CIS442D Advanced Business Analytics

Python Final Project
Facial Expression Recognition

Team Members:

Li Ding, Yining Xu, Zhangqi Li, Menglin Cui

1. Introduction

Using the feedback from the recognition of gamers' reaction to enhance the gaming experience on systems like the Microsoft Kinect, determining the drivers' stress level through analyzing their facial expression to alert if they should take a break… Facial expression analysis gradually takes part in our daily life as computers are becoming more human centered. Facial behavior has always played an essential role in the aspect of sensing human emotions, as well as speculating their intentions. In this case, we obtained our dataset from Kaggle.com, and dug into facial expressions from an image using deep learning.

Following is our report structure. Section 2 illustrates the datasets used to train and test our algorithm. Section 3 describes the problem with details. Section 4 describes how we preprocess the data in order to fit the models. Section 5 discusses the models we used, respectively are decision tree, random forest, support vector machine, and convolutional neural network. Section 6 presents our results, along with the analysis.

2. Dataset

Figure 1. Example images from the dataset

The data consists of 48x48 pixel grayscale images of faces. The seven categories into which the images are to be classified based on facial expression are Angry (category 0), Disgust (category 1), Fear (category 2), Happy (category 3), Sad (category 4), Surprise (category 5) and Neutral (category 6). The dataset contains 35887 examples, which are separated into the training set with 28,709 examples, and the public and the private test set with 3,589 examples respectively. In our project we used the public test set for validation and the private test set to report test error rate. Some example images from the dataset are shown in Figure 1 (each row represent 7 faces of the same expression). The number of examples of each class in the dataset are as listed in Table 1.

Table 1. Distribution of labels in training, public test and private test sets

Label	Training Set	Public Test Set	Private Test Set
Angry	3995	467	491
Disgust	436	56	55
Fear	4097	496	528
Happy	7215	895	879
Sad	4830	653	594
Surprise	3171	415	416
Neutral	4965	607	626

3. Problem Description

The goal of this project is to predict, from the grayscale picture of a person's face, which emotion the facial expression conveys. That is, we input a 48 by 48 grayscale image of a face and output the emotion conveyed by facial expression. The main problem we encountered in this project is that although humans recognize facial expressions virtually without any effort or delay, automatic facial expression recognition by machine is very challenging as faces vary from one individual to another quite considerably due to different age and ethnicity. So in the following part, we fit four models to the dataset, and try to find the best accuracy we can reach.

4. Preprocessing

4.1 Preprocessing for SVM, decision tree and random forest

The biggest challenge we faced in this dataset was too many features (48*48). If all these feature were fit into a model, it would be too slow to return the result. For our experiments, we provided Eigenfaces as inputs to SVM, decision tree and random forest. We performed Principal Components Analysis on the 2304 features on each image and extracted the top 65 principal components from them. Once this was done, the Training, PublicTest and PrivateTest data were then mapped to these principal components.

4.2 Preprocessing for convolutional neural network

We first reshaped the image to 48x48 and then added the depth to the input. Depth is only one since the images are in grayscale. We then made the pixel value floats in [0,1] by dividing them by 255. Next class vectors (emotions) were converted to binary class matrices. This is the only preprocessing done on the data before it is fed to our convolutional neural network. The convolutional neural network learns the appropriate features from this normalized data.

5. Learning Models

5.1 Decision Tree

We implemented a decision tree using the scikit-learn library in Python. From sklearn we built the decision tree classifier using default parameters and fit the tree using Training data. We also tried to modify the max_depth and min_samples_leaf parameters trying to find the highest accuracy.

5.2 Random Forest

We then implemented a random forest using the same library in Python. We set parameter max_feature as "sqrt", and both n_estimators and max_depth as 100.

5.3 SVM

Basically SVM is used for two-class prediction. The simplest way to extend SVMs for multiclass problems is using the so-called one-vs-one approach and classifying according to the largest number of 'votes'. We fit SVM with both linear and polynomial kernel and try to find a better accuracy.

5.4 Convolutional Neural Network

Next, we implemented convolutional neural networks for our problem. Convolutional neural networks are made up of neurons that have learnable weights and biases. Each neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity.

Convolutional Neural Networks take advantage of the fact that the input consists of images and they constrain the architecture in a more sensible way. In particular, unlike a regular Neural Network, the layers of a ConvNet have neurons arranged in 3 dimensions: width, height, depth. (Note that the word depth here refers to the third dimension of an activation volume, not to the depth of a full Neural Network, which can refer to the total number of layers in a network.) For example, the input images in CIFAR-10 are an input volume of activations, and the volume has dimensions 32x32x3 (width, height, depth respectively).

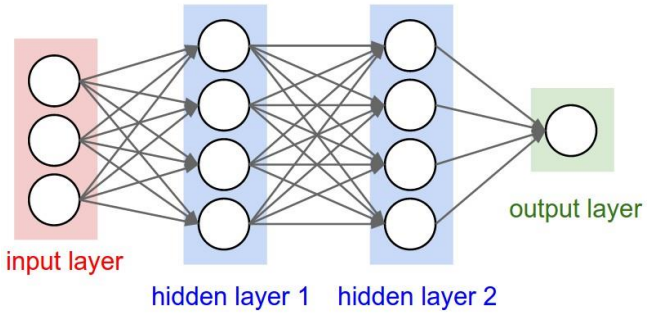

Figure 2. A regular 3-layer neural network

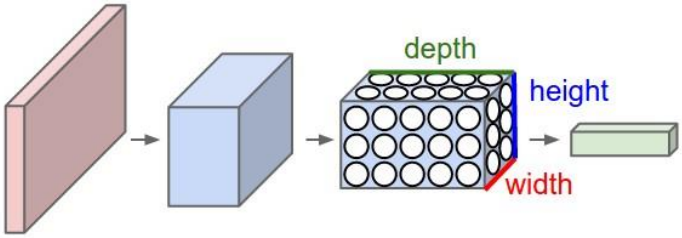

Figure 3. A ConvNet arranges its neurons in three dimensions (width, height, depth)

The network architecture is loosely based on the approach from the example CNN in Keras Github in our reference. The final architecture retained can be described as follows:

Convolution layers all have a depth of 32 (but it could also be a depth of 32 for the first layer and for instance a depth of 64 for the second in some cases). The max pooling layer being of size 2*2, it divides the height and width by two. The size of the convolution kernel in each layer is 3*3. The number of epochs is 10.

6. Result and Conclusion

In this section, we mainly compare the results accuracy of for four models, convolutional neural network, decision tree, random forest and SVM(multiclass). The results are shown in the graph: the highest accuracy comes from convolutional neural network, which is 51%.

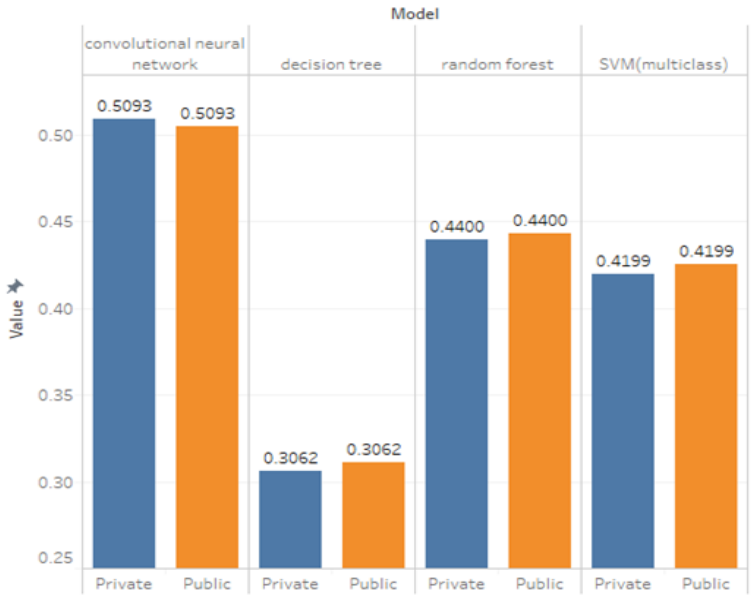

Figure 4. Model accuracy comparison

Comparing with the competition results posted on Kaggle, in the Public Leaderboard, where people use public dataset as test data to test the model, our highest accuracy of 0.5093 could rank at 25th of all the teams. In the private Leaderboard, where they use private data to test the winner, our accuracy could rank as 24th of all the teams.

Still, we have to consider the overfitting problem for the highest accuracy model. To test the degree of overfitting, we could make a matrix using each row to represent the percentage of correctly classified emotion and false positive emotions. By looking at that matrix, we can get an insight of which emotions are the most similar and hardest to distinguish. Then using that result, we can design specific classification methods for those emotions to improve accuracy, thus make this model usable in face of more data.

7. References

[1] https://keras.io/layers/core/
[2] http://ankivil.com/mnist-database-and-simple-classification-networks/
[3] http://cs231n.github.io/convolutional-networks/#conv
[4]https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge/discussion
[5] Tang, Yichuan. "Deep learning using linear support vector machines." Workshop on Challenges in Representation Learning, ICML. 2013.