Face Recognition using Machine Learning

Arun Alvappillai
UCSD
aalvappi@ucsd.edu

Peter Neal Barrina
UCSD
pbarrina@ucsd.edu

Abstract

In this paper, we proposed a facial recognition system using machine learning, specifically support vector machines (SVM). The first step required is face detection which we accomplish using a widely used method called the Viola-Jones algorithm. The Viola-Jones algorithm is highly desirable due to its high detection rate and fast processing time. Once the face is detected, feature extraction on the face is performed using histogram of oriented gradients (HOG) which essentially stores the edges of the face as well as the directionality of those edges. HOG is an effective form of feature extraction due its high performance in normalizing local contrast. Lastly, training and classification of the facial databases is done using the a multi-class SVM where each unique face in the facial database is a class. We attempt to use this facial recognition system on two sets of databases, the AT&T face database and the YALE B face database and will analyze the results.

1. Introduction

Face recognition has become a popular topic of research recently due to increases in demand for security as well as the rapid development of mobile devices. There are many applications which face recognition can be applied to such as access control, identity verification, security systems, surveillance systems, and social media networks.

Access control includes offices, computers, phones, ATMs, etc. Most of these forms currently do not use face recognition as the standard form of granting entry, but with advancing technologies in computers along with more refined algorithms, facial recognition is gaining some traction in replacing passwords and fingerprint scanners. Ever since the events of 9/11 there has been a more concerned emphasis on developing security systems to ensure the safety of innocent citizens. Namely in places such as airports and border crossings where identification verification is necessary, face recognition systems potentially have the ability to mitigate the risk and ultimately prevent future attacks from occurring. As for surveillance systems, the same point can be made if there are criminals on the loose. Surveillance cameras with face recognition abilities can aide in efforts of finding these individuals. Alternatively, these same surveillance systems can also help identify the whereabouts of missing persons, although this is dependent on robust facial recognition algorithms as well as a fully developed database of faces. And lastly, facial recognition has surfaced in social media applications on platforms such as Facebook which suggest users to tag friends who have been identified in pictures. It is clear that there are many applications the uses for facial recognition systems. In general the steps to achieve this are the following: face detection, feature extraction, and lastly training a model.

2. Description of Project

2.1. Face Detection

Facial detection via the Viola-Jones algorithm is a common method used due to its high detection rate and fast processing speed. The algorithm can be summed up in four steps: feature selection, feature evaluation, feature learning to create a classifier, and cascading classifiers.

Simple features are used, inspired by Haar basis functions, which are essentially rectangular features in various configurations. A two-rectangle feature represents the difference between the sum of the pixels in two adjacent regions of identical shape and size. This idea can be extended to the three-rectangle and four-rectangle features. In order to quickly compute these rectangle features, an alternate representation of the input image is required, called an integral image. The integral image can be represented by the following equation:

$$ii(x,y) = \sum_{x' \leq x, y' \leq y} i(x', y') \tag{1}$$

where the integral image is $ii(x,y)$ and the original image is $i(x,y)$. Essentially the integral image at the locations x, y is the sum of all the pixels to the left and above, including the point itself. The integral image representation can be computed with only one iteration through the entire input image, and allows a sum of a rectangular feature to be com-

puted using only four points. In reference to Figure 1, 1 is

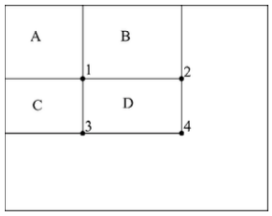

Figure 1. Four array references are required in order to determine the sum of rectangle D.

equal to the sum of the pixels in A, 2 is equal to the sum of pixels in A and B, 3 is is the sum of pixels in A and C, and lastly 4 is the sum of pixels in A, B, C and D. Knowing this, we can see that to compute the sum of pixels in D, it is simply (4+1) - (2+3). Instead of summing all of the individual pixels within the original image, we can simplify this computation by taking advantage of the integral image and thus reduce the time needed for the feature evaluation part of the algorithm.

The learning portion of the face detection algorithm uses Adaboost which basically uses a linear combination of weak classification functions to create a strong classifier. Each classification function is determined by the perceptron which produces the lowest error. However, this is characterized as a weak learner since the classification function does not classify the data well. In order to improve results, a strong classifier is created after multiple rounds of re-weighting a set weak classification functions. These weights of the weak classification functions are inversely proportional to their errors. The goal of this stage is to train the most relevant features of the face and to disregard redundant features. The last step of the Viola-Jones algorithm is a cascade of classifiers. The classifiers constructed in the previous step form a cascade. In this set up structure, the goal is to minimize the computation time and achieve high detection rate. Sub-windows of the input image will be determined a face or non-face with classifiers of increasing complexity. If a there is a positive result from the first classifier, it then gets evaluated by a second more complex classifier, and so on and so forth until the sub-window is rejected. By doing this, the structure utilizes the early stages of the cascade in order to reject as many negatives as possible. Figure 2 below shows a general diagram of the process.

As expected, there is an associated trade-off between the detection performance and the number of false positives. The perceptrons created from the AdaBoost can be tuned to address this trade-off by changing the threshold of the perceptrons. If the threshold is low, the classifier will have a high detection rate at the expense of more false positives. Conversely, if the threshold is high, the classifier will have a low detection rate however with fewer false positives. Ad-

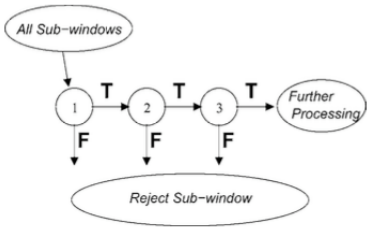

Figure 2. Schematic of the classifier cascade process flow.

justing this threshold may not keep the guarantees and training from AdaBoost therefore it is better to address the training process as a whole. This time a trade-off exists between performance in terms of high detection with low false positives and computation time. In general, higher detection rates and lower false positives are the result of classifiers which have more features. However, this also increases computation time. The detector is designed with specific constraints provided by the user which inputs the minimum acceptable detection rate and the maximum acceptable false positive rate. More features and layers are added if the detector does not meet the criteria provided.

2.2. Feature Extraction

Before we can identify faces, it is first necessary to specify what features of the face should be used to train a model. Once the Viola-Jones face detection runs, the face portion of the image is then used for feature extraction. It is important to select features which are unique to each face which are then used to store discriminant information in compact feature vectors. These feature vectors are the key part of the training portion of the facial recognition system and in our work we propose using HOG features. As mentioned previously, HOG features perform well because they store edges and edge direction. High quality local contrast normalization, course spatial binning and fine orientation binning are all vital to good HOG results. Extracting HOG features can be summarized with the following steps: calculate gradient of the image, calculate the histogram of gradients, normalize histograms, and finally form the HOG feature vector.

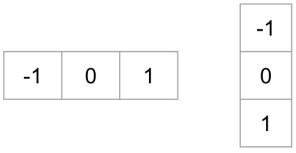

Figure 3. (Left) Vertical edge detector kernel. (Right) Horizontal edge detector kernel.

The first step requires computing the gradient of the input face image in both the x and y directions using 1x3 and

3x1 edge detector kernels shown in Figure 3. The horizontal kernel is applied to the input image to produce a horizontal gradient image which the vertical kernel is applied to producing a gradient image. The order in which the kernels are applied does not matter and the vertical kernel could be applied first to obtain the same result. We can see an example of the gradient image in Figure 4. As we can observe in the gradient image, the edges of the face are maintained and can be used for further processing.

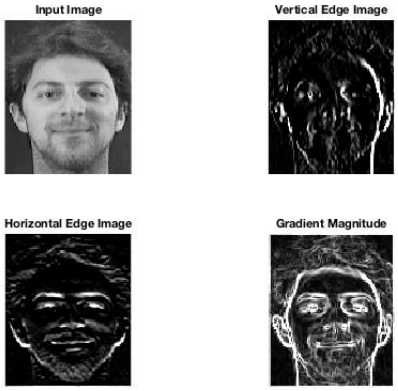

Figure 4. (Top left) Input image. (Top right) Vertical gradient image. (Bottom left) Horizontal gradient image. (Bottom right) Gradient image.

The next step in the processing is to calculate the histogram of the gradient image. The gradient image is broken up into a grid of cells where each cell is typically of size 8x8 pixels as shown in Figure 5. For each cell, a histogram of gradients is computed. To compute the histogram gradient two pieces of information are needed, the magnitude and direction of the gradient at each pixel. For a single cell, the histogram has nine channels with each channel corresponding to a range of directions from 0 to 180 degrees. Since we are using a range of directions from 0 to 180 and not 0 to 360, these are considered unsigned gradients. Each pixel in the corresponding cell selects a channel based on the direction of the gradient and votes for that channel based on the magnitude of the gradient. This process is repeated for each pixel in the cell until the histogram of gradients feature vector is completed.

Now that we have a HOG feature vector for each cell, the next part of the method is normalization. The purpose of normalizing the vectors is to ensure that the feature is invariant to any changes in light and contrast. Before normalizing the feature vectors, we first form blocks of cells in groups of 2x2. So essentially, each block is of size 16x16 pixels which results in four 9x1 HOG feature vectors. These four HOG feature vectors can then be concatenated to form one 36x1 feature vector for each block. Normalization is

Figure 5. (Left) Input image of face divided into cells. (Right) Gradient image of face divided into cells.

then performed on this 36x1 HOG feature vector using the following equation:

$$f = \frac{v}{\sqrt{\|v\|^2 + e}} \tag{2}$$

where v is the 36x1 HOG feature vector, $\|v\|$ is the norm of v, and e is a small constant. This normalization process is repeated for each block, and blocks are moved throughout the image overlapped such that every cell contributes to the HOG feature vectors more than once. In general, at least half a block size of overlap is desirable.

The last step in the feature extraction is to form the HOG feature vector. This is accomplished by concatenating the normalized 36x1 feature vectors in the previous step. So if you have a total of 100 unique block positions which each produce a 36x1 feature vector, the final concatenated HOG feature vector would be length (36)(100) = 3600. Figure 6 below shows an example of a face and its corresponding HOG feature vector.

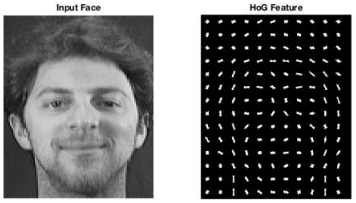

Figure 6. (Left) Input image. (Right) HOG feature of input image.

2.3. Training Model

Once the method of feature extraction for a face has been established, the next step is to train a model using

the extracted feature vectors of already identified faces in a database. The method of training employed attempts to utilize multiple instances of each distinct face in the gallery, such that the resulting model will be able to best match an input face to an identity from the gallery.

Support Vector Machines (SVM) are a popular training tool which can be used to generate a model based on several classes of data, and then distinguish between them. For the basic two-class classification problem, the goal of an SVM is to separate the two classes by a function induced from available examples. In the case of facial recognition, a class represents a unique face, and the SVM attempts to find what best separates the multiple feature vectors of one unique face from those of another unique face. If we represent all the feature vectors of both classes as data points with the same dimensionality: $(\mathbf{x}_1, y_1), ..., (\mathbf{x}_l, y_l)$ where $\mathbf{x}_i \in R^n, y_i \in -1, 1$, and l is the total number of data points, the SVM finds the hyperplane $\mathbf{w}x + b = 0$ that separates the largest possible fraction of points of the same class on the same side, while maximizing the distance from either class to the hyperplane. In other words, the set of vectors is optimally separated by the hyperplane if they are separated without error and the margin (distance between the hyperplane and the nearest data point of each class) is maximal. This is demonstrated in Figure 7.

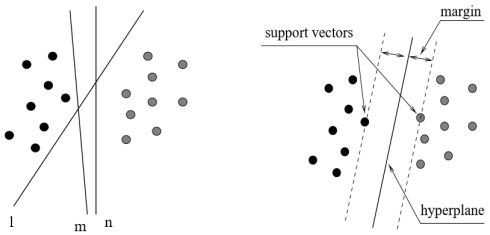

Figure 7. (Left) Arbitrary hyperplanes. (Right) Optimal Separating Hyperplane.

In order to obtain the optimal separating hyperplane (OSH), we first specify the constraints of the hyperplane:

$$y_i[(\mathbf{w} \cdot \mathbf{x}_i) + b] \geq 1, i = 1, ...l \tag{3}$$

$$min_{\mathbf{x}_i} y_i(\mathbf{w} \cdot \mathbf{x}_i + b) = 1 \tag{4}$$

The distance of a point $\mathbf{x}$ from the hyperplane is:

$$d(\mathbf{w}, b; \mathbf{x}) = \frac{|\mathbf{w} \cdot \mathbf{x} + b|}{||\mathbf{w}||} \tag{5}$$

The margin can be obtained from the above expression and constraints as $\frac{2}{||\mathbf{w}||}$. Therefore, the hyperplane that optimally separates the data is the one that minimizes:

$$\Phi(\mathbf{w}) = \frac{1}{2}||\mathbf{w}||^2 \tag{6}$$

When this expression is optimized using the Lagrange, the resulting OSH for linearly separable training data is given by:

$$\bar{\mathbf{w}} = \sum_{i=1}^{l} \bar{\alpha}_i y_i \mathbf{x}_i \tag{7}$$

$$\bar{b} = -\frac{1}{2}\bar{\mathbf{w}} \cdot [\mathbf{x}_r + \mathbf{x}_s] \tag{8}$$

$\mathbf{x}_r$ and $\mathbf{x}_s$ are support vectors which satisfy $\bar{\alpha}_r, \bar{\alpha}_s > 0, y_r = 1, y_s = -1$. For a new data point $\mathbf{x}$, the classification of that point is $f(\mathbf{x}) = sign(\bar{\mathbf{w}} \cdot \mathbf{x} + \bar{b})$. When the training data is not linearly separable, slack variables are introduced into the optimization.

Now that we have defined the process for two-class recognition using SVM, this basic building block can be extended to implement multi-class face recognition. Assuming there are multiple classes in the data set, a bottom-up binary tree is designed which applies a one-against-one strategy to classify between different pairings of classes. For the example shown in Figure 8, there are 8 classes which must be evaluated given an input test face. Each branch of the tree represents a two-class SVM from which results one "winner" that continues up the tree. Eventually, the unique class the model decides the input image best corresponds to will appear on top of the tree.

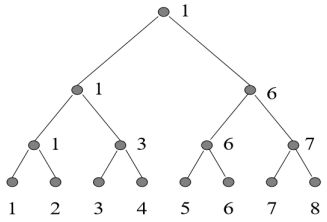

Figure 8. Binary tree structure for 8 classes

From the above derivations, it follows intuitively that the spread, quality, and amount of training vectors used for each SVM class pairing will have an impact on how well the resulting model identifies input faces. If the input face of a known person in the gallery varies from the examples in the gallery used to train the model (i.e. angle, expression), it is expected that the model will have more difficulty correctly recognizing the face. The extent to which this occurs is explored in our experiments with the ATT and Yale face databases.

3. Experiment

3.1. AT&T Dataset

The first dataset we used to train and test the facial recognition algorithm was the AT&T dataset of faces. The AT&T

set is composed of forty unique faces each with ten different images with. The composition of the images is mostly frontal view, consistent lighting conditions, and some varying expressions. When training the model using SVM, eight of the ten images for each person were used to create a class for each face while the remaining two images were reserved to test the model. Of the eighty images that were tested 73/80 faces were successfully recognized resulting in an accuracy of 91.25% which is desirable. When analyzing the cases which failed, we noticed that if the face was not facing forward and had a slight rotation to either the left or right then it was more susceptible to incorrectly identifying the face as shown in Figure 9 and Figure 10.

Figure 9. Incorrect facial recognition result.

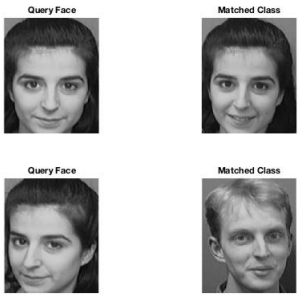

Figure 10. Incorrect facial recognition result.

In Figure 9, when the query faces in the left column are inputted into the model, they are incorrectly matched to the wrong person from the gallery. For both input images, the subject is not looking straight forward and directs both their gaze and face to the left. This observation is demonstrated more clearly in Figure 10, where the first input face is correctly matched because the subject's eyes and face are pointed forward. However, when the second input face is the same subject but they instead look into the camera sideways, the model fails to recognize the face correctly once again. The reason for this has to do with implementing a global approach or a component-approach to face recognition which will be explained later.

3.2. Yale B Dataset

The second dataset we explored testing was the Yale B Dataset. The subset we tested on included ten unique faces each with twenty images. The images are also frontal view with no changes in expression but have changes in lighting intensity. When training the model using SVM, sixteen of the twenty images were used to create a class for each person while the remaining four images were used for testing. Of the forty images which were tested, 39/40 faces were successfully recognized which results in an accuracy of 97.5%, an improvement from the previous dataset. This improvement in accuracy is due to two factors. The first reason why the accuracy increased is because the images of each face were consistent in position and facial expressions unlike the AT&T dataset which has varying expressions. Secondly, each unique face has more images used in training thus the model is more robust and holds more discriminant information of each face in the gallery.

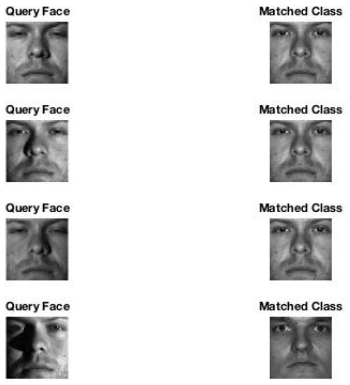

Figure 11. Incorrect facial recognition result.

In Figure 11, for the case that fails to recognize the face, we notice that the change in lighting plays a major factor. For the test image located at the bottom of the left column, the lighting environment causes the left half of the subject's face to be hidden in the shadows which drastically affects the feature vector which is extracted. Hence when trying to match the face to a face in the gallery the face recognition system struggles to identify the right face.

The reason for these errors from both datasets is tied closely towards the method of training we apply to generate the model and how we utilize SVMs. When developing facial recognition techniques, there is a global approach and a component-based approach. The global approach, which is used in this experiment, uses a single fea-

ture vector to represent the whole face image, and then use these vectors to train the model. Although global techniques work well for classifying frontal views of faces, they are not robust against pose changes since global features are highly sensitive to translation and rotation. The other approach meanwhile, which is component-based, attempts to deal with pose changes by allowing a flexible geometrical relation between the components in the classification stage. For example, by independently matching templates of the eyes, nose, and mouth instead of the entire face as a whole, the configuration of the components during classification is unconstrained and does not enforce a specific geometrical model of the face. In contrast to the global approach where a face detector extracts the face and propagates it to a set of SVM classifiers, in the component approach, the face detector instead extracts local components of the face. Then, these local components are fed independently into a set of SVM classifiers.

Despite the advantages in robustness that come with using local features, they also may require specialized classification algorithms to handle cases where the number of feature vectors returned by each image is not constant. Thus, if it is already known or made a requirement that all input faces must be facing forward, the global-approach is a simpler implementation of facial recognition and proves to produce sufficient results.

4. Conclusion

In this project, we implemented a facial recognition system using a global-approach to feature extraction based on Histogram-Oriented Gradient. We then extracted the feature vectors for various faces from the AT&T and Yale databases and used them to train a binary-tree structure SVM learning model. Running the model on both databases resulted in over 90% accuracy in matching the input face to the correct person from the gallery. We also noted one of the shortcomings of using a global approach to feature extraction, which is that a model trained using a feature vector of the entire face instead of its geometrical components makes it less robust to angle and orientation changes. However, when the variation in facial orientation is not large, the global-approach is still very accurate and simpler to implement than component-based approaches.

References

[1] Bernd Heisele, Purdy Ho, Jane Wu, Tomaso Poggio *Face recognition: component-based versus global approach*. Computer Vision and Image Understanding. February 2003.

[2] Guodong Guo, Stan Z. Li, Kap Luk Chan *Support vector machines for face recognition*. Image and Vision Computing. January 2001.

[3] Harihara Santosh Dadi, Gopala Krishna Mohan Pillutla *Improved Face Recognition Rate Using HOG Features and SVM Classifier*. IOSR Journal of Electronics and Communications Engineering, Vol. 11, Issue 4, pp 34-44, July 2016.

[4] Navneet Dalal and Bill Triggs *Histograms of Oriented Gradients for Human Detection*. Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. 2005.

[5] O. Deniz, G. Bueno, J. Salido, F. De La Torre *Face recognition using Histogram of Oriented Gradients*. Pattern Recognition Letters. 2011.

[6] P. Jonathon Phillips *Support Vector Machines Applied to Face Recognition*. National Institute of Standards and Technology,

[7] Qiang Zhu, Shai Avidan, Mei-Chen Yeh, Kwang-Ting Cheng *Fast Human Detection Using a Cascade of Histograms of Oriented Gradients*. Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Patter Recognition. 2006.