

CZ4042 Neural Network and Deep Learning

Project 1

Submitted by:

Brigitta Angelina (U1520174H)

Fang Ran (U1521819L)

Part A: Classification Problem

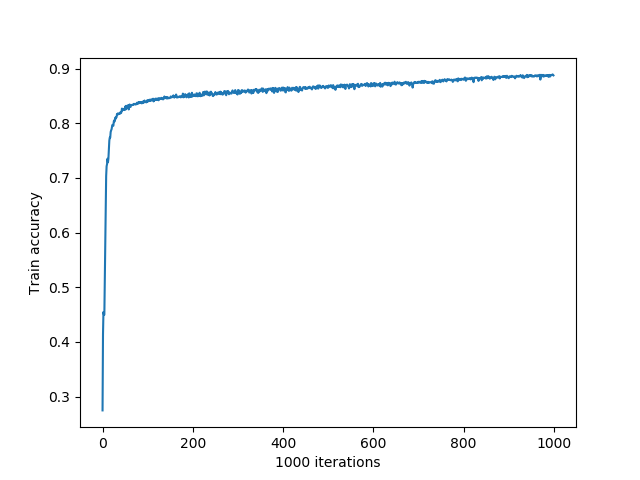
Introduction

The objective of the first part of this project is to build a neural networks to classify the landsat satellite data that is taken from <https://archive.ics.uci.edu/ml/datasets/Statlog+(Landsat+Satellite)>. The dataset contains multispectral values of pixels in a 3x3 neighbourhoods in satellite images and the classification associated with the central pixel in each neighbourhood. The aim is to predict class labels in the test dataset after training the neural network after training dataset.

The data provided has already been split into training dataset and testing dataset called sat\_train.txt and sat\_test.txt respectively. The dataset is a sub-area of a scene , consisting of 82 x 100 pixels. Each line of data corresponds to the 3 x 3 square neighbourhood of pixels which is the pixel values in 4 spectral bands of each of the 9 pixels in the 3 x 3 square neighbourhood and the number indicating the classification label of the central pixel. There are 7 classification labels, however there are no examples for number 6. Hence, appropriate solution has been made in the code.

Task 1: Design a 3 -layer feed forward neural network

As this is a classification problem, the 3-layer neural network consists of an input layer, a hidden perceptron layer and an output softmax layer. The neuron used for the hidden layer is 10 neurons, the input node is 36 according to the number of features taken from the data and the number of output labels is 6. The hidden layer used ReLu activation function while the output layer used softmax layer. The values of the learning rate is 0.01, weight decay parameters (𝛽) = 10−6 and batch size is 32.



This graph represented the training accuracy of the training data.

Task 2: Find optimal batch size

After training the 3-layer neural network with different batch sizes namely, 4,8,16,32,64. We plot the following graphs:

1. Training error by loss function against the number of epochs

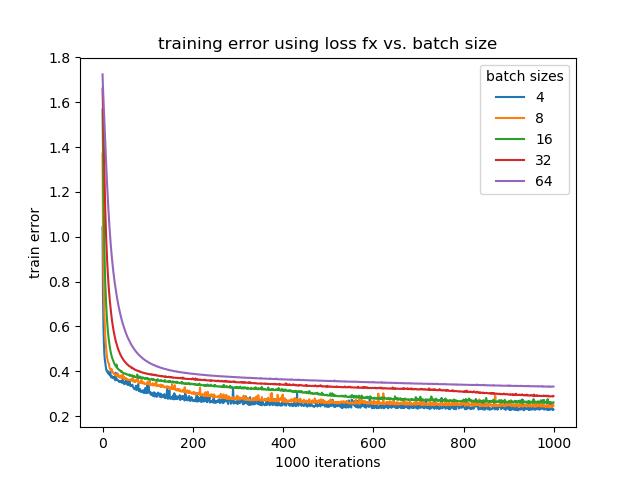


Fig 2.1

From this graph we can see the batch size that has the least error is batch size of 4 while batch size of 64 has the biggest training error.

1. Training error by using the accuracy function(1-accuracy) against the number of epochs

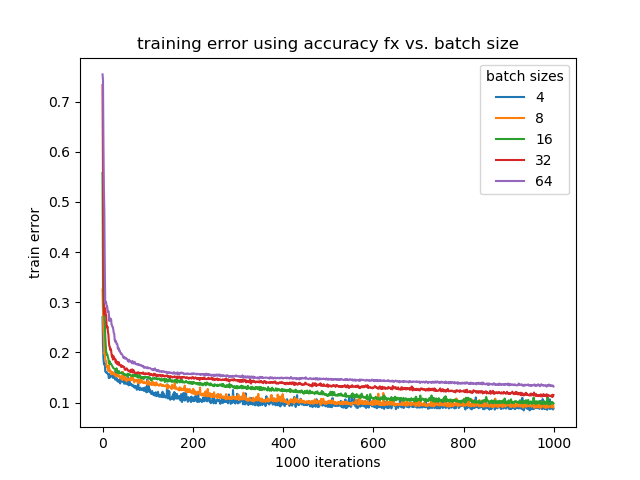


Fig 2.2

From this graph we can conclude that the best batch size that results in the least error is batch size of 4.

1. Test accuracy against the number of epochs

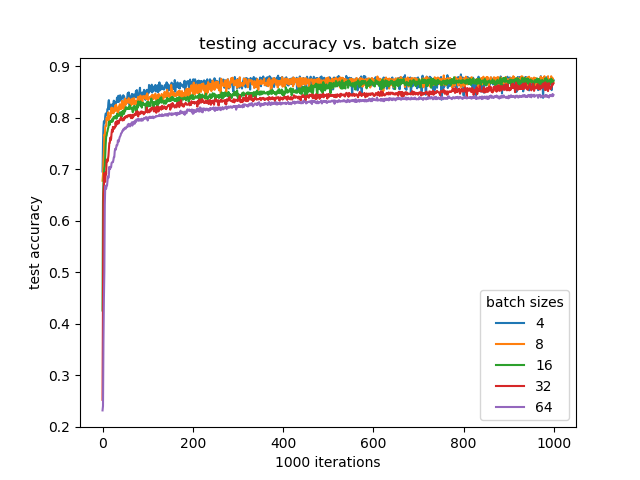


Fig 2.3

From this graph we can see that in the overall accuracy of all the different batches that we use excluding batch size of 64 all the batch sizes has similar accuracy.

1. Time taken for 1 epoch against batch sizes

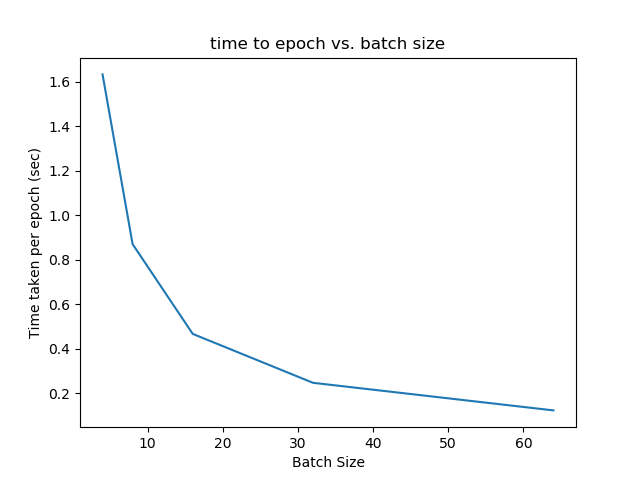


Fig 2.4

From this graph, we can surmise that as the number of batch size increases the time taken per epoch is decreasing.

In conclusion, the optimal batch size is 16. As we can see from figure 2.1, the batch size of 4,8, and 16 has similar convergence point while there is a bit of agap to the training error from batch size 32 and 64. In figure 2.2 we can also see the same thing as the graph in figure 1 as both of them is training error from different method. Then in figure 2.3, test accuracy of all the batch size except batch size 64 has similar result. But while it has similar result, batch size 32 in figure 2.3 has a little of gap then the other batch sizes when they has little to nothing gap. After figure 2.4 is taken into consideration, we think that batch size of 16 is the best. As even though it is not the fastest one but considering the time take for one epoch, training error and test accuracy, batch size 16 is the most optimal batch size.

Task 3: Find optimal number of hidden neuron

1. Training error by loss function against number of neurons

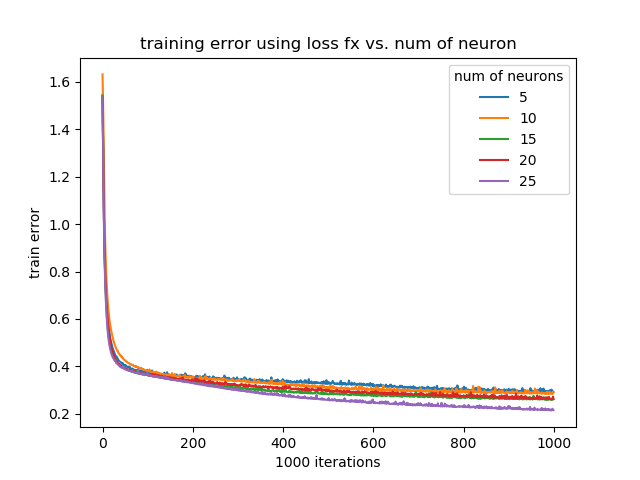


Fig 3.1

This graph is the training error acquired by evaluating the average of classification error.

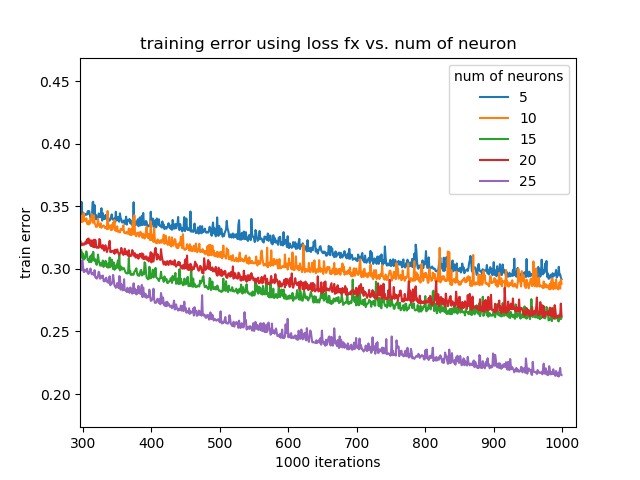


Fig 3.2

This is the zoom version of figure 3.1.We can look at the graph more clearly and find that according to the graph the best number of neurons is the biggest number there is which is 25 neurons.

1. Training error by the accuracy function(1-accuracy) against number of neurons

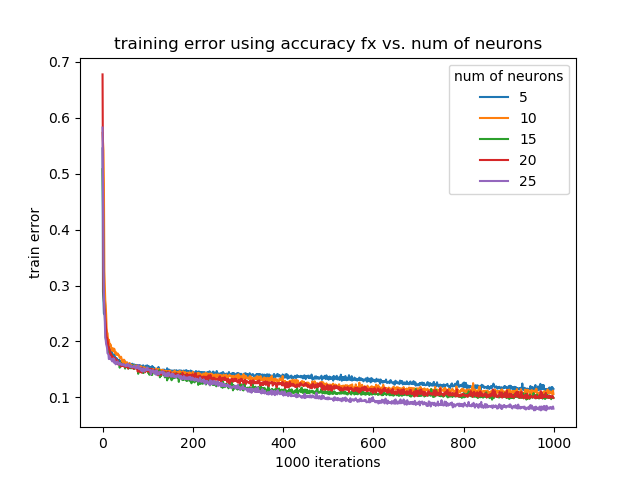


Fig 3.3

This is graph of training error is taken from calculating 1 - the accuracy of the the predicted data as the 1- accuracy is also the error.

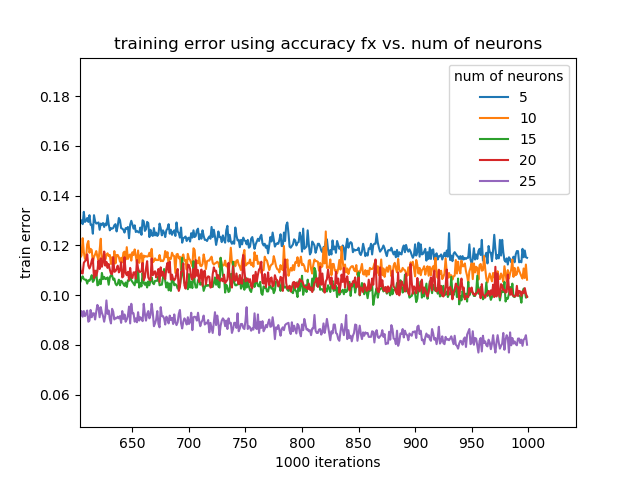


Fig 3.4

This is the zoom version of figure 3.3. We can look more clearly and find out that according to this graph the best number of hidden neurons is 25.

1. Test accuracy against the number of neurons

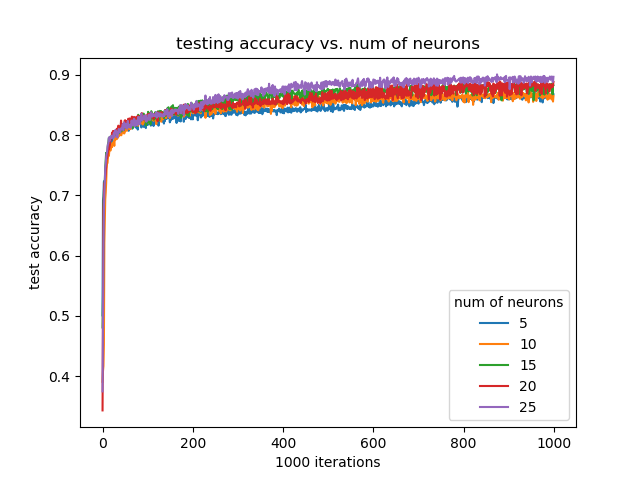


Fig 3.5

This graph represent the test accuracy as the iterations progresses.

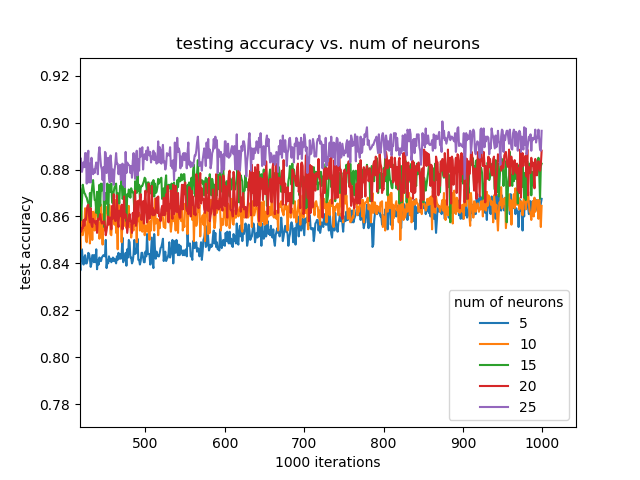


Fig 3.6

This is the zoom version of figure 3.5. According to this graph the best number of neurons is 25 neurons.

1. Time taken for 1 epoch for different number of neurons

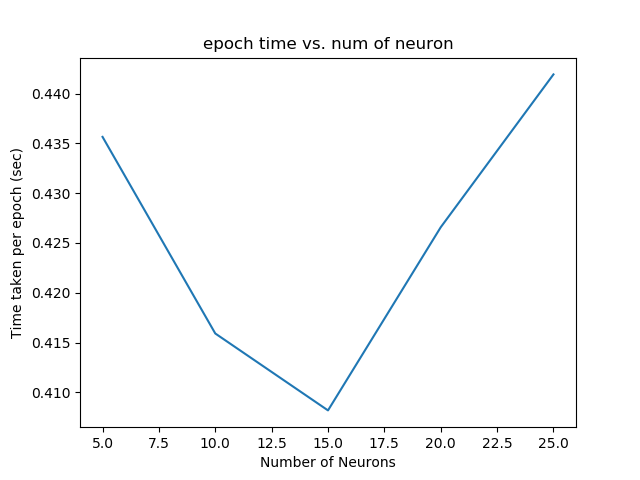


Fig 3.7

This figure shows the time take per epoch against the number of neurons used. We can surmise from this figure that 15 when the training is using 15 hidden neurons the time taken per epoch is fastest.

In conclusion, from figure 3.2, figure 3.4, and figure 3.5, the number of neurons that is the most optimal is 25 as it achieves the lowest error and the highest accuracy. However, we also have to take account of the time taken per epoch. When the number of neuron used increases, the network attempts to remember the training pattern as it tries to minimize the training error at the expense of generalization of unseen data therefore even though the highest number of neuron which is in this case 25 may receiver the lowest error and the highest accuracy it is not certain yet whether it is the most optimal or not. But as the accuracy and error of 25 neurons compared to 15 neurons has only a little differences considering time factor 15 achieves the fastest time per epoch at 0.408 sec per epoch while 25 neurons achieves 0.442 sec per epoch, if we calculate it per 1000 epochs they has a difference of 40 sec. Hence it is better to take the fastest one which is 15 neurons as it can try building the network with more data, increasing the accuracy rather than remembering the training patterns as what higher number of neurons do.

Task 4: Find the optimal weight decay parameter

1. Training error by the loss function against the weight decay parameters

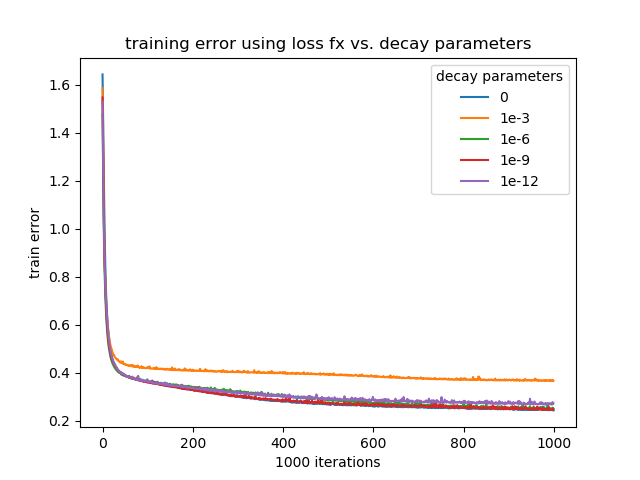


Fig 4.1

This is the figure taken by comparing all training error that is taken from the average of classification error according to 5 different weight decay parameters.

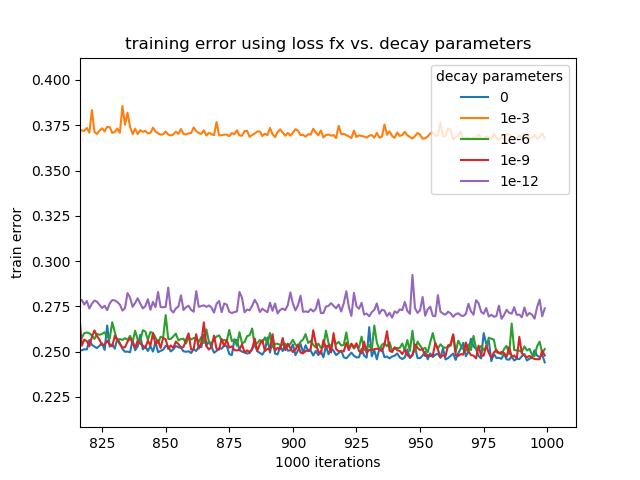


Fig 4.2

This is the zoom version of the figure 4.1. We can look more clearly that the weight decay parameters of 0, 1e-6, 1e-9 have similar results.

1. Training error by the accuracy function(1-accuracy) against the weight decay parameter

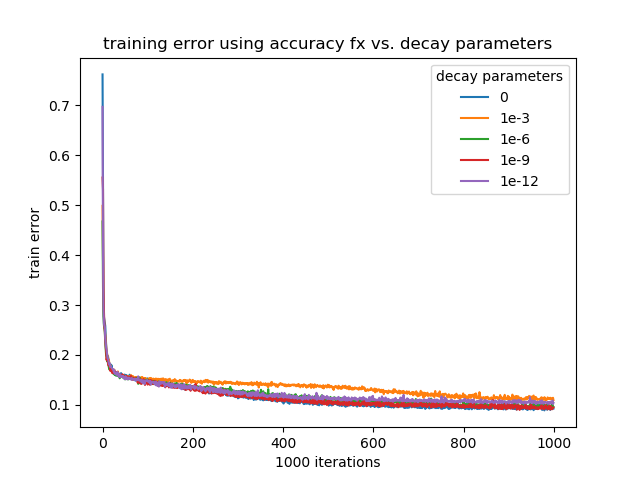


Fig 4.3

This is the training error taken from accuracy function.

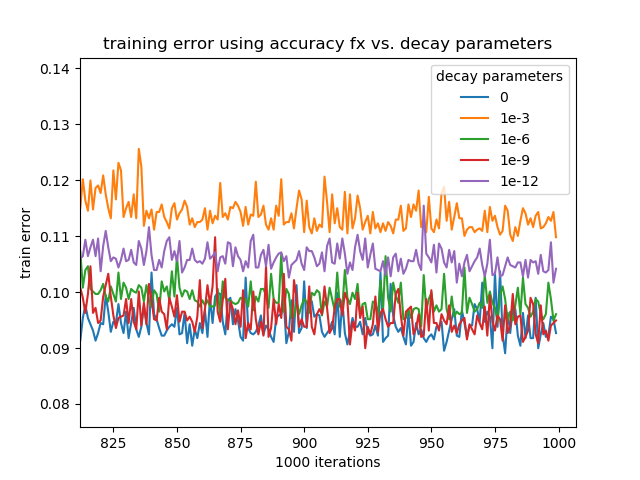


Fig 4.5

This is the zoom version of fig 4.4. From this figure we can surmise that 0, 1e-6, and 1e-9 have resulted in similar values.

1. Test accuracy against the weight decay parameter

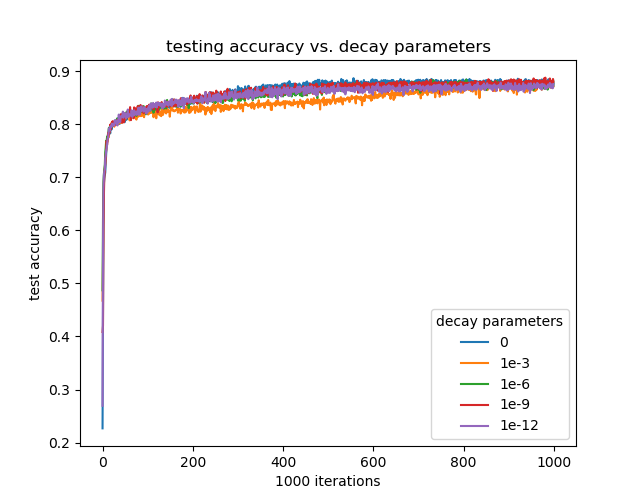


Fig 4.6

This figure shows the test accuracy by using the different weight decay parameters.

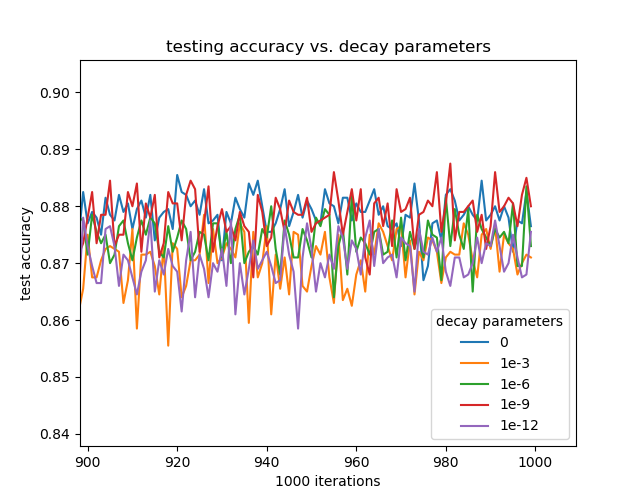


Fig 4.7

This figure is the zoom version of figure 4.6. From this graph we conclude that weight parameter of 1e-9 has the best test accuracy compared to other decay parameters.

In conclusion, 1e-9 is the most optimal weight decay parameter as looking through the training error the best decay parameter is either without it, 1e-6 or 1e-6. But in the test accuracy data which we can see clearly in fig 4.7 we can see that by using weight decay parameters of 1e-9, the test accuracy is the best.

Task 5: Design a 4-layer neural network

We design a 4-layer neural network with 2 hidden neuron layers, each consisting of 10 perceptrons. It was trained with a batch size of 32 and has a weight decay parameter of 10-6. After that we try to compare it with the 3-layer neural network and plot the following graph:

1. Test accuracy

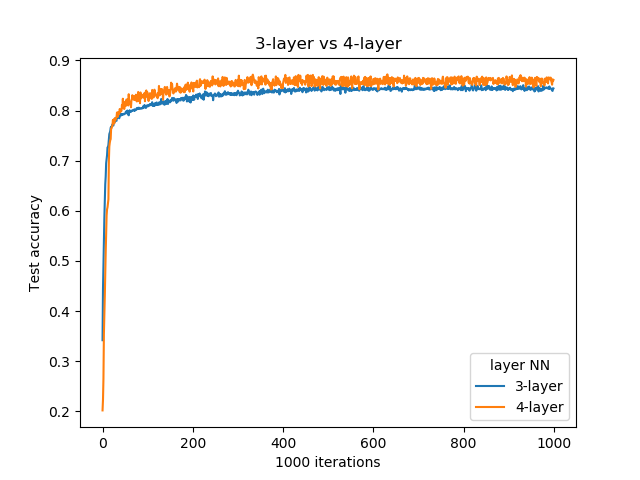


Fig 5.1

This figure shows the test accuracy of the 3-layer neural network compared to the test accuracy of the 4-layer neural network

1. Train accuracy

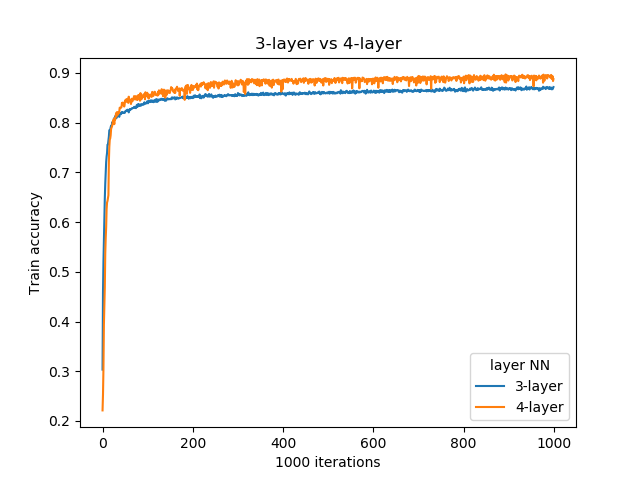


Fig 5.2

This graph shows the train accuracy of the 3-layer neural network with train accuracy of the 4-layer network

1. Train error using loss function

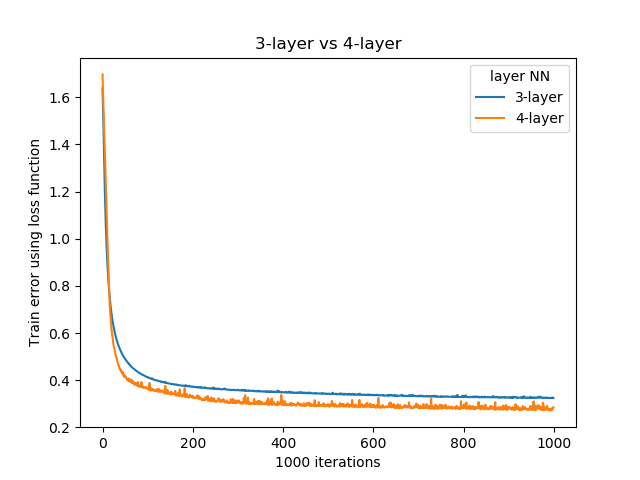


Fig 5.3

This graph shows the training error of 3-layer neural network and 4-layer neural network.

1. Train error using accuracy function(1-accuracy)

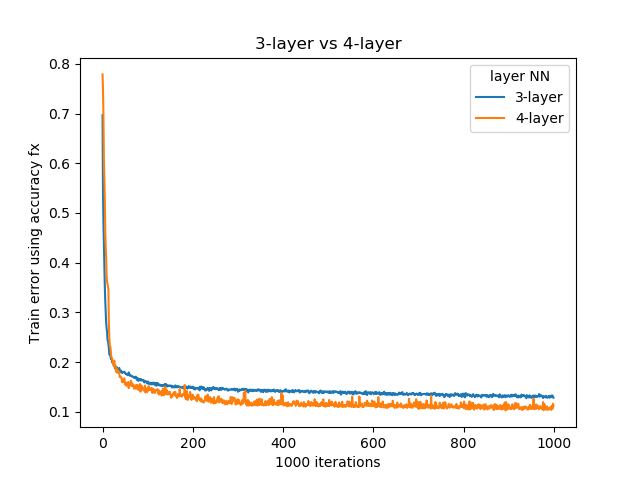


Fig 5.4

This graph shows the training error of 3-layer neural network and 4-layer neural network.

In conclusion, we can see that 4-layer neural network is better than 3-layer neural network in all aspect. As we can see from Figure 5.1,5.2,5.3 and 5.4 all of them shows that 4-layer neural network is better than a 3-layer neural network.

Part B: Regression Problem

Task 1: Design a 3-layer feed forward neural network

1. Validation error against epochs.

* Number of epochs used: 500
* Mini-batch gradient descent with a batch size of 32
* L 2 regularization at weight decay parameter β = 10 −3
* Learning rate α = 10 −7

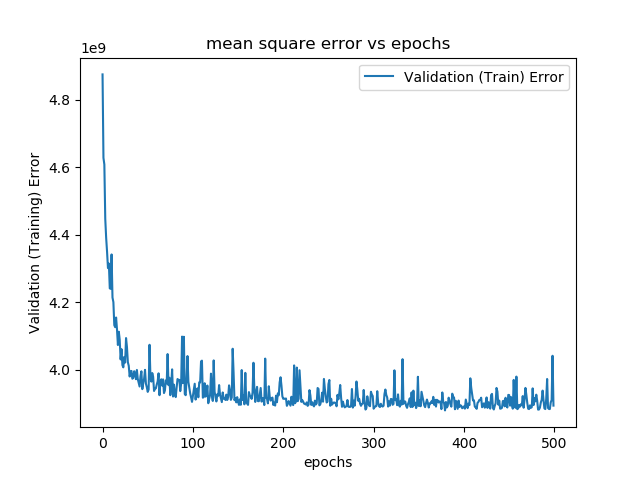


Fig 6.1

The error decreases with increasing epochs.

1. Plotting the predicted values and the target values for any 50 test samples

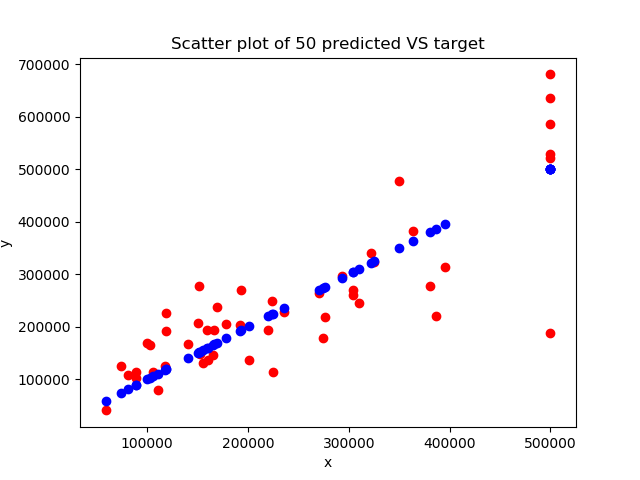


Fig 6.2

As shown here in this graph, the blue dots are the actual house prices. Our red dots (predicted values) are supposed to have a linear relationship since this model uses a linear output layer. The blue dots forms a straight line since they are the actual values while the red dots follows roughly around the blue dots.

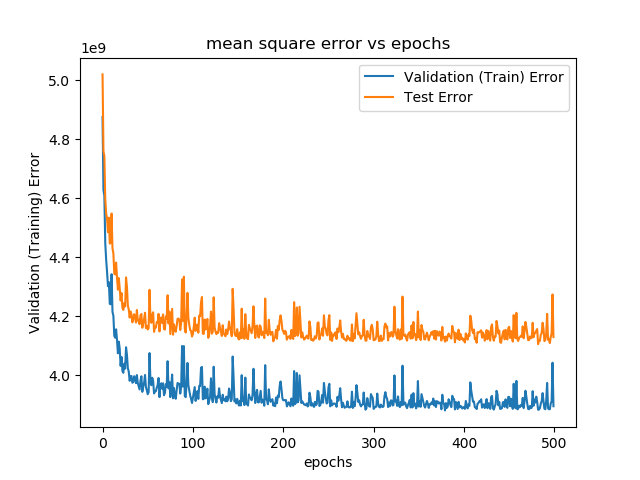


Fig 6.3

This graphs compares testing errors vs training errors.

Task 2: Find the optimal learning rate

Find the optimal learning rate for the 3-layer network designed using 5-fold cross-validation on validation data.

search space {0.5 × e −6 , e −7 , 0.5 ×e −8 , e −9 , e −10 }

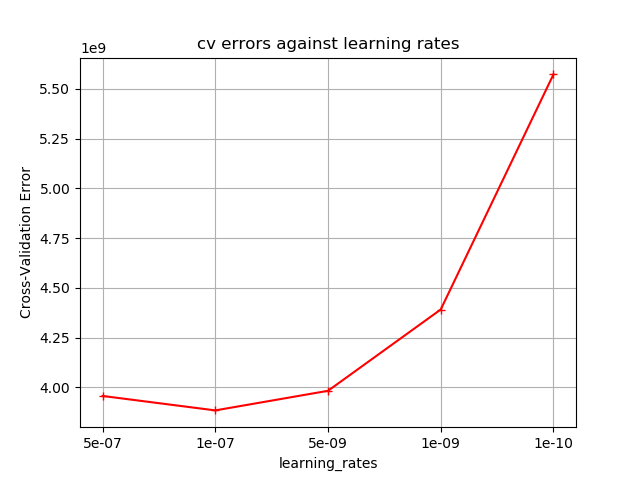


Fig 7.1

As shown in this figure, the best learning rate is 1e-07.

According to this optimal learning rate, we have plotted test errors against training epochs:

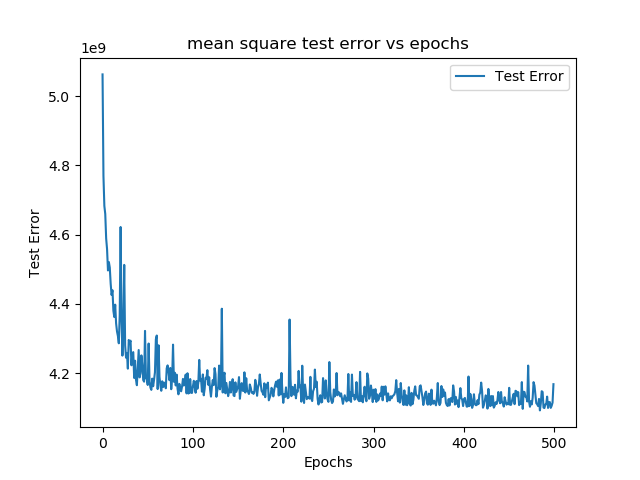


Fig 7.2

Task 3: Find the optimal number of hidden neurons

1. Find the optimal number of hidden neurons for the 3-layer network designed. Limit search space to: {20,40,60,80, 100}. Learning rate used: 1e-07.

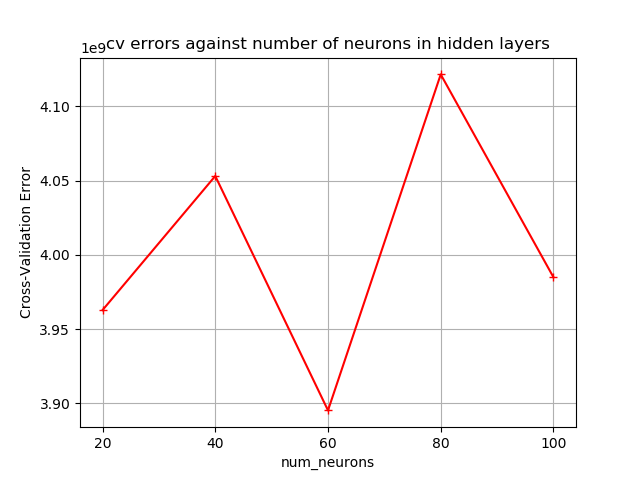


Fig 8.1

b) Plot the test errors against number of epochs for the network consisting of the optimal number of hidden neurons (60).

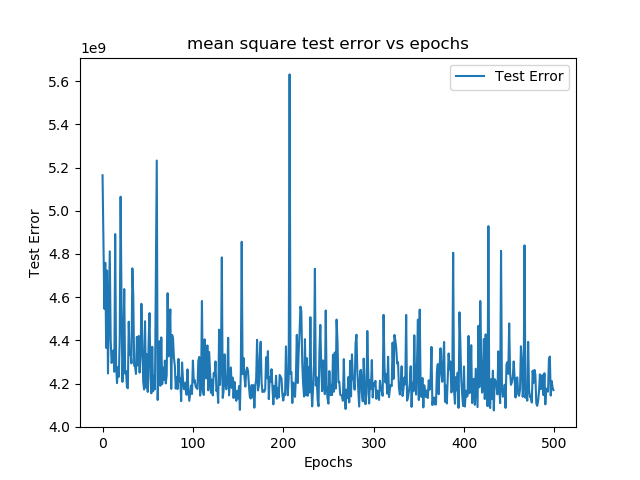


Fig 8.2

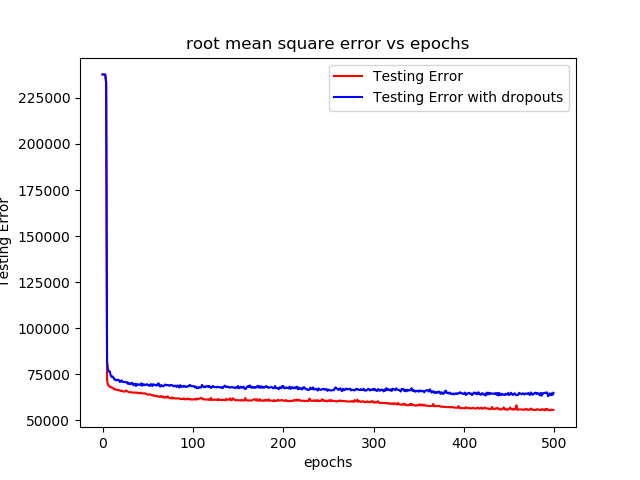
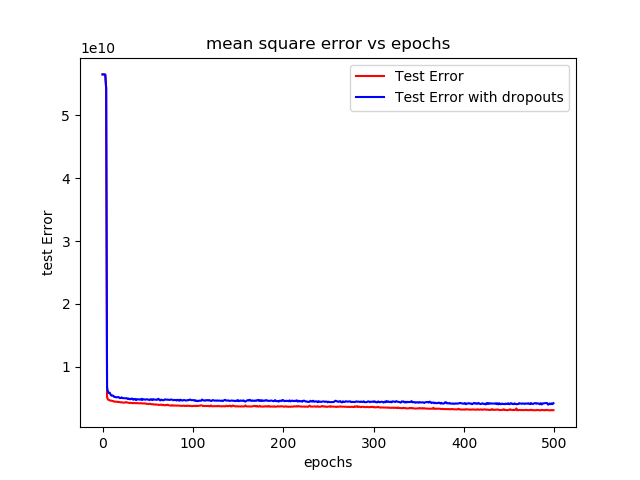
c) State the rationale behind selecting the optimal number of hidden neurons

In conclusion, we find that the most optimal number of neurons is 60. As we can see in figure 8.1, with 60 neurons used the cross validation error is the lowest compared to other number of neurons. As the number of neurons increases, the network tries to remember the training patterns to minimize the training error at the expense of its generalization ability of the unseen dat. Hence, we can surmise that as the number of neuron increases the reliability of the model decreases. Hence it can only be found by trial and error.

Task 4: Design a 4-layer and 5-layer neural network

Introduced dropouts of 0.9 probability.

For the 4-layer:

For the 5-layers: