INF552 Homework 5

Group members and contribution :
Krishna Akhil Maddali (Code development and report)
Yash Shahapurkar (Code development, research, report)
Myungjin Lee (research and report)

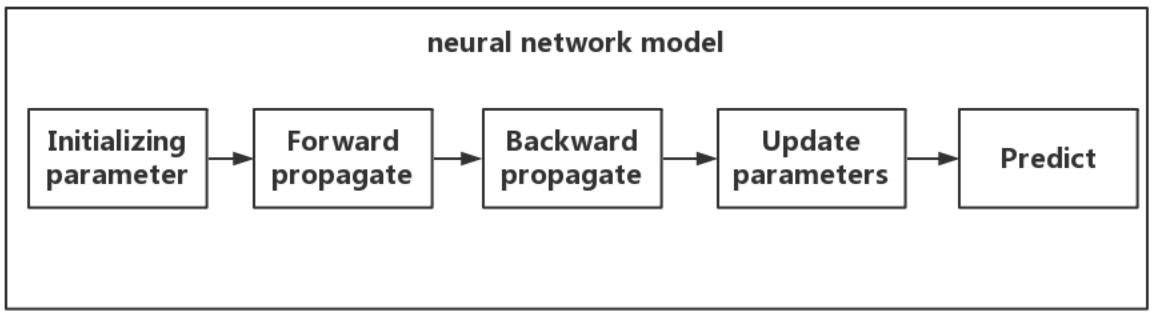
Part 1: Implementation

- ### Output after running the program

  Predictions [1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0]

  Training Accuracy 100%
  Testing Accuracy 89.1566265060241%

- ### Data structures
  - Arrays to store weights between layers and outputs of each layer.
  - Dictionary to store parameters.



- ## Code-level optimization
  - Use of arrays for vectorised implementation to calculate faster.
  - Using dot product instead of explicitly multiplying different values
  - After sorting out required columns, we create arrays for the features and target.
  - We initialize weights in a list which is converted to an array. We update this same array for new weights.
  - Initialize lists for iteration count and number of misclassifications. Append the value to the list in every iteration
  - Store predicted outputs in an array to compare with the targets

- ## Challenges
  - Load data : Since the given data set is in pgm type of file, we had to convert it to normal text file to store the data as an array. To deal with pgm file is not familiar with us, we have spent some time for searching and converting the data.

Part 2: Software Familiarization

Library : python Keras deep learning library

Prediction: [1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]

Training Accuracy 100%
Testing Accuracy 98.80%

The accuracy from the library is higher than our implemented code.
Also the library trains faster compared to the 2 mins of our implementation.

How to improve our code

- Accuracy : As the accuracy through library shows, our implementation has low accuracy predicting the output from the test data. In order to improve our code, we can modify the sigmoid function and increase the number of step size for enough convergence.
- Speed of the code : The performance of the code is a little bit slower than one by the library. Instead of having array, we could load data through data frame by pandas which is much faster than numpy array.

Part 3: Applications

1. **Image Processing and Character recognition**

Character recognition like handwriting has a lot of applications such as fraud detection and national security assessments. Image recognition is an ever-growing field with widespread applications from facial recognition in social media to satellite imagery processing for agricultural and defense usage.

2. **Forecasting**

Forecasting is required extensively in everyday business decisions in economic and stock market. More often, forecasting problems are complex, for example, predicting stock prices is a complex problem with a lot of underlying factors. ANNs, applied in the right way, can provide robust alternative, given its ability to model and extract unseen features and relationships. Also, unlike these traditional models, ANN doesn't impose any restriction on input and residual distributions.