Machine Learning Algorithms Analysis of Face Recognition

Lintong Yang, Haoran Han

E-mail: han.h@husky.neu.edu

E-mail: yang.lin@husky.neu.edu

1. Abstract

Machine Learning Algorithms Analysis aims to validate existing algorithms in real-world context. Face Recognition contains many type of technology, such as Face Detection, Face Alignment, Face Attribute, Face Feature Extraction, Face Compare, Face Verification, Face Cluster. This paper mainly considers the algorithm about Face Compare. The input of Face Compare are features of two faces and output is the similarity of the two faces. Face Verification, Face Recognition, Face Retrieval are based on Face Compare and added some methods to implement. Face Compare Algorithm costs short time. The human face plays an important role in our social interaction, conveying people`s identity. Using the human face as a key to security, biometric face recognition technology has received significant attention in the past several years due to its potential for a wide variety of applications in both law enforcement and non-law enforcement. Face Recognition is a type of machine learning technology which is capable of identifying or verifying a person from a digital image or a video frame from a video source. One of the ways to do this is by comparing selected facial features from the image and a face database. It is typically used in security systems and can be compared to other biometrics such as fingerprint or eye iris recognition systems. Recently, it has also become popular as a commercial identification and marketing tool.

2. Introduction

As early as the 1950s, cognitive scientists began to study face recognition in the 1960s. Research on the application of face recognition engineering was officially launched.

In 1991, the famous “characteristic face” method first introduced principal component analysis and statistical feature technology into face recognition, achieving remarkable practical results. This idea was further developed in the follow-up study.

In the first decade of the 21st century, with the development of machine learning theory, scholars have successively explored the application of genetic algorithms, support vector machines (SVM), advanced nature, manifold learning and kernel methods in face recognition. Until now, face recognition has been widely used in different fields, like face payment, social media, public transit and etc.

Related works

2.1 Template matching

There are two main methods, fixed templates and deformed templates. The method of fixing the template is to first design one or several reference templates, and then calculate a certain measure between the test sample and the reference template, and determine whether the test sample is more than a threshold value. This method is relatively simple and used more often in early systems. However, due to the large variations in facial features, it is difficult to obtain an effective template to represent the commonality of human faces. The deformation template is in principle the same as the fixed template, but it contains some non-fixed elements. One method is to manually construct parametric curves and surfaces to represent certain non-fixed features in the face, such as eyes, nose, and lips. . Another method is that the system automatically generates an adaptive curve or surface to form a deformed face template. The detection method is: elastically matching the template with the test image, adding a penalty mechanism, and using a certain energy function to express the matching degree.

2.2 Neural Networks

In essence, neural networks are also a sample-based learning method. The use of neural networks for face detection has made great progress. The MIT scholar first clustered the face sample set and the non-face sample set to test the distance between the sample and the subsamples of the face sample set and the non-face sample set as a measure of classification using a multi-layer perceptron (The MLP) network acts as a classifier. The researchers of CMU directly use the image as the input of the neural network to design a neural network classifier with unique structure that is suitable for face features, and optimize the detection result through the feedforward neural network. Raphael Feraud and others used multiple neural networks: Multilayer Perceptron (MLP) and Constrained Generative Model (CGM) to implement a fast and accurate face detection method that can be applied to face image retrieval in WEB. . Shang-Hung Lin and others trained three Probabilistic Decision Based Neural Networks (PDBNNs) for face detection, eye positioning, and face recognition to implement a complete face recognition system.

2.3 Method Based on HMM

The Markov model is a discrete-time finite state automata. Hidden Markov model (HMM) means that the internal state of the Markov model is invisible to the outside world, and the outside world can only see the output values at each moment. For face mode, we can divide it into a sequence of forehead, eyes, nose, mouth, and chin. Face patterns can be detected by orderly recognition of these areas, which is exactly what Hidden Markov Models can easily achieve. Samaria et al. proposed an algorithm for face detection using the HMM

model. They used the structural information of the human face region as the state transition condition of the hidden Markov model. In addition, face recognition algorithms based on AdaBoost, methods based on color information, methods based on shape analysis, and multi-modal information fusion methods have been extensively studied and experimented at home and abroad.

3. Code with Documentation

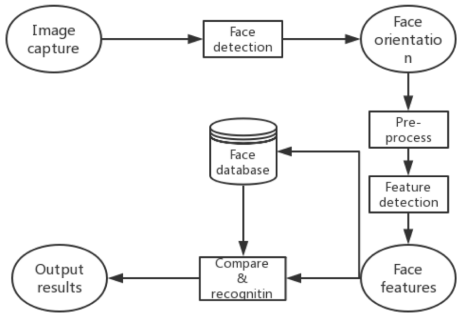
In this part of paper, the graphs of workflow and code will shows the whole process of the project.

3.1 Dataset

The project use sets of photographs of famous people such as movie stars and singer stars as data source. From these photos, the code could do face detection, face comparation more easily and get result more obviously.

3.2 Steps of code

This part will show the whole process of the project`s code and the paper will introduce the process with details. The picture1 shows the whole workflow of the project.



picture1: workflow of project

3.3 Step 1: Find all faces

In the first step, the project finds all faces in a picture, such as iphone`s camera application. The project uses Histogram of Oriented Gradients (HOG) method to capture all faces.

For this method, we first convert the picture from colored to black and white, because we don`t need the colored data to detect faces. This picture shows how we convert images from RGB to grey image.

picture 3.3.1: grey image of Emma

And then the program will look for every pixel in the picture we detect. For each pixel,

the program also check the other pixels around this one. The goal is to compare gradients of this pixel and the pixels surround. We use an arrow to represent the gradients for the color change to dark. This are the formulas used to calculate direction gradient of each pixel.

$$M(x,y) = \sqrt{I_x^2 + I_y^2}$$

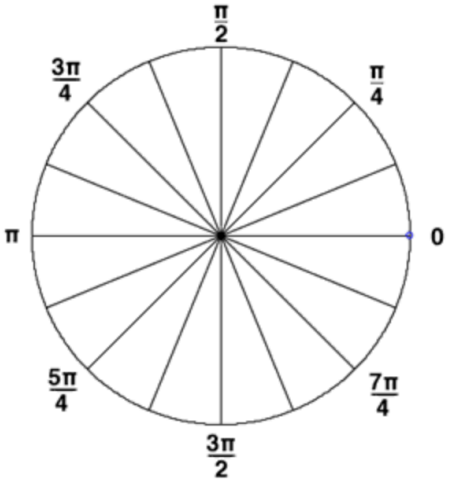
$$\theta(x,y) = \tan^{-1}\frac{I_y}{I_x} \in [0, 360^\circ) or [0, 180^\circ]$$

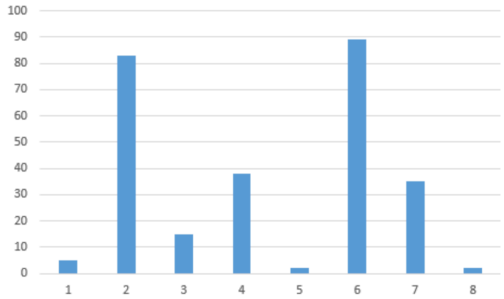
formula 3.3: HOG formula

I_x and I_y represent the gradient values in the horizontal and vertical direction, $M(x,y)$ represents the magnitude of the gradient, and (x,y) represents the direction of the gradient.

We divided the image into several "cell cells", by default we set the cell to 8*8 pixels. Suppose we use 8 bin histograms to measure the gradient information of 6*6 pixels. 360 degrees and gradient direction of the cell is divided into eight directions, for example: if the pixel gradient direction is 0-22.5 degrees, the histogram of the first bin count plus one, in this way, each pixel within the cell to use in the histogram weighted gradient direction projection range) (maps to the fixed point of view, you can get this cell gradient direction histogram, is that the cell characteristic vector and gradient size 8 d as a projection of the weights.

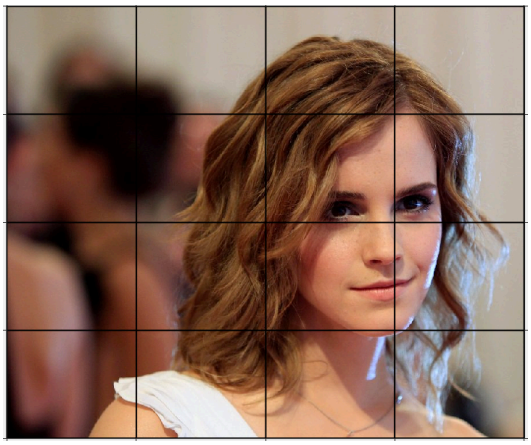
The cell unit is combined into a large block, and the normalized gradient histogram of the block is normalized.





Due to the change of local lighting and foreground - background contrast, the range of gradient intensity is very large. This requires normalization of gradient strength. Normalization can further compress light, shadows, and edges.

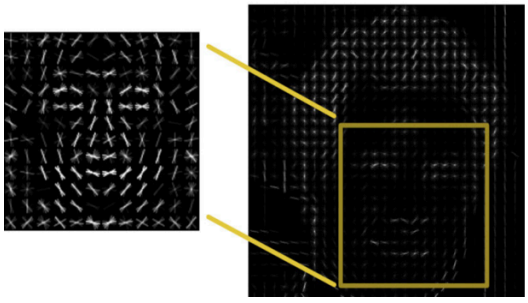
Combine cell units into large, spatially connected intervals (blocks). In this way, the characteristic vector of all cells in a block is connected in series to get the HOG feature of the block. These intervals are overlapped.



picture 3.3.2

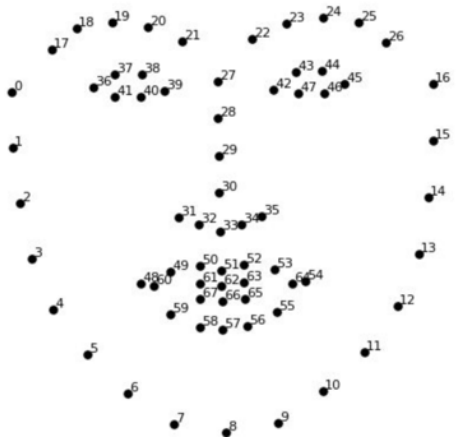
In each of the cubes, the program will calculate how many gradients there are in each main direction (how many points, pointing to the right, pointing to the right, etc.). And then we're going to replace that little square with the arrow that's the strongest direction.

The end result is that we transform the original image into a very simple form of expression that captures the basic structure of the face in a simple way. This picture shows the result of the process.



3.4 Step 2: Different positions of faces

In step 2, the problem we have to deal with is, for a computer, the same face in different directions is different. Humans can easily recognize that both images are the same, but the computer thinks these two pictures are two completely different people. To solve this, we will try to distort each image so that the eyes and lips are always in the sample place in the image. This will make it easier to compare the differences between faces in the next steps. To do this, we'll use an algorithm called face landmark estimation. There are many ways to do this, but this time we will use a method invented by Vahid Kazemi and Josephine Sullivan in 2014. The basic idea of this algorithm is to find 68 people face the prevalence of particular point (called feature points, the landmarks), including the top of the jaw, external contours of each eye, each internal contours of the eyebrows, etc. Next the program trains a machine learning algorithm that allows it to find 68 specific points in any face. We will place 68 feature points on each face. The picture shows the 68 landmarks need to be captured.

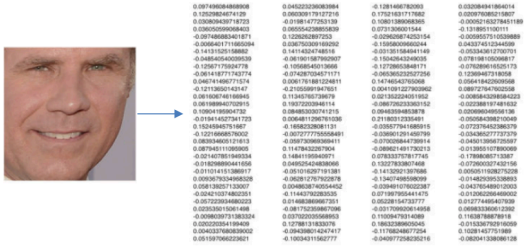


Now, we know where the eyes and mouth are, and we rotate, scale, and cut the

image so that the eyes and mouth are as close to the center as possible. We're not going to do any fancy 3d distortion, because it distorts the image. We will only use the basic image transformations that keep the images relatively parallel, such as rotation and scaling (called affine transformations). Now, no matter which side of the face we face, we can move our eyes and mouth to roughly the same position. This will make our next step more accurate.

3.5 Step 3: Coding the face

The simplest method of face recognition is to directly compare the unknown human face we found in step 2 with the face images we have already marked. It must be the same person when we find that the unknown face looks similar to a previously marked face. There is actually a huge problem with this approach. A site like Facebook, which has billions of users and trillions of photos, is not likely to go around and compare each of the previously labeled faces, which is too long. We need to recognize faces in milliseconds, not hours. The method we need is to extract some basic measurements from each person's face. Then we can measure the unknown face in the same way and find the known face closest to the measurement. This picture shows the process of this part.



For example, we can measure the size of each ear, the distance between the eyes, the length of the nose, and so on. So the solution is to train a deep convolutional neural network. However, it's not about identifying objects in the picture, and this time we're training it to generate 128 measurements for the face. And by comparing the 128 measurements from different images, we can find difference and get the same image. In training process, three different steps:

1. Load the face training image of a known person.

2. Load another photo of the same person.

3. Load another person's photo.

The algorithm then looks at its own measurements for the three images. Then, adjust the neural network slightly to ensure that the first and second generated measurements are close, while the second and third generated measurements are slightly different.

After repeating the process millions of times for millions of people, the neural network learned how to reliably measure 128 measurements per person. For any ten different photos of the same person, it should give roughly the same measurements.

This is the graph of 128 measurements

value of Emma image.

-0.1246	-0.098	0.29288	-0.1788
0.13702	0.08022	0.08616	-0.0282
-0.0181	0.00524	-0.1726	-0.1274
-0.051	0.13054	-0.103	-0.048
-0.1808	-0.0546	0.1866	0.0051
0.05222	-0.1892	-0.1538	-0.1234
-0.0231	0.25139	-0.0738	-0.0362
-0.0991	0.0188	0.15466	0.01751
0.18069	-0.1987	-0.1754	-0.1947
-0.1878	-0.0288	-0.2472	0.08896
0.15734	0.18038	-0.2287	-0.0943
-0.0702	0.31737	0.06776	-0.0568
-0.3592	0.24496	0.41852	-0.1063
-0.0007	-0.0818	0.21672	-0.0648
-0.0606	0.02782	-0.0148	-0.1638
0.23831	-0.0255	0.03795	0.09494
-0.2702	0.22745	-0.0619	0.19536
-0.1787	-0.354	-0.0534	-0.3364
-0.1108	0.04807	-0.0839	0.13722
-0.164	0.14909	0.16367	0.14458
0.04154	0.11218	-0.0472	0.12208
0.05795	0.08541	-0.0673	0.17612
0.09158	0.16597	-0.134	0.06031
0.13219	-0.1633	0.13484	0.04233
-0.1881	0.02901	0.2344	0.05752
-0.345	0.20641	-0.029	-0.118
-0.0394	-0.1709	-0.0575	-0.1295
-0.1472	0.04947	0.33964	-0.0861
-0.088	0.08956	0.03555	0.09217
-0.1665	-0.081	-0.0886	-0.0149
0.06768	0.02359	-0.0149	0.05854
0.07243	-0.0436	0.01992	0.05165

picture: the 128 measurement values

3.6 Step 4: Face comparation

This last step is actually the simplest step in the whole process. All we have to do is find the person in the database that is closest to our test image.

The program can achieve this by any basic machine learning classification algorithm. We don't need fancy deep learning skills. The program uses a simple linear SVM classifier, but there are many other classification algorithms that can be used.

What the project needs to do is to train a classifier, which can take measurements from a new test image and find the person that matches the most. The classifier runs only a few milliseconds, and the result of the classifier is the person's name!

4. Results

After we have implemented our project code, what we need to do is to do the experiment and test our project code. We use the pictures of Emma Waston and Miranda Kerr to build the project`s database and we first train the picture of Emma and get the 128 measurement values of Emma`s picture. And then we train the picture of Kerr and the program also get its 128 measurement values. Second, we use the code to compare these two images, found that they are different people. But when we put the same people`s picture in database, the output of program is right. This is the result of our experiment.

```
In [11]: # calculate euclidean distance and judge if they are the same people
         comparePersonData(person_data1, person_data2)

         0.869438026398012
         It's not the same person
```

From the experiment we can draw the conclusion that our project`s code id runnable and right. Our machine learning algorithm of face recognition can recognize face rightly.

5. Discussion

Software setup and packages import. During the process we implement our project, there are several problems we met. The first one is to install necessary software and import some packages such as dlib, opencv etc. We are confused about how to import these packages. But after many tries and searching materials online, we setup environment successfully.

Training images and building model. Before we did the experiment. We need to set up our model and train our neural network. It

cost a lot of time.

Reference

1. Face Recognition Algorithm one: （Eigenface Method）. (n.d.). Retrieved February 08, 2018, from http://blog.csdn.net/smartempire/article/details/21406005 [Accessed 8 Feb. 2018].
2. Phillips, P., Moon, H., Rauss, P., & Rizvi, S. (n.d.). The FERET evaluation methodology for face-recognition algorithms. *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. doi:10.1109/cvpr.1997.609311
3. Hwang, B., Roh, M., & Lee, S. (n.d.). Performance evaluation of face recognition algorithms on asian face database. *Sixth IEEE International Conference on Automatic Face and Gesture Recognition, 2004. Proceedings.*doi:10.1109/afgr.2004.1301544
4. Face database. (2016, April 18). Retrieved from https://blog.csdn.net/lilai619/article/details/51178971
5. J. Deutscher, A. Blake, and I. Reid. Experiments with a new boosting algorithm. In *CVPR*, 2000.
6. C. Liu, H.-Y. Shum, and C. Zhang. Hierarchical shape modeling for automatic face localization. In *ECCV*, 2002.
7. K. Okuma, A. Taleghani, D. Freitas, J. J. Little, and D. G. Lowe. A boosted particle filter: Multitarget detection and tracking. In *ECCV*, 2004.
8. F. Porikli and O. Tuzel. Object tracking in low-frame-rate video. *SPIE Image and Video Communications and Processing*, 5685:72–79, 2005.