I. Introduction:

In Part A, we are required to design a three-layer feedforward neuron network, and to explore the optimal parameters including the training batch-size, number of hidden neurons and value of decay (learning rate). We further compare the performance between the four-layer network and the three-layer network.

In the following paragraphs, Section 2 introduces the design of the three-layer network and the validation and testing method for this specific application. Section 3 discusses the optimal value of batch-size for minibatch training approach. Section 4 explores and discusses the optimal number of neurons in the hidden layer. Section 5 tests the selection of decay value for this model. Last but not least, Section 6 compares performance between the four-layer network with fixed parameters, and the optimal three-layer model determined by the previous sections.

II. Implementation of a three-layer network for classification

We have built a three-layer feedforward network using the TensorFlow package in Python. The network includes an input layer, a hidden layer, and an output layer. The input layer takes in one, or a batch of vectors with 36 elements, indicating that the number of features (*NUM\_FEATURES*) for each record equals to 36. The hidden layer consists of *n* neurons, where the value of *n* (*NUM\_HIDDEN*) is initiated as 10. The output layers returns one or a batch of vector with 6 elements within range [0, 1], indicating the probability that this record falls in the six categories respectively. The output factor determines the classification result, which is the corresponding class of the greatest probability among these 6 values.

The model is represented by TensorFlow tensors. The input layer is implemented as a 2-dimensional tensor X with the shape of ( ,*NUM\_FEATURES*). The hidden layer is implemented as a Weight Matrix W having the shape of (*NUM\_FEATURES*, *NUM\_HIDDEN*), and a Bias Vector b that has a shape of (*NUM\_HIDDEN*, 1). The synaptic input is calculated as X×W+b, which is a matrix as (*NUM\_HIDDEN*, 1). Since the hidden layer is designed as a perceptron, the synaptic input is then passed into the sigmoid function and is outputted as h.

Similarly, the output layer is implemented as a Weight Matrix V with the shape of (*NUM\_HIDDEN*, *NUM\_CLASSES*), and a Bias Vector c that has a shape of (*NUM\_CLASSES*, 1). The synaptic input is calculated as h×V+b, and then passed into the sigmoid function to transfer from the range of real number into the range of [0, 1], representing the probability of each class and is outputted as y.

During the evaluation procedure, the cost function is then calculated as the sum of the cross entropy and the L2 regularization loss, and the accuracy is determined by the ratio of the accurate prediction. The optimizing and back-propagation procedure are handled by the build-in Gradient Descent Optimizer (*tf.train.GradientDescentOptimizer*) to minimize the cost function.

The parameters are initialized as the following:

* *NUM\_FEATURES* = 36
* *NUM\_CLASSES* = 6
* *NUM\_HIDDEN* = 10
* *LEARNING\_RATE* = 0.01
* *EPOCHS* = 5000
* *BATCH\_SIZE* = 32
* *BETA* = pow(10, -6)

To evaluate the performance during the training procedure, the training accuracy and test accuracy are calculated for every 100 epochs. The model will be saved if the test accuracy is improved comparing to the best accuracy in the previous testing.

III. Determine the Optimal Batch Size for the Three-Layer Network

We searched for the optimal batch size (*BATCH\_SIZE*) for the neural network within the search space as S={4, 8, 16, 32, 64} for the Mini-Batch Gradient Descent Model. Instead of feeding in the whole chunk of data into the model, the training data are divided into smaller batches, each of which has a fixed number (*BATCH\_SIZE*) of records. The model will be updated after each batch. Therefore, the smaller the *BATCH\_SIZE* is, the more updates will be needed and hence the longer the training will be. If the *BATCH\_SIZE* equals to 1, the training approach is considered as a Stochastic Gradient Descent (SGD) Model. On the other hand, if the *BATCH\_SIZE* equals to the length of the input data, the model is called as a Batch Gradient Descent (GD) Model. For any number of *BATCH\_SIZE* in between, the model is categorized as a Mini-Batch Gradient Descent Model.

The training errors and test accuracies are recorded as the following graphs show.

|  |  |
| --- | --- |
|  |  |
| Training Error for Batch Number = 4 | Test Accuracy for Batch Number = 4 |
|  | |
|  |  |
| Training Error for Batch Number = 8 | Test Accuracy for Batch Number = 8 |
|  | |
|  |  |
| Training Error for Batch Number = 16 | Test Accuracy for Batch Number = 16 |
|  | |
|  |  |
| Training Error for Batch Number = 32 | Test Accuracy for Batch Number = 32 |
|  | |
|  |  |
| Training Error for Batch Number = 64 | Test Accuracy for Batch Number = 64 |

For different batch sizes, the time taken to train the network for one epoch are plotted as the following.

|  |  |
| --- | --- |
|  |  |
| Training Time for Batch Number = 4 (Scatter) | Training Time for Batch Number = 4 (Box Plot) |
|  | |
|  |  |
| Training Time for Batch Number = 8 (Scatter) | Training Time for Batch Number = 8 (Box Plot) |
|  | |
|  |  |
| Training Time for Batch Number = 16 (Scatter) | Training Time for Batch Number = 16 (Box Plot) |
|  | |
|  |  |
| Training Time for Batch Number = 32 (Scatter) | Training Time for Batch Number = 32 (Box Plot) |
|  | |
|  |  |
| Training Time for Batch Number = 64 (Scatter) | Training Time for Batch Number = 64 (Box Plot) |

The average training time for different batches are:

|  |  |
| --- | --- |
| Batch Size | Average Training Time per Batch (s) |
| 4 | 1.1366786398887634 |
| 8 | 0.6407565122127533 |
| 16 | 0.36728416357040405 |
| 32 | 0.19519758477211 |
| 64 | 0.10925680437088013 |

The best test accuracy for different batches and total training time are:

|  |  |  |
| --- | --- | --- |
| Batch Size | Best Test Accuracy | Total Training Time |
| 4 | 0.843 | 1:55:26 |
| 8 | 0.8325 | 0:58:27 |
| 16 | 0.8235 | 0:30:10 |
| 32 | 0.811 | 0:16:04 |
| 64 | 0.7965 | 0:08:47 |

Considering the Best Tech Accuracy, the model with Batch Size 4 performs the best. The training time is inversely proportional to the batch size. Moreover, for most models, the accuracy converges after 2000 epochs. Therefore, we will adopt the *BATCH\_SIZE* to be 4, and *EPOCHS* to be 2000 in the following part.

IV. Determine the Optimal Number of Hidden Neurons

We searched for the optimal number of hidden neurons (*NUM\_HIDDEN*) for the neural network within the search space as S={5 ,10, 15, 20, 25}. The models are built upon the setting described in the previous section

The parameters are initialized as the following:

* *NUM\_FEATURES* = 36
* *NUM\_CLASSES* = 6
* *LEARNING\_RATE* = 0.01
* *EPOCHS* = 2000
* *BATCH\_SIZE* = 4
* *BETA* = pow(10, -6)

The training errors and test accuracies are recorded as the Figure 3 shows.

|  |  |
| --- | --- |
|  |  |
| Training Error for # Hidden Neuron = 5 | Test Accuracy for # Hidden Neuron = 5 |
|  | |
|  |  |
| Training Error for # Hidden Neuron = 10 | Test Accuracy for # Hidden Neuron = 10 |
|  | |
|  |  |
| Training Error for # Hidden Neuron = 15 | Test Accuracy for # Hidden Neuron = 15 |
|  | |
|  |  |
| Training Error for # Hidden Neuron = 20 | Test Accuracy for # Hidden Neuron = 20 |
|  | |
|  |  |
| Training Error for # Hidden Neuron = 25 | Test Accuracy for # Hidden Neuron = 25 |

For different batch sizes, the time taken to train the network for one epoch are plotted as in Figure 4.

|  |  |
| --- | --- |
|  |  |
| Training Time for # Hidden Neuron = 5 (Scatter) | Training Time for # Hidden Neuron = 5 (Box Plot) |
|  | |
|  |  |
| Training Time for # Hidden Neuron = 10 (Scatter) | Training Time for # Hidden Neuron = 10 (Box Plot) |
|  | |
|  |  |
| Training Time for # Hidden Neuron = 15 (Scatter) | Training Time for # Hidden Neuron = 15 (Box Plot) |
|  | |
|  |  |
| Training Time for # Hidden Neuron = 20 (Scatter) | Training Time for # Hidden Neuron = 20 (Box Plot) |
|  | |
|  |  |
| Training Time for # Hidden Neuron = 25 (Scatter) | Training Time for # Hidden Neuron = 25 (Box Plot) |

The average training time for different batches are:

|  |  |
| --- | --- |
| # Hidden Neuron | Average Training Time per Batch (s) |
| 5 | 1.1233651428222655 |
| 10 | 1.1224233508110046 |
| 15 | 1.130188461661339 |
| 20 | 1.1459117535352707 |
| 25 | 1.1415876120328903 |

The best test accuracy for different batches and total training time are:

|  |  |  |
| --- | --- | --- |
| # Hidden Neuron | Best Test Accuracy | Total Training Time |
| 5 | 0.8335 | 0:37:27 |
| 10 | 0.8415 | 0:37:26 |
| 15 | 0.8245 | 0:37:41 |
| 20 | 0.8325 | 0:38:13 |
| 25 | 0.8195 | 0:38:04 |

Considering the Best Tech Accuracy, the model with 10 Hidden Neurons performs the best. There is no significant difference in the training time among different number of hidden neurons.

V. Determine the Optimal Decay Parameter of the Hidden Layer

We searched for the optimal decay parameter of hidden neurons  (*BETA*) for the neural network within the search space as S={0 ,10^-3, 10^-6, 10^-9, 10^-12}. When *BETA* equals to 0, it is equivalent to disabling the regularization. The models are built upon the setting described in the previous section. In order to achieve the convergence of the model, we set the number of epochs (*EPOCHS*) to 3000 instead.

The parameters are initialized as the following:

* *NUM\_FEATURES* = 36
* *NUM\_CLASSES* = 6
* *NUM\_HIDDEN* = 10
* *LEARNING\_RATE* = 0.01
* *EPOCHS* = 3000
* *BATCH\_SIZE* = 4

The training errors and test accuracies are recorded as the Figure 5 shows.

|  |  |
| --- | --- |
| C:\Users\lzcai\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\C1D12285.tmp | C:\Users\lzcai\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\9BA8FBDB.tmp |
| Training Error for BETA = 0 | Test Accuracy for BETA = 0 |
|  | |
| C:\Users\lzcai\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\42125DA1.tmp | C:\Users\lzcai\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\2A171157.tmp |
| Training Error for BETA = | Test Accuracy for BETA = |
|  | |
| C:\Users\lzcai\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\AF971C7D.tmp | C:\Users\lzcai\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\AF702093.tmp |
| Training Error for BETA = | Test Accuracy for BETA = |
|  | |
| C:\Users\lzcai\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\71FBBB19.tmp | C:\Users\lzcai\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\E2FCE58F.tmp |
| Training Error for BETA = | Test Accuracy for BETA = |
|  | |
| C:\Users\lzcai\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\CBD45575.tmp | C:\Users\lzcai\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\1293DC4B.tmp |
| Training Error for BETA = | Test Accuracy for BETA = |

The best test accuracy for different batches and total training time are:

|  |  |  |
| --- | --- | --- |
| Decay Parameter | Best Test Accuracy | Total Training Time |
| 0 | 0.838 | 1:11:27 |
|  | 0.775 | 1:17:45 |
|  | 0.7705 | 1:14:46 |
|  | 0.8475 | 1:13:56 |
|  | 0.844 | 1:18:26 |

Considering the Best Tech Accuracy, the model with BETA = performs the best. There is no significant difference in the training time among different number of hidden neurons.

We discovered that the decay parameter may restrain the fitting of the model when it is comparative large. As the decay parameter tends to 0, the test accuracy tends to increase. This may imply that, instead of preventing overfitting, the decay parameter may hinder the correctness of the model when it is too large.

V. Comparison of the 4-layer Network with the 3-layer Network

We are going to compare the performance of the 4-layer network and the 3-layer network in the same application. The 4-layer network are initialized with the number pf hidden neurons (*NUM\_HIDDEN*) being 10 per layer, the batch size (*BATCH\_SIZE*) being 32, and the decay parameter being .

The training errors and test accuracies are recorded as the following graphs show.

|  |  |
| --- | --- |
|  |  |
| Training Error for 4-layer Network | Test Accuracy for 4-layer Network |

The best test accuracy for different batches and total training time are:

|  |  |
| --- | --- |
| Best Test Accuracy | Total Training Time |
| 0.7505 | 0:10:33 |

In comparison, the result of the 3-layer network with same parameter (*BATCH\_SIZE =* 32 records/batch, BETA = ) are as the following:

|  |  |
| --- | --- |
|  |  |
| Training Error for Batch Number = 32 | Test Accuracy for Batch Number = 32 |

Its best test accuracy and total training time:

|  |  |
| --- | --- |
| Best Test Accuracy | Total Training Time |
| 0.811 | 0:16:04 |

There is an essential boost of accuracy during the 1500-2000 epoch. However, comparing to the 3-layer network, the accuracy of the 4-layer network is significantly lower than the 3-layered network. This is partially due to the over excessive complexity of the model of the 4-layer network and hence it causes overfitting of the training data.