FACE VERIFICATION SYSTEM

JIACHEN XIE; YICHENG WANG; QIHUA CHEN

PB16001698; PB16001699; PB16060171

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

This is the report for the project of Introduction to Machine Learning. In this project, we implement a face verification system to verify whether two facial images belong to the same person, using A subset of the Labeled Faces in the Wild (LFW) dataset. Our face verification system includes three steps. First, we extract the feature of input using a pre-trained Light CNN;Second, we use PCA to compress data; Finally, a SVM trained with SMO Algorithm is implemented as classification model to predict label.

1 INTRODUCTION

Face verification, which is the task of determining whether a pair of face images are from the same person, has been an active research topic in computer vision for decades. It has many important applications, including surveillance, access control, image retrieval, and automatic log-on for personal computer or mobile devices. However, various visual complications deteriorate the performance of face verification, such as occlusion, illumination and messy background. The Labeled Faces in the Wild (LFW) dataset is well known as a challenging benchmark for face verification. The dataset provides a large set of relatively unconstrained face images with complex variations in pose, lighting, expression, race, ethnicity, age, gender, clothing, hairstyles, and other parameters, which is difficult for automatic face verification.

Modern face verification methods are mainly divided into two steps: extracting low-level features, and building classification models. For the methods in the first step, for example, low-level features such as SIFT, LBP, and Gabor are handcrafted. For the methods in the second step, Random forest or Logistic regression can perform well. We can also use an end to end deep convolutional neural network to automatically extract features and predict label. However, training a deep neural network need large datasets, and it may also cause tremendous time for Parameter tuning. Therefore, in this project, we use a pre-trained face recognition network to extract feature, then PCA and SVM to produce reliable prediction result with limited data and tuning time.

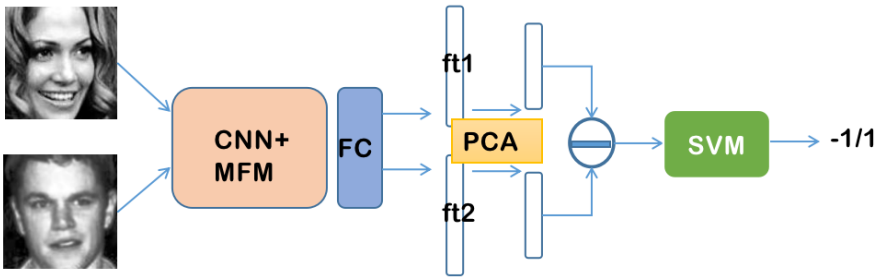

Figure 1: The pipeline of our model

2 METHODS

2.1 Feature Extraction

The subset of LFW dataset contains 1,600 pairs matching faces and 1,600 pairs mismatching faces. Every face can be seen as a sample. However, lots of random factors disturb the useful information of the samples such as illumination, noise, pose, etc. In order increase the efficiency of samples for the next step of PCA, we extracted features from the data. These features will replace the raw face data as the samples. CNN has great performance on feature extraction and usually provides high-level features.

We do not have enough data for CNN-training, thus we introduced a pre-trained Light-CNN Face Extraction model for feature extraction. This CNN model is not very deep with only 5 convolution layers, 4 Max-Feature-Map(MFM) layers, 4 Max-Pooling layers and a Fully-Connect layer. The structure shown in Fig.2(a). Each convolution layers follows a max-out operation, which separates the output feature maps into two parts and retain the greater element on corresponding position(Fig.2(b)). MFM layer is similar with the max-out operation mentioned above,

using only a set of weight and bias parameters (Fig.2.(c)). This layer can be seen as a kind of activation function.
We crop the 250*250 image to 128*128 with the face in the center and send it into the CNN model. The output is a 256-dimension vector. This vector will be used in the following PCA dimension reduction.

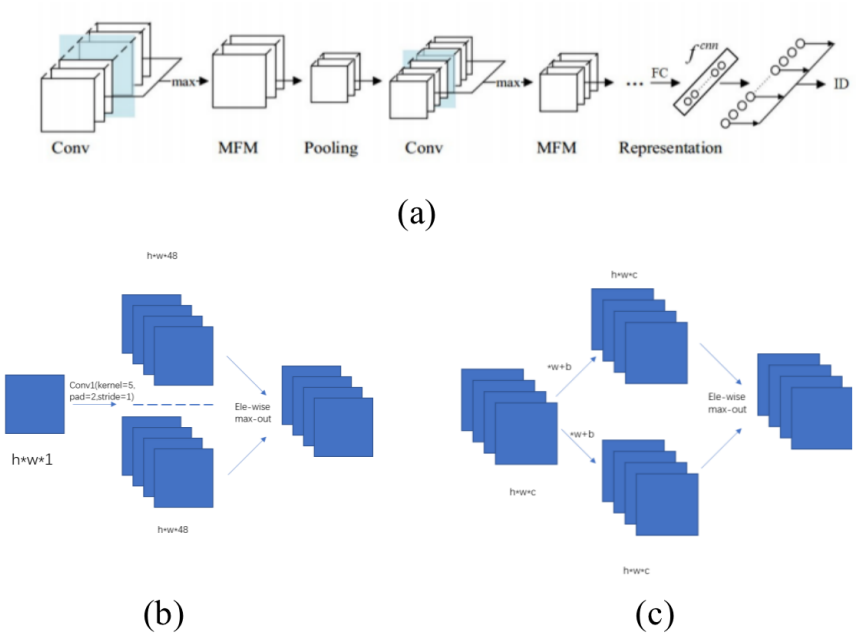

Figure 2: The structure of Light CNN.

2.2 PCA

Since we can't assume the feature is linearly separable, we use Radial Basis Function kernel for svm instead of linear kernel. However, RBF kernel can't perform well when the dimension of input features is high, so we choose to reduce the dimension through PCA. To reduce the dimension of the original feature space, PCA is used to find the projection direction which is the most effective representation of the original data by Maximize the sample variance. Assume $G \in \Re^{d*K}$ is the projection matrix which project $d \to K$, $x_i \in \Re^d$ is one of the samples, then the information preserved by the projected data instances is:

$$\frac{1}{n-1}\sum_{i=1}^{n}\|z_i - \hat{z}\|^2$$

where

$$z_i = GG^T x_i; \hat{z} = \frac{1}{n-1}\sum_{i=1}^{n} z_i$$

We can demonstrate that $[u_1, u-2...u_k]$ is an optimal solution of G, where $u_1, u-2...u_k$ of normalized sample matrix $\hat{X}$ according to first K largest singular values.

$$\hat{X}\hat{X}^T = U\Sigma U^T$$

except feature reducing, there is another important function of PCA: since we project the test set using the projection matrix obtained from training set, it can transfer the trained model to a new domain by modulating the statistics by PCA.

2.3 SVM

We know that the learning problem of SVM can be formalized to solve convex quadratic programming problem with global optimal solution. At present, many fast implementation algorithms have been proposed. In this paper, a widely used SMO algorithm was proposed by Platt in 1998.Reviewing the derivation process of machine learning course, SMO algorithm should solve the dual problem of convex quadratic programming as follows:

$$\min_{\vec{\alpha}} \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j K(\vec{x_i}, \vec{x_j}) \alpha_i \alpha_j - \sum_{i=1}^{N} \alpha_i$$

$$0 \leqslant \alpha_i \leqslant C, \forall i$$

$$\sum_{i=1}^{N} y_i \alpha_i = 0$$

In this problem, the variable is a Lagrange multiplier, a variable α_i corresponds to a sample point (x_i, y_i), and the total number of variables is equal to the training sample capacity N.

2.3.1 *Introduction To SMO Algorithm*

SMO is a heuristic algorithm, its basic idea is: if the solutions of all variables meet the KKT conditions of this optimization problem, then the solution of this optimization problem will be obtained. Because KKT is a necessary and sufficient condition for this optimization problem, otherwise, two variables are selected to fix other variables, and a quadratic programming problem is constructed for these two variables. The solution of this quadratic programming problem about these two variables should be closer to the solution of the original quadratic programming problem, because this will make the objective function value of the original quadratic programming problem smaller. What's more, at this time, sub The problem can be solved by analytic method, which greatly improves the operation speed of the whole algorithm. There are two variables in the sub-problem, one is the one that violates the most serious KKT condition, and the other is determined automatically by the constraint condition. In this way, SMO algorithm decomposes the original problem into sub-problems and solves the sub-problems to solve the original problem.
Therefore, the whole SMO algorithm consists of two parts: the analytic method of solving quadratic programming with two variables [1] and the heuristic method of selecting variables [1].

2.3.2 *Heuristics for Choosing Which Multipliers To Optimize*

The following part only briefly introduces the second part: variable selection method, while the first part: to solve two Lagrange multipliers, please refer to Platt [1] for more details. SMO algorithm chooses two variables in each sub-problem, at least one of which violates KKT condition.

SELECTION OF THE FIRST VARIABLE In SMO algorithm, the process of selecting the first variable is called the outer loop. The outer loop selects the sample points that violate the KKT condition most seriously in the training samples, and takes the corresponding variables as the first variable. Specifically, check whether the training sample points (x_i, y_i) meet the KKT conditions, that is

$$\alpha_i = 0 \Leftrightarrow y_i g(x_i) \geqslant 1$$

$$0 < \alpha_i < C \Leftrightarrow y_i g(x_i) = 1$$

$$\alpha_i = C \Leftrightarrow y_i g(x_i) \leqslant 1$$

The test is carried out in the range of ϵ. In the test process, the outer loop first traverses all the sample points that meet the conditions$0 < \alpha_i < C$: the support vector points on the interval boundary, and tests whether they meet the KKT condition. If all the sample points meet the KKT condition, then it traverses the whole training set to test whether they meet the KKT condition.

SELECTION OF THE SECOND VARIABLE SMO algorithm calls the process of selecting the second variable inner loop. Suppose that the first variable α_1 has been found in the outer loop, and now we need to find the second variable α_2 in the inner loop. The second variable selection criteria is to make α_2 change sufficiently. Refer to these two formulas (16)(17) in Platt [1], α_2^{new} depends on $|E_1 - E_2|$. In order to speed up the calculation, a simple way is to choose α_2 to make its corresponding $|E_1 - E_2|$, because α_1 has been determined and E_1 has also been determined. If E_1 is positive, select the smallest E_i as E_2; if E_1 is negative, select the largest E_1 as E_2. To save computation time, all E_i values are saved in a list.

2.3.3 *Summary To SMO Algorithm*

Input: training set $T = \{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}$, where $x_i \in \mathcal{R}^n$, $y_i \in \{-1, +1\}$, $i = 1, 2, ..., N$, accuracy ϵ;
Output: approximate solution $\hat{\alpha}$.

1. Initialize value $\alpha^{(0)} = 0$, let $k = 0$;
2. Optimization variables $\alpha_1^{(k)}, \alpha_2^{(k)}$, analytic solution to the optimization problem of two variables [2], get the optimal solution $\alpha_1^{(k+1)}, \alpha_2^{(k+1)}$, update α to $\alpha^{(k+1)}$;
3. If the shutdown conditions are met within the accuracy ϵ range:

$$\sum_{i=1}^{N} y_i \alpha_i = 0$$

$$0 \leqslant \alpha_i \leqslant C, i = 1, 2, ..., N$$

$$y_i g(x_i) = \begin{cases} \geqslant 1 & \{x_i | \alpha_i = 0\} \\ = 1 & \{x_i | 0 < \alpha_i < C\} \\ \leqslant 1 & \{x_i | \alpha_i = C\} \end{cases}$$

   where,

$$g(x_i) = \sum_{j=1}^{N} \alpha_j y_j K(x_j, x_i) + b$$

   then turn to 4; or let $k = k + 1$, turn to 2;
4. Take $\hat{\alpha} = \alpha^{(k+1)}$.

3 RESULTS

We explored the effectiveness of every part of our model in the experiments below.

3.1 Feature Extractor

We use a Light CNN to extract feature from images, which is pre-trained using large scale face recognition task. Since the dataset and task is different from ours, we should demonstrate the effectiveness of the extracted feature. The L_1 distance between two features are shown in figure2.

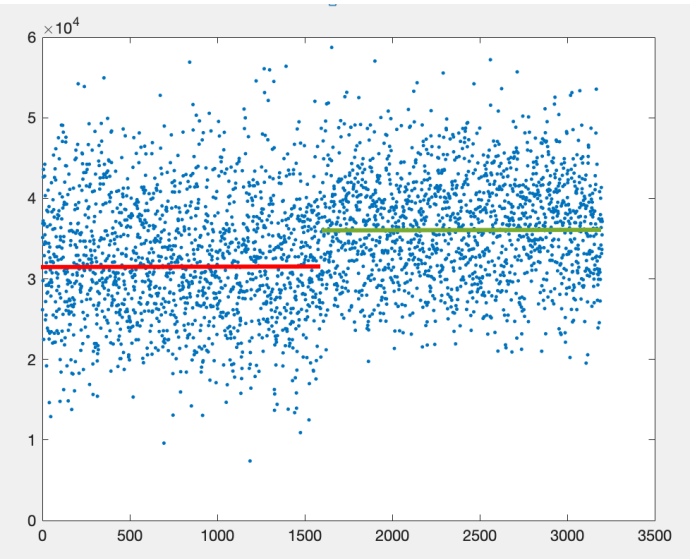

Figure 3: The first 1600 samples are from matched pairs, and the last 1600 are from mismatched pairs. We can see obvious difference between the distribution of L_1 distance of matched pairs and mismatched pairs.

The difference between two distributions is easy to see, when no obvious difference can be observed when we calculate L_1 distance between images on pixel level. We can obtain an accuracy of 63% if we separate the data using the middle line, which is equal to the case that, we use a linear classifier and set all weight to 1.

3.2 PCA result

3.2.1 *information preserved*

The ratio of information preserved by different projection dimensions is shown below.

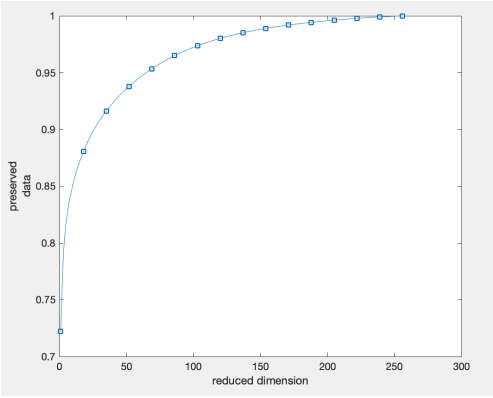

Figure 4: The ratio of information preserved by different projection dimensions

It can be seen that the first principle contains information more than 70%, and 90% of information are preserved in first 26 dimensions. However, since the difference between people's face are preserved by details, so we choose K=65, which preserve 95% of information.

3.2.2 *eigenface*

Since we didn't adopt CNN to extract the feature at the beginning, we implemented PCA directly on the original images. Note that in order to increase the performance of PCA, we use face landmark produced by a deep learning method, and turn all there pixels outside face to 0. The result is interesting. The first 8 eigen vector of cov matrix can be shown as eigenface, as in figure 5(a). The eigenface seems meaningful, each of them stand for an attribute of face. Then we use top 10% feature to reconstruct the image, and the result show that it preserved enough informstion for verification.

(**a**) (**b**)

Figure 5: The eigface and reconstruction quality of PCA

However, PCA is unsupervised, it treat every information equally, which means in many cases the component preserved are useless for our task face verification, and some times even misleading, for example, the component which is sensitive to the direction of the face. WE need a feature extractor that can extract features that are useful to our task, so we then add the cnn module before PCA.

3.3 SVM Prediction

When we use a linear kernel function, the performance of SVM is limited on the training dataset with accuracy approxiemately 60%, no matter how we tune the hyper parameter. This shows the feature is hardly linearly separable.
When we use RBF kernel function, the training converged soon with an accuracy above 95% on the training set. However, the accuracy on the validation dataset is low, which is a sign of over fitting. The convergence is so quick that, no matter how we tune the hyper parameters, the model convergent to nearly the same point, with no obvious increase of accuracy on the validation set.

4 DISCUSSION

This task is still challenging for our carefully designed model. The feature extractor can't fully extract it's ability, since LFW dataset is different from the dataset it trained on, since that dataset is manually aligned. The quality of feature extracted can severely influence the performance of SVM. The method to improve the performance include, (1) manually align all the dataset; (2) train the feature extractor using our dataset; (3) use a classifier that is not sensitive to the quality of input features. Although we spend lots of time considering how to make the feature better, the result is still not satisfying. This reflect how convinient it is to automatically extract feature using a neural network.