Programming Assignment 5

Project Report by Arpit Parwal & Yeon-soo Park

1. Implementation Algorithms

a. Data structures

i. Neural Network

Our Neural networks consist of the following components.

1) An input layer, X (List)
2) 1 hidden layer of size 100
3) Output layer, Y (List)
4) A set of weights and biases between each layer (Input - hidden/hidden - Output) Written as **W** and **b** (np.array)
5) Activation function for each hidden layer. Here, we use the Sigmoid activation function as suggested.

We used a **Numpy array** to store data points and to initialize the initial weights and biases. And we perform a dot product to multiply the input by asset weights and to compute every backpropagation.

b. Explanation of Modules

i. Neural Network

- **Main Function**

  This main function reads the dataset from the given train/test. list file in NumPy array. Then, we get the number of inputs from the dataset and pass this value to the constructor of the Neural Network class. In the class, we create a construct function def __init__ to initialize the weights vector W including threshold **b and learning rate.** We set the size of W as the number of inputs + 1 (bias **b**). We set the learning rate as 0.1 and epoch as 1000 which was given.

- **Train Model**
  - **Forward**

    This model takes input values **data points** as an argument and performs the weighted aggregation of **data points**
    (dot product between **w.x = w * x**) and call evaluate function that returns the value as a prediction. Evaluate function takes input values **x** as an argument and for every observation in **x** and returns the result between 0 and 1. Next, We update weight based on the result.

- **Backward**

  *** Weight Update Formula [W1 = W0 + L(O-Y)*X] ***
  (w1= weight updated, w0 = current weight, L = learning rate,
  O = label, Y = actual output, X = input)

  Here, we set L as 0.1. O(label) is given from dataset (0 or 1(down)) and Y is the result out from the model (0 or 1 from the activation function)

  Sigmoid Activation Function is $\theta(s) = 1/(1+e-s)$.
  The derivative of Sigmoid Function is $\theta'(s) = \theta(s)(1-\theta(s))$.

- **Predict Function**
  Each data point is evaluated exactly once in every iteration to evaluate the trained model performance.

- **Print Output Function**
  This prints the result of weight and accuracy after the final iteration.

c. Challenges

i. Neural Network

- **Evaluating model performance**
  Since it is difficult to find the most appropriate functional form which has direct implications on accuracy, we have to discover optimal weights, learning rate, and epochs via an empirical optimization procedure by tuning the neural network. However, the learning rate and epochs are already given in this assignment, so it's very hard to figure out the optimal parameters and evaluate our model performance.
- **Computation and maintaining dimensions**
  Since Deep Learning libraries such as Tensorflow and Keras make it easy to build neural networks without fully understanding the inner workings, It is required to get a deeper understanding of Neural Networks to implement it from scratch.

d. Code-level optimization

i. Neural Network

- To reduce computational load, we use the NumPy array to calculate dot products between weights and input data.
- Define classes for both models and functions to reduce the size of the program and repetitive tasks.
- Using Numpy Array for matrix calculation and as a comprehension over explicit for loops in certain places, making code more compact and execution faster.
- In the training procedure, we pick up the data randomly rather than sequentially from the data set in every training epoch. Because we found that if putting in the data sets in sequence and repeating, the neural network would be less well-trained so that the accuracy of prediction would be lower

e. Output

i. Neural Network

```
++++  PREDICTIONS   ++++

No.1: A/A_down_1.pgm Predict=True, Output value=[1.]==> Match(Y)
No.2: A/A_down_2.pgm Predict=True, Output value=[1.]==> Match(Y)
No.3: A/A_hold_1.pgm Predict=False, Output value=[0.]==> Match(Y)
No.4: A/A_hold_10.pgm Predict=False, Output value=[0.]==> Match(Y)
No.5: A/A_stop_1.pgm Predict=False, Output value=[0.]==> Match(Y)
No.6: A/A_stop_4.pgm Predict=False, Output value=[0.]==> Match(Y)
No.7: A/A_up_1.pgm Predict=False, Output value=[0.]==> Match(Y)
No.8: A/A_up_10.pgm Predict=False, Output value=[0.]==> Match(Y)
No.9: B/B_down_1.pgm Predict=False, Output value=[0.]==> Match(N)
No.10: B/B_down_2.pgm Predict=False, Output value=[0.]==> Match(N)
No.11: B/B_hold_1.pgm Predict=False, Output value=[0.]==> Match(Y)
No.12: B/B_hold_2.pgm Predict=False, Output value=[0.]==> Match(Y)
No.13: B/B_stop_1.pgm Predict=False, Output value=[0.]==> Match(Y)
No.14: B/B_stop_2.pgm Predict=False, Output value=[0.]==> Match(Y)
No.15: B/B_up_1.pgm Predict=False, Output value=[0.]==> Match(Y)
No.16: B/B_up_4.pgm Predict=False, Output value=[0.]==> Match(Y)
No.17: C/C_down_1.pgm Predict=False, Output value=[0.]==> Match(N)
No.18: C/C_down_2.pgm Predict=False, Output value=[0.]==> Match(N)
No.19: C/C_hold_1.pgm Predict=False, Output value=[0.]==> Match(Y)
No.20: C/C_hold_2.pgm Predict=False, Output value=[0.]==> Match(Y)
No.21: C/C_stop_2.pgm Predict=False, Output value=[0.]==> Match(Y)
No.22: C/C_stop_3.pgm Predict=False, Output value=[0.]==> Match(Y)
No.23: C/C_up_1.pgm Predict=False, Output value=[0.]==> Match(Y)
No.24: D/D_down_1.pgm Predict=False, Output value=[0.]==> Match(N)
No.25: D/D_down_2.pgm Predict=False, Output value=[0.]==> Match(N)
No.26: D/D_hold_1.pgm Predict=False, Output value=[0.]==> Match(Y)
No.27: D/D_hold_2.pgm Predict=False, Output value=[0.]==> Match(Y)
No.28: D/D_hold_6.pgm Predict=False, Output value=[0.]==> Match(Y)
No.29: D/D_stop_1.pgm Predict=False, Output value=[0.]==> Match(Y)
No.30: D/D_stop_2.pgm Predict=False, Output value=[0.]==> Match(Y)
No.31: D/D_up_1.pgm Predict=False, Output value=[0.]==> Match(Y)
No.32: D/D_up_3.pgm Predict=False, Output value=[0.]==> Match(Y)
No.33: E/E_down_1.pgm Predict=False, Output value=[0.]==> Match(N)
No.34: E/E_hold_1.pgm Predict=False, Output value=[0.]==> Match(Y)
```

```
No.35: E/E_hold_5.pgm Predict=False, Output value=[0.]==> Match(Y)
No.36: E/E_stop_1.pgm Predict=False, Output value=[0.]==> Match(Y)
No.37: E/E_stop_2.pgm Predict=False, Output value=[0.]==> Match(Y)
No.38: E/E_up_1.pgm Predict=False, Output value=[0.]==> Match(Y)
No.39: E/E_up_2.pgm Predict=False, Output value=[0.]==> Match(Y)
No.40: F/F_down_1.pgm Predict=False, Output value=[0.]==> Match(N)
No.41: F/F_down_4.pgm Predict=False, Output value=[0.]==> Match(N)
No.42: F/F_hold_1.pgm Predict=False, Output value=[0.]==> Match(Y)
No.43: F/F_hold_2.pgm Predict=False, Output value=[0.]==> Match(Y)
No.44: F/F_stop_2.pgm Predict=False, Output value=[0.]==> Match(Y)
No.45: F/F_stop_5.pgm Predict=False, Output value=[0.]==> Match(Y)
No.46: G/G_down_2.pgm Predict=False, Output value=[0.]==> Match(N)
No.47: G/G_down_3.pgm Predict=False, Output value=[0.]==> Match(N)
No.48: G/G_hold_4.pgm Predict=False, Output value=[0.]==> Match(Y)
No.49: G/G_stop_2.pgm Predict=False, Output value=[0.]==> Match(Y)
No.50: G/G_stop_5.pgm Predict=False, Output value=[0.]==> Match(Y)
No.51: G/G_up_2.pgm Predict=False, Output value=[0.]==> Match(Y)
No.52: G/G_up_5.pgm Predict=False, Output value=[0.]==> Match(Y)
No.53: H/H_down_2.pgm Predict=False, Output value=[0.]==> Match(N)
No.54: H/H_hold_10.pgm Predict=False, Output value=[0.]==> Match(Y)
No.55: H/H_hold_2.pgm Predict=False, Output value=[0.]==> Match(Y)
No.56: H/H_hold_5.pgm Predict=False, Output value=[0.]==> Match(Y)
No.57: H/H_stop_5.pgm Predict=False, Output value=[0.]==> Match(Y)
No.58: H/H_stop_6.pgm Predict=False, Output value=[0.]==> Match(Y)
No.59: H/H_up_5.pgm Predict=False, Output value=[0.]==> Match(Y)
No.60: I/I_hold_2.pgm Predict=False, Output value=[0.]==> Match(Y)
No.61: I/I_hold_5.pgm Predict=False, Output value=[0.]==> Match(Y)
No.62: I/I_down_3.pgm Predict=True, Output value=[1.]==> Match(Y)
No.63: I/I_stop_5.pgm Predict=False, Output value=[0.]==> Match(Y)
No.64: I/I_stop_6.pgm Predict=False, Output value=[0.]==> Match(Y)
No.65: I/I_up_2.pgm Predict=False, Output value=[0.]==> Match(Y)
No.66: I/I_up_3.pgm Predict=False, Output value=[0.]==> Match(Y)
No.67: J/J_down_5.pgm Predict=True, Output value=[1.]==> Match(Y)
No.68: J/J_down_6.pgm Predict=False, Output value=[0.]==> Match(N)
No.69: J/J_hold_2.pgm Predict=False, Output value=[0.]==> Match(Y)
No.70: J/J_hold_3.pgm Predict=False, Output value=[0.]==> Match(Y)
No.71: J/J_stop_7.pgm Predict=False, Output value=[0.]==> Match(Y)
No.72: J/J_stop_8.pgm Predict=False, Output value=[0.]==> Match(Y)
No.73: J/J_up_1.pgm Predict=False, Output value=[0.]==> Match(Y)
```

```
No.74: J/J_up_2.pgm Predict=False, Output value=[0.]==> Match(Y)
No.75: K/K_down_2.pgm Predict=True, Output value=[1.]==> Match(Y)
No.76: K/K_down_3.pgm Predict=True, Output value=[1.]==> Match(Y)
No.77: K/K_hold_1.pgm Predict=False, Output value=[0.]==> Match(Y)
No.78: K/K_hold_2.pgm Predict=False, Output value=[0.]==> Match(Y)
No.79: K/K_hold_3.pgm Predict=False, Output value=[0.]==> Match(Y)
No.80: K/K_stop_1.pgm Predict=False, Output value=[0.]==> Match(Y)
No.81: K/K_stop_2.pgm Predict=False, Output value=[0.]==> Match(Y)
No.82: K/K_stop_1.pgm Predict=False, Output value=[0.]==> Match(Y)
No.83: K/K_stop_2.pgm Predict=False, Output value=[0.]==> Match(Y)

++++++++


++++  SUMMARY   ++++
Total correct predictions: 70
Total size of data: 83
Accuracy: 84.34%
```

3. Software Familiarization

a. Existing Libraries

i. **Neural Networks**

- **Pytorch**
  For a huge dataset, we can use a parallel data library to wrap modules. It will be parallelized over the batch dimension so that we can leverage multiple GPUs easily. We can customize the parameters; choose the number of layers, learning rate, activation functions, and batch size.
- **Tensorflow**
  This network consists of multiple layers of neurons, the first of which takes the input but TensorFlow does not support inline matrix operation so every time has to copy a matrix and it is slow compared to other libraries.

b. Comparisons

i. **Neural Networks**

- **Improve the random point selection.**
  We can improve our code to automatically generate different weight vectors and learning rates to get the best result in different variables.
- **Different activation function**
  Implementation of tanh function to make the curve steeper and to get more optimized result

4. Applications

a. **Character Recognition**
Character recognition is one of the most popular research topics in computer vision and image processing. Automatic recognition of characters is a very complex task due to viewpoint variations, occlusions, background interference, movement variability of the same action and ambiguity between different actions. Character recognition like handwriting has a lot of applications in fraud detection.

b. **Forecasting**
We can overcome traditional forecasting models in terms of taking into account these complex relationships. Forecasting is required every day for business decisions so Neural networks can provide robust automatic models to predict it.

5. Individual Contribution

Student Name	Programming Part	Documentation
Arpit Parwal **aparwal@usc.edu**	• Backpropagation • Model Training	The data structure, Modules, Challenges, improvements,
Yeon-soo Park **yeonsoop@usc.edu**	• Getting data from images to the array. • Running prediction	Comparison, improvements, Code-level optimization, Applications