify an image for both of the window sizes examined. For each window size (20×20 & 12×21), we give the timing results for SVM classifiers which after training had approximately 300, 600 and 900 support vectors. For the classifiers trained with AdaBoost, since we explicitly control how many features are used, we show timing results using 10, 100, 500 & 1000 features.[3]

Table 3: Timing Results for 3 SVMs with 300, 600 & 900 support vectors, and for Ada-Boost with 10, 100, 500 & 1000 features.

Window Size	Classification Method		Time (μ sec)
20×20	SVM	339 support vectors	581
		654 support vectors	1107
		897 support vectors	1515
	AdaBoost	10 features	0.16
		50 features	0.87
		500 features	9.47
		1000 features	19.53
12×21	SVM	338 support vectors	392
		656 support vectors	769
		899 support vectors	1025
	AdaBoost	10 features	0.16
		50 features	0.81
		500 features	9.43
		1000 features	20.29

As can be seen in Table 3, the difference in timings between the fastest SVM (with approximately 330 support vectors) and the AdaBoost classifier which gives

[3] It should be noted that for the timing results we used SVM-Light (Joachims, 1999) in its original form, and we compared it to unoptimized AdaBoost code.

comparable performance (with 500 features) is significant; the AdaBoost classifiers are approximately only 1.6% (9.47/581) as expensive to run with 20×20 images and approximately 2.4% (9.43/392) as expensive to run with 12×21 images. In both cases, there is approximately a 50 times improvement in speed.

There is little change in speed with window size for the AdaBoost classifier; this is because the number of features that is examined is independent of window size. With SVMs, this is not the case, since the entire image is examined. Note that the most accurate SVM result obtained is not based on the number of support vectors alone; rather, the number of support vectors varies with the training parameters. The best performance was obtained in the 20×20 case with 339 support vectors, and in the 12×21 case with 656 support vectors.

Also shown in Table 3 is the speed of AdaBoost when the number of features examined is reduced; if our application only requires lower accuracies, significant speed gains can be obtained. Also given for reference is the speed with 1000 features.

Conclusions and Future Work

We have presented a method to distinguish between male and female faces that matches (and slightly exceeds) the performance obtained with SVMs. However, the classification is achieved with a fraction of the computational expense; the classifiers presented here are 1-2 orders of magnitude faster (approximately 50 times) than SVMs.

Due to space restrictions, we are unable to present the extensive studies measuring the robustness of the classifiers to translation, scale and rotation. To summarize: in our tests, we varied the angle of the face from ±45°, scaled the images by 0.2 to 5, and examined translations from ±3 pixels in X and Y. Despite the fact that AdaBoost was used to select only individual pixels to compare, in every case, the AdaBoost classifiers performed as well, or better than, the SVM classifiers. Of course, the larger the variation, the larger the performance degredation.

We achieve 80% accuracy in identification with less than 10 pixel comparisons and 90% accuracy with less than 50 pixel comparisons. Results which match those of SVMs are obtained with as few as 500 comparison operations on a 20×20 image. These results support earlier work which has found pixel comparisons are effective in determining the pose of a face.

There are at least three areas for immediate future exploration. The first is to evaluate this approach in the context of a complete face detection system. The images used here from the standard FERET database are fairly clean and well aligned; using this system to classify faces that are found with a face detection system (Rowley, Baluja & Kanade, 1998) will require the classifier to handle much more variability in the data. The second future direction is to explore the use of different features. For example, following numerous previous approaches, box-like features may prove to be useful. Finally, the third direction is to explore the use of new types of classifiers. In this study, we used very simple pixel comparisons that mapped directly to a classifier; however, more complex transforms of the pixels may provide benefits as well. It will be interesting to measure and understand the tradeoffs between weak-classifier complexity and number of classifiers in terms of accuracy, robustness and speed.

Acknowledgements

Portions of the research in this paper use the FERET database of facial images collected under the FERET program (Phillips et al., 2000).

References

Baluja, S, Sahami, M., Rowley, H., "Efficient Face Orientation Discrimination" *International Conference on Image Processing, 2004.*

Gutta, S., Wechsler H., and Phillips, P. J. "Gender and ethnic classification". *IEEE Int. Workshop on Automatic Face and Gesture Recognition*, pages 194-199, 1998.

Joachims, T. "Making Large-Scale SVM Learning Practical". *Advances in Kernel Methods – Support Vector Learning*, 1999.

Moghaddam, B. and Yang, M.H. "Learning Gender with Support Faces". *IEEE T.PAMI* Vol. 24, No. 5, May 2002.

Phillips, P.J., Moon, H., Rizvi, S.A., and Rauss, P. "The FERET Evaluation Methodology for Face Recognition Algorithms". *IEEE PAMI*, Vol. 22, p. 1090-1104, 10, 2000.

Rowley, H A., Baluja, S., and Kanade, T. "Neural Network-Based Face Detection". *T. PAMI* Vol. 20, No. 1, pages 23-38, January 1998.

Shakhnarovich, Gregory, Viola, Paul A., and Moghaddam, Baback. "A Unified Learning Framework for Real Time Face Detection and Classification". *Int. Conf. on Automatic Face and Gesture Recognition*, 2002.

Viola, Paul and Jones, Michael J. "Robust real-time object detection". *Proceedings of the IEEE Workshop on Statistical and Computational Theories of Vision, 2001*.

Wu, Jianxin, Rehg, James M., and Mullin, Matthew D. "Learning a Rare Event Detection Cascade by Direct Feature Selection". *NIPS 16*, 2003.