Robust Object Detection Using Boosted Learning[*]

Peter S. Carbonetto
Department of Computer Science
University of British Columbia
Vancouver, British Columbia
pcarbo@cs.ubc.ca

April 29, 2002

1 Introduction

This project is based on two face detection algorithms developed by Paul Viola and Michael Jones [6],[7]. Using a training set of positive and negative images (where the positive instances contain human faces), the algorithms use AdaBoost [3] to build a robust scale-invariant image classifier for face detection. My motivation for exploring the Viola and Jones face detection framework is to gain experience with boosting and to explore issues and obstacles concerning the application of machine learning to object detection.

In this first part of this project I discuss the motivation behind the proposed algorithm. In the second part I outline the implementation details of my algorithm and some considerations for making it efficient. In the third and final part I outline the results of my training and testing using the faces data set and look at future extensions to the project.

2 Considerations on a Framework for Object Detection

Before taking an in-depth look at the object detection model proposed by by Paul Viola and Michael Jones, let's first consider some preliminary requirements for constructing a general framework for object detection.

First of all, we cannot make any assumptions about scene structure; to do so would place an unrealistic constraint. People can detect faces far away in large crowds or in cluttered city scenes. To satisfy this condition we need a robust, simple and unbiased representation that can characterize objects under

[*]To appear in the first annual University of British Columbia Poster Workshop on Machine Learning

a variety of situations. Edge information, for example, will not do the job since it is highly sensitive to the composition of the image. From the faces shown in figure 1, it is clear that finding consistent patterns in the intensity or texture of the faces would be difficult. Instead, we can rely on the discerning the differences in the intensity between the faces and the background, and consistencies in the intensity gradients gradients in the faces themselves.

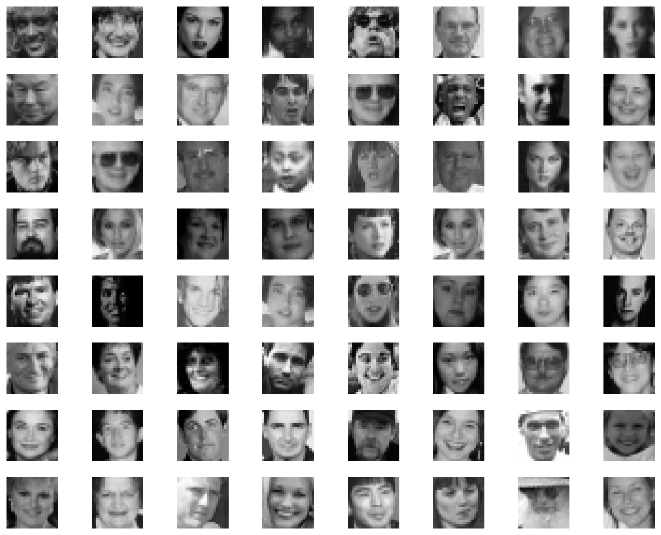

Figure 1: A random sample from the faces training set.

In their framework on object detection, Papageorgiou *et al* address the problem of detecting single static images in unconstrained and unstructured scenes [2]. They propose the Harr wavelet as a basis for image representation. Wavelets are commonly used in graphics to functionally decompose and compress images. Informally, we can define wavelets as piecewise-constant "wave" functions that form a basis for two-dimensional images [4]. Harr wavelets appear to be a respectable candidate for a general image representation since they are intensitive to mean changes in intensity and scale. See section 3.1 for a further discussion on wavelets.

Any biases in scene structure should result only from the learning algorithm. This seems natural to obtain biases since humans bias their judgements through learned information about the environment. On the other hand, we don't want to constrain our model by encorporating this *a priori* into the model. From this, we can also see that the selection of training examples is an important factor in the success of the algorithm. A well-formed data set of positive and negative examples should be sampled from commonly-encountered – but diverse – situations.

Given that we want a general object detection model, we should focus our efforts on poorly classified examples – in other words, images that are very close to together in feature space. This, in part, motivates the use of the AdaBoost learning algorithm.

We cannot have a practical model without a consideration of efficiency. Viola and Jones pay special attention to the run-time performance of the fast detector, which motivates their proposal for the attentional cascade [6],[7]. This is discussed in more detail in section 4.3.

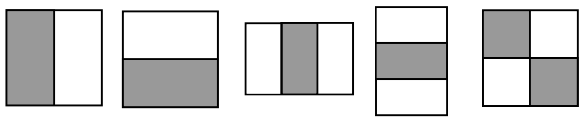

Figure 2: The set of Harr wavelets used as training features.

3 Implementation of Boosted Learning for Face Detection

3.1 Wavelets

Viola and Jones propose a set of five different Harr wavelets for the image basis, as shown in Figure 2. They do not give a justification for this selection but intuitively this overcomplete set of basis functions provides more structure than the standard Harr basis without compromising computational efficiency. A wavelet function is calculated by finding the difference of the pixels in the white area from the pixels in the gray area (see Figure 2). A single coefficient represents the height of the wave. We can think of a coefficient of low absolute value as indicating an area of almost uniform area, while a high coefficient represents a strong change in the intensity. As we will see later on, the task of the learning algorithm is to identify a set of wavelets (*i.e.* a template) that consistently distinguishes the faces in an image.

For efficiency, we compute the integral of the image. The integral of a single pixel is simply defined to be the sum of all the pixels above and to the left, including itself. Using the following pair of recurrences, the integral image can be computed in a single pass:

$$s(x,y) = s(x,y-1) + i(x,y) \tag{1}$$

$$ii(x,y) = ii(x-1,y) + s(x,y) \tag{2}$$

where $ii(x,y)$ is the integral of the pixel, $s(x,y)$ is the cumulative row sum, and $s(x,-1) = 0$ and $ii(-1,y) = 0$.

Once we have the integral of the image, computing a wavelet function can be done in constant time. The following formuli are used to compute the five features. For an explanation of these equations, see Figure 3.

two-rectangle feature	$a - 2b + c - d + 2e - f$
three-rectangle feature	$a - 2b + 2c - d - e + 2f - 2g + h$
four-rectangle feature	$a - 2b + c - 2d + 4e - 2f + g - 2h + i$

One drawback of using the overcomplete set of wavelets is that the number of possible features in a single image is very large. In a 24 x 24 image, there are 86,400 two-rectangle wavelets, 55,200 three-rectangle wavelets and 20,736 four-rectangle corner wavelets for a total of 162,336 features.

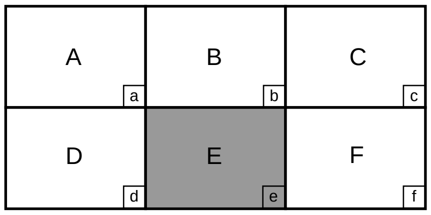

Figure 3: To compute the two-rectangle horizontal feature E - F, we note that the the sum of the pixels in box E can be described using the pixels of the integral image, $a + e - b - d$, and F is $b + f - c - e$.

3.2 The Learning Algorithm

I will first motivate and describe the overall learning process using AdaBoost. Section 6 elaborates on the weak classifiers.

The motivation for boosting is derived from the observation that finding several crude rules for classification can often be easier than finding a single, highly accurate rule. Abstractly, the final classifier produced by the boosting algorithm is a set of weighted features that accurately classifies the two sets of labeled images such that the features with the highest weights are relatively good at classifying the labeled examples.

The principle is to modify the distribution of training set at each round in order to place an emphasis on the more difficult examples. At each round, the boosting algorithm must make a selection from the pool of weak classifiers. Since the base learner's job is to minimize the error over the distribution of training examples, the best classifier is the one that results in the lowest error. Once AdaBoost makes its choice the weak classifier is assigned a measure of importance $\alpha_t = 0.5 * ln(\frac{1-e_t}{e_t})$ where e_t is the training error of the weak classifier at round t.

In essence, AdaBoost is a procedure for minimizing the error over the training set,

$$error = \sum_i \exp\left(-y_i \sum_t \alpha_t h_t(x_i)\right) \tag{3}$$

where x_i is a training example, y_i is the label on the example (either 0 or 1 in this case), α_t is the weight corresponding to classifier t, and $h_t(x_i)$ is the result of the weak classifier t applied to the training example. AdaBoost follows a steepest descent search to minimize the above equation, where the gradient is identified through the weights assigned to the weak classifiers. If we make our choice for α_t in accordance with the above equation (we can make this choice because we have a binary classification task), then the error on the training set is bounded by

$$error \leq \exp\left(-2\sum_t (0.5 - e_t)^2\right) \tag{4}$$

This means that training error drops exponentially with respect to the number of training rounds T. As long as the best base classifier is at least better than random, the upper bound on the error is $\exp(-2T(0.5 - e_t)^2)$.

One advantage with AdaBoost is that it has no parameters to tune other than the number of rounds, T. Another desirable property of AdaBoost is its ability to focus its efforts on the outliers – examples that are hard to categorize. However, a large number of outliers can impede the performance of the boosting algortihm. Fortunately, this is does not appear to be a problem for the face detector.

The performance of AdaBoost depends on the quality of the training set. Boosting is prone to overfitting and noise, and can fail if the training set is not sufficiently general. Again, the importance of supplying a strong training set is emphasized.

For a full description of the AdaBoost algorithm, see [3].

3.3 The Training Set

The positive examples for the training set were provided by Matt Flagg at Georgia Tech University, although I believe Michael Jones is the original author of this data set. The data set consists of a total of 4096 24 x 24 unnormalized images of faces.

I was unable to obtain the original set of 10,000 negative examples so I generated it myself. I hand-picked a list of 25 JPEG images from the Internet. The images were large and reasonably complex, consisting of scenes such as office spaces, crowds and city scapes. I designed a Matlab program to partition the large images into identically-sized 24 x 24 windows and then manually expunged any images that appeared to be faces. Since I did not make use of the full set of images, I randomized the order of the images in the data file. The final result was a set of 7872 unnormalized non-faces.

3.4 The Weak Learners

The Harr wavelet functions described above are the foundation for our weak classifiers. The basic idea behind the classifier h_j is to find a threshold θ_j which best separates the value of the feature f_j of the positively-labeled images from the negatively-labeled images. Thus, we can define the weak classifier as

$$h_j(x) = \begin{cases} 1, & \text{if } p_j f_j < p_j \theta_j \\ 0, & \text{otherwise} \end{cases}$$

where p_j is 1 if the positive examples are classified below the threshold or -1 is the positive examples are classified above the threshold.

In order to complete the classification task, we still have to find the optimal threshold θ_j [5]. If we define $C_0 = \{ x_i \mid y_i = 0 \}$ to be the set of negative examples and $C_1 = \{ x_i \mid y_i = 1 \}$ to be the set of positive examples then we can set the threshold using the following equation:

$$\frac{1}{2}\left(\frac{1}{|C_0|} \sum_{x \epsilon C_0} f_j(x) \; + \; \frac{1}{|C_1|} \sum_{x \epsilon C_1} f_j(x) \right) \tag{5}$$

The parity p_j is simply the sign of the difference of the two means in the above equation. This is a simple and not very satisfying way for selecting the threshold, but we can compensate for deficiencies in this selection process through the boosting algorithm. Nevertheless, it would be interesting to see if the overall scheme could be improved by introducing more effective weak learners. For example, one could maximise the margin between the weighted positive and negative examples instead.

3.5 Implementation Issues for the Learning Algorithm

To learn a reasonably general classifier, the AdaBoost algorithm has to have a data set of several thousand examples. Due to memory and time considerations, the largest training data set used consists of 500 faces and 1500 non-faces. Viola and Jones remarked that their initial training runs using a data set of approximately 14,000 images took several weeks to complete! The greatest retarding factor is the large number of memory references.

The good news is that the time complexity of the learning algorithm is linear with respect to the size of the training set, so adding more examples will not significantly increase the time for training. The computation of the unweighted Harr wavelet features in each round is redundant. To reduce the computational cost for each boosting round, I precompute the features beforehand, so the limiting factor is the amount of memory available in the system. Using a data set of 2000 images, storing the precomputed features takes approximately 1.3 GB of memory. In retrospect this may have been a bad decision – it is possible that recomputing the features is faster since the significant reduction in memory would produce much fewer cache misses, and cache misses can slow down the computation by an order of a thousand.

To avoid overfitting due to differences in lighting conditions, I normalized the variance of the training examples. To normalize the variance, each pixel in the image is divided by the standard deviation σ, calculated using the following equation:

$$\sigma^2 = \frac{1}{N}\sum_{i,j} x_{ij}^2 - \left(\frac{1}{N}\sum_{i,j} x_{ij}\right)^2 \tag{6}$$

where N is the number of pixels in the image and x_{ij} is the pixel at location (i, j). To avoid storing the normlalized images in memory using a floating-point representation, I simply stored the standard deviations in a single array and achieved the effect of normalization by dividing the feature values by the respective standard deviations.

The final strong classifier consists of an array of T elements, where each element has the following information: the type of feature, the height and width of the feature, the (x, y) location of the feature in the 24 x 24 sub-window, the threshold θ_t, the parity p_t, the importance weight α_t. To get an idea at the time it takes for training, learning a 30-feature classifier using a training set of 1500 examples on a convential top-end machine took 50 minutes.

3.6 Implementation Issues for the Face Detector

In this case, the speed of the face detector is the most important factor. Before proceeding with face detection using the strong classifier, the program first computes the integral image and the integral squared image. We need the latter to efficiently compute the standard deviation of the scaled sub-windows. (More precisely, we can find the standard deviation using a total of eight pixel references using this technique.)

In addition, the program calculates beforehand the coordinates of the pixel references for each feature classifier. In this way, during the actual detection phase the pixel references only need to be shifted along the height and width of the image. Storing all this information for each subwindow scale takes up a negligible amount of memory. In this implementation, the sub-window is increased by a factor of 1.25 at every scale. As in training, we post-multiply the features by the standard deviation instead of applying the variance normalization directly to the image pixels.

4 Results and Futher Considerations

4.1 Training Results

Figure 4 shows the error of the minimum weak classifiers increases nonmonotonically on subsequent rounds as the distribution of the training examples become more difficult to classify. As expected, the strong classifier error on the training set decreases as more weak classifiers are included.

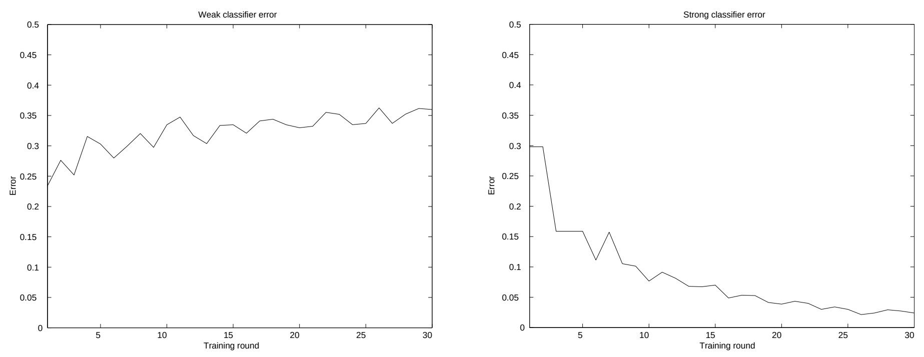

Figure 4: The graph on the left shows the error of the weak classifier h_t. On the right is the strong classifier produced after t rounds. In this instance, the final classifier was constructed on 30 rounds of training.

4.2 Test Results

At this point I have no final test results because the training is expected to take four days to complete. Early test results indicate that a reasonably accurate face classifier requires a very low final error on a large training set. Consider the following: if we scan a 256 x 256 image at a scale factor of 1.25, then there is a total of 284,143 potential matches. If we have a training error of 0.01, we could expect on the order of 2000 false positives. A portion of these errors would be elimitated because the face detector combines intersecting detections (this is based on the observation that the classifier used a training set in which the faces are not perfectly aligned so it is flexible on the order of plus/minus a few pixels in both dimensions), but clearly a very low error is needed on the order of 0.001.

4.3 The Attentional Cascade

Due to a lack of time, I was unable to implement the attentional cascade algorithm proposed in the same paper by Paul and Viola [6],[7]. However I would like to briefly discuss the motivation for the cascade and provide some arguments in favour of this model for object detection.

The attentional cascade is motivated by two observations stemming from the experiences in working on this project: 1.) the first few features in the strong classifier are much better at accepting or rejecting images than the rest of the features, and 2.) we can adjust the threshold of the weak classifiers to accomodate a better detection rate at the expense of an increased false positive rate. Since an overwhelming majority of sub-windows are negative, the goal of the attentional cascade is to speed up the scanning of images by rejecting as many negative instances as possible using the smallest number of features.

The cascade acts as a decision tree (or more precisely a decision stump), such that a given image must be accepted by the classifier at each and every level in order to be classified as a positive example. Ideally one would like to achieve high detection rates at the expense of an increased number of false positives in the upper levels of the cascade, for the purposes of quickly rejecting large portions of an image with a minimal number of computations. From inspection of the following equations, one can see that a multi-level attentional cascade tends toward a low false positive rate and a high false negative rate:

$$false\ positive\ rate = \prod_{i=1}^{K} f_i \qquad detection\ rate = \prod_{i=1}^{K} d_i$$

where f_i and d_i are the false positive rate and detection rate for the ith classifier, respectively.

Viola and Jones propose a scheme for trading off these two competing interests by adjusting the feature thresholds.

The attentional cascade is not grounded theoretically but it holds a lot of promise because it can be trained using standard techniques and it shows significant gains in efficiency.

4.4 Further Considerations

The proposed scheme provides a very robust face detection algorithm because the features can be scaled and translated and the learned classifiers are robust and insensitive to lighting conditions. The Harr wavelet functions in combination with the integral image technique incur a very small computational cost so this simple scheme can be used to rapidly scan large images. Boosting holds promise for additional applications in object detection (and perhaps computer vision in general) for two principal reasons: one, it is a very general procedure because it requires little a priori information, and secondly, there is a lot of room for introducing improvements.

One problem with the boosting algorithm is that it depends on a large pool of training examples to learn a robust classifier. This is due to the nature of the face detector problem: there is a wide variety of faces and even more non-faces. A data set of 10,000 images cannot possibly encompass all the different negative examples and, at the same time, increasing the training set even further is clearly not feasable. Perhaps active learning could be pursued to remedy this problem. Active learning, by weeding out easily-classified examples early on in training, would allow for a large data set with a significantly reduced penalty in training time.

To widen the scope of application, especially to the field of robotics, I believe the first move should be to develop an online version of the learning algorithm. An intelligent face detection framework is not feasible unless it is able to learn from supervised experiences after its intensive preliminary training.

References

[1] Michael Oren, C. Papageorgiou, P. Sinha, E. Osuna & T. Poggio. Pedestrian detection using wavelet templates. *Conference on Computer Vision and Pattern Recognition*, 1997.

[2] Constantine P. Papageorgiou, Michael Oren & Tomaso Poggio. *A General Framework for Object Detection. International Conference on Computer Vision*, Januaary 1998.

[3] Robert E. Schapire. The Boosting approach to machine learning: an overview. *MSRI Workshop on Nonlinear Estimation and Classification*, 2002.

[4] Eric Stollnitz, Tony DeRose & David Salesin. Wavelets for computer graphics: a primer part 1. *IEEE Computer Graphics and Applications*, Vol. 15, No. 3, May 1995, p. 76-84.

[5] Kinh H. Tieu. Boosting sparse representations for image retrieval. *Masters thesis*, 2000.

[6] Paul Viola & Michael Jones. Robust real-time object detection. *Second International Workshop on Statistical Learning and Computational Theories of Vision-Modeling, Learning, Computing and Sampling*, July 2001.

[7] Paul Viola & Michael Jones. Rapid object detection using a boosted cascade of simple features. *Conference on Computer Vision and Pattern Recognition*, 2001.