# **Part B**

The regression problem in Part B uses a Kaggle dataset containing 8 pieces of data – GRE SCore, TOFEL Score, University Rating, Statement of Purpose (SOP), Letter of Recommendation (LOR), undergraduate GPA (CGPA) and research experience and the Chance of Admission. SOP and LOR are assigned scores up to 5 and research experience is simply a 1 or 0 indicating whether or not the student had research experience.

The goal of Part B was to use that to predict the chance of admission based on the first seven columns of data (serial number is ignored).

## Preparation of data

The dataset was split into a 70:30 ratio into test and train sets as specified by the project manual. The train\_test\_split function from sklearn was used to achieve this.

x\_train**,** x\_test**,** y\_train**,** y\_test **=** train\_test\_split**(**features**,** y**,** test\_size**=**0.3**,** random\_state**=**42**)**

# **Question 1**

A 3-layer feedforward neural network was built which consisted of an input layer, a hidden-layer of 10 neurons with ReLU activation functions, and an output layer with a linear activation function.

Mini-batch gradient descent was used with batch size of 8. L2 regularization of 10-3 is used to reduce overfitting issues on our loss function. A learning rate of 10-3 is used.

## Setup of Weights, Biases and Neurons

The weights and biases of the hidden layer and output layer are as follows:

|  |
| --- |
| weights **=** **{**  'h1'**:** tf**.**Variable**(**tf**.**random\_normal**([**NUM\_FEATURES**,** NUM\_NEURON**])),**  'out'**:** tf**.**Variable**(**tf**.**random\_normal**([**NUM\_NEURON**,** 1**]))**# 1 ouput label  **}**  biases **=** **{**  'b1'**:** tf**.**Variable**(**tf**.**random\_normal**([**10**])),**  'out'**:** tf**.**Variable**(**tf**.**random\_normal**([**1**]))**  **}** |

And the neural net was defined as follows:

|  |
| --- |
| **def** neural\_net**(**x**):**  #hidden layer 1  layer\_1 **=** tf**.**add**(**tf**.**matmul**(**x**,** weights**[**'h1'**]),** biases**[**'b1'**])**  layer\_1 **=** tf**.**nn**.**relu**(**layer\_1**)** #activation function    out\_layer **=** tf**.**matmul**(**layer\_1**,** weights**[**'out'**])** **+** biases**[**'out'**]**  **return** **(**out\_layer**)** |

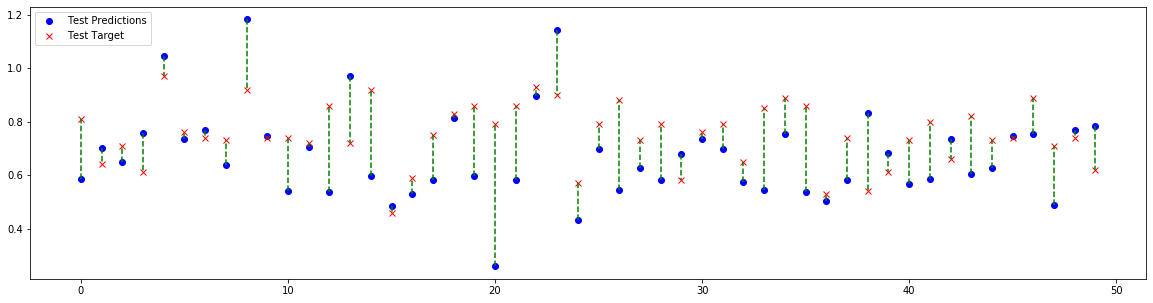
## Results

Figure 1: Results with all 7 features



The figure above is a plot between the test and train error over the number of epochs. The approximate epoch at which test error is minimum is around the 1200th epoch.

Figure 2: Predicted vs target



The figure above shows the predicted values and target values of a random set of 50 test samples.

At the 1500h epoch, the train error and test error are as follows:

train error 0.0340372, test error 0.040109

# **Question 2**

The table below shows the 8x8 correlation matrix. The diagonal values are all 1 because it is the value of correlation with itself which we can ignore.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **GRE Score** | **TOEFL Score** | **University Rating** | **SOP** | **LOR** | **CGPA** | **Research** | **Chance of Admit** |
| GRE Score | 1 | 0.835977 | 0.668976 | 0.612831 | 0.557555 | 0.83306 | 0.580391 | 0.80261 |
| TOEFL Score | 0.835977 | 1 | 0.69559 | 0.657981 | 0.567721 | 0.828417 | 0.489858 | 0.791594 |
| University Rating | 0.668976 | 0.69559 | 1 | 0.734523 | 0.660123 | 0.746479 | 0.447783 | 0.71125 |
| SOP | 0.612831 | 0.657981 | 0.734523 | 1 | 0.729593 | 0.718144 | 0.444029 | 0.675732 |
| LOR | 0.557555 | 0.567721 | 0.660123 | 0.729593 | 1 | 0.670211 | 0.396859 | 0.669889 |
| CGPA | 0.83306 | 0.828417 | 0.746479 | 0.718144 | 0.670211 | 1 | 0.521654 | 0.873289 |
| Research | 0.580391 | 0.489858 | 0.447783 | 0.444029 | 0.396859 | 0.521654 | 1 | 0.553202 |
| Chance of Admit | 0.80261 | 0.791594 | 0.71125 | 0.675732 | 0.669889 | 0.873289 | 0.553202 | 1 |

By rearranging the data, we can get a better understanding of the correlations of features. In the table below, we can see that CGPA, GRE Score and TOEFL score have the highest correlations with the Chance of Admit.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **1st** | **2nd** | **3rd** | **4th** | **1st\_Val** | **2nd\_Val** | **3rd\_Val** | **4th\_Val** |
| GRE Score | TOEFL Score | CGPA | Chance of Admit | University Rating | 0.835977 | 0.83306 | 0.80261 | 0.668976 |
| TOEFL Score | GRE Score | CGPA | Chance of Admit | University Rating | 0.835977 | 0.828417 | 0.791594 | 0.69559 |
| University Rating | CGPA | SOP | Chance of Admit | TOEFL Score | 0.746479 | 0.734523 | 0.71125 | 0.69559 |
| SOP | University Rating | LOR | CGPA | Chance of Admit | 0.734523 | 0.729593 | 0.718144 | 0.675732 |
| LOR | SOP | CGPA | Chance of Admit | University Rating | 0.729593 | 0.670211 | 0.669889 | 0.660123 |
| CGPA | Chance of Admit | GRE Score | TOEFL Score | University Rating | 0.873289 | 0.83306 | 0.828417 | 0.746479 |
| Research | GRE Score | Chance of Admit | CGPA | TOEFL Score | 0.580391 | 0.553202 | 0.521654 | 0.489858 |
| Chance of Admit | CGPA | GRE Score | TOEFL Score | University Rating | 0.873289 | 0.80261 | 0.791594 | 0.71125 |

This is justifiable since graduate admissions tend to be based on undergraduate GPA, GRE score and TOEFL score all of which are metrics that allow admission officers to gauge your prior educational performance.

## Question 3

The sklearn feature\_selection library was used for implementing Recursive Feature Elimination (RFE). The code excerpt below is the implementation of it in Python.

|  |
| --- |
| #Stores feature ranking from rfe.  rfe\_ranking **=** **[]**  **for** n **in** range**(**5**,** 7**):**  X\_train**,** X\_test**,** y\_train**,** y\_test **=** train\_test\_split**(**features**,**y**,** test\_size **=** 0.3**,** random\_state **=** 42**)**  model **=** LinearRegression**()**  rfe **=** RFE**(**model**,**n**)** #second arg is number of features to select  X\_train\_rfe **=** rfe**.**fit\_transform**(**X\_train**,**y\_train**)**  X\_test\_rfe **=** rfe**.**transform**(**X\_test**)**  model**.**fit**(**X\_train\_rfe**,**y\_train**)**  score **=** model**.**score**(**X\_test\_rfe**,**y\_test**)**  score\_list**.**append**(**score**)**    rfe\_ranking**.**append**(**rfe**.**ranking\_**)** |

The code runs twice, the first run selects five features while the second run selects six features. The results are appended in the rfe\_ranking list and are as follows:

Rfe\_ranking[0] = [2 1 1 3 1 1 1]

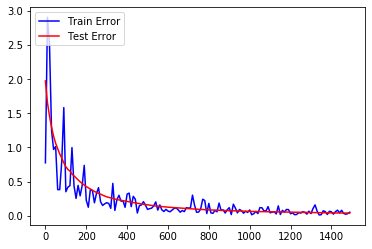
Rfe\_ranking[1] = [1 1 1 2 1 1 1]

The numbers ranging from 1 to 3 denote the importance of the feature. The index represents the order of features in the dataset – the element in the first index corresponds to GRE Score, followed by TOEFL score so on and so forth.

The values in Rfe\_ranking[0] tells us that the GRE Score and Statement of Purpose were not as important as the other features.

By running the neural network with GRE Score and SOP features removed, we get the following results:

Figure 3: RFE 5 features



If however 6 features were selected instead, only the feature SOP is dropped. Figure 4 shows the results. The training error for 6 features is noisier compared to the 5feature model at the beginning but settles at around the 800th epoch.

Compared with using all 7 features however, using 6 features is significantly noisier for training in later epochs.

Figure 4: RFE 6 Features

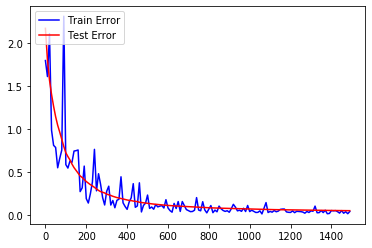


Figure : 7 features



### Accuracy

Figure 6: Predicted vs target (5 Features)

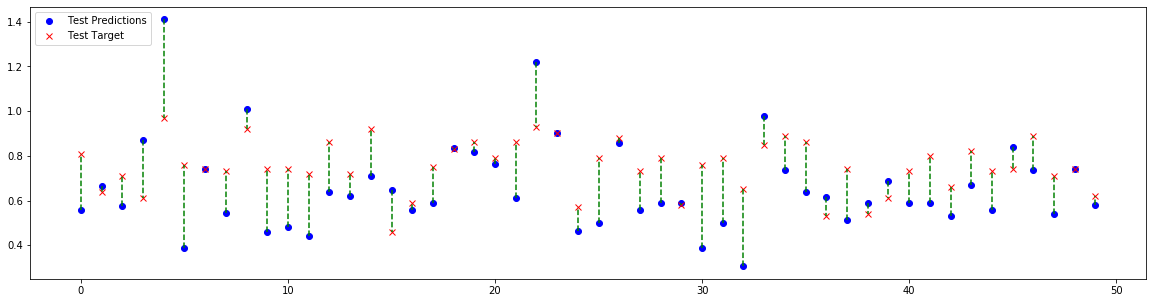


Figure 7: Predicted vs target (6 Features)

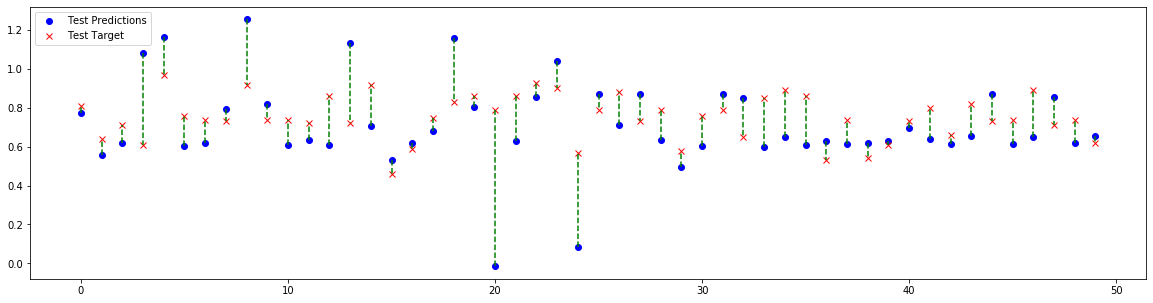
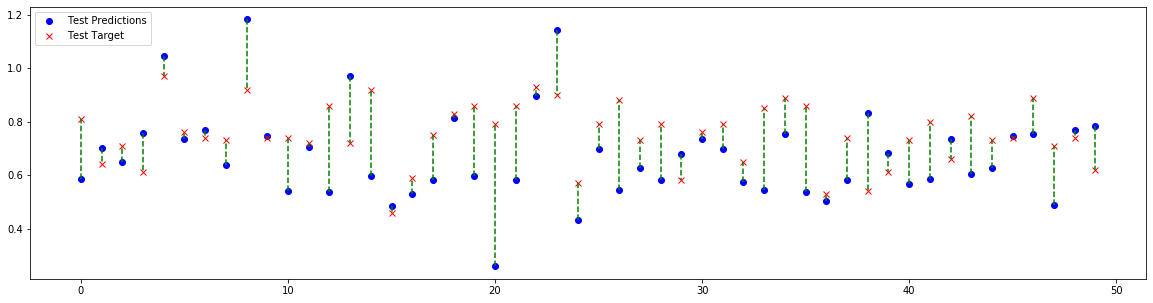


Figure : Predicted vs target (7 Features)



Comparing the accuracy, we can see that the model trained from using 5 features is the most inaccurate (Figure 6) compared to Figure 7 and Figure 8. The model trained using only 6 features is more accurate than the one with 7 features albeit only slightly.

## Question 4

This question requires the implementation of a four and five-layer neural network with the hidden layers having 50 neurons each. The learning rate remains the same at 10-3. Dropout at a 0.2 probability rate is introduced to prevent overfitting as we have a lot of neurons in the hidden layers. Note that we are **using the optimal feature set** selected in Question 3 (6 features, SOP dropped).

## Results

Figure 9: 3 layer, 10 neurons

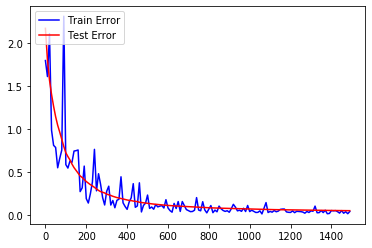


Figure 10: 4 layer, 50 neurons

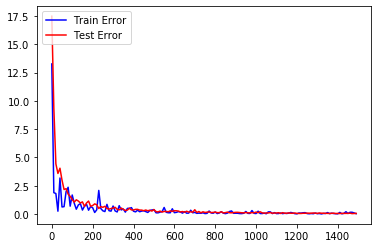
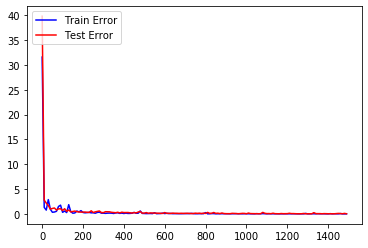


Figure 11: 5 layer, 50 neurons



The three models being compared are as follows:

* 3-layer, 10 neurons in hidden-layer
* 4-layer, 50 neurons in hidden-layer
* 5-layer, 50 neurons in hidden-layer

Moving from the 3-layer model to the 4-layer model, the immediate observation is that the 3-layer model train error is nosier and settles slower than the 4-layer model. While both the 3-layer and 4-layer model test errors converges to roughly the same value, the 4-layer model converges must faster.

The 5-layer network is better than the 3-layer and 4-layer models in terms of train error and test error as it converges at around the first few epochs.

Figure 12: 3-layer, 10 neurons predictions

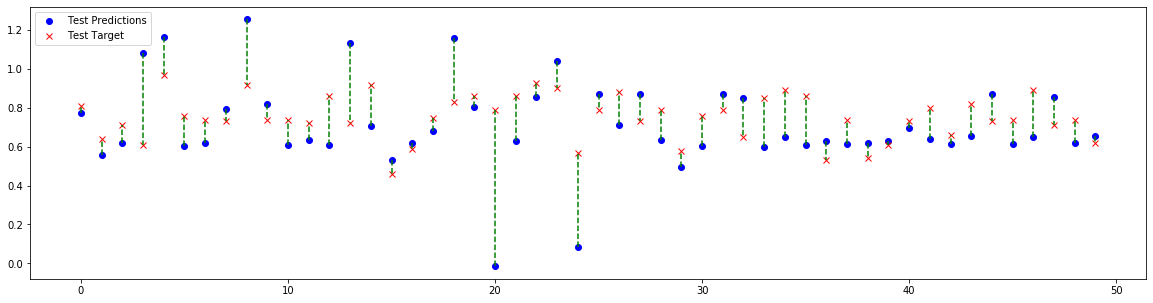


Figure 13: 4-layer, 50 neurons predictions

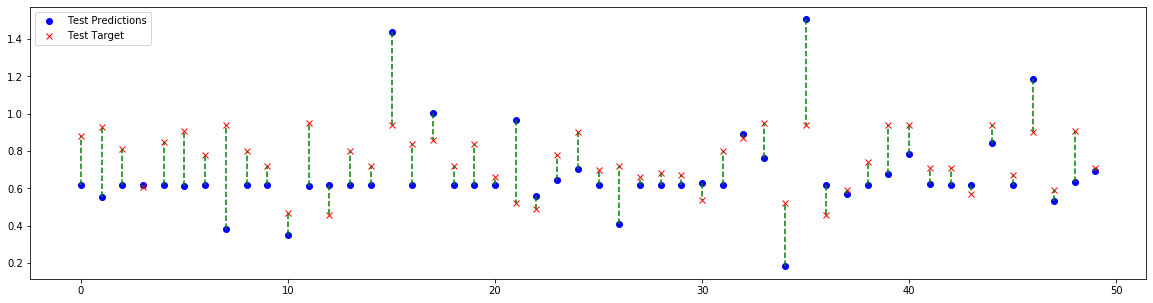
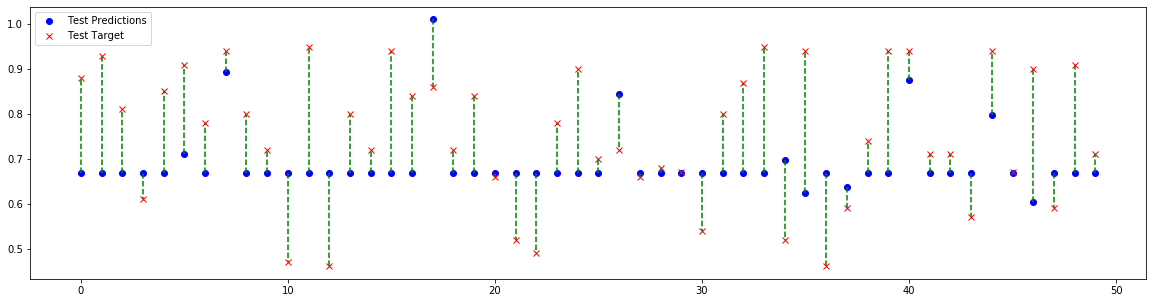


Figure : 5-layer, 50 neuron predictions



The predictions however appear to be better for the 3-layer, 10 neuron model. This might be because of overfitting. More hidden layers and neurons make it easier for the model to memorize the training dataset and hence it wasn’t making predictions but rather giving out the actual target value corresponding to the input.