INF552: Programming Assignment 5

Student: Jiashi Chen USCID: 4684194123

Part 1: Implementation

The code is in ***Jiashi_Chen_Homework_5_Code.ipynb*** file.

Output (Since model initializes the weight randomly, the output may be a bit different each time):
Accuracy of test data: 89%
Predictions of test data:

IMG	PRED	IMG	PRED	IMG	PRED
gestures/A/A_down_1.pgm	1	gestures/D/D_stop_1.pgm	0	gestures/H/H_stop_5.pgm	0
gestures/A/A_down_2.pgm	1	gestures/D/D_stop_2.pgm	0	gestures/H/H_stop_6.pgm	0
gestures/A/A_hold_1.pgm	0	gestures/D/D_up_1.pgm	0	gestures/H/H_up_5.pgm	0
gestures/A/A_hold_10.pgm	0	gestures/D/D_up_3.pgm	0	gestures/I/I_hold_2.pgm	0
gestures/A/A_stop_1.pgm	0	gestures/E/E_down_1.pgm	1	gestures/I/I_hold_5.pgm	0
gestures/A/A_stop_4.pgm	0	gestures/E/E_hold_1.pgm	0	gestures/I/I_down_3.pgm	1
gestures/A/A_up_1.pgm	0	gestures/E/E_hold_5.pgm	0	gestures/I/I_stop_5.pgm	0
gestures/A/A_up_10.pgm	0	gestures/E/E_stop_1.pgm	0	gestures/I/I_stop_6.pgm	0
gestures/B/B_down_1.pgm	1	gestures/E/E_stop_2.pgm	0	gestures/I/I_up_2.pgm	0
gestures/B/B_down_2.pgm	0	gestures/E/E_up_1.pgm	0	gestures/I/I_up_3.pgm	1
gestures/B/B_hold_1.pgm	0	gestures/E/E_up_2.pgm	1	gestures/J/J_down_5.pgm	1
gestures/B/B_hold_2.pgm	0	gestures/F/F_down_1.pgm	0	gestures/J/J_down_6.pgm	0
gestures/B/B_stop_1.pgm	0	gestures/F/F_down_4.pgm	1	gestures/J/J_hold_2.pgm	0
gestures/B/B_stop_2.pgm	0	gestures/F/F_hold_1.pgm	0	gestures/J/J_hold_3.pgm	0
gestures/B/B_up_1.pgm	0	gestures/F/F_hold_2.pgm	0	gestures/J/J_stop_7.pgm	0
gestures/B/B_up_4.pgm	0	gestures/F/F_stop_2.pgm	0	gestures/J/J_stop_8.pgm	0
gestures/C/C_down_1.pgm	1	gestures/F/F_stop_5.pgm	0	gestures/J/J_up_1.pgm	0
gestures/C/C_down_2.pgm	1	gestures/G/G_down_2.pgm	1	gestures/J/J_up_2.pgm	0
gestures/C/C_hold_1.pgm	0	gestures/G/G_down_3.pgm	1	gestures/K/K_down_2.pgm	1
gestures/C/C_hold_2.pgm	0	gestures/G/G_hold_4.pgm	0	gestures/K/K_down_3.pgm	1
gestures/C/C_stop_2.pgm	0	gestures/G/G_stop_2.pgm	0	gestures/K/K_hold_1.pgm	0
gestures/C/C_stop_3.pgm	0	gestures/G/G_stop_5.pgm	0	gestures/K/K_hold_2.pgm	0

gestures/C/C_up_1.pgm	0	gestures/G/G_up_2.pgm	1	gestures/K/K_hold_3.pgm	0
gestures/D/D_down_1.pgm	0	gestures/G/G_up_5.pgm	0	gestures/K/K_stop_1.pgm	0
gestures/D/D_down_2.pgm	0	gestures/H/H_down_2.pgm	0	gestures/K/K_stop_2.pgm	0
gestures/D/D_hold_1.pgm	0	gestures/H/H_hold_10.pgm	0	gestures/K/K_stop_1.pgm	0
gestures/D/D_hold_2.pgm	0	gestures/H/H_hold_2.pgm	0	gestures/K/K_stop_2.pgm	0
gestures/D/D_hold_6.pgm	0	gestures/H/H_hold_5.pgm	0		

The **data structure** I use is *matrix* from numpy module and *dataframe* from pandas module. Dataframe could help me store the orginal data. I use the matrix to store the gray scale of the images, classification label and weight, which makes it easier for me to do matrix operations.

Any code-level **optimizations** I perform is that I try to use some vector and matrix operations to get the values at once when acculating variable *x* from variable *s*. This will be much faster than calculating the value one by one for all perceptron node. I also encapsulate each step of calculation into a function, which is good for my maintenance and finding bug.

The **challenges** I face is that how to make the code work more efficiently. At first, I use for loop to calculate the value, which is inefficent. Now I use matrix operations to solve this problem. I pay a lot of efforts to find out the difference between the operation of array in numpy and the operation of matrix in pandas when it comes to the same operator. And when I acculate the delta, some of matrix needs to do dot mulitiply while others needs to do mulitiply like matrix operation, which also confused me for a while.

Part 2: Software Familiarization

MLPClassifier in Sklearn package offers a good implementation of the Back Propagation algorithm for Feed Forward Neural Networks.

How to use

I used this package to fit the data in ***Jiashi_Chen_Homework_5_Code.ipynb*** file too.

API for Multi-layer Perceptron classifier.(https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html#sklearn.neural_network.MLPClassifier)

```
from sklearn.neural_network import MLPClassifier  ##
Obtain necessary package

mlp = MLPClassifier(solver='sgd', alpha=0,
hidden_layer_sizes=(100,), activation='logistic',
learning_rate_init=0.1, max_iter=1000)  #Get classifier.
mlp.fit(X, y)  #Fit the model
mlp.predict(X_test)  #Predicted label based on trained
Neural Network
```

Comparsion and Improvement

After I read the API document of Sklearn, I find it provides various paramteres in order to fine tune the performance of the neural network. And I could try to add some of them in my code. For example, I am only required to use sigmoid function as activation function in this assignment. I could also add tanh function which I also learned from class into my activation function list. And about the learning rate, I keep it as 0.1 in my code so far. While we can choose "constant", "invscaling", "adaptive" in Sklearn module which can improve the efficiency a lot. Last but not the least, I could also add option to run L2 penalty like what the sklearn does just in case the data scale is very different.

Part 3: Applications

Neural Networks in Music[1]

The investigation of neural information structures in music is a rather new, exciting research area bringing together different disciplines such as computer science, musicology, mathematics and cognitive science. One of its objectives is to develop neural network models which are able to learn from music compositions structure elements characterizing the personal style of a composer. This has been a starting point of research activities for some institutes during the last years. The impression has resulted that artificial neural networks, when carefully developed, are sensitive, powerful, and innovative tools if capturing the specificity of a certain composer's style is intended.

Neural Networks in Face Detection[2]

This article present a neural network-based face detection system. A retinally connected neural network examines small windows of an image, and decides whether each window contains a face. The system arbitrates between multiple networks to improve performance over a single network. They use a bootstrap algorithm for training the networks, which adds false detections into the training set as training progresses. This eliminates the difficult task of manually selecting non-face training examples, which must be chosen to span the entire space of non-face images. Comparisons with other state-of-the-art face detection systems are presented; their system has better performance in terms of detection and false-positive rates.

References

[1] Dominik Hernel and Wolfram Menzel. Learning Musical Structure and Style with Neural Networks[J]. Computer Music Journal, 22(4):44-62.

[2] Henry A. Rowley, Shumeet Baluja, Takeo Kanade. Neural Network-Based Face Detection[C]// Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. IEEE Computer Society, 1996.