CZ4042 Neural Network

Project 1

Completed by:
Kyle Huang Junyuan (U1721717G)
Nelson Ko Ming Wei (U1721410B)

Introduction

Project 1 is split into two sections - Part A and Part B. It is attempted and completed by Kyle Huang Junyuan and Nelson Ko Ming Wei respectively.

Part A

Part A is a classification problem on the Cardiotocography dataset which contains measurements of fetal heart rate (FHR) and uterine contraction (UC) features. There are a total of 2126 cardiotocograph in this dataset with 21 input features and 1 output NSP label with values 1, 2 and 3.

With the cardiotocographs classified by three expert obstetricians, a consensus classification label is assigned with respect to morphologic pattern and fetal state. These classifications are **Normal (N)**, **Suspect (S)** and **Pathologic (P)**.

The aim of this section is to accurately predict the N, S and P class labels by training the neural network on the provided training dataset.

Part B

Part B is a regression problem on the Graduate Admissions Prediction dataset which contains several parameters such as the GRE score (out of 340), TOEFL score (out of 120), university rating (out of 5), strengths of Statement of Purpose and Letter of Recommendation (out of 5), undergraduate GPA (out of 10), research experience (either 0 or 1). These parameters are closely considered during the application for Master Programs.

The aim of this section is to accurately predict the final parameter which represents the chance of getting an admit (ranging from 0 to 1).

Methods

Part A

Pre-processing the Dataset

The dataset contains a total of 2126 samples. Before performing any training or testing, the dataset will first undergo the following steps:

1. Scaling the input features (x):

$$X_{scaled} = \frac{X - Xmin}{Xmax - Xmin}$$

```
def scale(X, X_min, X_max):
    return (X - X_min)/(X_max - X_min)
```

As the 21 input features have a varying spectrum of values, it is important to scale the input features according to the minimum and maximum value as seen in the dataset. This allows a standardised set of input attributes that are proportional to one another.

Unscaled input variables can result in a slow or unstable learning process. This is especially true because the weights used in the model will be tuned according to the loss function to achieve a better result. Any inconsistencies in the input variables can cause undesirable changes to the way the neural network learns.

2. One-hot matrix

N: $\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$ S: $\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$ P: $\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$

```
trainY = np.zeros((train_Y.shape[0], NUM_CLASSES))
trainY[np.arange(train_Y.shape[0]), train_Y-1] = 1
```

Since Part A represents a categorical problem, there needs to be a value that represents each category.

The original target values are represented in an integer encoding format such as 1, 2 and 3 which corresponds to N, S and P respectively. However, these values have an order to them which the neural network might wrongly learn these increments.

To solve this, a one-hot encoding is used so that no ordered relationship gets taken into account by the neural network.

3. Shuffling

```
s = np.arange(trainX.shape[0])      # Create an index array
np.random.shuffle(s)                # Shuffle the index array

trainX, trainY = trainX[s], trainY[s]
```

It is very crucial that the entire dataset is shuffled in the beginning. This is to avoid inducing any unwanted biases into the neural network when certain samples get trained into the neural network in an ordered way. Shuffling the dataset beforehand will ensure that the training process will be carried out fairly.

K-Fold Cross-validation

K-Fold Cross-validation is one of the many methods to help evaluate the skill of a machine learning model. It is a resampling procedure usually used in occasions where the provided dataset has a limited number of samples.

In this project, a 5-fold cross-validation is used. Therefore, the dataset will be divided into five parts.

The general procedure involves the following steps:

1. Shuffle the dataset
2. Split the dataset into 5 groups
3. For each of the 5 combinations
    a. Set a portion of it as the test dataset
    b. Train the model using the remaining dataset
    c. Evaluate the module using the test dataset
    d. Retain the evaluation scores and discard the model

Batch sizes

A batch size refers to the number of samples being processed before the model gets updated again.

For example, a batch size of 4 would mean that 4 samples will get processed by the model first before updating the weights and biases.

Regularisation

There are generally two types of regularisations - Lasso Regression (L1) and Ridge Regression (L2).

In this project, an L2 Regularisation is used for the loss function. It adds a "squared magnitude" of coefficient as a penalty term to the loss function.

Part B

Regression Problem
Pre-Processing the Dataset

For the dataset of the Admission Prediction problem, the input X data are scaled of the purpose for faster training and avoid being stuck in a local optima. Hence, it speeds up the convergence of gradient descent. The standardization formula below is used for this problem:

$$X_{Scaled} = \frac{X - \mu_X}{\sigma_X}$$

$$= \left(\frac{X_1 - \mu_{X_1}}{\sigma_{X_1}} \quad \frac{X_2 - \mu_{X_2}}{\sigma_{X_2}} \quad \dots \quad \frac{X_7 - \mu_{X_7}}{\sigma_{X_7}} \right)$$

The Data is split into Training and Test dataset at 70:30 ratio.

Model

A 3-layer neural net is built using the methods below:

1. All weights are initialized with truncated normal distribution
2. All biases are initialized to zero
3. ReLU activation function is used for the hidden layer
4. Mean-squared error is used as the loss function
5. L2 regularization is used to applied with the loss
6. Mini-batch gradient descent is used for training

Experiments and Results

Part A

Question 1

Design a feedforward neural network which consists of an input layer, one hidden layer of 10 neurons with ReLU activation function, and an output softmax layer. Assume a learning rate α = 0.01, L2 regularization with weight decay parameter β = 10−6, and batch size = 32. Use appropriate scaling of input features.

(a) **Use the training dataset to train the model and plot both accuracies on training and testing data against epochs.**

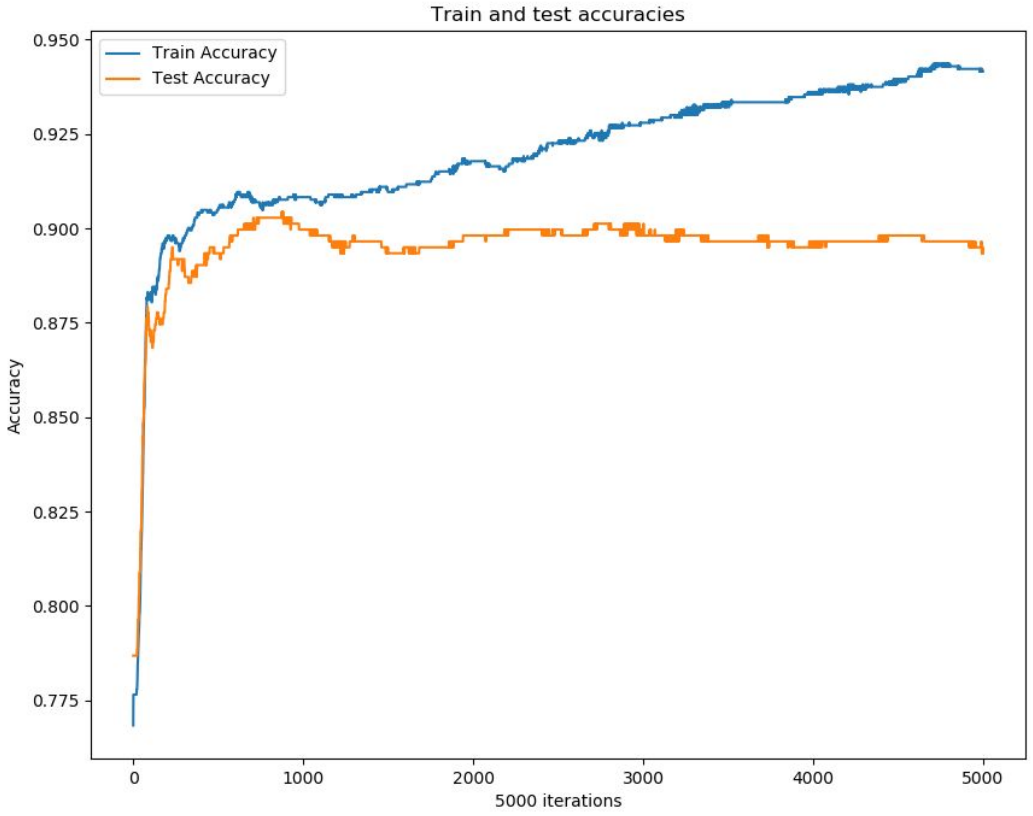

Figure A1: Train and test accuracies against 5000 iterations

In the above Figure A1, the train accuracy is seen to rise to a maximum of 94% as it approaches the 5000th iteration.

However, it is observed that even as train accuracy increases, the test accuracy does not increase along at the same rate. The test accuracy reaches its maximum value (approximately 90%) around the 1000th iteration. After that, it starts to decline to an accuracy of about 88%.

(b) **State the approximate number of epochs where the test error converges.**

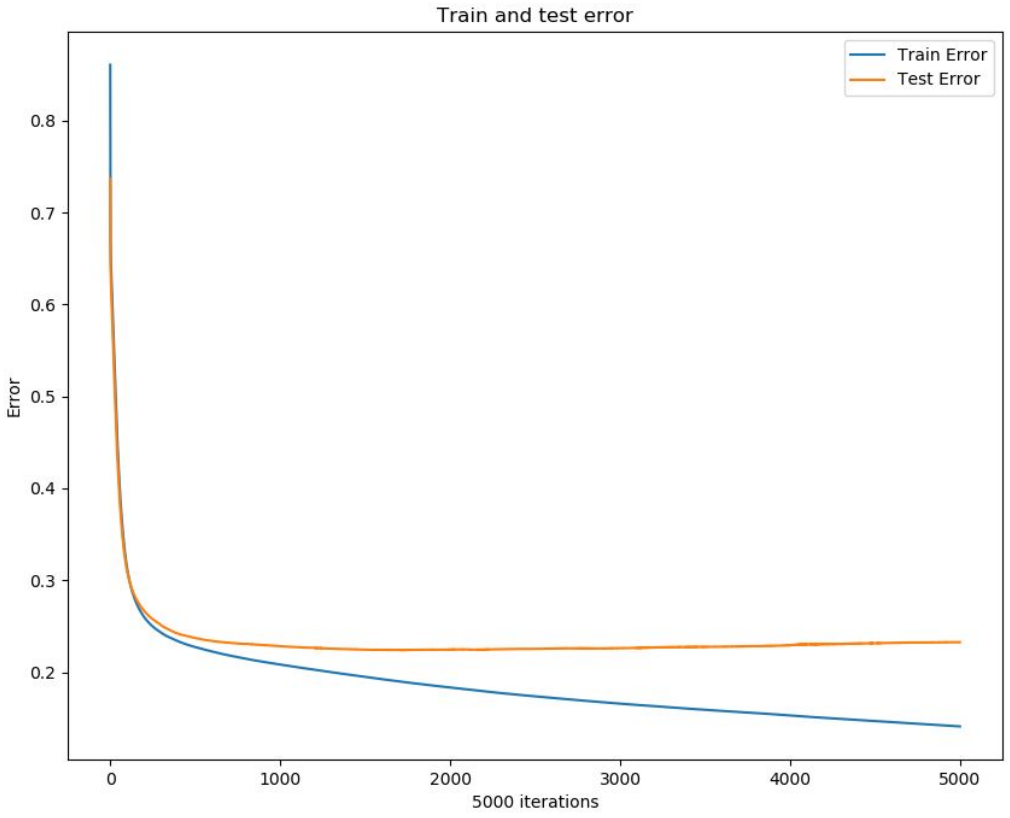

Figure A2: Test error against 5000 iterations

Based on the above Figure A2, the test error converges around the 1000th iteration. Beyond that, it is observed that the test error starts increases slightly. This behaviour corresponds to the above Figure A1 where the test accuracy stops increasing after the 1000th iteration.

The increase in error is most likely attributed to overfitting. An overfit model learns the training dataset too well. However, it does not perform well on a separate hold out sample such as the 30% test set that has been set aside in this case.

Question 2

Find the optimal batch size by training the neural network and evaluating the performances for different batch sizes.

(a) **Plot cross-validation accuracies against the number of epochs for different batch sizes. Limit search space to batch sizes {4, 8, 16, 32, 64}.**

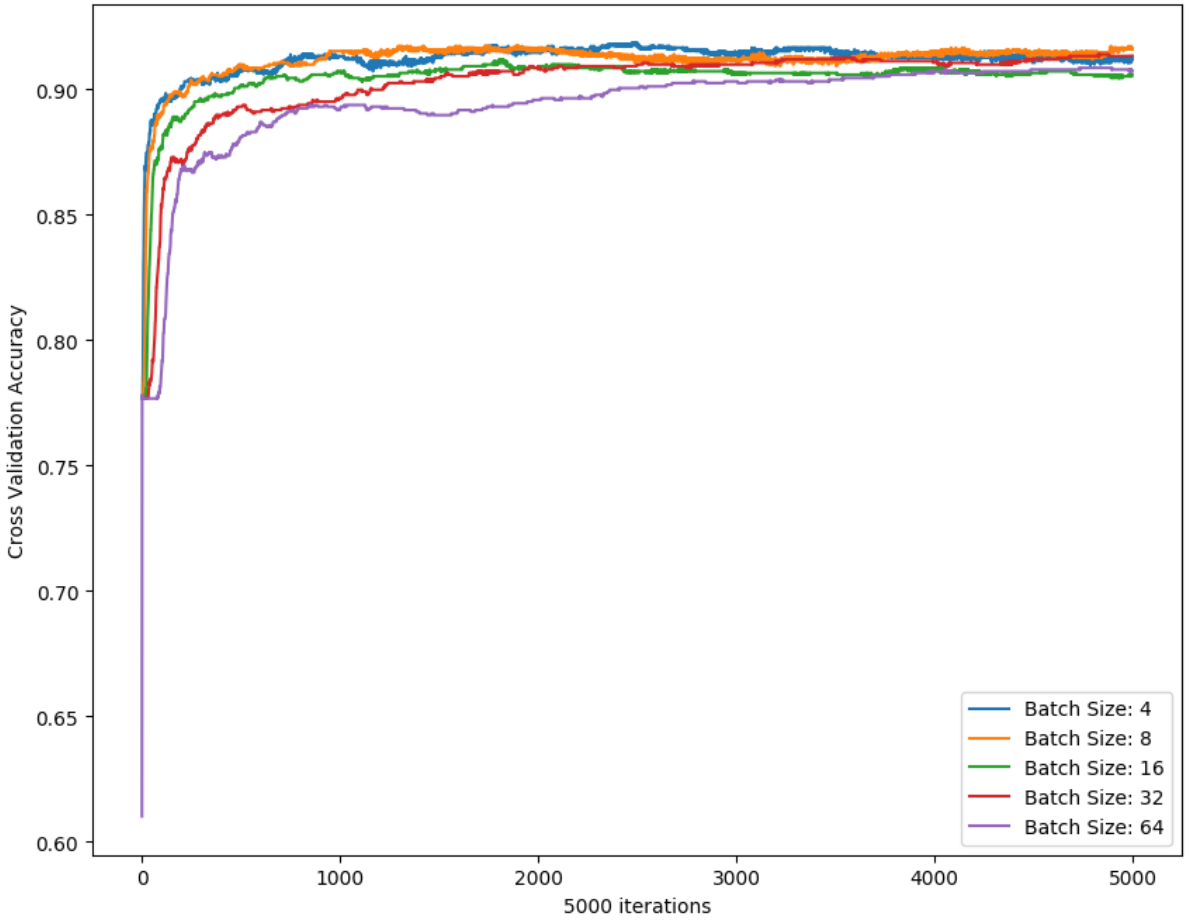

Figure A3: Cross-validation accuracies for different batch sizes

In the above Figure A3, the cross-validation accuracies are computed for the different batch sizes of 4, 8, 16, 32, 64. A batch size of 4 would mean that 4 samples will get processed by the model before updating the weights and biases.

It is observed that as the batch size increases, the cross-validation accuracy will take a larger number of iterations to converge. For contrast, a batch size of 4 will converge around the 1000th iteration while a batch size of 64 will only converge around the 4000th iteration.

(b) **Plot the time taken to train the network for one epoch against different batch sizes.**

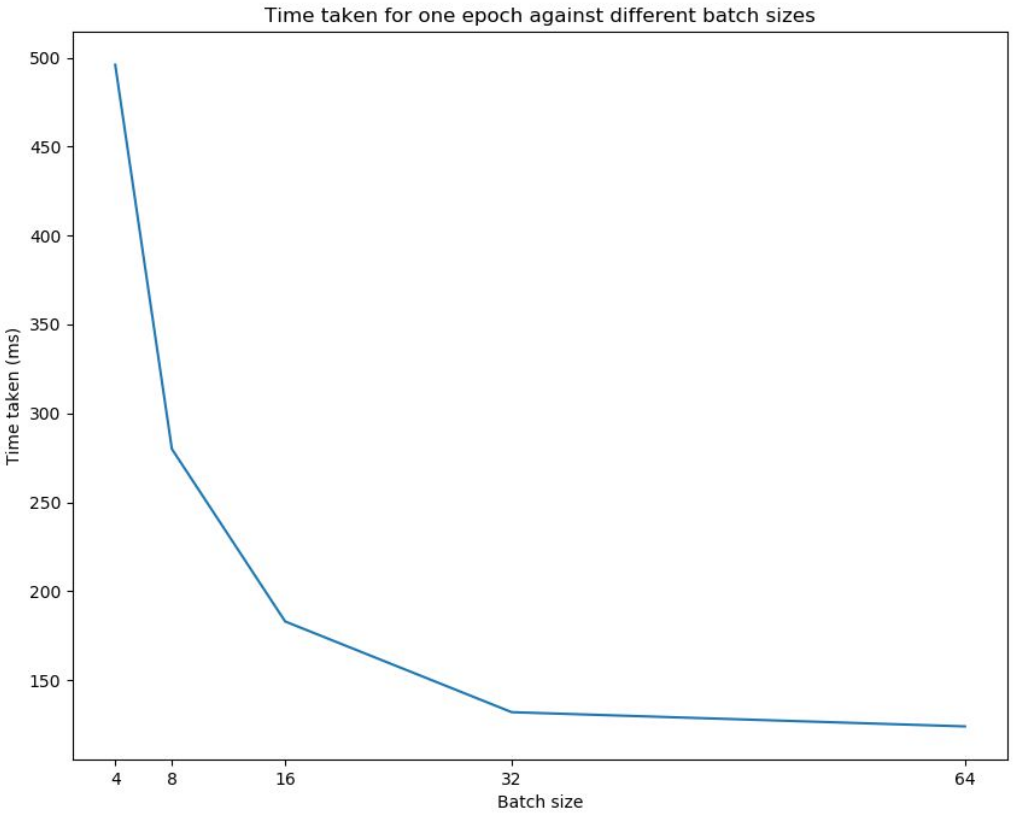

Figure A4: Time taken for different batch sizes

In the above Figure A4, it is observed that as the batch size increases, the time taken to run 1 epoch decreases drastically. For contrast, a batch size of 4 will take about 500ms while a batch size 64 will only take about 100ms.

This decrease in time is expected and explained in Question 2(a). A lower batch size would mean that the weights and biases will be required to be updated more frequently than the larger batch size.

(c) **Select the optimal batch size and state reasons for your selection.**

Based on the above results in Figure A3, a batch size of 32 will converge around the 3500th iteration and meet a similar accuracy as the batch size of 4. The batch size of 32 is also selected because of the shorter time taken at 100ms as compared to 500ms for a batch size of 4.

Even as the batch size of 32 requires 3500 iterations as compared to 1000 iterations for the batch size of 4, the overall time taken will still be shorter if a batch size of 32 is used.

As such, the optimal batch size is selected at 32.

(d) **Plot the train and test accuracies against epochs for the optimal batch size.**

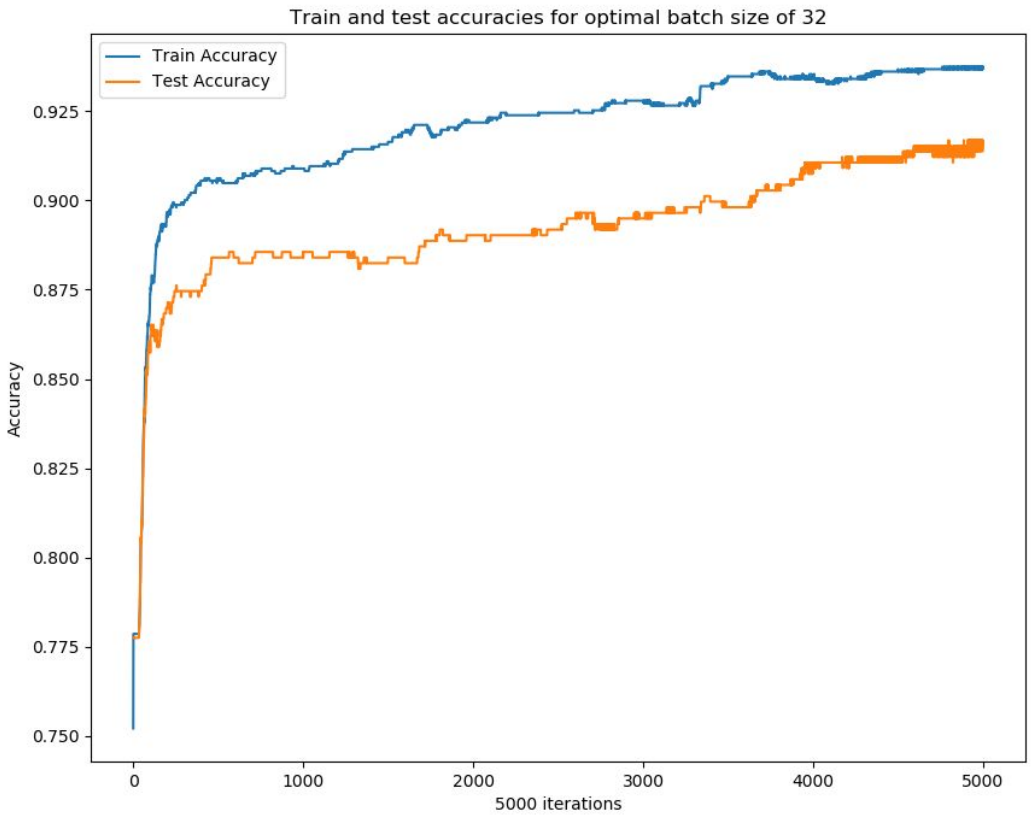

Figure A5: Train and test accuracies for optimal batch size of 32

In the above Figure A5, the test accuracies shows a noticeable improvement as it goes beyond the 3500th iteration. However, as discovered in Figure A2, an increase in test accuracy is not enough to determine whether the model is actually more accurate. The test error should be referenced as well.

Question 3

Find the optimal number of hidden neurons for the 3-layer network designed in part (2).

(a) **Plot the cross-validation accuracies against the number of epochs for different number of hidden-layer neurons. Limit the search space of number of neurons to {5,10,15,20,25}.**

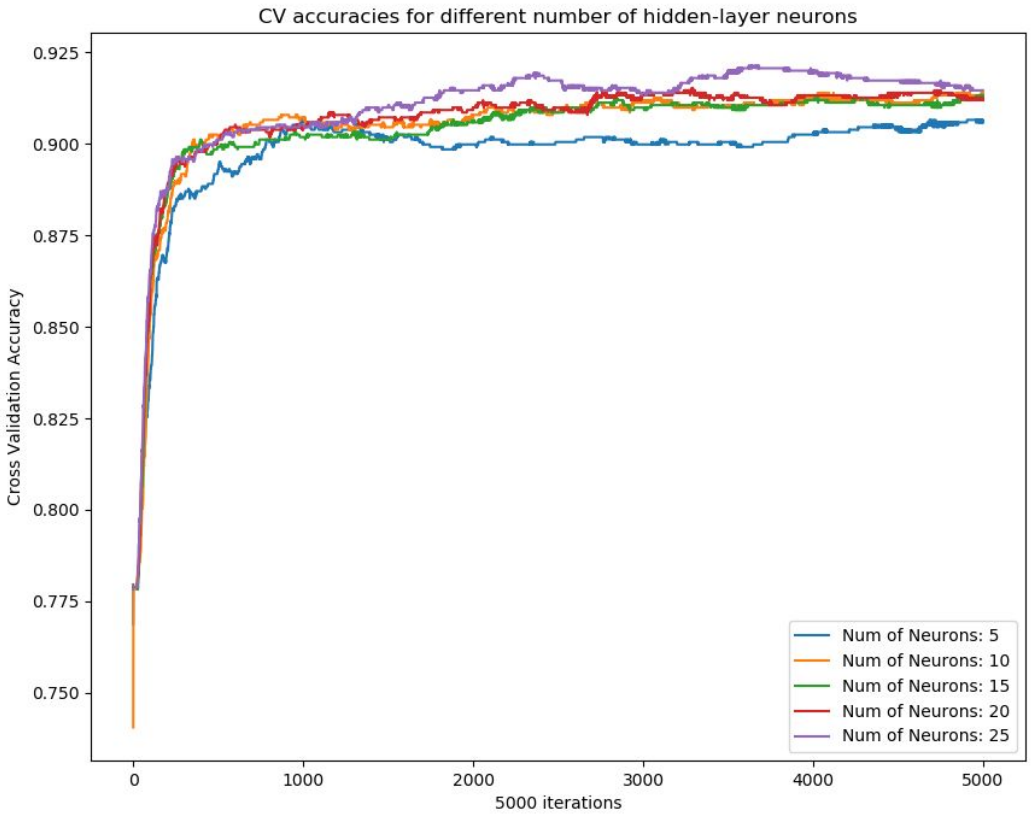

Figure A6: Cross-validation accuracies for different number of neurons in hidden layer

In the above Figure A6, the cross-validation accuracies are computed for the different number of neurons in the hidden layer such as 5, 10, 15, 20, 25.

(b) **Select the optimal number of neurons for the hidden layer. State the rationale for your selection.**

It is observed that as the number of neurons increases, the cross-validation accuracy shows a slight increase as well. For contrast, the 25-neuron network managed to reach the highest maximum accuracy of about 92% at the 2500th iteration while the 5-neuron network only managed to reach about 90% accuracy at the 1000th iteration.

As such, the optimal number of neurons in the hidden layer is selected at 25.

(c) **Plot the train and test accuracies against epochs with the optimal number of neurons.**

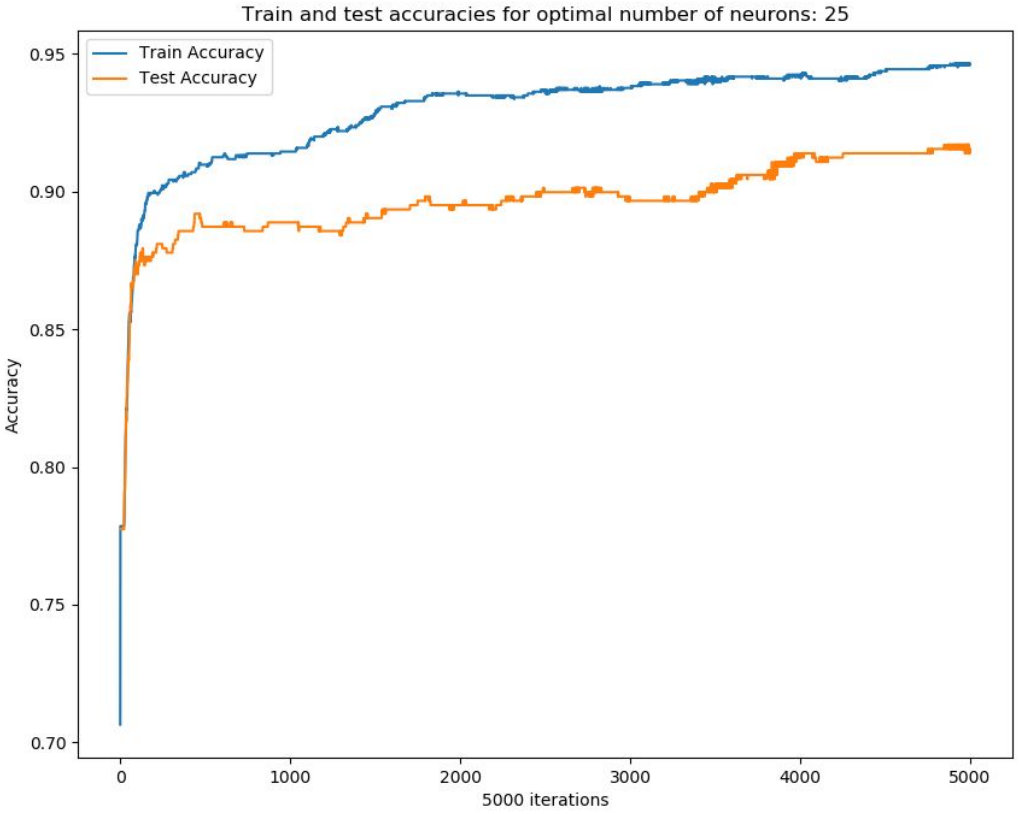

Figure A7: Train and test accuracies for optimal number of neurons in the hidden layer at 25

In the above Figure A7, the test accuracies shows a similar improvement to that of Figure A5 as it goes beyond the 3500th iteration.

Question 4

Find the optimal decay parameter for the 3-layer network designed with optimal hidden neurons in part (3).

(a) **Plot cross-validation accuracies against the number of epochs for the 3-layer network for different values of decay parameters. Limit the search space of decay parameters to {0, 10−3, 10−6, 10−9, 10−12}.**

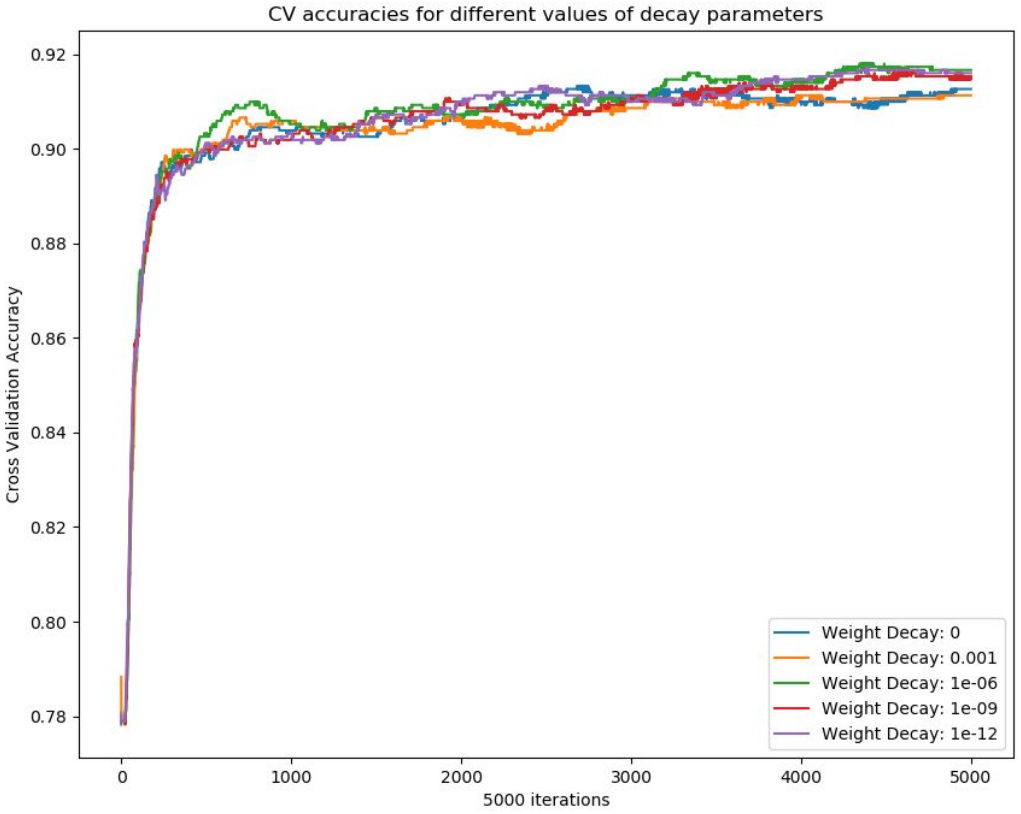

Figure A8: Cross-validation accuracies for different values of decay parameters

In the above Figure A8, the different cross-validation accuracies from each weight decay parameter are relatively similar to one another.

(b) **Select the optimal decay parameter. State the rationale for your selection.**

Upon a closer look, the weight decay of 1e-06 seems to rise above the rest of the other weight decay parameters at multiple points. Noticeably at the 1000th, 1900th, and 3500th iteration.

As such, the optimal decay parameter is set at 1e-06.

(c) **Plot the train and test accuracies against epochs for the optimal decay parameter.**

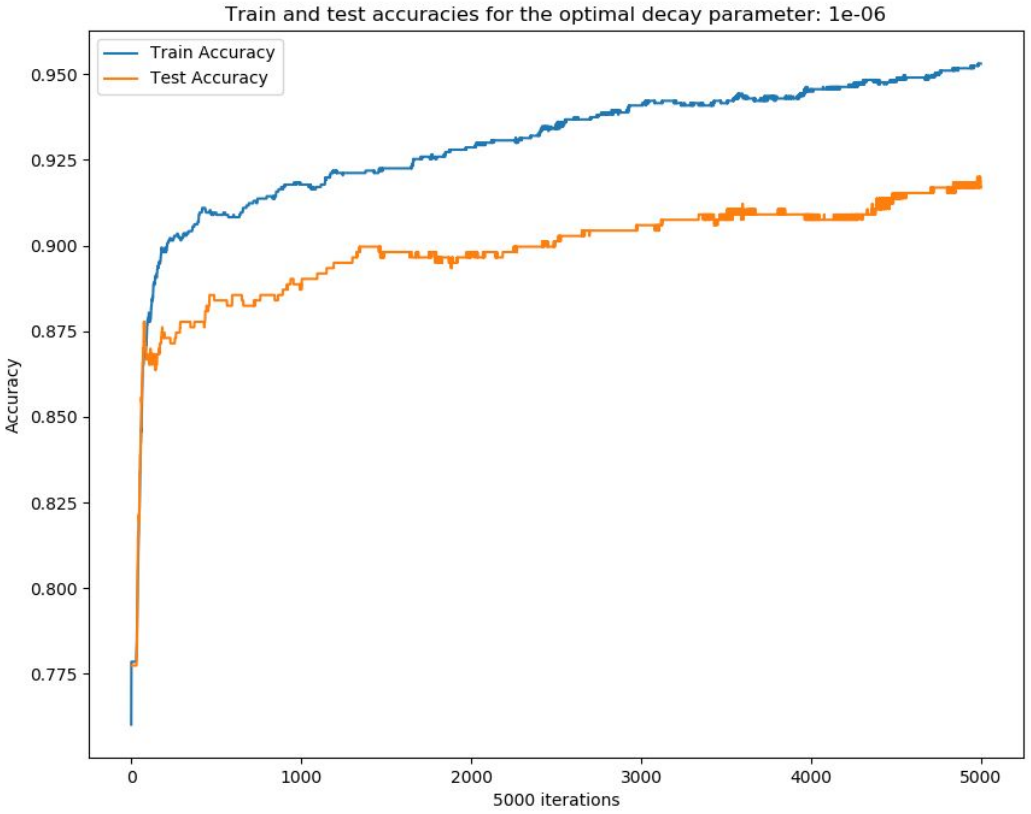

Figure A9: Train and test accuracies for the optimal decay parameter at 1e-06 (also the optimal 3-layer network)

In the above Figure A9, the test accuracies shows a similar improvement to that of Figure A7 as it goes beyond the 3500th iteration.

Question 5

After you are done with the 3-layer network, design a 4-layer network with two hidden layers, each consisting of 10 neurons, and train it with a batch size of 32 and decay parameter 10-6.

(a) **Plot the train and test accuracy of the 4-layer network.**

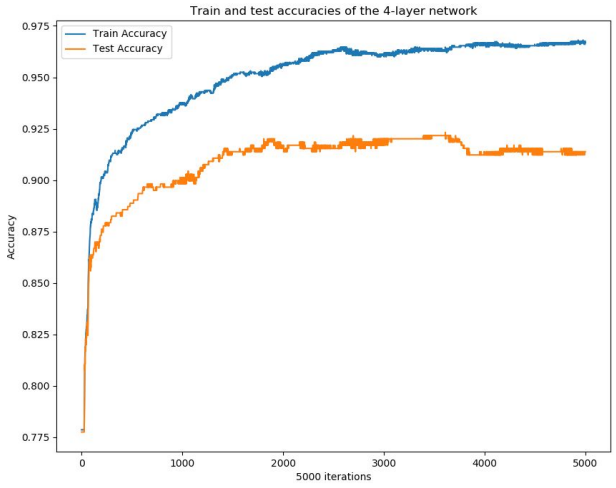

Figure A10: Train and test accuracy of the 4-layer network

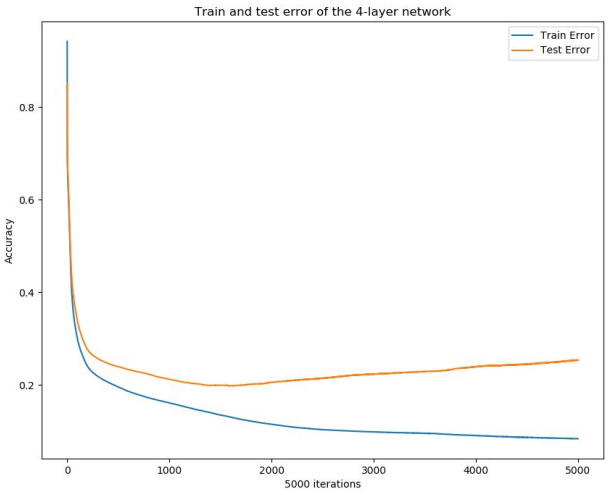

Figure A11: Train and test error of the 4-layer network

(b) **Compare and comment on the performances of the optimal 3-layer and 4-layer networks.**

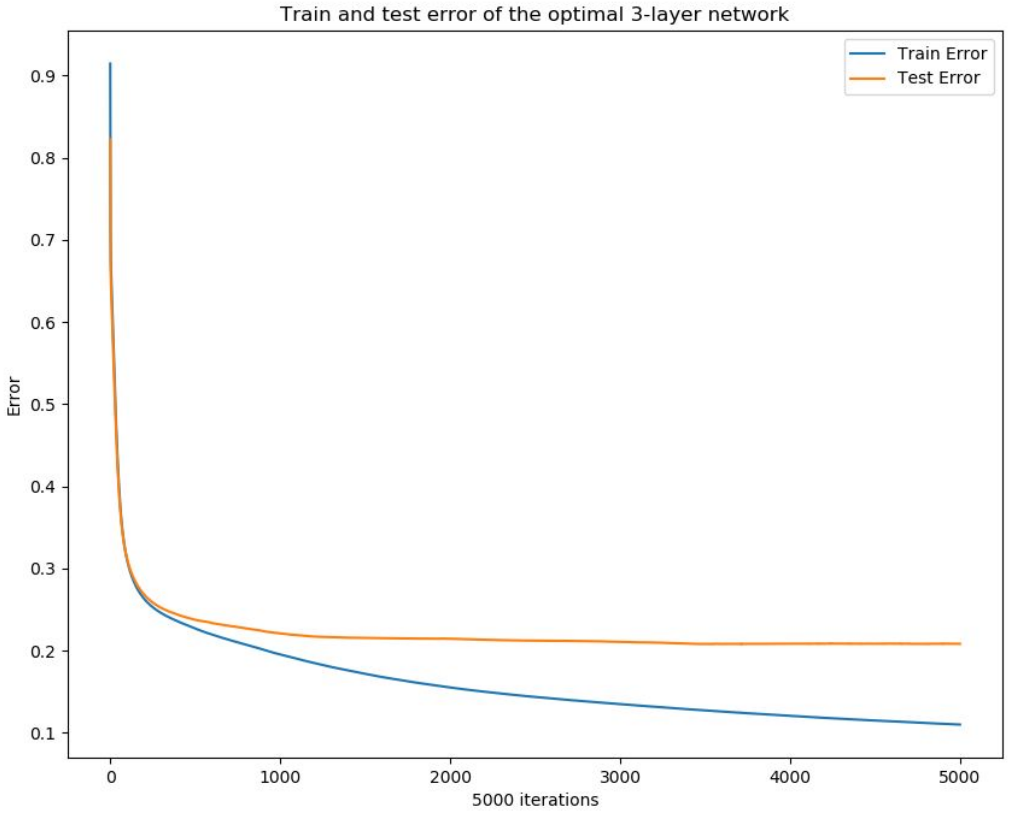

Figure A12: Train and test error of the optimal 3-layer network

In the 4-layer network, there is a sudden drop in the test accuracy at the 3500th iteration (as seen in Figure A10). The 4-layer network also shows an odd increase in the test error after the 1500th iteration (as seen in Figure A11). This behaviour hints towards an overfitted model.

On the other hand, the 3-layer network seemed to have fared better. This is evident as the test accuracy eventually increases above 90% (as seen in Figure A9) at the 4000th iteration. As this increase is taking place, the test error remains steady at about 22% (as seen in Figure A12).

As such, the 3-layer network clearly performs better than the 4-layer network.

Part B

Question 1

(a) Qn

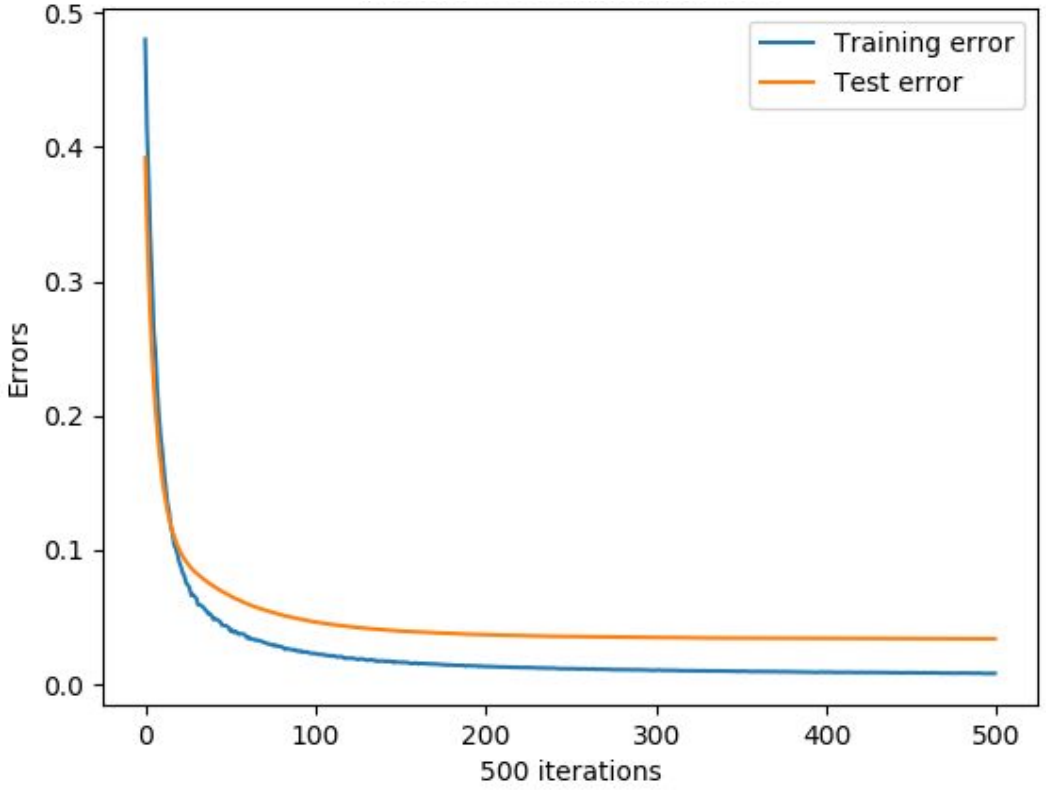

Figure B1: Train and Test error of 7 features against 500 iterations

(b) For the experiment, approximating number of epochs where the test error is minimum will be at 500 iterations. The chosen number of iterations is based on multiple runs and observation of the test error. The minimum error obtained from the figure is 0.043(3sf).

(c)

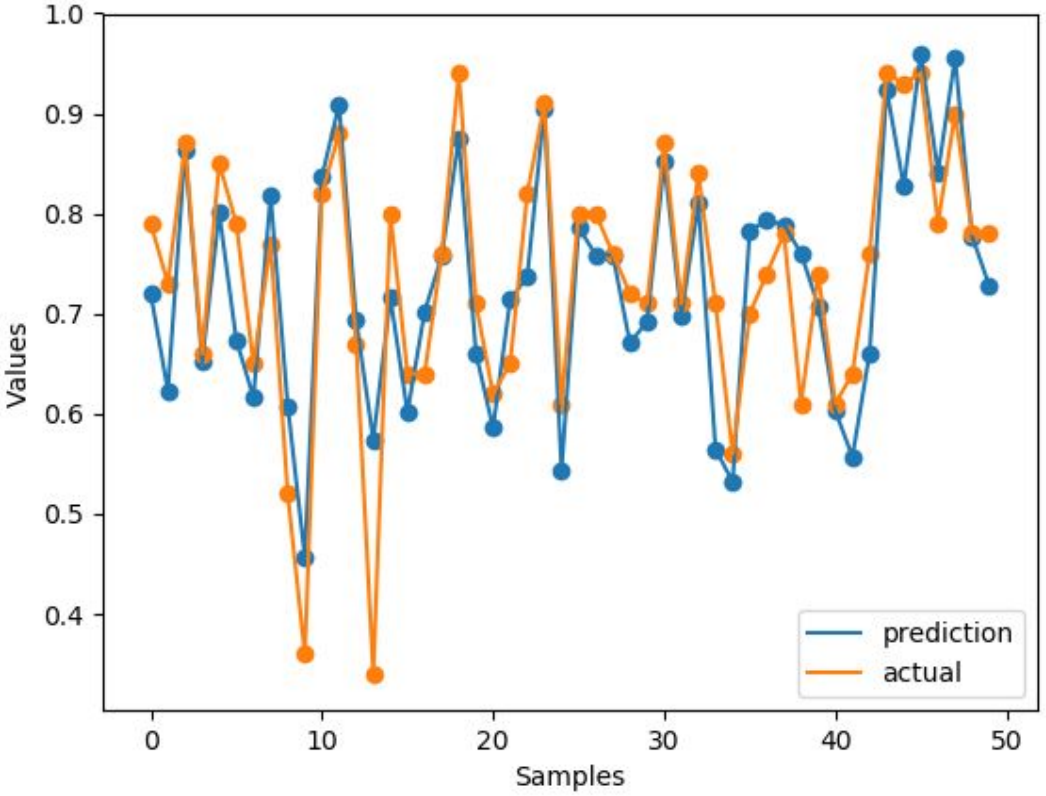

Figure B2: Predicted values and target values of 50 test samples

Question 2

(a)

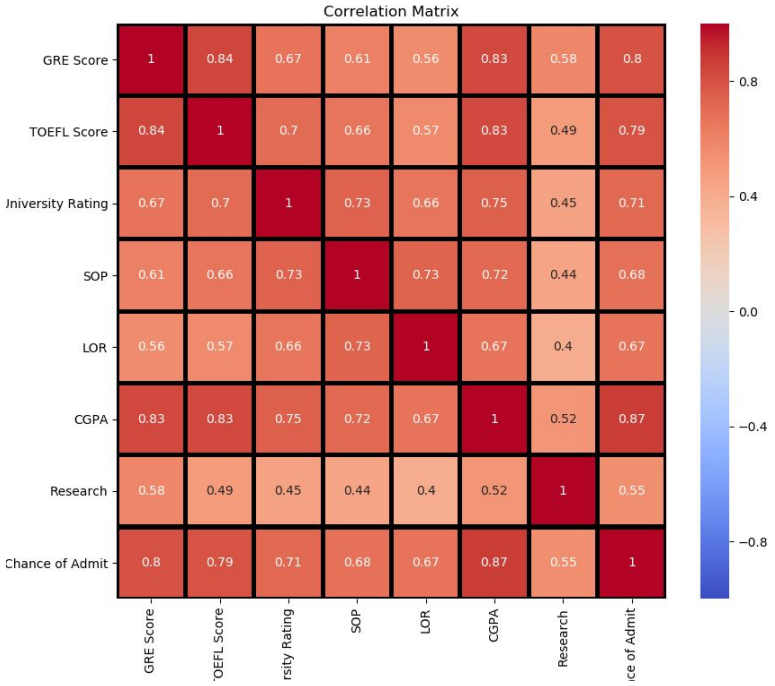

Figure B3: 8x8 Correlation Matrix between features

(b) According to the correlation matrix, there are 3 features that has high positive high correlations coefficient to each other and also to the Chances of admit. These features are 'GRE Score' , 'TOEFL Score' and 'CGPA'. First, CGPA with highest correlation with Chance of Admit. It is justifiable due to the fact that Cumulative Grade Point Average (CGPA) is a measure of a student's academic performance. Hence, in all universities have a minimum CGPA requirement for admission and is a determining factor for student getting their chances of enrolling in the university of their choice. Similarly to the GRE and TOEFL Score, The Graduate Record Examinations (GRE) where most of the schools in North America set as an admission requirement and Test of English as a Foreign Language (TOEFL) is used as a factor for enrolling into english speaking universities for non-native speakers.

(c) CGPA have the highest correlation

Question 3

REF method using linear regression model to define rankings for features. The result below shows the support and ranking for 7 features and which feature should remove.

[True True True False True True True]
[1 1 1 2 1 1 1]

The result shown that the fifth column stated “False” this correspond to the feature “SOP” and giving a ranking of 2. Hence, removing this feature and experiment whether it will cause minimum drop in performance.

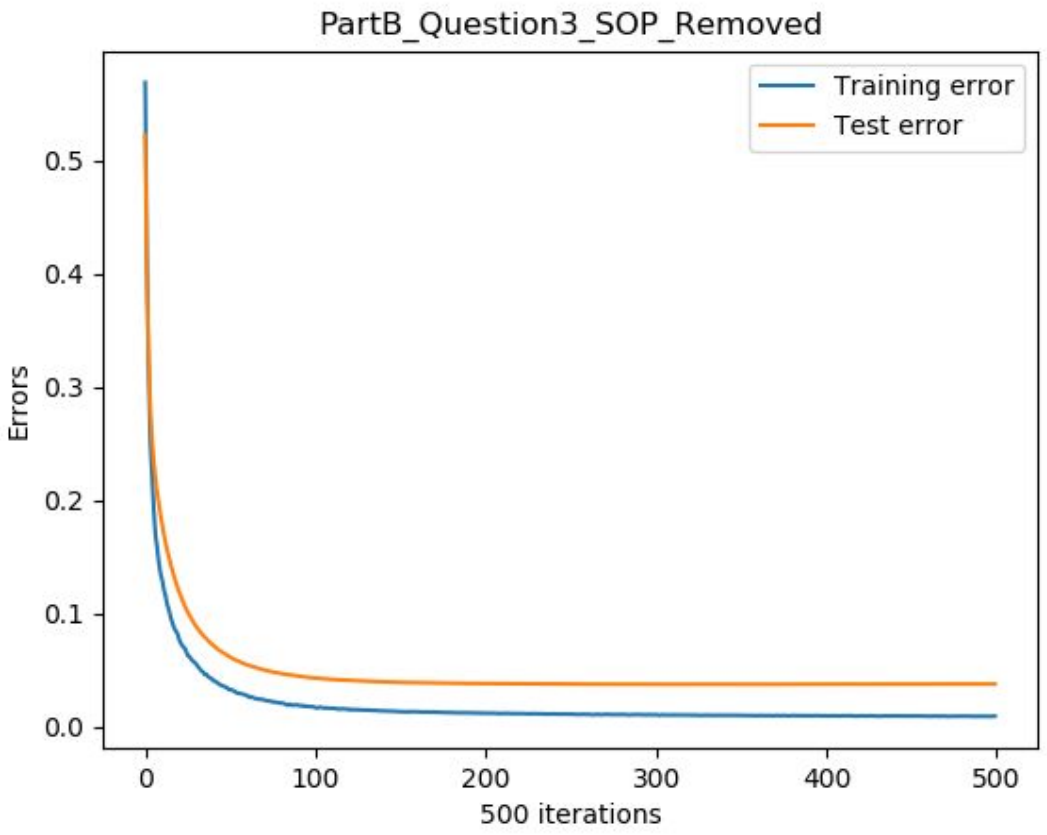

Figure B4: Train and Test error of 5 features (“SOP” removed)

After running 10 experiments of removing “SOP” feature the errors steady at about 0.0424(4sf). Hence, it was observed that it did little improvement to previously having 7 features.

The next feature to remove will be “University Rating” based on the REF model. The support and ranking are given:

[True True False False True True True]

[1 1 2 3 1 1 1]

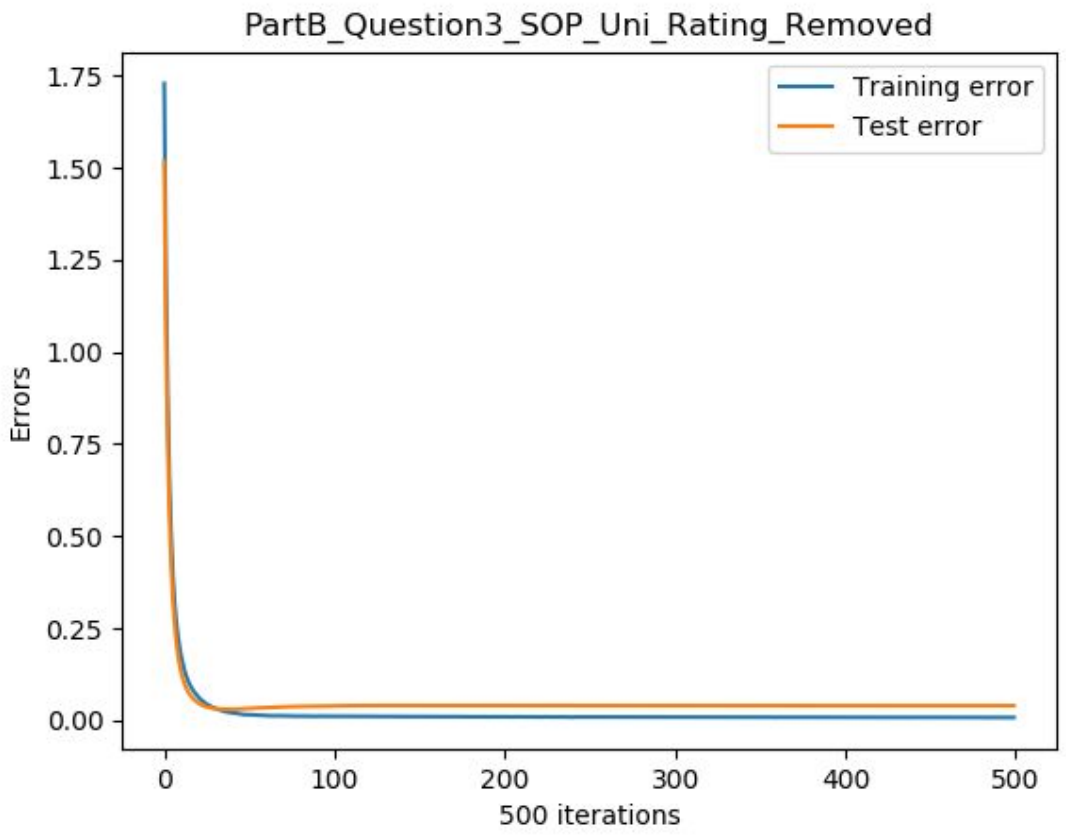

Figure B5: Train and Test error of 5 features (“SOP” & “University Rating” removed)

Finally, after running the experiment on 5 features it shows there is a drop of error where the error reach steady about 0.0409(4sf). This shows that indeed removing “SOP” and “University Rating” can reduce the errors.

In this case, from the above experiment it shows there is a drop of errors from 7 features to 5 features input. To support our findings that 5 input features could be the optimum number of features to use, the REF Linear Regression model is called to check. The result is shown below.

Optimum number of features: 5
Score with 5 features: 0.789103

Therefore, the optimum number of features will be 5.

Performance		
All Input Features	Removed “SOP”	Removed “University Rating”
Errors: 0.0430	Errors: 0.0424	Errors: 0.0409

Question 4

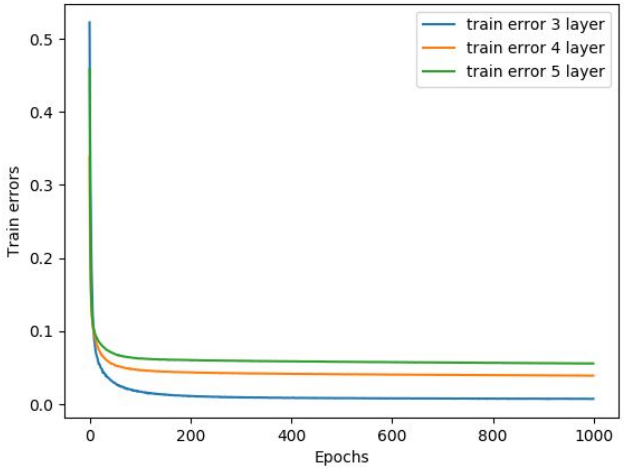

Figure B6: Training errors against Epoch for different number of layers

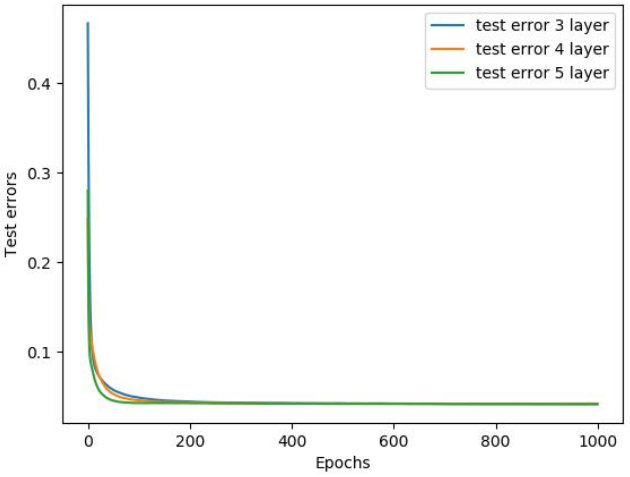

Figure B: Training errors against Epoch for different number of layers

	Test error	Test error (with dropout)
4-Layer network	0.0422	0.0520
5-Layer network	0.0420	0.0478

From the observation, the test error decreases as the number of layers increases due to the fact of deeper networks having more neurons for training. Therefore, it is likely to generalize

more features from the dataset and giving better performance. From the above table, the test error with dropout have are higher than without dropout. This explains that dropout is not suitable to implement in these networks. This is because that possibly when the network is small relative to the dataset, is not necessary to have regularization.

Conclusion

At the end of this project, the following points summarise our discoveries:

- Scaling the input data can help to speed up convergence in gradient descent
- Shuffling the index of the dataset is more efficient than changing the actual order of it
- The dataset should be split into its respective train and test set beforehand so that the test set will not be seen by the model during training
- Choosing the batch size is often a trade-off between accuracy and efficiency. A smaller batch size might be able to converge with a lower number of iterations but take a longer to compute while a larger batch size may need a higher number of iterations to reach the same accuracy
- Increasing the number of hidden layers and neurons in them may not necessarily improve the performance of the model
- Regularisation techniques are normally used to improve models when overfitting occurs