TECHNOLOGICAL UNIVERSITY SINGAPORE

NEURAL NETWORK

LAB 1

Abstract

[Draw your reader in with an engaging abstract. It is typically a short summary of the document.
When you're ready to add your content, just click here and start typing.]

Laura Nolling Jensen & Søren Goddiksen Graabæk

N1800155C e.ntu.edu.sg & N1800154F@e.ntu.edu.sg

Introduction

This paper describes the development of two neural networks as a solution to the ”Project 1”-assignment in the course 4042 Neural Networks and Deep Learning. One is used for classification of the center pixel from 3x3 neighborhoods in a satellite image. The other network is used for regression to predict housing prices from other attributes in the data. The paper is therefore divided in two sections - Part A and Part B, where the working method, experiment results and part conclusions are given

Methods and Experiments

Part A: Classification problem

This project aims at building neural networks to classify the Landsat satellite dataset:
Https://archive.ics.uci.edu/ml/datasets/Statlog+(Landsat+Satellite)

The dataset contains multispectral values of pixels in a 3x3 neighborhoods in satellite images and class labels of the centre pixels in each neighborhood. The aim is to predict class labels in the test dataset after training the neural network on training dataset.
Training data: 4435 samples
Test data: 2000 samples

Read the data from the two files: training data from sat_train.txt and testing data from sat_test.txt. Do not use the data in the test dataset during training. It is reserved for computation of final performance measures. Think of it as unseen data during all of your work.

Each data sample is a row of 37 values: 36 input attributes (4 spectral bands x9 pixels in the neighborhood) and the class label. There are 6 class-labels: 1, 2, 3, 4, 5, 7

1) Design a feedforward neural network which consists of: an input layer, one hidden perceptron layer of 10 neurons and an output softmax layer. Assume a learning rate $\alpha = 0.01$, L2 regularization with weight decau parameter $\beta = 10^{-6}$, and batch size 32. Use appropriate scaling of input features.

Ans Q1
The source code for question 1 is “PartAq1.py”. On figure 1 is the training error versus epochs and accuracy versus epochs. We can see that the training error is converging to a small error, while the test accuracy is oscillating until about 2000 iterations, where it starts overfitting.

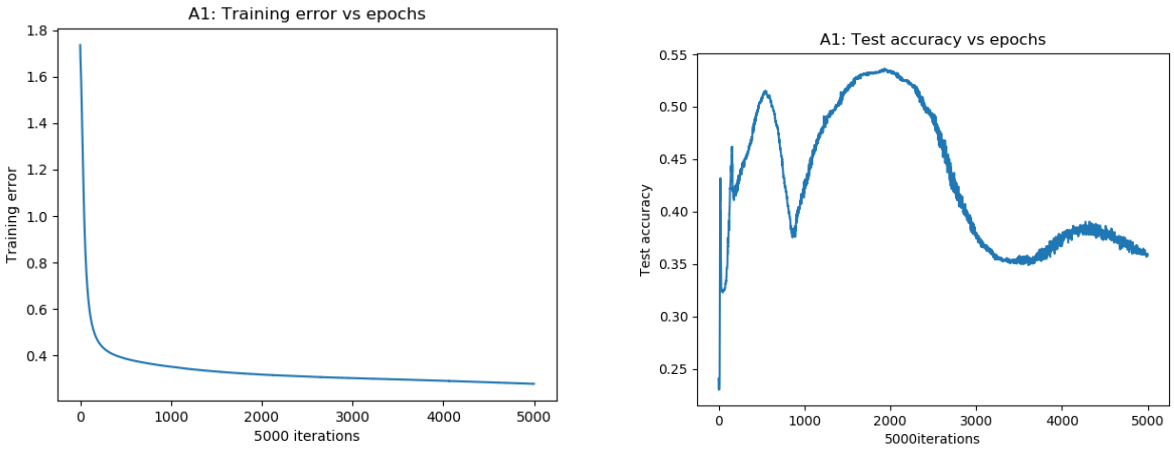


Figure 1: Training error vs epochs (left) and accuracy vs epochs(right)

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

   a. Plot the training errors and test accuracies against the number of epochs for the 3-layer network for different batch sizes. Limit search space to $s = \{4,8,16,32,64\}$
   b. Plot the time taken to train the network for once epoch against different batch sizes
   c. State the rationale for selecting the optimal batch size

   Use the batch size for the rest of the experiments.

Ans Q2

The source code for question 2 is "PartAq2.py". The "Training error vs epochs" and "test accuracy vs epochs" for the 5 different batch sizes are plotted in Figure 2 to 6. The average epoch time for each of the batch sizes is plotted in Figure 7. If we look at the accuracy for batch size 4, 8 and 16, we see that the network starts overfitting before the training error converges, which means that they would be a bad choice. The accuracy of batch size 32 oscillates while the accuracy for batch size 64 is smoother. Because the training data size is so large, 4435, compared to the batch sizes, the increasing of the batch size from 32 to 64 will not have a big impact on the training. A Batch size of 64 is therefore chosen as the optimal.

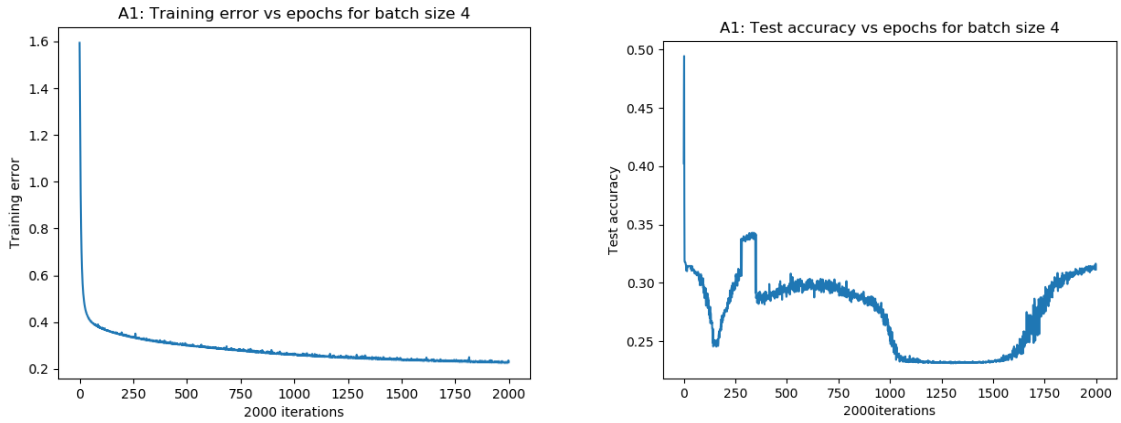


Figure 2: Batch size: 4. Training error vs epoch (left), and accuracy vs epoch (right)

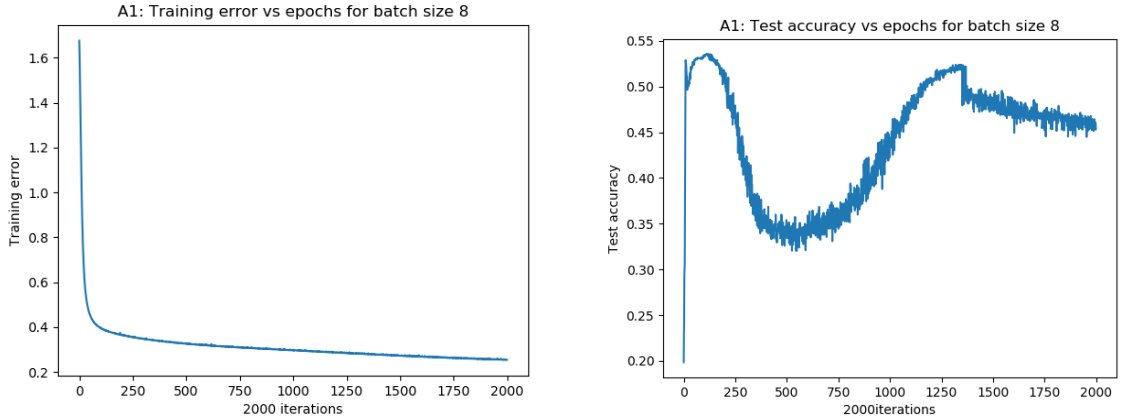


Figure 3: Batch size: 8. Training error vs epoch (left), and accuracy vs epoch (right)

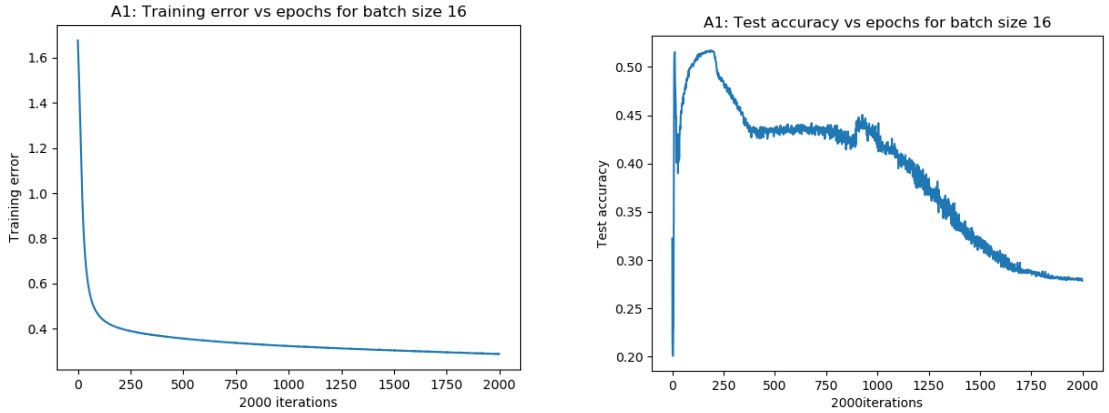


Figure 4: Batch size: 16. Training error vs epoch (left), and accuracy vs epoch (right)

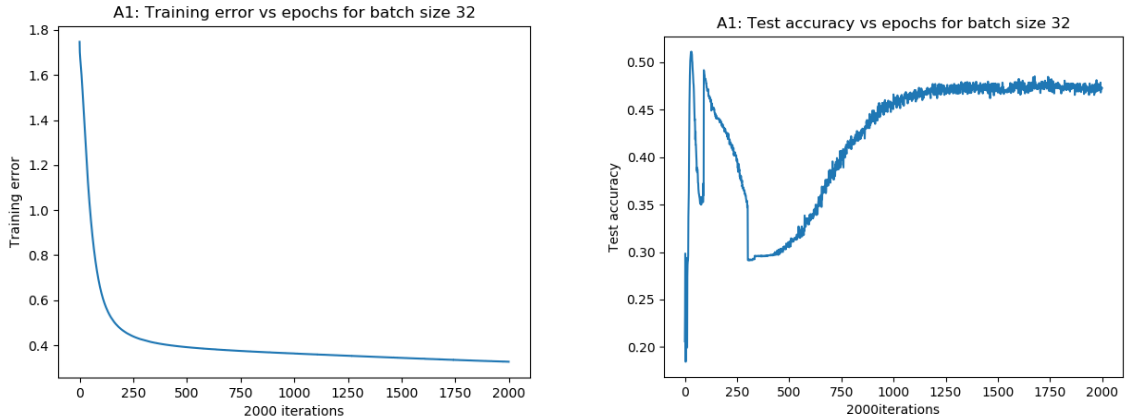


Figure 5: Batch size: 32. Training error vs epoch (left), and accuracy vs epoch (right)

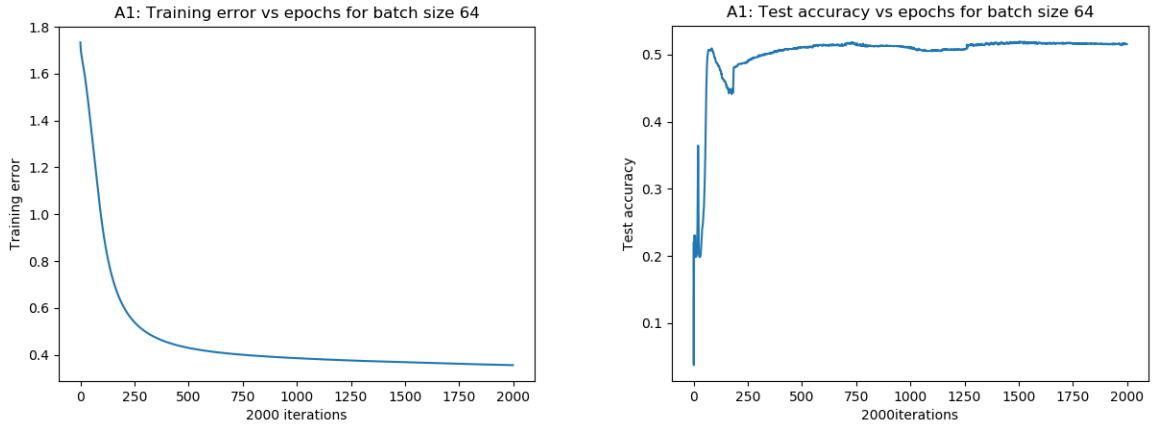


Figure 6: Batch size: 64. Training error vs epoch (left), and accuracy vs epoch (right)

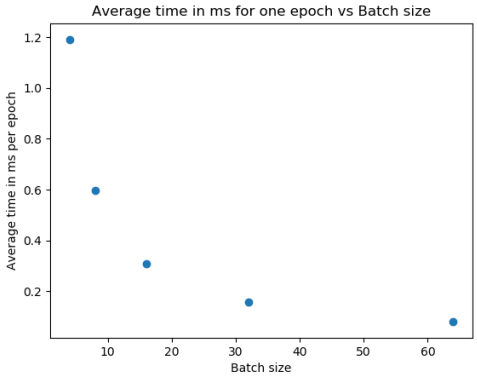


Figure 7: Average epoch time vs batch size

3) Find the optimal number of hidden neurons for the 3-layer network designed in part (2).
    a. Plot the training errors and test accuracies against the number of epochs for 3-layer neurons. Limit the search space for the number of hidden neurons to $S = \{5,10,15,20,25\}$
    b. Plot the time to train the network for each one epoch for different number of hidden-layer neurons
    c. State the rationale for selecting the optimal number of hidden neurons

Ans Q3

The source code for question 3 is “PartAq3.py”. The “training error vs epochs” and “accuracy vs epochs” for the 5 different numbers of hidden neurons are plotted in Figure 8 to 12, and the average epoch time for each of the batch sizes is plotted on Figure 13. We can see that 10 neurons have a very bad test accuracy over all. 5 hidden neurons have a decent test accuracy around 1000 epochs, but the training error has not yet converged. The same can be said about 20 and 25 hidden neurons. 15 hidden neurons have a stable test accuracy and have a good average time per epoch as shown in Figure 13. 15 hidden neurons are therefore selected as the optimal number.

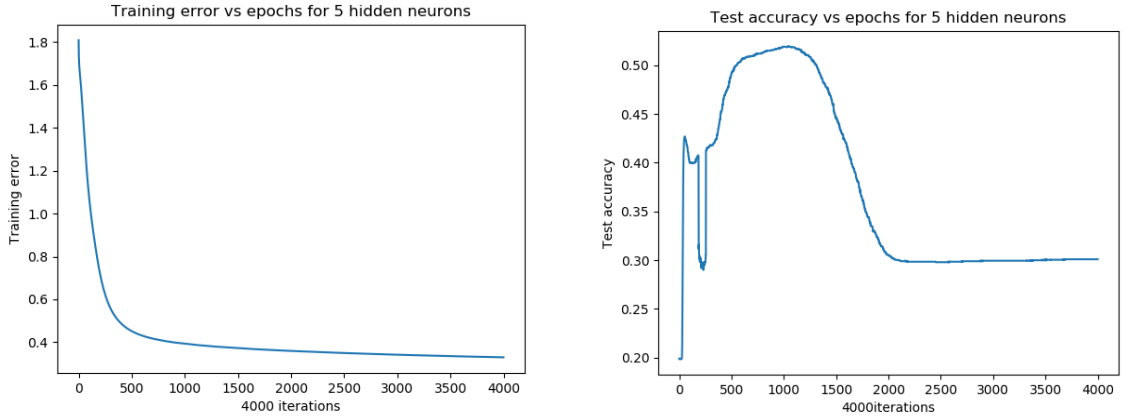


Figure 8: Hidden neurons: 5. Training error vs epoch (left), and accuracy vs epoch (right)

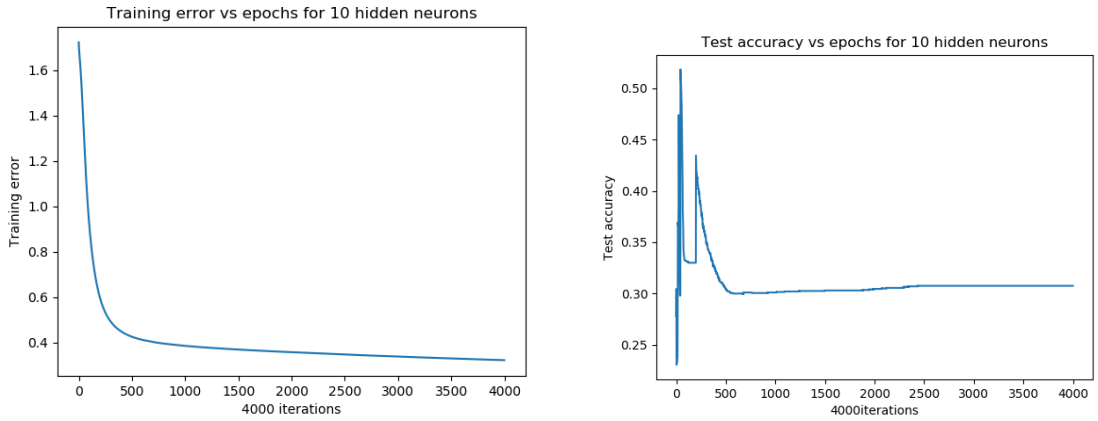


Figure 9: Hidden neurons: 10. Training error vs epoch (left), and accuracy vs epoch (right

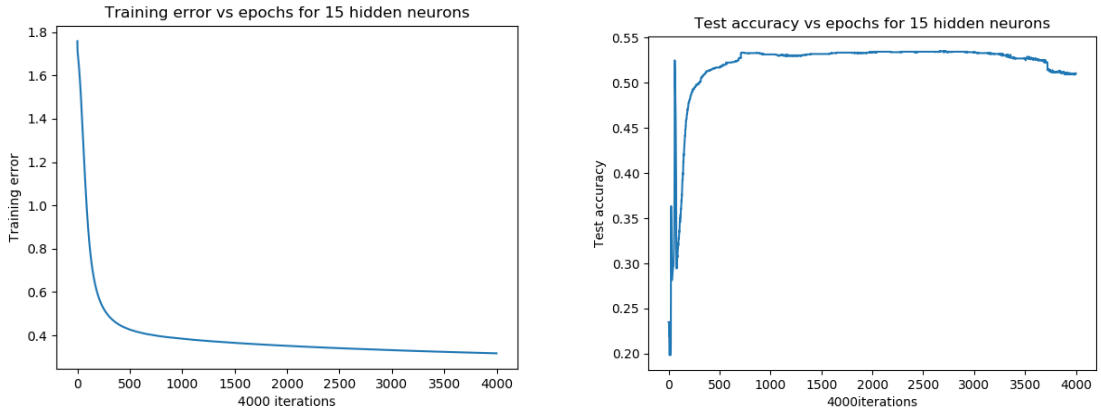


Figure 10: Hidden neurons: 15. Training error vs epoch (left), and accuracy vs epoch (right)

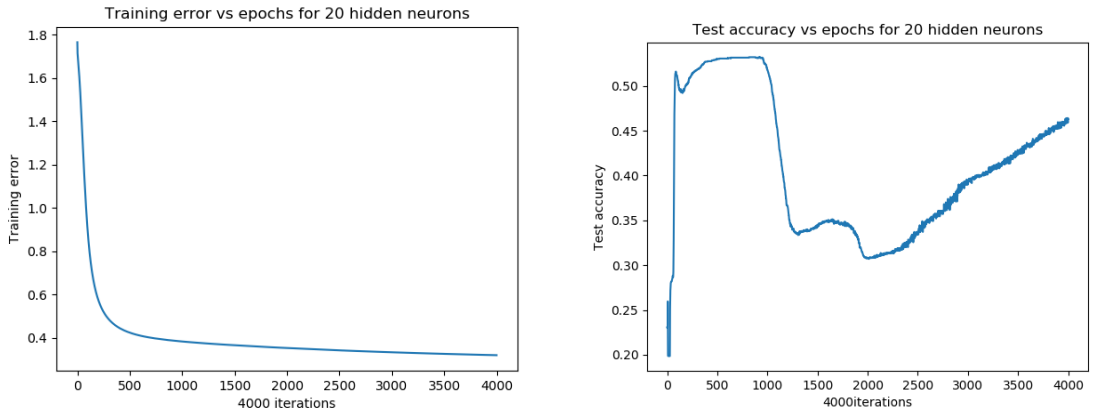


Figure 11: Hidden neurons: 20. Training error vs epoch (left), and accuracy vs epoch (right)

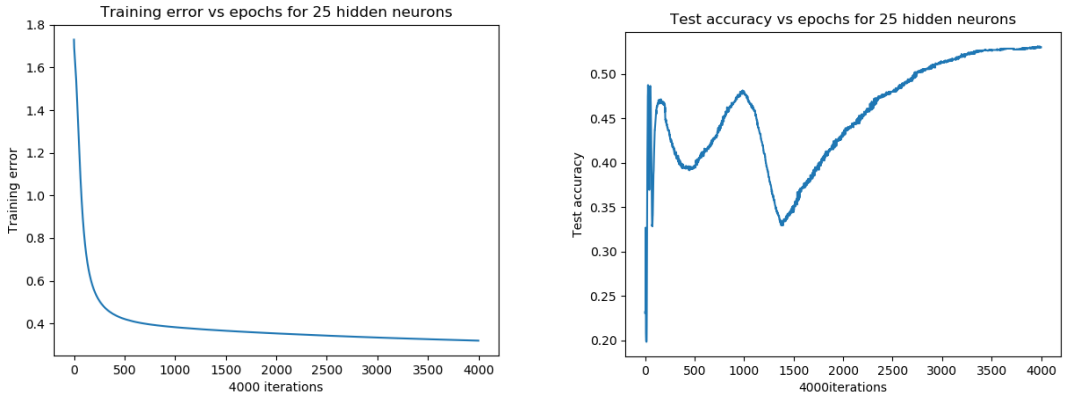


Figure 12: Hidden neurons: 25. Training error vs epoch (left), and accuracy vs epoch (right)

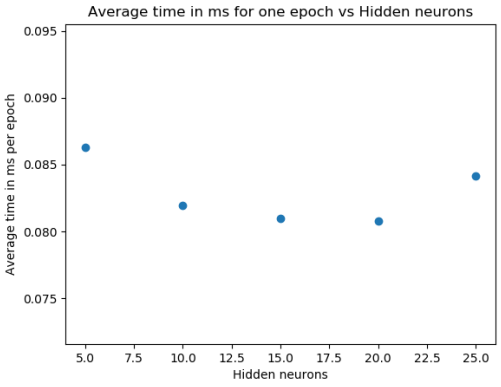


Figure 13: Average epoch time vs hidden neurons

4) Find the optimal decay parameter for the 3-layer network designed with the optimal number of hidden neurons in part (3).
    a. Plot the training errors against the number of epochs for the 3-layers network for different values of decay parameters in search space $S = \{0, 10^{-3}, 10^{-6}, 10^{-9}, 10^{-12}\}$
    b. Plot the test accuracies against the different values of decay parameter.

c. State the rationale for selecting the optimal decay parameter

Ans Q4

The source code for question 4 is "PartAq4.py". The "training error vs epochs" and "accuracy vs epochs" for the 5 different decay parameters are plotted in Figure 14 to 18. We can see that for a decay parameter 10^{-6} and 10^{-12} the accuracy is dropping at around 1000 iterations. For a decay parameter of 0 the sudden drop is around 2000 iterations. For the network with a decay parameter of 10^{-3} and 10^{-9} the accuracy is kind of stable after 500 iterations, where the network with a decay parameter of 10^{-3} has a bigger drop at about 100 interactions compared to the network with a decay parameter of 10^{-9}. The network with a decay parameter of 10^{-9} is therefore selected as the optimal.

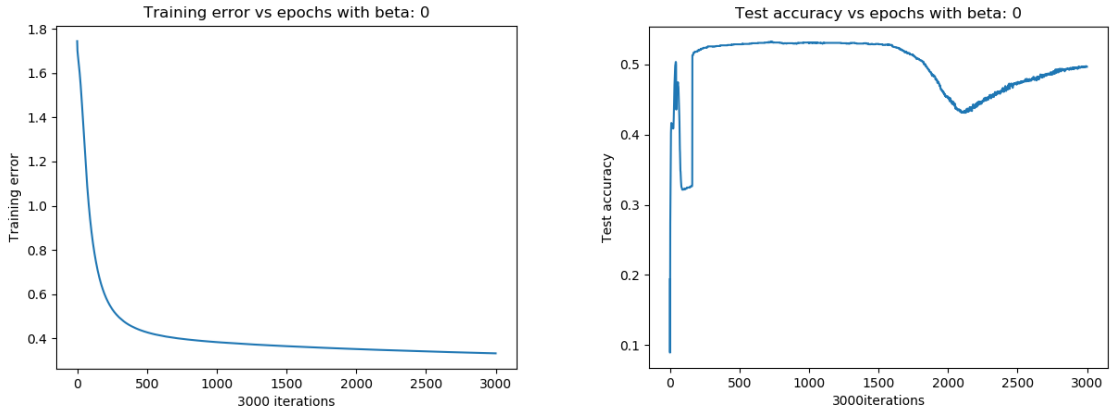


Figure 14: Decay parameter: 0. Training error vs epoch (left), and accuracy vs epoch (right)

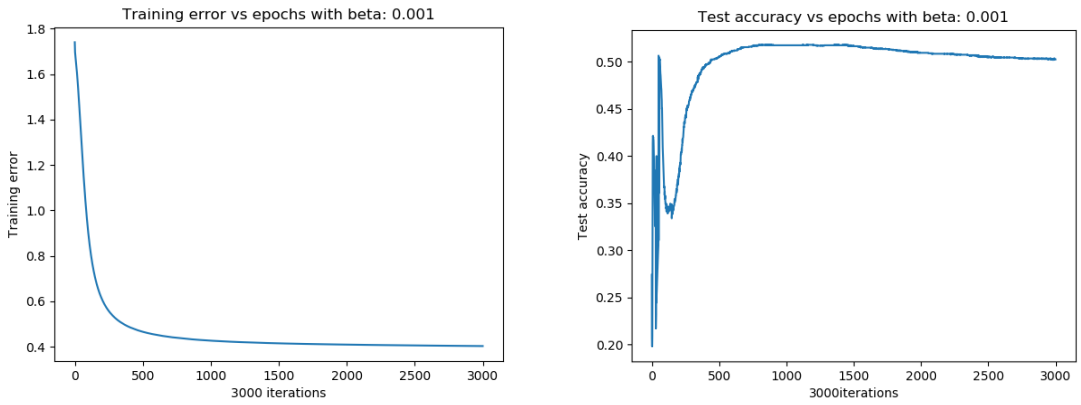


Figure 15: Decay parameter: 10^(-3). Training error vs epoch (left), and accuracy vs epoch (right)

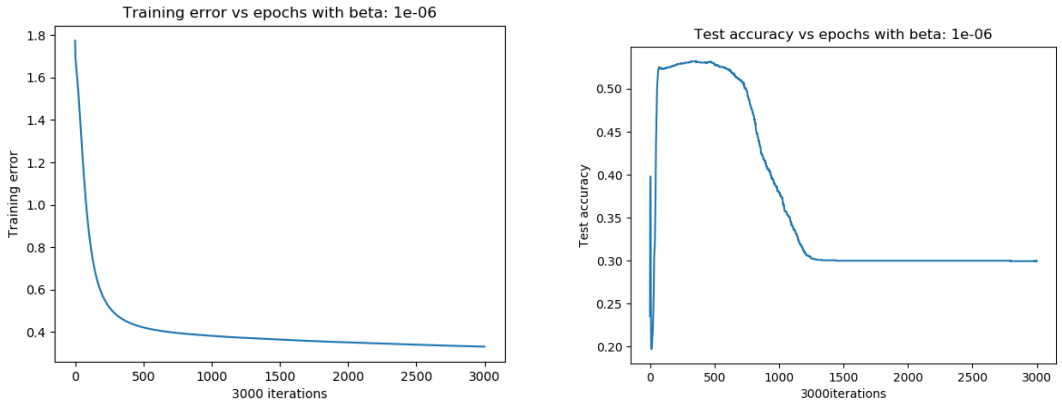


Figure 16: Decay parameter: 10^(-6). Training error vs epoch (left), and accuracy vs epoch (right)

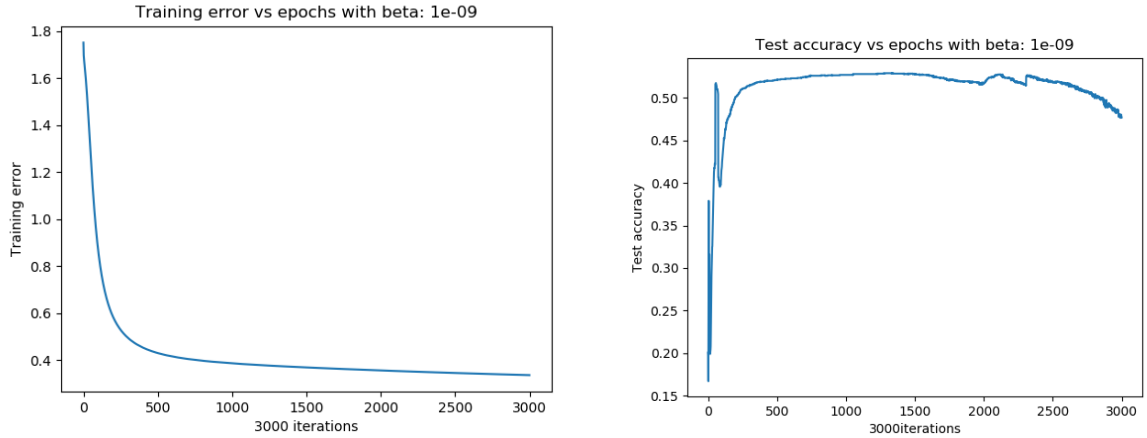


Figure 17: Decay parameter: 10^(-9). Training error vs epoch (left), and accuracy vs epoch (right)

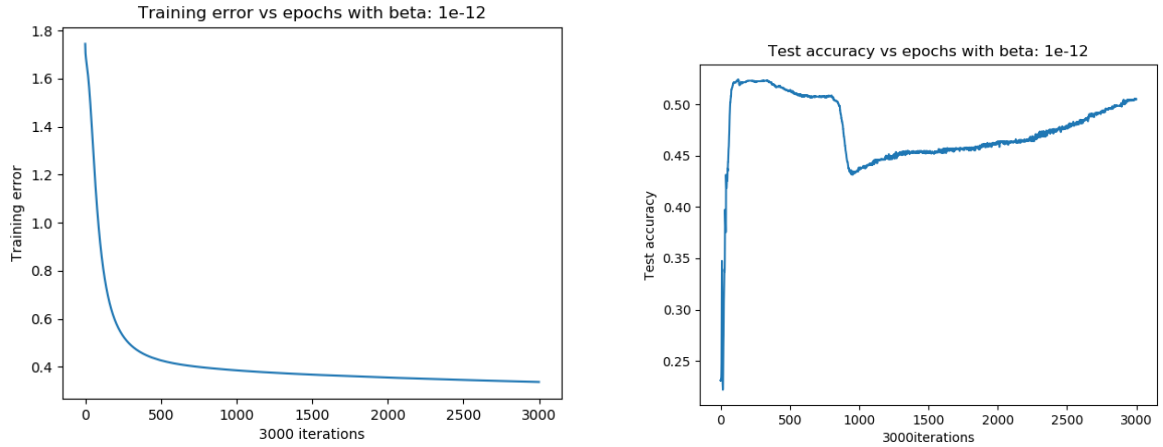


Figure 18: Decay parameter: 10^(-12). Training error vs epoch (left), and accuracy vs epoch (right)

5) After you are done with the 3-layer network, design a 4-layer network with two hidden layers, each consisting of 10 perceptrons, trained with a batch size of 32 and decay parameter 10^{-6}
    a. Plot the training and test accuracy of the 4 layer network
    b. Compare and comment on the performances of 3-layer and 4-layers networks

Ans Q5

The source code for question 5 is “PartAq5.py”. The “training error vs epochs” and “accuracy vs epochs” for the 4-layer network are plotted in Figure 19, and the optimal 3-layer is plotted in Figure 20. If we compare the graphs we see that the training error for the 3- layer network is converging faster than the 4- layer network, and the accuracy of the 3-layer network is more stable than the 4-layer network. We can therefore conclude that the 3-layer network is the best for this dataset.

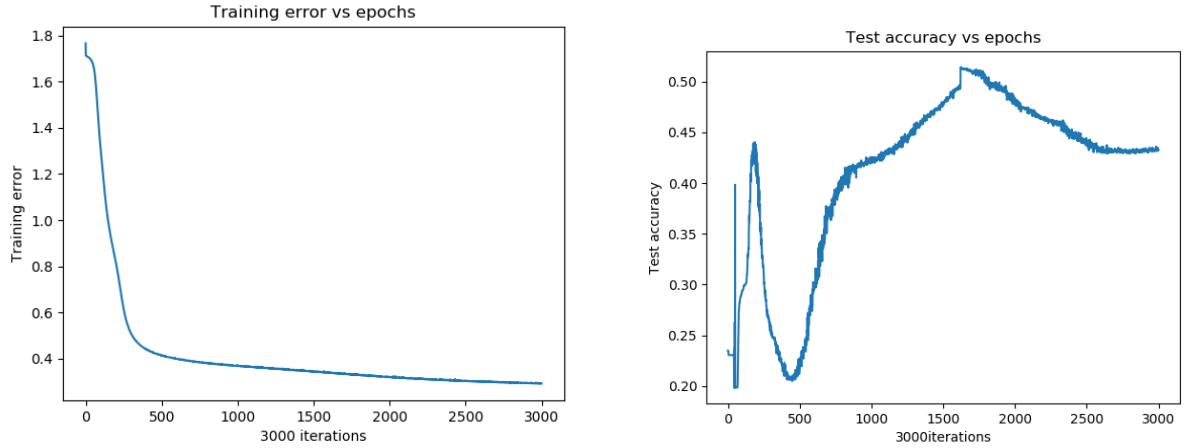


Figure 19: Training error vs epochs (left) and test accuracy vs epochs (right) for a 4-layer neural network

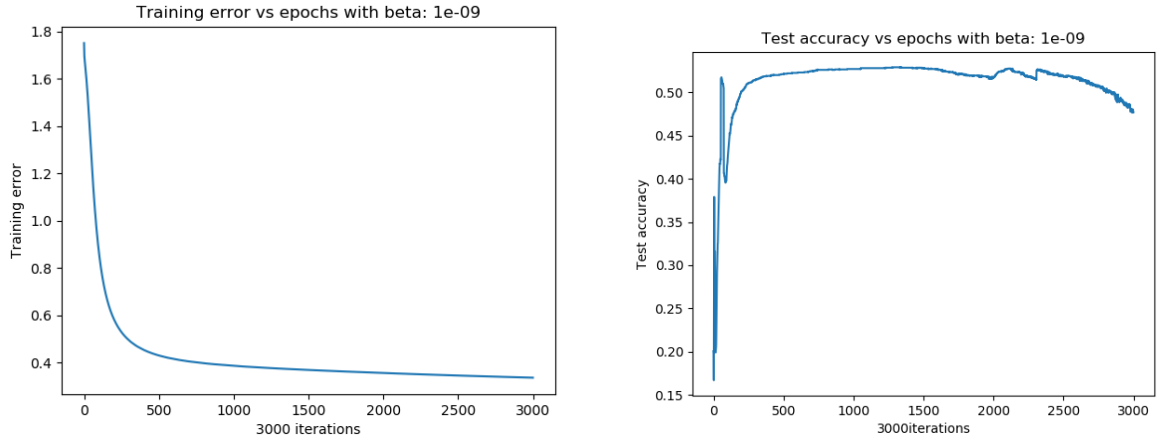


Figure 20: Training error vs epochs (left) and test accuracy vs epochs (right) for the optimal 3-layer NN. Beta: 10^(-9), Hidden neurons: 15, batch size: 64

Part B: Regression problem

This assignment uses the data from the California Housing database that contains attributes of housing complexes in California such as location, dimensions, etc., and their corresponding prices. The aim is to predict the housing prices from the other attributes in the data. This task also involves model selection and you will have to select the parameters giving the lowest validation error for prediction.
Read the data from the file 'california_housing.data'. Each data sample is a row of 9 values: 8 input attributes and the median housing price as targets. Divide the dataset at 70:30 ratio for validation and testing datasets. For selection of the best models, use 5-fold cross-validation on the validation data. The performances should be evaluated on the test data.

1) Design a 3-layer feedforward neural network containing: An input layer, a hidden-layer of 30 neurons having ReLu activation function, and a linear output layer. Use mini-batch gradient descent with a batch size of 32, L_2 regularization at weight decay parameter $\beta = 10^{-3}$ and a learning rate $\alpha = 10^{-7}$ to train the network.
    a. Use the validation dataset to train the model and plot validation errors against epochs.
    b. Plot the predicted values and target values for any 50 test samples.

Ans Q1:
The source code for Question 1 is "PartBq1.py". The network is trained with the whole training set with over 600 epochs and the validation error is plotted against the epochs in Figure 21. The "predictions vs targets" is plotted in Figure 22, where the left figure is the prediction from the known data and the right figure is the predictions from the test data. We can see that the cross-validation error is not smooth, which means that the learning rate needs to be adjusted. We see the error on figure 22 (right) is really big, but it is only a miss prediction of 10^8 ~ $10.000.

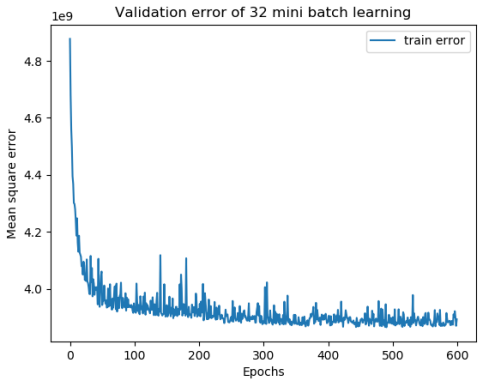


Figure 21: Cross-Validation error vs Epochs

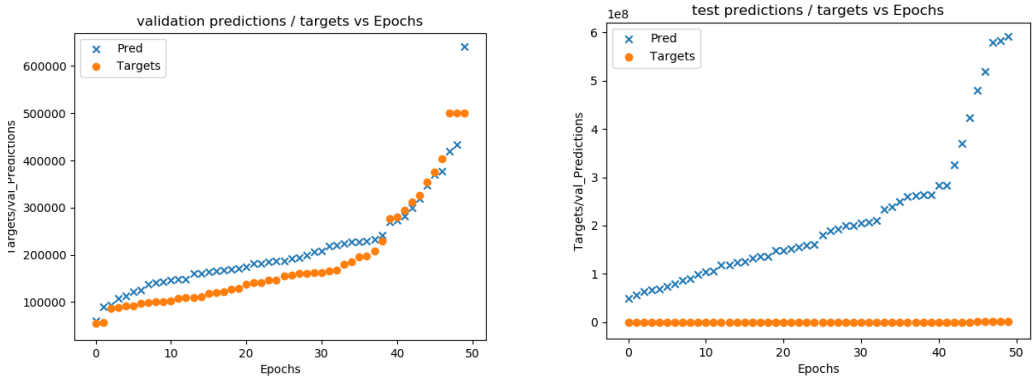


Figure 22: Validation predictions and labels (right) and test prediction and labels (left)

2) Find the optimal learning rate for the 3-layer network designed using 5-fold cross-validation on validation data. Let the search space be: $\{0.5 \cdot 10^{-6}, 10^{-7}, 10^{-8}, 10^{-9}, 10^{-10}\}$
   a. Plot cross-validation errors achieved different learning rates
   b. For the optimal learning rate, plot the test error against training epochs

Ans Q2

The source code for Question 2 is "PartBq2.py". The "cross-validation error of the 5 fold training vs epochs" are plotted in Figure 23 to 27, where the left plot is the error for each individual fold and the right plot is the average.

We can see in Figure 23 and 24, where the learning rate is $0.5 \cdot 10^{-6}$ and 10^{-7}, respectively, that the plot is not smooth which means that the learning rate is too high. On Figure 27, where the learning rate is 10^{-10}, the mean square error is not converging doing the epoch time, which means that the learning rate is too low. If we look at Figure 25 and 26 we see that the mean square error for a learning rate on $0.5 \cdot 10^{-8}$ and 10^{-9} converge to about the same mean square error. The test error for learning rate $0.5 \cdot 10^{-8}$ and 10^{-9} is therefore plotted on figure 28 to be compared.

On Figure 28 we see that the network with learning rate $0.5 \cdot 10^{-8}$ has a smaller window where it does not overfit compared to the network with learning rate 10^{-9}.

A learning rate on 10^{-9} is therefore selected as the optimal learning rate.

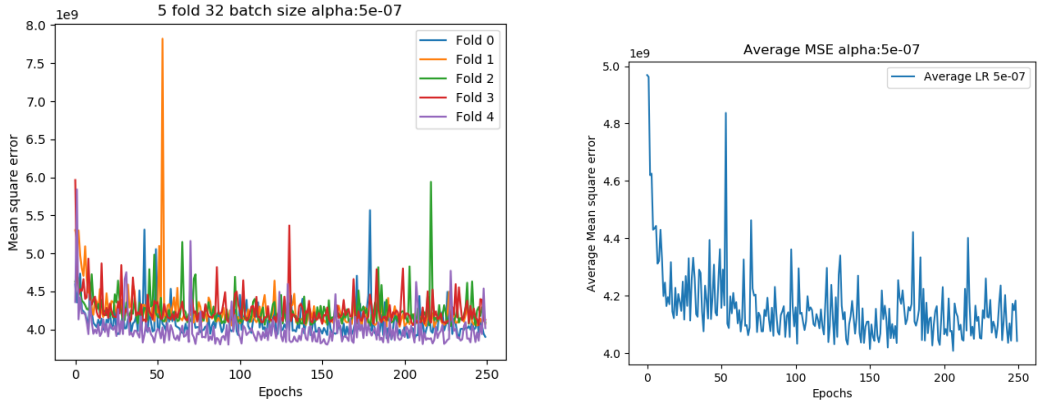


*Figure 23: Learning rate 0.5*10^(-6). Mean square error per fold vs Epoch (left) and the avarege Mean square error vs Epoch (right)*

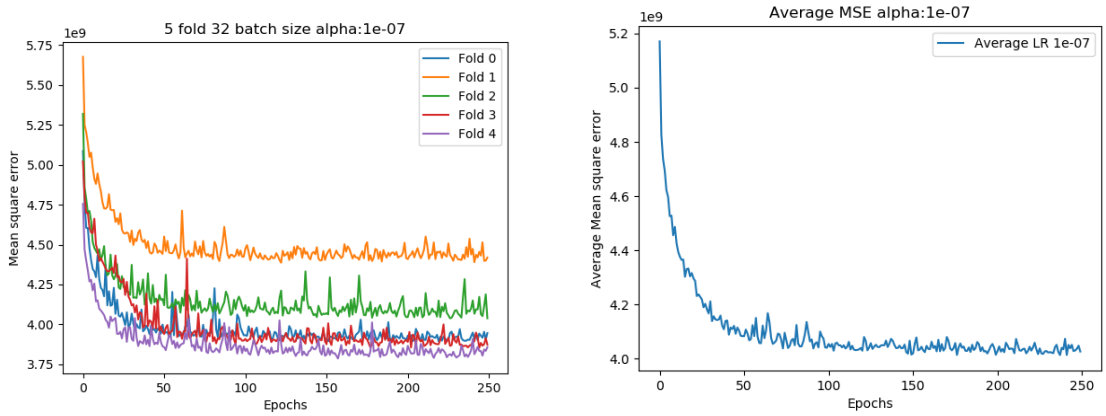


Figure 24: Learning rate 10^(-7). Mean square error per fold vs Epoch (left) and the avarege Mean square error vs Epoch (right)

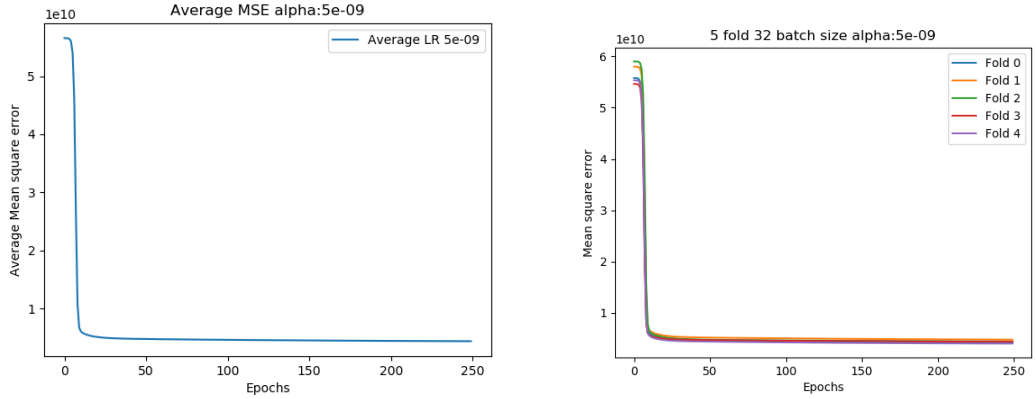


*Figure 25: Learning rate 0.5*10^(-8). Mean square error per fold vs Epoch (left) and the avarege Mean square error vs Epoch (right)*

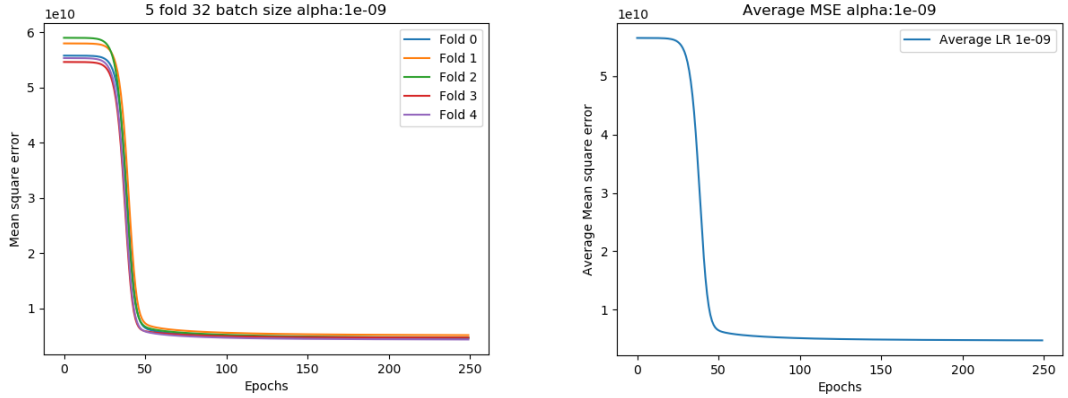


Figure 26: Learning rate 10^(-9). Mean square error per fold vs Epoch (left) and the avarege Mean square error vs Epoch (right)

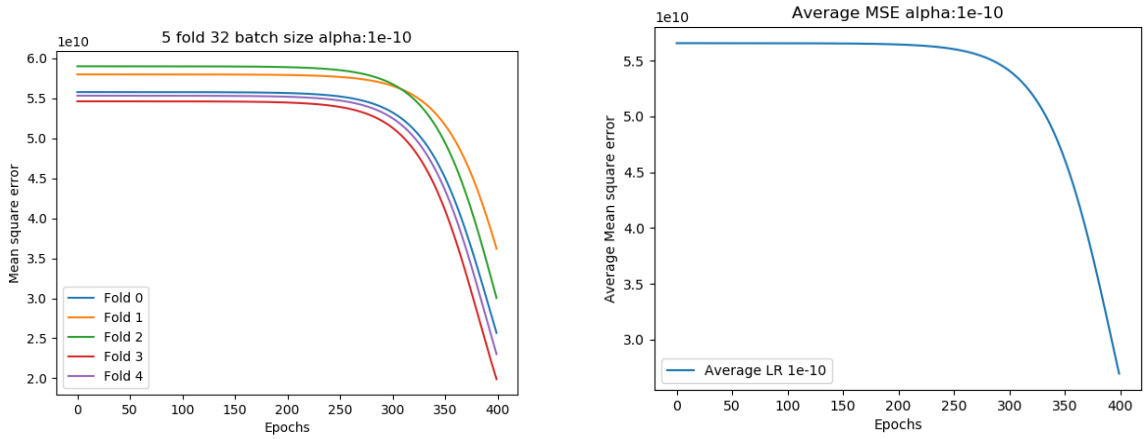
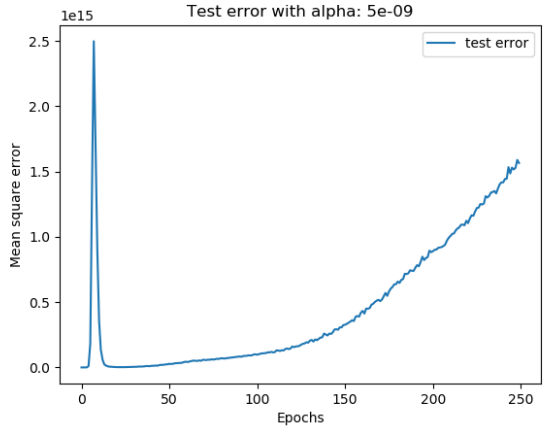
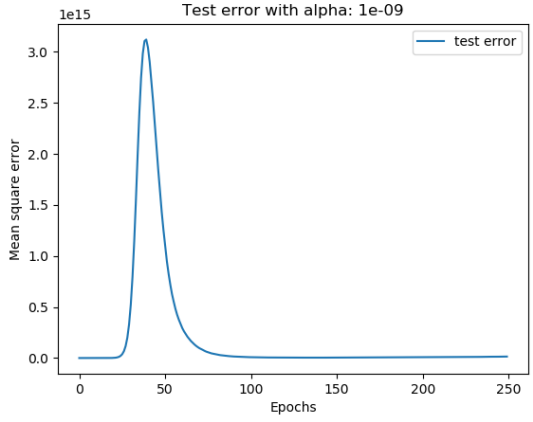


Figure 27: Learning rate 10^(-10). Mean square error per fold vs Epoch (left) and the avarege Mean square error vs Epoch (right)




```
step 0, error: train 5.6541e+10, test 5.25166e+10
step 100, error: train 4.48137e+09, test 9.74651e+13
step 200, error: train 4.28812e+09, test 8.89738e+14
step 249, error: train 4.23035e+09, test 1.56517e+15
```




```
step 0, error: train 5.65433e+10, test 5.43409e+10
step 100, error: train 4.9317e+09, test 8.11652e+12
step 200, error: train 4.70354e+09, test 7.9018e+12
step 249, error: train 4.65975e+09, test 1.44291e+13
```

*Figure 28: Test error for learning rate 0.5*10^(-8) (top) and 10^(-9) (button) and their test error*

3) Find the optimal number of hidden neurons for the 3-layer network designed. Limit search space to: {20, 40, 60, 80, 100}. Use the learning rate from part (2).
    a. Plot the cross- validation errors against the number of hidden - layer neurons.

b. Plot the test errors against number of epochs for the network consisting of the optimal number of hidden neurons.
c. State the rationale behind selecting the optimal number of hidden neurons

Ans Q3

The source code for Question on is "PartBq3.py". The cross-validation error of the 5-fold training versus epochs are plotted on Figure 29 to 33, where the left plot is the error for each individual fold and the right is the average error. We can see that the plots are close to identical and it is therefore necessary to select the optimal number of neurons by comparing the test square errors for every number of hidden neurons, which is plotted on Figure 34. We can see that 20 hidden neurons result in the lowest test square error.

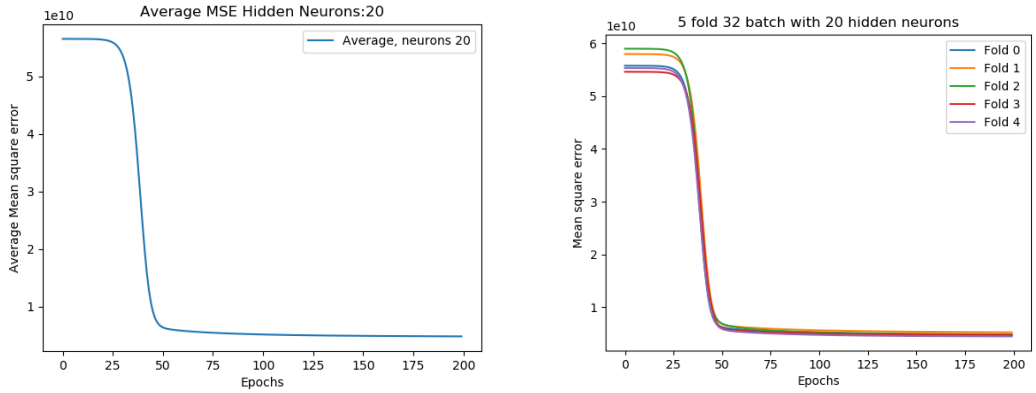


Figure 29: Hidden neurons 20. Mean square error per fold vs Epoch (left) and the avarege Mean square error vs Epoch (right)

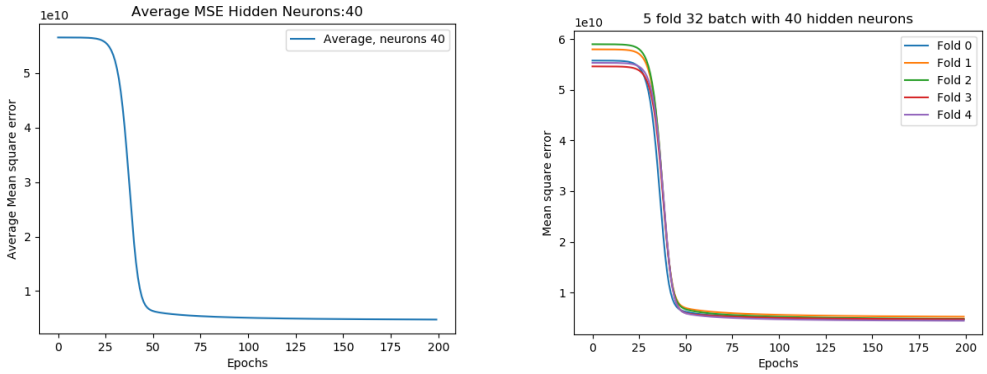


Figure 30: Hidden neurons 40. Mean square error per fold vs Epoch (left) and the avarege Mean square error vs Epoch (right)

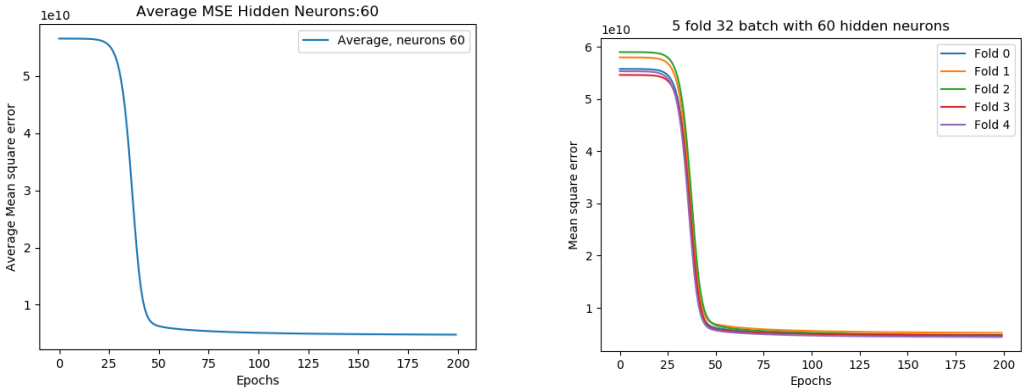


Figure 31: Hidden neurons 60. Mean square error per fold vs Epoch (left) and the avarege Mean square error vs Epoch (right)

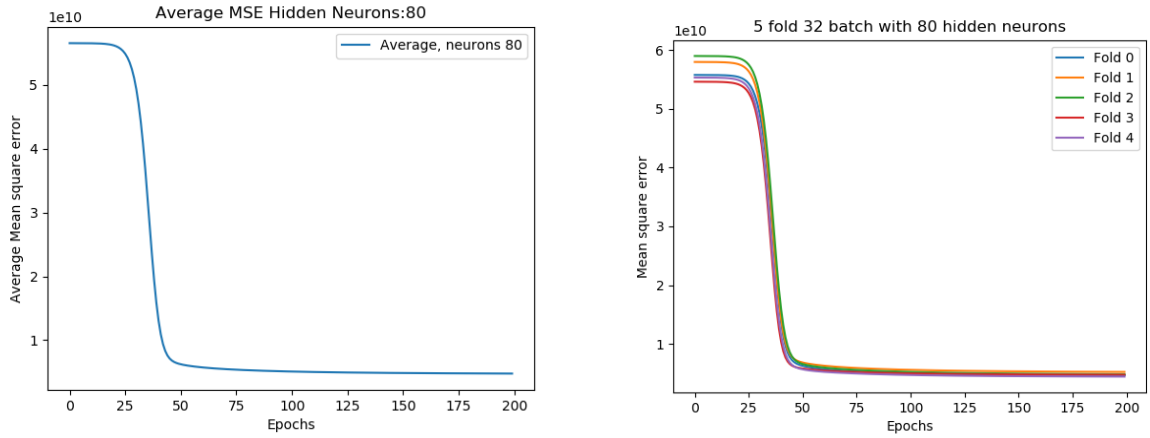


Figure 32: Hidden neurons 80. Mean square error per fold vs Epoch (left) and the avarege Mean square error vs Epoch (right)

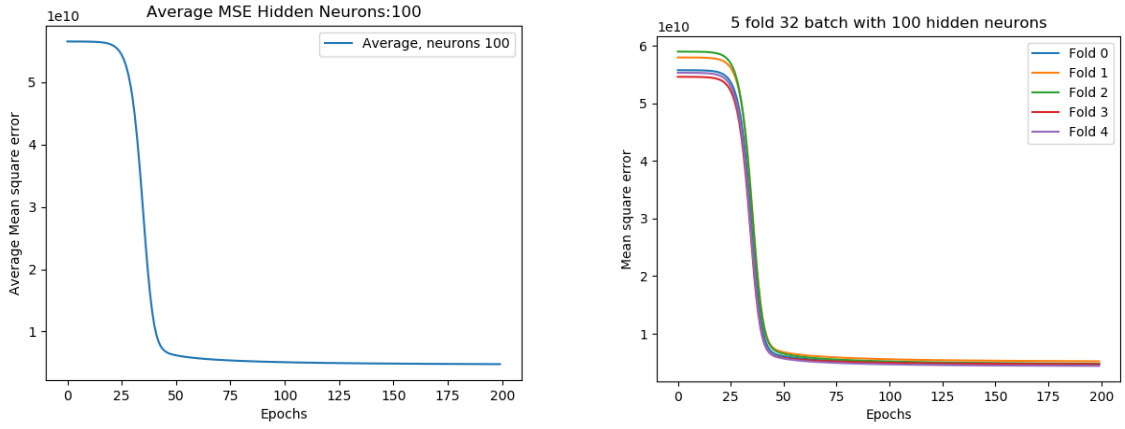
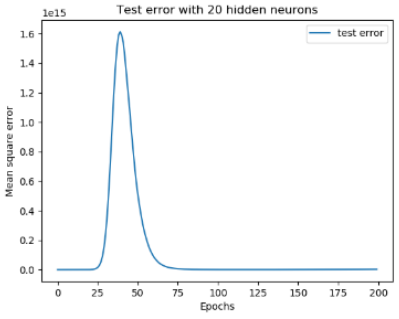
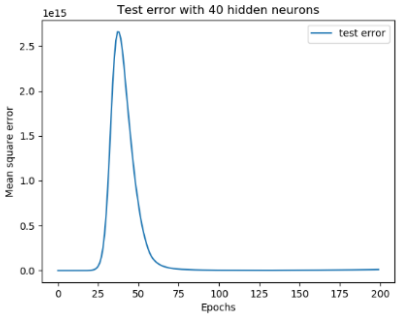


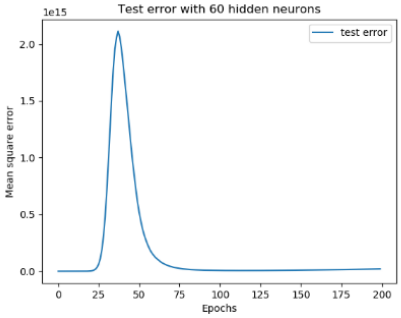
Figure 33: Hidden neurons 100. Mean square error per fold vs Epoch (left) and the avarege Mean square error vs Epoch (right)



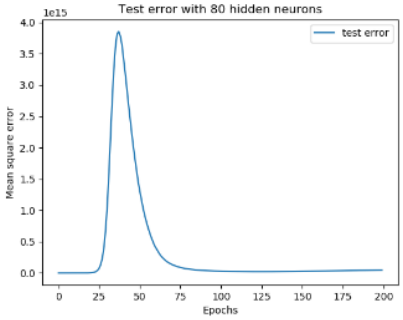

```
step 0, error: train 5.65435e+10, test 5.45635e+10
step 100, error: train 4.94225e+09, test 7.93938e+11
step 199, error: train 4.7013e+09, test 3.18212e+12
```



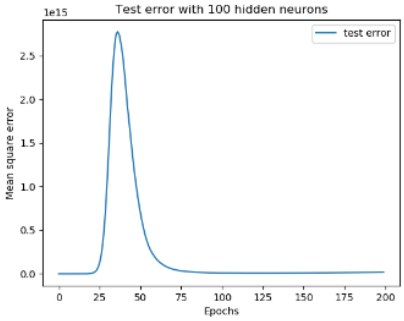

```
step 0, error: train 5.6543e+10, test 5.4333e+10
step 100, error: train 4.93274e+09, test 3.36545e+12
step 199, error: train 4.71058e+09, test 1.15646e+13
```




```
step 0, error: train 5.6543e+10, test 5.40918e+10
step 100, error: train 4.92422e+09, test 7.12287e+12
step 199, error: train 4.69717e+09, test 2.06369e+13
```




```
step 0, error: train 5.65427e+10, test 5.39547e+10
step 100, error: train 4.91968e+09, test 2.87457e+13
step 199, error: train 4.7027e+09, test 4.7116e+13
```




```
step 0, error: train 5.65429e+10, test 5.42381e+10
step 100, error: train 4.91265e+09, test 9.99443e+12
step 199, error: train 4.69153e+09, test 1.90933e+13
```

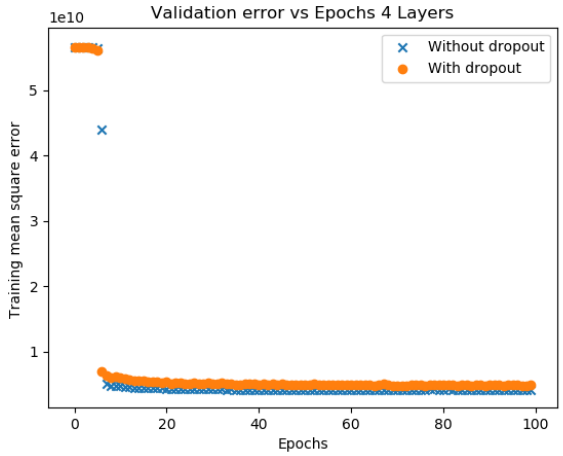
Figure 34: Mean square error versus Epochs for 20, 40, 60, 80 and 100 hidden neurons.

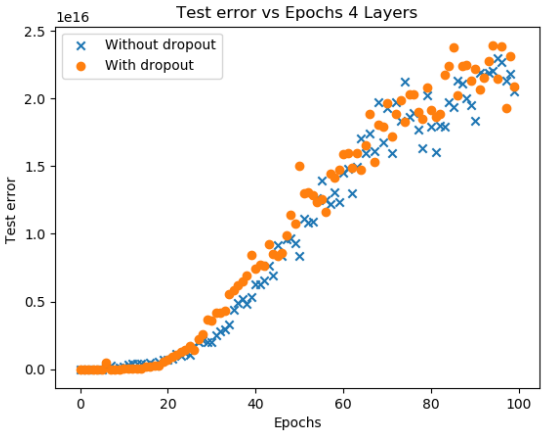
4) Design a four-layer neural network and a five-layer neural network, with the first hidden layer having the number of neurons found in step (3) and other hidden layers having 20 neurons each. Use a learning rate of $\alpha = 10^{-9}$ for all layers. Train four-layer and five-layer networks on validation data and compare their test errors on test data with those on the three-layer networks.

Introduce dropouts (with a keep probability of 0.9) to the layers and report test errors with and without dropouts.

Ans Q4

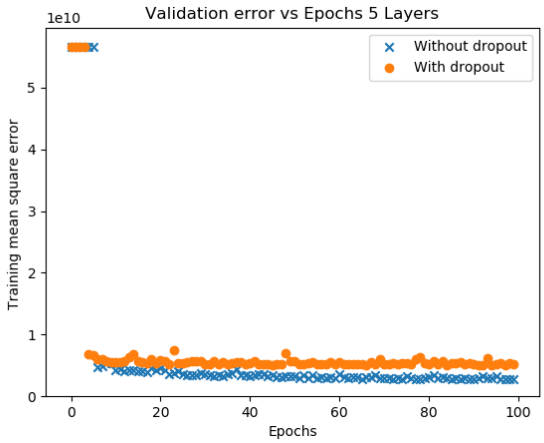
The source code for Question on is "PartBq4.py". The "validation error vs epochs" and the "test error vs epoch" is plotted on Figure 35 for the 4- layer network and Figure 36 for the 5-layer system.
We can see that the validation error converges really fast, a few epochs for both the 4- and 5- layer network. Both networks start overfitting around the 15 epochs for the 4-layer and around the 10 epochs for the 5-layer. A 3-layer network will therefore be a better option.

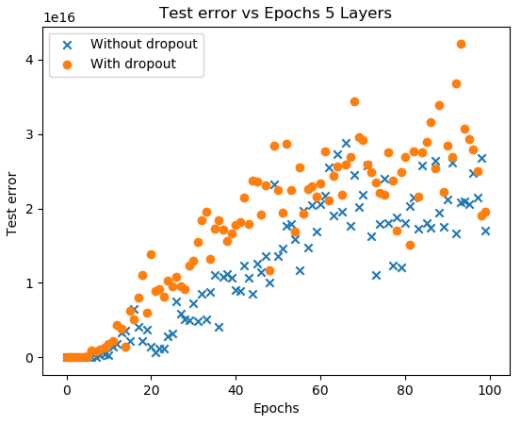





```
step 0, error: train 5.65434e+10, test 5.50713e+10
step 10, error: train 4.65558e+09, test 2.03944e+14
step 20, error: train 4.30863e+09, test 7.10264e+14
step 30, error: train 4.20795e+09, test 1.96842e+15
step 40, error: train 4.10865e+09, test 6.28303e+15
step 0, error: train 5.65433e+10, test 5.46701e+10
step 10, error: train 5.92067e+09, test 7.62531e+12
step 20, error: train 5.29205e+09, test 7.14556e+14
step 30, error: train 5.10392e+09, test 3.57045e+15
step 40, error: train 4.93181e+09, test 7.41882e+15
```

Figure 35: Training error vs epochs for 4-layer (left) and test error vs Epochs (left)






```
step 0, error: train 5.65436e+10, test 5.51201e+10
step 10, error: train 4.2063e+09, test 3.5793e+14
step 20, error: train 4.61488e+09, test 1.44388e+15
step 30, error: train 3.52082e+09, test 7.22899e+15
step 40, error: train 3.21795e+09, test 8.98738e+15
step 0, error: train 5.65434e+10, test 5.46157e+10
step 10, error: train 5.50438e+09, test 1.82117e+15
step 20, error: train 5.85144e+09, test 1.38198e+16
step 30, error: train 5.25973e+09, test 1.29876e+16
step 40, error: train 5.32019e+09, test 1.77567e+16
```

Figure 36: Training error vs epochs for 4-layer (left) and test error vs Epochs (left)

Conclusion

It can be concluded that for both the classification problem and the regression problem, that a simple 3- layer network is better than a 4- or 5- layer problem.
It has been observed that network that have a lot of training data – the regression problem – suffer earlier from overfitting, and it is therefor necessary to run through less epochs, and thereby make the training faster. The all experiments in partB have been carried out with 600 epoch the first time the experiments was done, which yielded some confusing at first. All the experiments was then redone with about 200 epochs which gave the above results.