Gender Classification in Three Methods

Chenghao Du(N11726887)
Ruiqi Cao(N19515630)

Abstract

Nowadays automatic gender classification based on human facial appearance has been applied in many real-life applications, such as personalized advertisement, customer service management, and entertainment. In this project, we use three popular approaches, KNN, CNN and SVM to classify the images by gender, and compare these three approaches on running time and test accuracy. The results show that neural networks approach gives the best accuracy of 97.5%, while SVM and LDA-KNN yields 94% and 91% respectively. On the other hand, LDA-KNN(~8s) has far less time complexity than CNN(~500s) and SVM(~50s).

I. Introduction

Gender classification is an important study for many real-life applications, such as customer service management and national crime intelligence service. There are numerous studies [1, 2] and several large datasets on the gender classification based on a single face image. Among these studies, neural Network is widely used, which showed good performance. So in this project, we want to study the gender of images by three different approaches, which are CNN, KNN and SVM, and compare their performance mainly on running time and test accuracy.

II. Background

Since there are many datasets on the gender classification, we chose the dataset from "Labeled Faces in the Wild Home". The face images from this public dataset are all crawled from IMDB and Wikipedia website and we used part of this dataset to classify gender by CNN, LDA-KNN, and SVM.

III. Methods

A. Pre-processing

To make it easier for our classifiers to extract key features, we want to normalize the training set to eliminate the influence of head orientation, face tilt, and illumination condition.

For each picture in the training set, we first employ HOG based linear detectors in Dlib to extract facial landmarks. This detector is able to quickly pinpoint the ROI (Rectangle of Interest) that contains a single face as well as its 68 vital landmarks including eye corners, nose tip, chin, mouse corners, etc. Based on these coordinates we: (1) estimate the orientation of the heads in training set and filter out those that are not facing front-on towards the camera, (2) rotate and align a face according to their eye coordinates and crop out the area of interest.

Figure 1 Landmarks on 3D Head Model 2D Picture and Caculated Direction Vector

The method in step (1) is described Satya Mallick (2016). According to his article, if coordinates of same landmarks are known for both a 3D head and its projection on a 2D (camera) plane, we can reversely compute the direction vector of the projection given the camera coefficients, as shown in Figure 1. In realistic scenario, the 3D head model of a specific person can be replaced by an average head model, as still yields quite accurate result for the purpose of head pose study. Similarly, the camera coefficients are also easy to estimate from a picture itself. Therefore, the (yaw, pitch, row) of a face can be easily determined from the results from Dlib's HOG detector. In our experiment, we filtered out the pictures with yaw and pitch greater than ± 10 degrees. This decision is reasonable to reduce the problem size and for those pictures lying out of this range it is trivial to apply similar filters and train same classifiers as shown in our paper.

In step (2), a face first is rotated so that the segment joining two eyes becomes horizontal. Then a square containing the face is cropped out with eyes at a fixed height, so that the eyes of all images are properly aligned (Figure 3). The side length of the ROI is dictated by the product of a factor ω and the length l of the segment joining two eyes. As shown in following Figure 2, larger choices of ω will include more information in ROI.

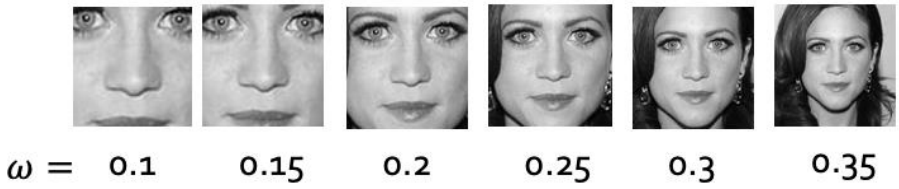

Figure 2 ROI variation with ω

Figure 3 Aligned Faces

Figure 4 Original Image, Image After Histogram Equalization, Image After CLAHE

The final step of preprocessing is illumination fix. Empirically, the illumination condition plays an important role in computer vision problems. Therefore, here we tried two different illumination fix technique: Histogram Equalization and Contrast-limited Adaptive Equalization. Histogram Equalization tries to enhance the contrast of the whole image by expanding the most intensive area in histogram of original picture, while contrast-limited Adaptive Histogram Equalization (CLAHE) uses multiple histograms to better bring out local details. (Figure 4)

In the end, a training set with 200 female samples and 200 male samples are generated. The test set contains 200 pictures in total, of which there are 100 females and 100 males. Each picture is in grayscale format and the size is 200*200.

B. KNN-LDA/SVM based on PCA

With the training images properly aligned, we can leverage the Principle Component Analysis method to extract the “eigenfaces” of training samples. This is a very classic feature extraction technique in face recognition which tries to identify the special directions (subspace) in sample space where the variance of training samples’ projection is maximized. In our application PCA not only efficiently reduces the dimension but also reduce the influence of noise. Note that PCA is an un-supervised learning method, that is the eigenfaces contains the most significant features both male and female samples. In the following step, LDA, we will try to identify the gender-specific features.

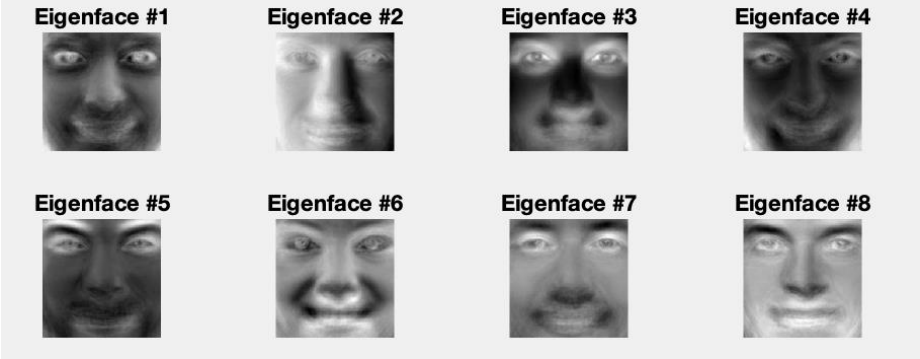

Figure 5 An Illustration of Eigenfaces

LDA is in some sense similar to PCA, as it still wants to identify a special direction in feature space. The difference is that this direction is to maximize the distance between labeled classes. In our case LDA points out the direction where both genders are best separated. After projecting all samples in this direction, we can use a simple threshold as a classifier. Yet a better approach is to use a K nearest neighbor algorithm. LDA might looks coarse in some ways as it collapses the whole feature space into one dimension. Therefore, in comparison, we use an SVM based on the result of PCA.

C. CNN

In this project, my initial architecture for my CNN model is listed below:

1. Convolutional Layer #1: applies 32 3x3 filters with Relu activation function.
2. Convolutional Layer #2: applies 32 3x3 filters with Relu activation function.
3. Pooling Layer: applies max pooling using a 2x2 filter and stride of 2.
4. Dense Layer #1: 128 neurons with Relu activation function.
5. Dense Layer#2: 2 neurons with Softmax function.

D. SVM

The reason we use SVM is because it is a non-probabilistic binary linear classifier. The core idea of SVM is building an hyperplane which maximize the margin as the expression shown in Figure 6. Once a boundary is established, most of the training data is redundant. All it needs is a core set of points which can help identify and set the boundary.

$$\left[\frac{1}{n}\sum_{i=1}^{n}\max\left(0, 1 - y_i(w \cdot x_i - b)\right)\right] + \lambda\|w\|^2.$$

Figure 6 SVM Classifier Expression

Another reason we chose SVM is the kernel trick. Since we used HOG, LBP+HOG, and Haar-like algorithm to extract features from photos. All these features have high-dimensional variables which cannot be separated very well by linear model. In this project, we compared the difference between:

Linear kernel: $k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top \mathbf{x}'$

RBF kernel: $k(\mathbf{x}, \mathbf{x}') = \left(1 + \mathbf{x}^\top \mathbf{x}'\right)^d$ for any $d > 0$

Polynomial kernel: $k(\mathbf{x}, \mathbf{x}') = \exp\left(-||\mathbf{x} - \mathbf{x}'||^2 / 2\sigma^2\right)$ for $\sigma > 0$

IV. Experiments

A. KNN-LDA/SVM based on PCA

There are several parameters to tune in the preprocessing and model building step. Therefore, we employed a 10-fold cross validation for our experiment in this paragraph. First, as shown in Figure 2, different choices of ω have significant impact on the information containing in the pictures. For instance, when $\omega = 0.15$ the picture only contains the eyes, nose, and mouth, while we can clearly see her hair style if $\omega = 0.35$. Intuitively, women's hair tends to be longer than men's, which can potentially be a feature. However, the cross validations (Figure 7) show that when $\omega = 0.2$, the images provide enough image for the purpose of classification and our model might not be expressive enough to capture more complex features (including hairstyle).

Secondly, the number of eigenfaces generated from PCA must be determined. As Figure 8 shows, there is a sweet zone for the choice of number of eigenfaces. Too little eigenfaces makes the model not expressive enough, while too many eigenfaces incurs noise. In our experiment, we selected 80 according to the result.

At last, we want to determine the best parameter K in K-NN. Per our validation results, 5 is the optimized value for the k in our experiment. With these tuned parameters, the final accuracy on the 200-image test set is 91%. In comparison, the tuned linear SVM has an accuracy of 88%. Furthermore, we tested both classifiers on a 1300-sample dataset. The cross-validation accuracy is 90% for PCA based KNN/LDA and 88% for SVM. Furthermore, as of workload, a 10-fold cross validation of SVM on 400-image dataset takes on average 48.5s to finish while KNN/LDA takes on average 8.6s to finish.

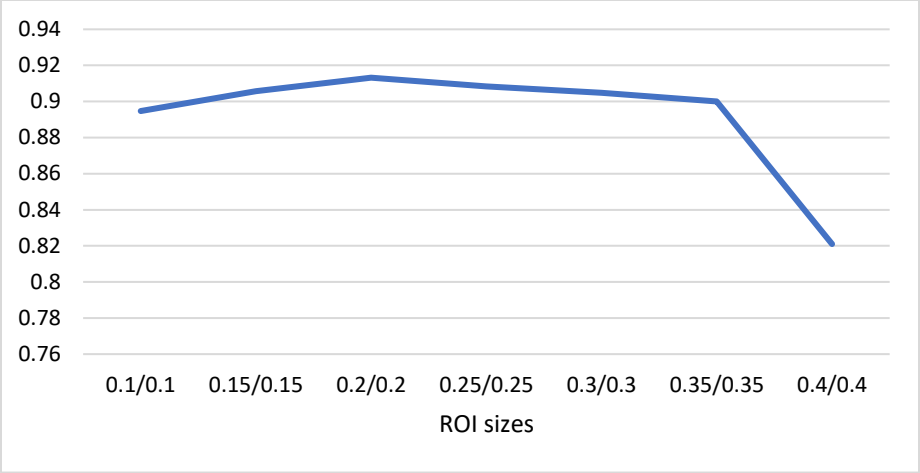

Figure 7 Validation Accuracy variation with the ROI size

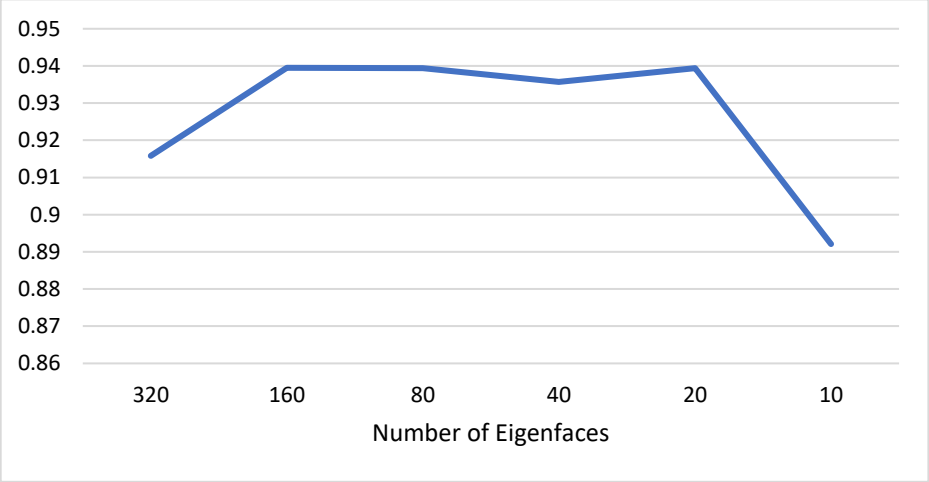

Figure 8 Validation Accuracy w.r.t number of eigenfaces

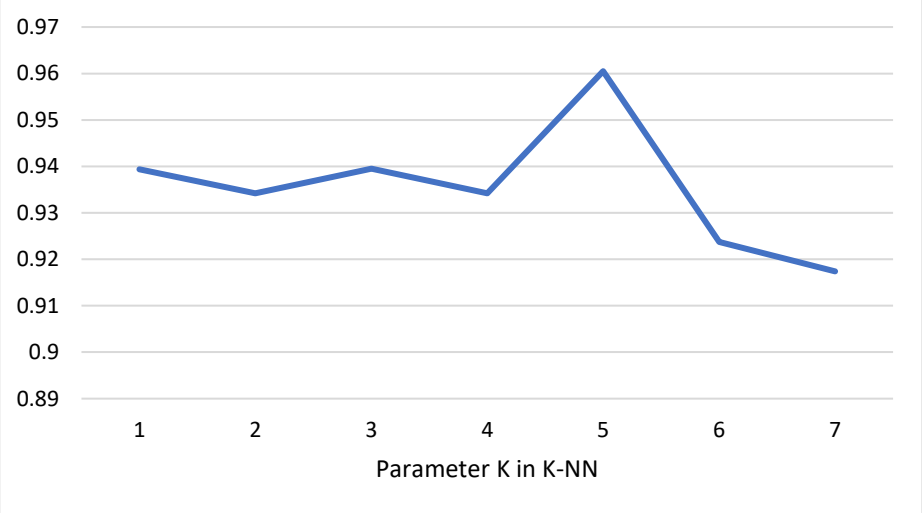

Figure 9 Validation Accuracy w.r.t K in K-NN

B. CNN Experiment

When implementing CNN model in this project, the TensorFlow was used as our tool. In the training part, we only used 400 preprocessed grayscale images, which include 200 male images and 200 female images. Besides, 200 preprocessed grayscale images including 100 male images and 100 female images were used for testing process. At first, I applied my initial architecture to build my CNN model, which includes two convolutional layers, one pooling layer and two dense layers. Under this model, we got ~70.5% accuracy on test images. Then I knew that dropout is an important and simple way to prevent neural networks from overfitting. The key idea is to randomly drop units (along with their connections) from the neural network during training [4]. Dropout was added at the beginning of the first dense layer. After tuning the dropout from 0.1 to 0.9, the accuracy was improved significantly, which was ~95%. Besides, sometimes normalization may have also a great improvement on the model [5]. Mean subtraction and standard deviation division were applied in the normalization stage and we got the best accuracy of ~97.75% in this model.

	Initial architecture	Tuning parameter	Normalization
Accuracy	70.5%	95%	97.75%

C. SVM Experiment

The team used the OpenCV library to extract feature from photos. The following are a basic introduction and comparison between three features we use.

HOG(Histogram of Oriented Gradients): In the HOG feature descriptor, the histograms of directions of gradients are used as features. Gradients of an image are useful because the magnitude of gradients is large around edges and corners and they can represent more information about object shape than flat regions.

LBP(Local binary patterns): Just like HOG algorithm, LBPs also compute a local representation of texture. This local representation is constructed by comparing each pixel with its surrounding neighborhood of pixels. For example, forming labels for the image pixels by thresholding the 3 x 3 neighborhood of each pixel with the center value and considering the result as a binary number. The histogram of these $2^8 = 256$ different labels can then be used as a texture descriptor.

Haar-like: Different from HOG and LBP algorithm, Haar-like feature considers adjacent rectangular regions at a specific location in a detection window, sums up the pixel intensities in each region and calculates the difference between these sums. Then we can use these difference to categorize subsections of an image.

The first experiment in SVM compared the accuracy by using different kernel function and different feature.

	Linear kernel	RBF kernel	Polynomial kernel
HOG	40%	40%	82.5%
LBP+HOG	67.5%	85%	90.5%
Haar-like	71%	82.5%	94%

Based on the experiment results, we can find that Haar-like feature always better than other two algorithm no matter what kernel function in SVM. And it also took the shortest time to train the data.
We also modified the feature dimension for HOG and LBP+HOG, and compared the relationship between training time and the size of feature dimension.

HOG(Polynomial kernel)

Dimension	32	64	128	256
Time(s)	0.9	1.3	5.1	13.2

LBP+HOG(Polynomial kernel)

Dimension	16	64	256	1024
Time(s)	0.7	2.3	11.5	134.6

Since SVM is very sensitive to the dimension of input data, the training time has a exponential relationship with variable dimension. When the team changed LBP feature to 1024 dimensions, it took about two minutes to get the result but the accuracy was even worse. Based on these result, we can conclude that using Haar-like feature and Polynomial kernel as the kernel function in SVM can get the most accuracy result and the training time for this case is still the best one.

V. Discussion

A. K-NN Discussion

The classic KNN has a reasonable performance on gender classification task. It even out performs SVM in our test cases. Moreover, it involves far less computation compared with SVM.

B. CNN Discussion

During tuning the dropout parameter, I found there was a great reduce on accuracy when I added some dropout on the input layer, which means the beginning of first layer. But in some experiments [4] dropout in the lower layer helps. The main reason is that our training images are highly preprocessed and there are seldom noisy inputs, which have a great effect of overfitting for highly fully connected layer. In the future, we could try more layers in the CNN model and find a good tradeoff between training time and accuracy.

C. SVM Discussion

SVM is very sensitive to the dimension, when the input dimensions increase, the running time will also increase dramatically. Although LBP+HOG algorithm can extract a lot details in a picture, it also needs a lot of data to represent these information. One way to improve this feature is optimize the original picture and find a suitable extract window.

Reference

[1] E. Eidinger, R. Enbar, and T. Hassner. Age and gender estimation of unfiltered faces. IEEE Transactions on Information Forensics and Security, 9(12):2170–2179, 2014. 1.

[2] R. Rothe, R. Timofte, and L. V. Gool. Dex: Deep expectation of apparent age from a single image. In IEEE International Conference on Computer Vision Workshops (ICCVW), December 2015.

[3] S. Karamizadeh, S. M. Abdullah, M. Halimi, J. Shayan and M. j. Rajabi, "Advantage and drawback of support vector machine functionality," 2014 International Conference on Computer, Communications, and Control Technology (I4CT), Langkawi, 2014, pp. 63-65.

[4] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. JMLR, 2014. 5

[5] M. Liang and X. Hu. Recurrent convolutional neural network for object recognition. In CVPR, 2015. 2, 4.

[6] OpenCV Tutorial on Face Recognition:
https://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_tutorial.html

[7] S. Mallick, Head Pose Estimation using OpenCV and Dlib:
https://www.learnopencv.com/head-pose-estimation-using-opencv-and-dlib/