Minh Tran – HW5: Neural Network

1. Part 1: Implementation:

a. Neural network:

- Libraries:

Pycharm IDE is used with Python 3.6 as the programming language. Basic libraries such as numpy are used to facilitate the program

- Neural network structure:

✓ Type: Feed Forward Back Propagation Neural network

✓ 1 input layer (960 neurons corresponding to the number of pixels in a pgm image)

✓ 1 hidden layer (100 neurons)

✓ 1 output layer (1 neuron: 1 for "down gesture" or 0 for "not down gesture")

- Neural network details:

✓ Learning rate: 0.1

✓ Activation function: sigmoid

✓ Training epoch: 1000

✓ Weight initialization randomly: between -0.01 and 0.01

✓ Error function: standard squared error

- Data structure:

✓ The given file list is read by the function parse_file_list. Input includes a list file directory, either training data or test data. Output includes 2 lists: one has a list of files that are labeled 1 (down gestures) and another has a list of files that are labeled 0 (not down gestures)

✓ Input data is read by the function read_pgm_image. Input includes a pgm image and output includes a list of lists of values between 0 and 255.

- Files included:

✓ main.py - Contains the implementation of neural network from scratch

✓ mlnn.py: contains class "Model" to build neural network

- init__: bias b, weight W, choice of activation function, choice of output function
- Weight and bias are initialized as list of lists to contain values for all layers
- Function calculate_loss: perform forward propagation computation of neural network to calculate loss
- Function predict: perform forward propagation computation of neural network to calculate the predicted class
- Function train: implement batch gradient descent using backpropagation.

✓ output.py: contains softmax and least square error implementation. Each class contains predict, loss and diff function to perform needed functionalities

✓ Layer.py: contains activation function "tanh" and "sigmoid". Each class has backward and forward computations

✓ Folder named "gestures": contain all the images in subfolders with labeled names

✓ Downgesture_test.list: txt file containing names of all images for testing

✓ Downgesture_train.list: txt file containing names of all images for training

- Code-level optimization:

✓ The last layer of the network is modified from least square error to softmax to allow multiclass prediction

✓ Regularization parameter lambda is added to the weight calculation to avoid overfitting

✓ Another popular activation function "tanh" is included in the output.py file

- Challenges:

✓ The neural network can be improved with minibatch stochastic gradient descent. Each epoch does not have to scan the whole training data

✓ Input can be normalized if needed

✓ L2 and dropout regularization can be added

✓ Other activation functions such as ReLU, Linear can be added

-
- Results:

```
C:\ProgramData\Anaconda3\python.exe C:/INF552/Assignment/hw5_neural_network/main.py
start reading input text files
start reading input pictures files
start shuffling and splitting data
Loss after iteration 0: 0.243746
Loss after iteration 100: 0.162069
Loss after iteration 200: 0.126770
Loss after iteration 300: 0.097500
Loss after iteration 400: 0.074038
Loss after iteration 500: 0.056187
Loss after iteration 600: 0.042197
Loss after iteration 700: 0.035505
Loss after iteration 800: 0.031065
Loss after iteration 900: 0.023958
training accuracy:
1.0
testing accuracy:
0.9397590361445783
```

The training accuracy is **1** while the testing accuracy is **0.94**

2. Part 2: Software familiarization

a. Sklearn for Neural network:

✓ Scikit-Learn offers useful built-in packages to implement neural network with the ability to tune different parameters

✓ 3 simple steps are needed:

- Import data
- Assign labels based on given name
- Create model using the module MLPClassifier
- Fit the model by providing the images and labels
- Calculate the accuracy rate on test data

✓ Results

- Run file neuralNetworkSK_MT.py
- The testing accuracy is 0.87

3. Applications

Neural network is capable of modeling complex and nonlinear processes. ANN can also generalize, learning from initial inputs and their relationships, then inferring unseen association. ANN does not impose or assume any restriction on input variables.

Therefore, they have become an extremely powerful tool to solve problems ranging from classification, clustering, regression, pattern recognition (national security assessment), dimension reduction, prediction (sales forecast), translation, natural language processing, anomaly detection (bank fraud detection) and computer vision.