CSC523 - Machine Learning

Assignment

Face Recognition Challenge

Dipkumar Patel
Dept. of Information and Communication Technology
School of Engineering and Applied Sciences, Ahmedabad University
Ahmedabad
dipkumar.p.btechi15@ahduni.edu.in

Heet Gorakhiya
Dept. of Information and Communication Technology
School of Engineering and Applied Sciences, Ahmedabad University
Ahmedabad
heet.g.btechi15@ahduni.edu.in

Kuldeep Jitiya
Dept. of Information and Communication Technology
School of Engineering and Applied Sciences, Ahmedabad University
Ahmedabad
kuldeep.j.btechi15@ahduni.edu.in

Grishma Shah
Dept. of Information and Communication Technology
School of Engineering and Applied Sciences, Ahmedabad University
Ahmedabad
grishma.s.btechi15@ahduni.edu.in

***Abstract*—This is a report on our attempt on solving the very well known face recognition problem in the Machine Learning paradigm. Our approach is based on the Support Vector Machine (SVM) Algorithm which is a classifying algorithm for ML framework. In this document we discuss our approach and its outcomes as well as our journey through the assignment, i.e., what different methods we used before we decided our final approach as well as the challenges we faced implementing them. In this assignment, we take a data set of 588 images, each with one of the 43 people's faces and on that we apply our SVM Machine Learning algorithm to classify and identify each person uniquely. To test the results and accuracy of face detection, we give a different picture of a person's face as an input and find out how many times it identifies the person correctly.**

Keywords—Support Vector Machine, classification problem, Face Recognition, Machine Learning.

I. Introduction

Face Recognition is one of the most relevant applications in image analysis. It's a true challenge to build an automated system which equals human ability to recognize faces. The fundamental difference in human and computer based face recognition is that humans can identify a small number of faces to 100% accuracy, but don't do so well with a large number of faces, say, a thousand or so. Computers, on the other hand, have huge computing abilities. That, coupled with a large amount of memory, helps a well-trained computing system to recognize a large number of faces with lesser accuracy than human brain. Due to the recent advancements in the various fields like pattern recognition, bio-metric authentication and computer vision, this problem has recently gained a lot of popularity and is now studied as a principal component of the Machine Learning paradigm.

As this document focuses on the face recognition from pixel based images, we first need to identify the problems that are likely to be encountered while solving the problem. One of the problem is the problem of illumination. The images which have low illumination will have more greyscale pixel values while the ones with high illumination will low greyscale values. This will affect the accuracy of face recognition. The other problem is the pose problem; i.e., the orientation of the face in the image. Different images of the same face with different poses require higher intelligence level to identify as one and the same. The third problem is the background. This problem occurs when the face occupies less part of the image than the background, because then the different background's RGB pixel values will have a great impact on the image identification.

II. Existing approaches in Face Recognition - the current state of art

There are three main currently available approaches for Face Detection. They are:

A. Principal Component Analysis (PCA)

Principle Component Analysis is by far the most commonly used methodology for face recognition. PCA method focuses on feature extraction as its principal approach and utilizes different classifier algorithms for discriminating between features and based on that, it trains the classifier such that face recognition can be done to a decent accuracy.
The feature extraction method is common to all the variants of PCA. For this, the individual image is taken as an image vector and then dimensions of the image vector are reduced so that computations can be easier.

There are two variants of PCA which uses different classifiers: PCA using LDA and PCA Eigenface Implementation.

Eigenface Implementation: This approach projects the input image vector on the Eigenspace and finds similarity by the "closeness" of the vector with available Eigenfaces[1].

LDA Implementation: In the LDA, the main focus is on finding the best separation of means of a higher dimensional data by projecting them onto lower dimensional axes. This way, dimensions are reduced one by one by finding the best possible projected separation of means [2].

B. Support Vector Machine (SVM)

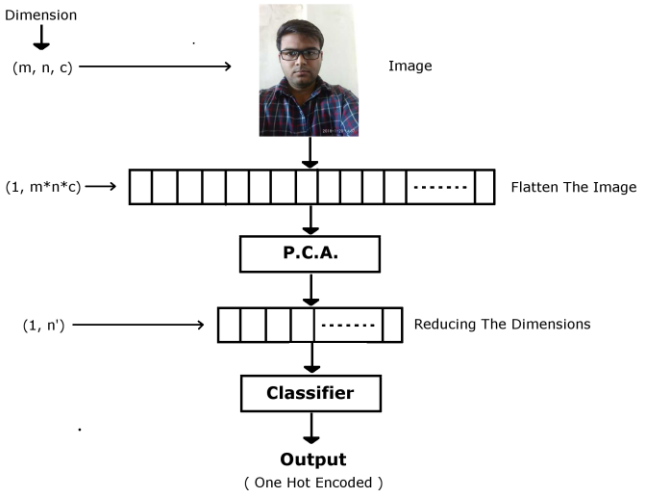
Support Vector Machine is a classification algorithm based on a supervised learning structure, which means the input training data needs to be labelled for it to work. The SVM learns a hyperplane based model, which can also be said to be a linear classification model in two dimensions[2]. There are four types of SVMs, namely – The Maximal Margin Classifier, Kernelized MMC, Soft-margin MMC and the Kernelised Soft-margin MMC. The most commonly and widely used is the Kernelized Soft-margin Maximal Margin Classifier, which is also used in the keras library for SVM implementation.

C. Deep Learning based Implementations

Input Image has too much extra information which are not useful always. Machine learning approaches deal with hand crafted features extraction. So, success of Machine learning approaches is based on the accuracy of hand-crafted feature extraction. Many common feature extraction includes HOG (histogram of oriented gradients), Fisher Vector Faces descriptor, scale invariant feature transform (SIFT) etc. These features are given as input to Neural Networks and then trying to recognize faces. Due to the rise of powerful CNN (Convolutional Neural Network), many people have tried different CNN for Face recognition. VGG Face[3] paper has trained VGG network on different dataset able to achieve more than 97% accuracy in YouTube face dataset. Face recognition with limited data is one-shot learning problem. For this problem, Facenet[4] introduces triplet loss and two CNN based learning where it tries to minimize loss between to similar images and maximize loss between different images.

III. OUR JOURNEY OF SOLVING THE PROBLEM - VARIOUS METHODS USED AND THEIR DRAWBACKS

We incorporated different methods to solve the Face Recognition Challenge. The general framework of the approach is shown in the following block diagram:



As shown in the block diagram, the image is first converted into image vector. Then, we implement PCA and reduce the dimensions. Afterwards, the classifier gives the output, which is the final detection.

The following is the full account of the assignment progress and the decisions made based on the drawbacks and performance metrics of each method used, each of which is based on the above general framework:

A. Approach 1 – PCA and Neural Network Framework

1) The first approach was decided as PCA and Neural Network training, i.e. the Classifier is a simple Feed Forward Neural Network. In this approach, after training the NN, the Train and Test gave 88% prediction accuracy (the percentage score given is based on the metrics.categorical_accuracy function used in the keras library for NN in python), while this approach was very unreliable for Valid image detection. It gave very situational output predictions and its accuracy was 10% based on the 10 Valid images we tested it on. It was observed that the output remained same for the same Valid image, hence the reason for this behaviour was found to be the large contribution of background.

2) The version 2.0 of the framework edits the dataset images by cropping them to remove the contribution of background so that only the pixels of faces are analysed. This was done by implementing Haar Cascading of the OpenCV framework. But this resulted in over-fitting of the dataset, hence the prediction accuracy still remained the same as before. It did not work on Valid Images. Here, the cropped face images are converted to 100*100 pixels.

3) The version 3.0 of the framework adds the next improvement, which was adding "dropout" function to the NN. Dropout is a regularization technique for neural network models proposed by Srivastava, et al. in their 2014 paper "Dropout: A Simple Way to Prevent Neural Networks from Overfitting"[3]. This function normalises the back-propagated decision values in the hidden layer of NN so that overfitting can be reduced. But the major drawback of this variant was that we had to train the NN more times than the previous approach, to validate the functionality of NN so the dropout function was rendered useless, because more number of training iterations re-introduced overfitting. The improvement in the accuracy is very less, i.e. the new accuracy is still only 20%

B. Approach 2 – PCA and SVM Framework

After the failure of different approaches which used FFNN, we changed the classifier from FFNN to SVM classifier. We implemented the SVM classifier function in the sklearn library (python). The output of the PCA dimensionally reduced vectors were fed to the SVM classifier function and the outputs were tested. This classifier gave 88% accuracy for Train and Test and upto 85% accuracy for Valid. Hence, this was selected as the final approach.

C. Other Possible Miscellaneous Approaches

We tried VGG face net which was trained on celebface dataset and then we applied transfer learning using pre-trained weight of VGG face. Training VGG face was computationally expensive but given selected images we have accuracy over 90% within 25 minutes of training[3].

Other approach is the implementation of RandomForestClassifier function from the sklearn library in python. It gives 50% accuracy when used as a classifier in our general framework.

Lastly, we checked if using just a FFNN was enough to get any accuracy, but since the FFNN's output was very highly dependent on independent values and the FFNN was not learning at all, the accuracy is very situational and can be considered to be 0% accuracy.

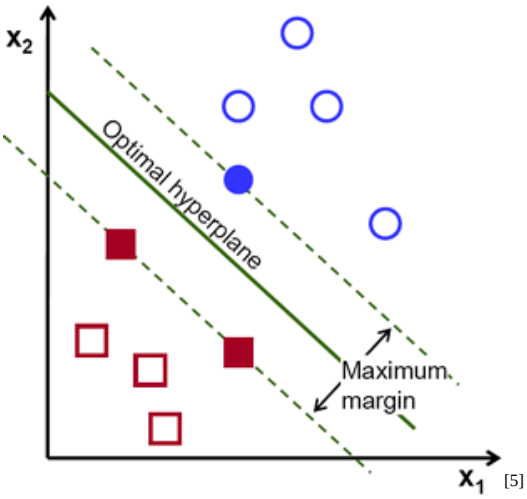
IV. Final Approach and Results

The decided approach is using a combined PCA dimensional reduction with SVM as a classifier.

The SVM Model/Algorithm:

A. Intuitive Understanding

This model defines a hyper-plane which classifies the outputs into two different regions. The main focus here is to find the hyper-plane in such a way that it has the most distance from both regions.



[5]

B. Mathematical Framework

1) The hyperplane function is represented as:

$$f(x) = \beta_0 + \beta^T x,$$ [5]

Where β and β_0 represent Weight Vector and Bias respectively.

2) The optimal hyperplane can be represented in an infinite number of different ways by scaling of β and β_0. As a matter of convention, among all the possible representations of the hyperplane, the one chosen is:

$$|\beta_0 + \beta^T x| = 1$$ [5]

Where $\mathbf{x}$ symbolizes the training examples closest to the hyperplane. In general, the training examples that are closest to the hyperplane are called **support vectors**. This representation is known as the **canonical hyperplane**.

3) Now, we use the result of geometry that gives the distance between a point $\mathbf{x}$ and a hyperplane (β, β_0) :

$$\text{distance} = \frac{|\beta_0 + \beta^T x|}{\|\beta\|}.$$ [5]

4) For the canonical hyperplane, the numerator is equal to one and the distance to the support vectors is:

$$\text{distance}_{\text{support vectors}} = \frac{|\beta_0 + \beta^T x|}{\|\beta\|} = \frac{1}{\|\beta\|}.$$ [5]

5) As intuitively described, the Margin (M) is twice the distance to the closest examples:

$$M = \frac{2}{\|\beta\|}$$ [5]

6) Finally, the problem of maximizing M is equivalent to the problem of minimizing a function L(β) subject to some constraints. The constraints model the requirement for the hyperplane to classify correctly all the training examples χ_i . Formally,

$$\min_{\beta,\beta_0} L(\beta) = \frac{1}{2}\|\beta\|^2 \text{ subject to } y_i(\beta^T x_i + \beta_0) \geq 1 \ \forall i,$$

[5]

where y_i represents each of the labels of the training examples

This is called **Sequential Minimal Optimization (SMO)** algorithm.

C. *Final Results*

The outputs of the classifier are shown in the following Screenshots:

```
PCA_output_n = 600
from sklearn.decomposition import PCA
pca = PCA(n_components=PCA_output_n)
```

```
pca.fit(X_data)
print (pca.explained_variance_ratio_.sum())
X_temp_data = pca.transform(X_data)
```

```
1.0
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_temp_data,Y, test_size=0.1)
```

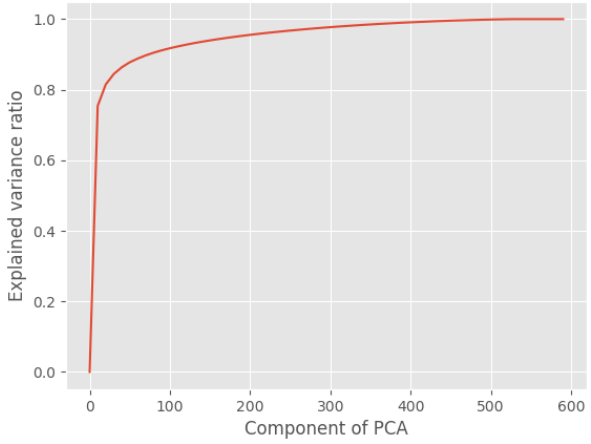
```
from sklearn import svm
clf = svm.SVC(gamma=0.0005,C=150,kernel='rbf',degree=5,probability=True)
clf.fit(X_train, y_train)
clf.score(X_test,y_test)
```

```
0.90740740740740744
```

```
test_wild_images = os.listdir(os.getcwd() + '/test_in_wild/')
for i in test_wild_images:
    temp =predict_output('/test_in_wild/'+i)
    #print (i,' =>',temp)
```

```
/test_in_wild/201501034_2.jpg => ['201501032']
/test_in_wild/201501038_2.jpeg => ['201501039']
/test_in_wild/201501038_2.jpg => ['201501038']
/test_in_wild/201501038_3.jpg => ['201501038']
/test_in_wild/201501067_1.jpg => ['201501067']
/test_in_wild/201501071_1.jpeg => ['201501097']
/test_in_wild/201501086_2.jpg => ['201501086']
/test_in_wild/201501086_3.jpg => ['201501086']
/test_in_wild/201501101.jpg => ['201501101']
/test_in_wild/201501121_1.jpeg => ['201501121']
```

The actual accuracy is of 85% on Valid images, but we can see that 7 out of 10 images are detected correctly. This variation is due to the tweaking of PCA and gamma parameters of the function. The following is the graph showing how variance ratio (i.e. accuracy score) of the output changes with PCA component count:



REFERENCES

[1] Turk, Matthew A., and Alex P. Pentland. "Face recognition using eigenfaces." *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR'91., IEEE Computer Society Conference on*. IEEE, 1991.

[2] Melišek, Ján Mazanec—Martin, and Miloš Oravec—Jarmila Pavlovicová. "Support vector machines, PCA and LDA in face recognition." *J. Electr. Eng* 59 (2008): 203-209.

[3] Schroff, Florian, Dmitry Kalenichenko, and James Philbin. "Facenet: A unified embedding for face recognition and clustering." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.

[4] Parkhi, Omkar M., Andrea Vedaldi, and Andrew Zisserman. "Deep Face Recognition." *BMVC*. Vol. 1. No. 3. 2015.

[5] OpenCV Documentation: https://docs.opencv.org/2.4/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html