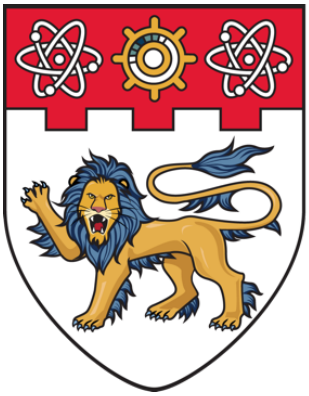


NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

CZ4042 NEURAL NETWORKS
PROJECT 1 REPORT

NGUYEN HUY ANH (U1420871B)
LEE VIC SON (U1423115H)

School of Computer Science and Engineering

Academic Year 2017/18, Semester 1

Part A: Classification

Introduction

The Landsat satellites are part of a continuous effort between NASA and the US Geological Survey to acquire land remote sensing data. The Multispectral Scanner (MSS) sensor aboard the satellites acquires imagery of the earth in four different spectral bands — two of these are in the visible region (corresponding approximately to green and red regions of the visible spectrum) and two are in the (near) infra-red.

The database consists of the multi-spectral values of pixels in 3×3 neighbourhoods in a satellite image, and the classification associated with the central pixel in each neighbourhood. The aim is to predict this classification, given the multi-spectral values.

The whole data set contains 6435 lines. In each line, the multi-spectral values of a neighborhood are specified: the four values for the top-left pixel are given first, followed by the top-middle pixel and then the top-right pixel, and so on with the pixels read out in sequence left-to-right and top-to-bottom. These multi-spectral values are integer-coded from 0 (black) to 255 (white). The class value of the centre pixel is coded as another integer:

Integer	Class
1	red soil
2	cotton crop
3	grey soil
4	damp grey soil
5	soil with vegetation stubble
6	mixture class (all types present)
7	very damp grey soil

No line of class 6 is present in the data set. The training set contains 4435 lines, while the testing set contains 2000 lines.

Multilayer feed-forward neural networks are suitable to solve classification problems like this, because it can approximate an arbitrary nonlinear function to a high degree of accuracy, given enough training data.

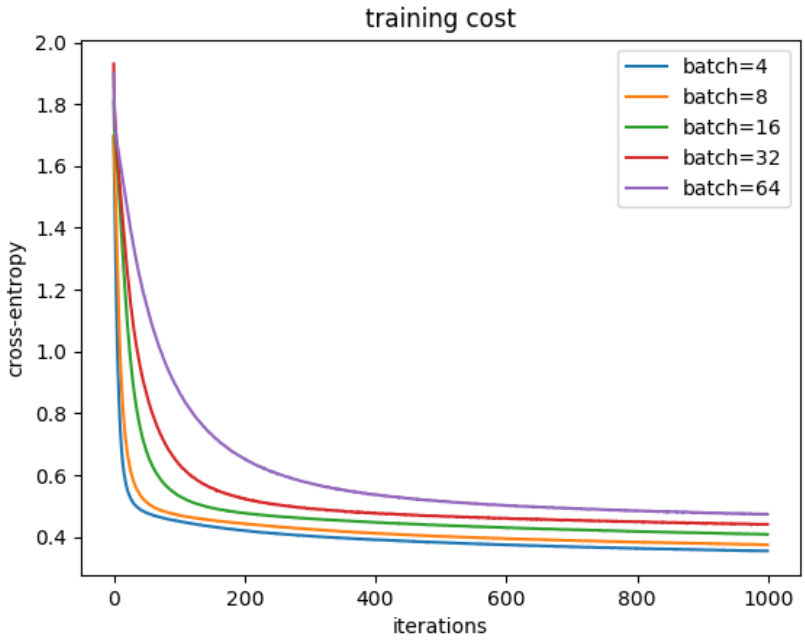
Experiments

We implemented the neural network as a function `train_test()` taking three optional parameters: `batch_size`, `hl_neuron` (the hidden layer neuron count) and `decay`. Each experiment is

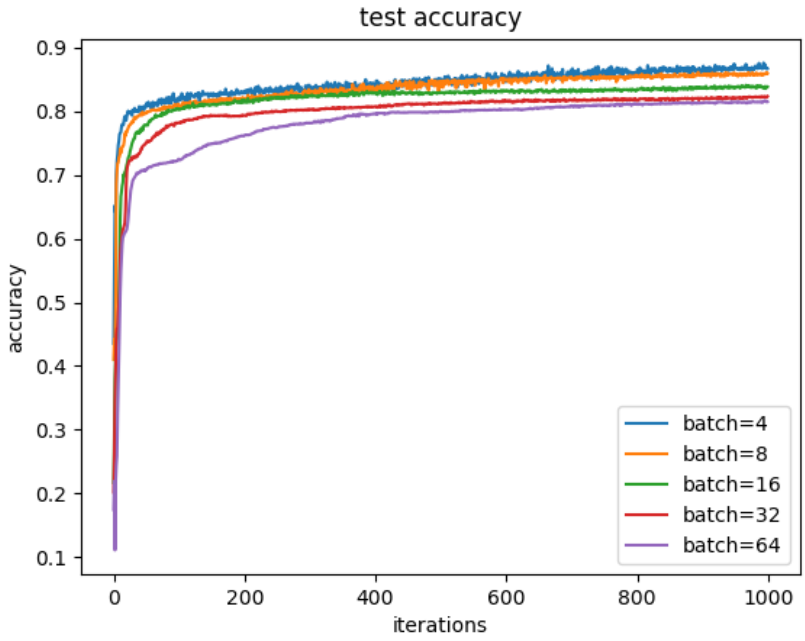
performed by utilizing the search function `search()`, which takes two additional inputs, the search parameter (either `batch_size`, `hl_neuron` or `decay`) and the search space associated with it. The function would loop through the possible values of the parameter, and train and test the neural network accordingly; graphs are stored and displayed after each experiment.

Experiment 1: Determining the optimal batch size (Question 2)

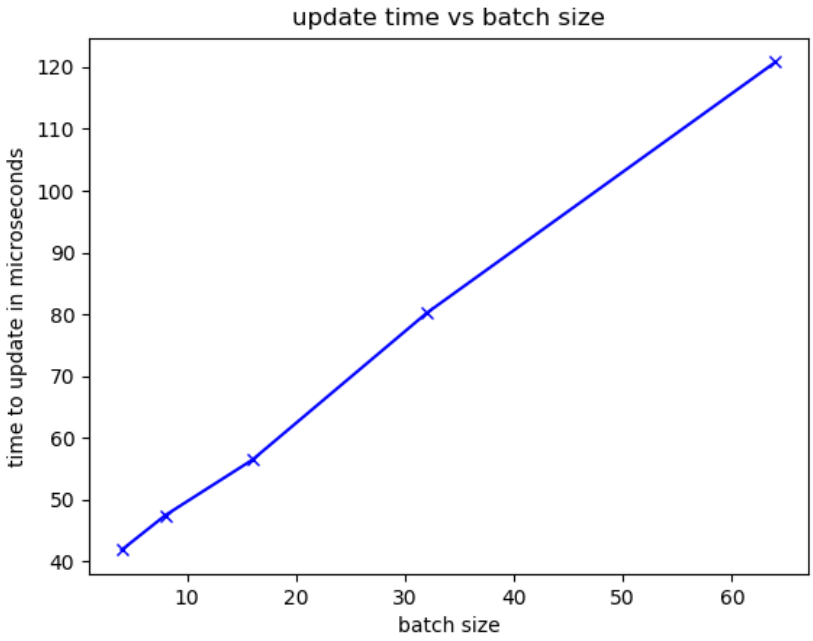
The parameter `batch_size` is varied in the search space $[4, 8, 16, 32, 64]$. Plotting the training errors against the number of epochs for different batch sizes:



Plotting the test accuracy against the number of epochs for different batch sizes:



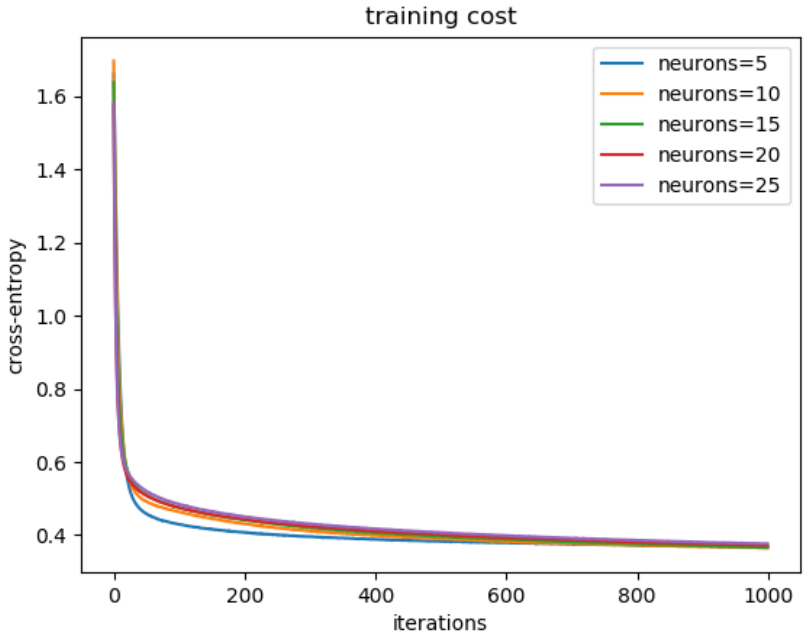
Plotting the time taken to update parameters for different batch sizes:



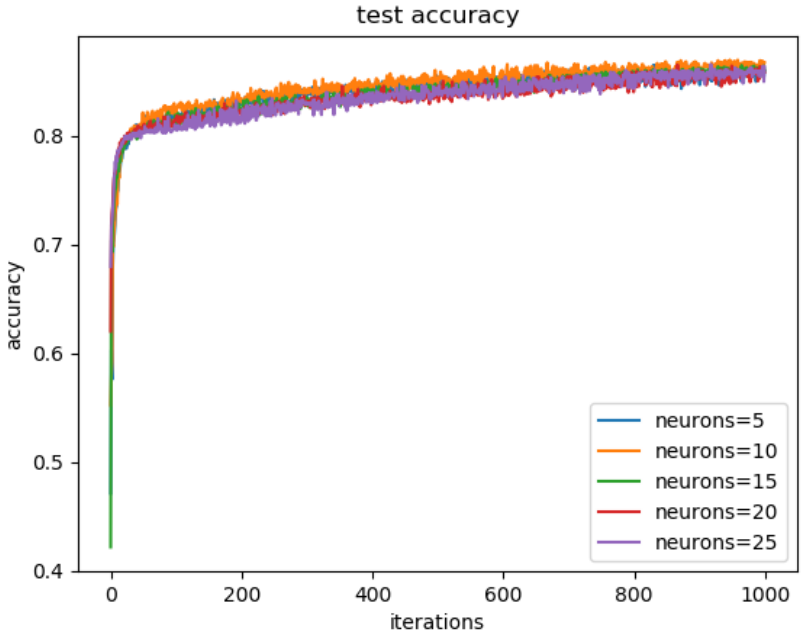
We decided to choose batch size of 4, for the highest test accuracy.

Experiment 2: Determining the optimal number of hidden neurons (Question 3)

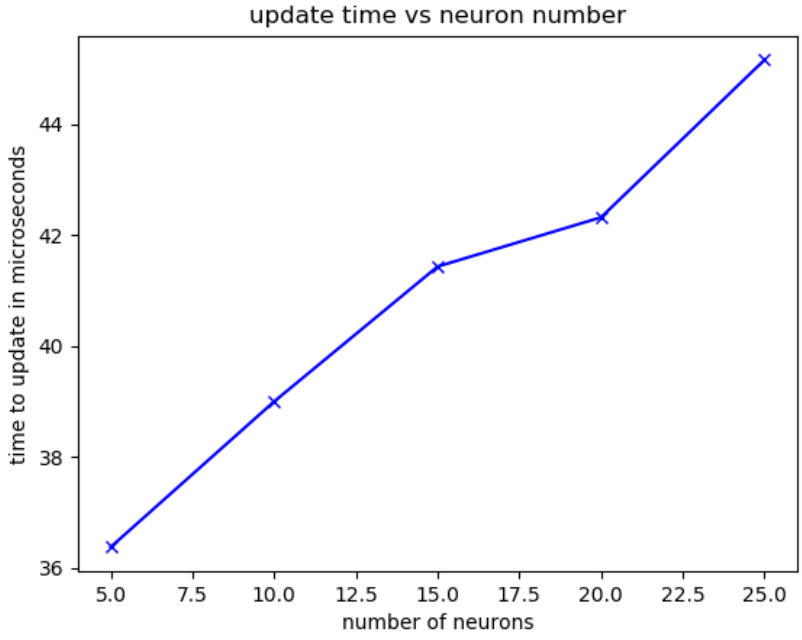
The parameter `hl_neuron` is varied in the search space $[5, 10, 15, 20, 25]$. Plotting the training errors against the number of epochs for different hidden neuron counts:



Plotting the test accuracy against the number of epochs for different hidden neuron counts:



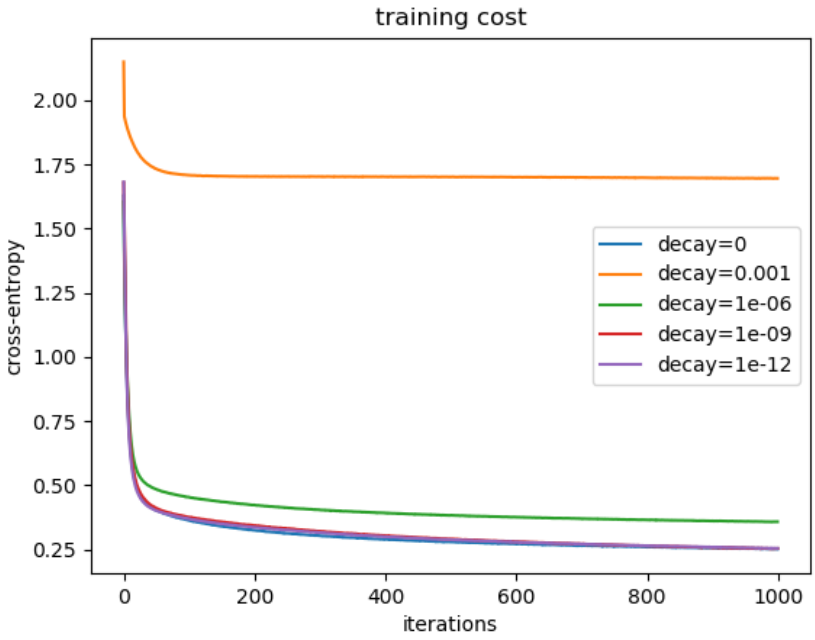
Plotting the time taken to update parameters for different hidden neuron counts:



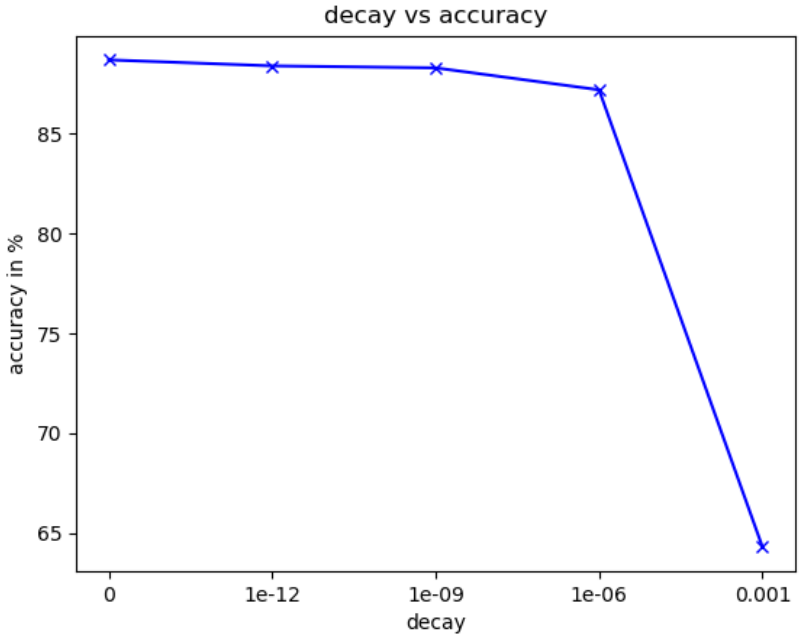
We decided to choose 10 hidden neurons, for the highest test accuracy.

Experiment 3: Determining the optimal decay parameter (Question 4)

The parameter `decay` is varied in the search space $[0, 10^{-12}, 10^{-9}, 10^{-6}, 10^{-3}]$. Plotting the training errors against the number of epochs for different decay parameters:



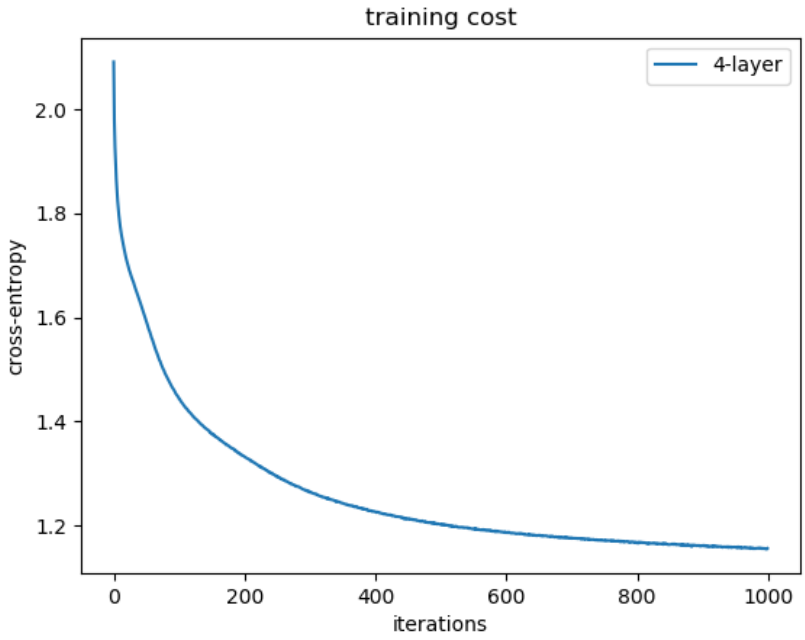
Plotting the test accuracy against the decay parameters:



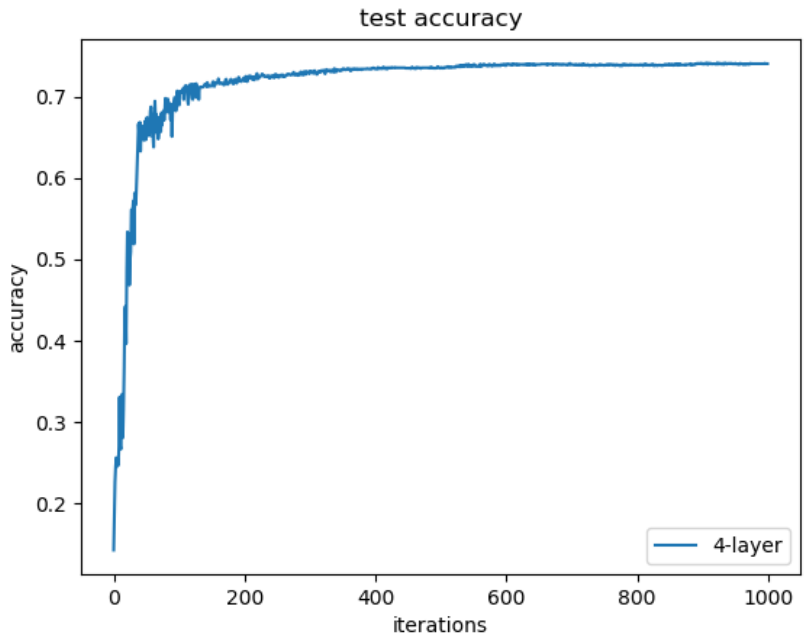
We decided to choose decay parameter 10^{-9}, for the highest accuracy.

Experiment 4: The 4-layer neural network (Question 4)

Plotting the training errors against the number of epochs for the 4-layer network:



Plotting the test accuracy against the number of epochs for the 4-layer network:



The 3-layer network achieves a higher accuracy compared to the 4-layer network. This shows that more layers do not neccesarily lead to higher performance in a feedforward neural network.

Part B: Approximation

Introduction

The California Housing data set consists of information collected from the 1990 Census conducted by the United States Census Bureau. More specifically, the data set focused on the median housing prices and 8 independent variables associated with the housing price for all the block groups in California.

A block group is the smallest geographical unit for which the United States Census Bureau publishes data. The geographical size of block groups vary inversely with their population density, and they usually have a population size of 600 to 3000 people.

The entire data set has 20640 lines of data, and the variables (listed in order) are:

- Longitude
- Latitude
- Housing Median Age
- Total Rooms
- Total Bedrooms
- Population
- Households
- Median Income
- Median House Value

The longitudes and latitudes were calculated from the centroids of each block group, and any block groups that did not have statistics for one or more variables were excluded from the final data set.

The aim of this part of the project is to predict the median housing price of a block group, given the other 8 variables as input — a regression problem. The data was split into train and test sets, with each having 14322 and 6138 lines respectively (7:3 ratio). The train set was further split into 5 parts to perform 5-fold cross-validation on the models tested, with the final accuracy of the selected model obtained from testing with the test set.

The use of multilayer feedforward networks to solve nonlinear regression problems like these has been well documented. The only difference between using these networks for classification

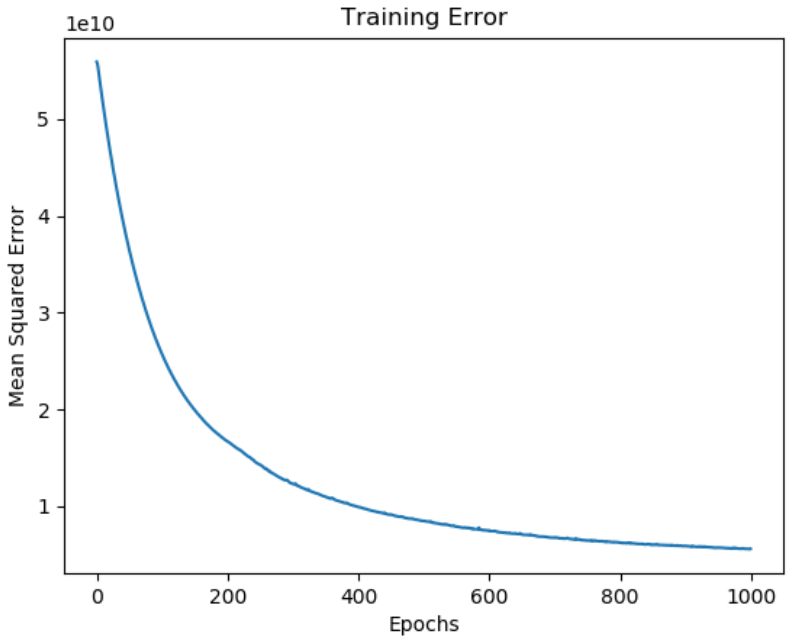
problems and regression problems is simply the activation function used in the output neuron (s). As such, neural networks perform similarly well on regression problems.

Experiments

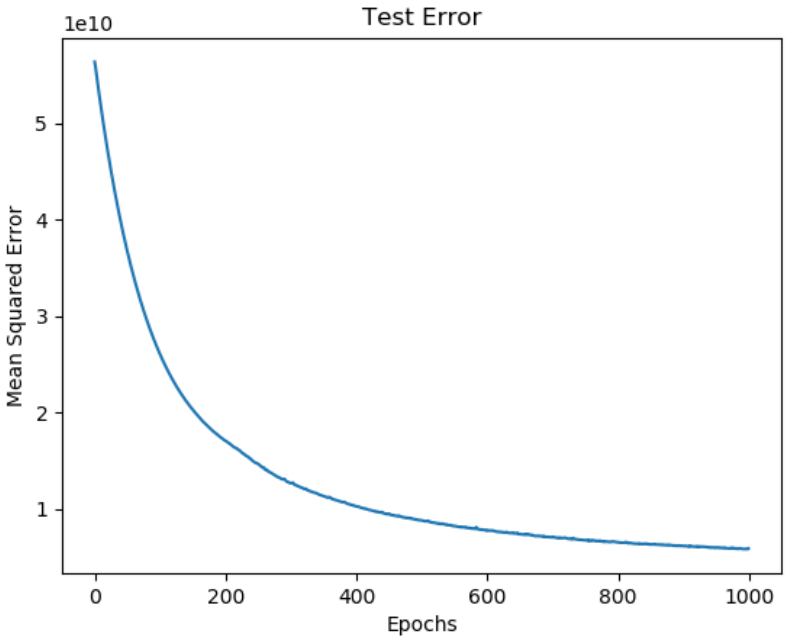
We implemented the neural network as two functions `nn_3_layer()` and `cross_validation()`, each taking two optional parameters: `learning_rate` and `no_hidden` (the hidden layer neuron count). Each experiment is performed by utilizing the search function `search()`, which takes two additional inputs, the search parameter (either `learning_rate` or `no_hidden`) and the search space associated with it. Depending on the flags set, the function would either loop through the possible values of the parameter and perform cross-validation on the training set, or train and test the neural network on the whole data set. Graphs are stored and displayed after each experiment.

Experiment 1: Building the reference 3-layer network (Question 1)

Plotting the training errors against the number of epochs:

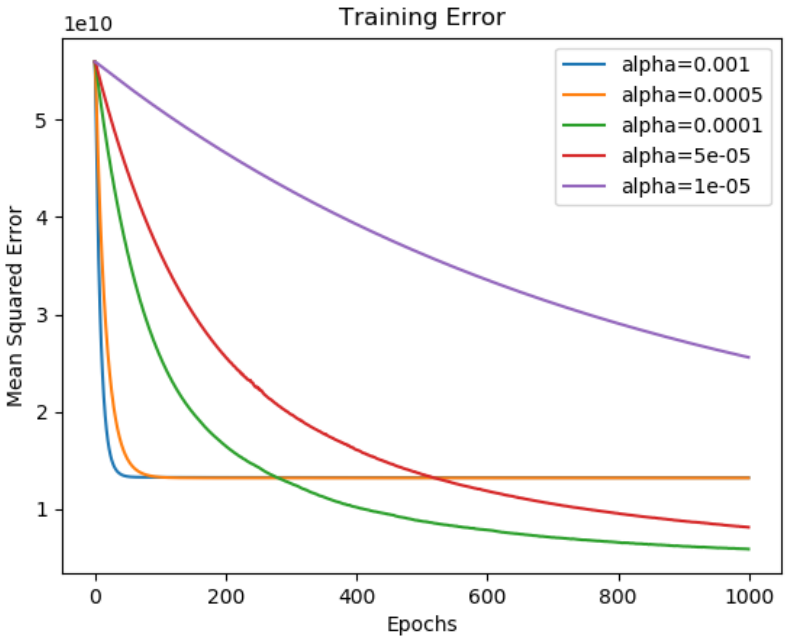


Plotting the test errors against the number of epochs:

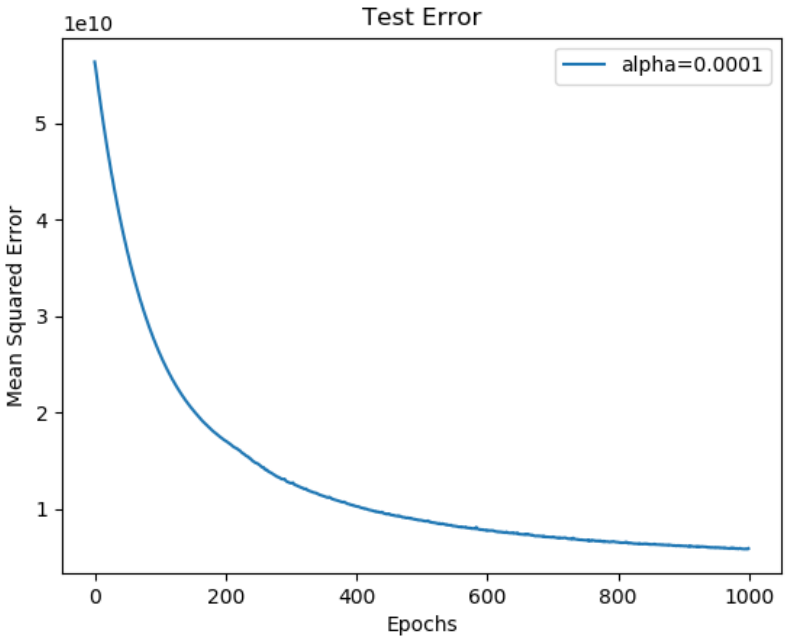


Experiment 2: Determining the optimal learning rate (Question 2)

The parameter `learning_rate` is varied in the search space $[10^{-5}, 0.5 \times 10^{-4}, 10^{-4}, 0.5 \times 10^{-3}, 10^{-3}]$. Plotting the training errors against the number of epochs for different learning rate:

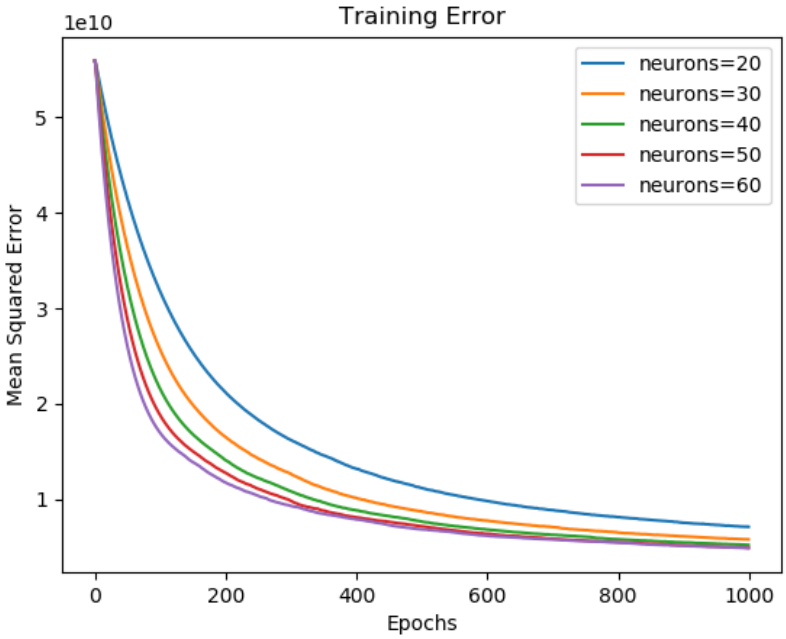


We decided to pick a learning rate of 10^{-4} for the best accuracy.
Plotting the test errors against the number of epochs for the optimal learning rate:



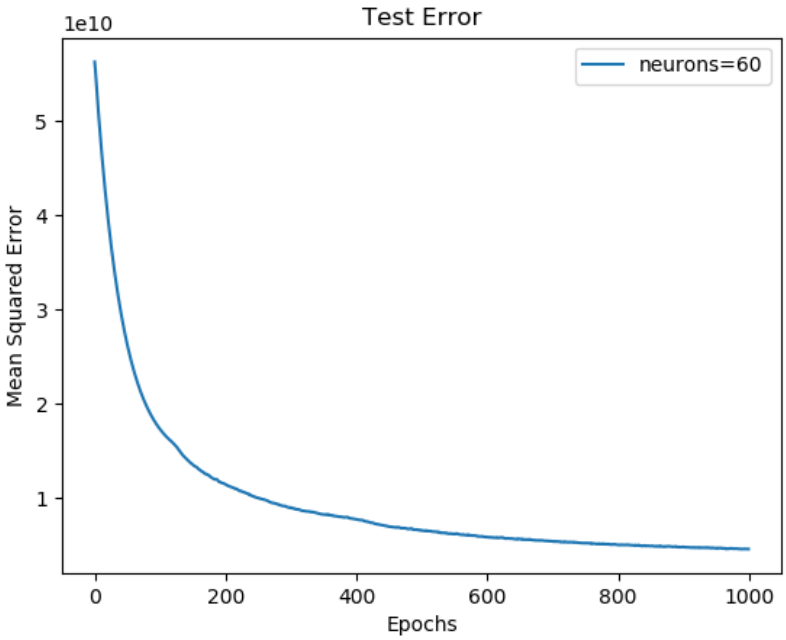
Experiment 3: Determining the optimal number of hidden neurons (Question 3)

The parameter `no_hidden` is varied in the search space $[20, 30, 40, 50, 60]$. Plotting the training errors against the number of epochs for different numbers of hidden neurons:



We decided to choose 60 hidden neurons for the best accuracy.

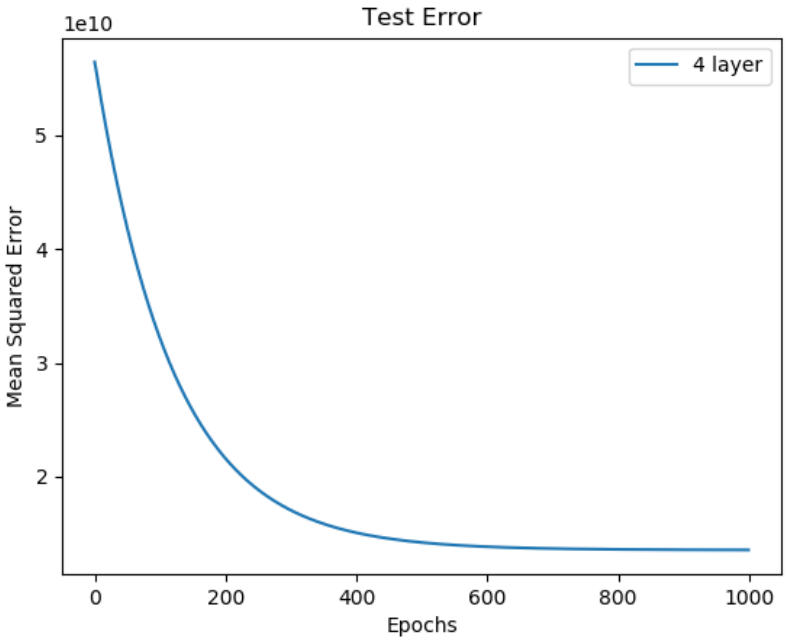
Plotting the test errors against the number of epochs for the optimal number of hidden neurons:



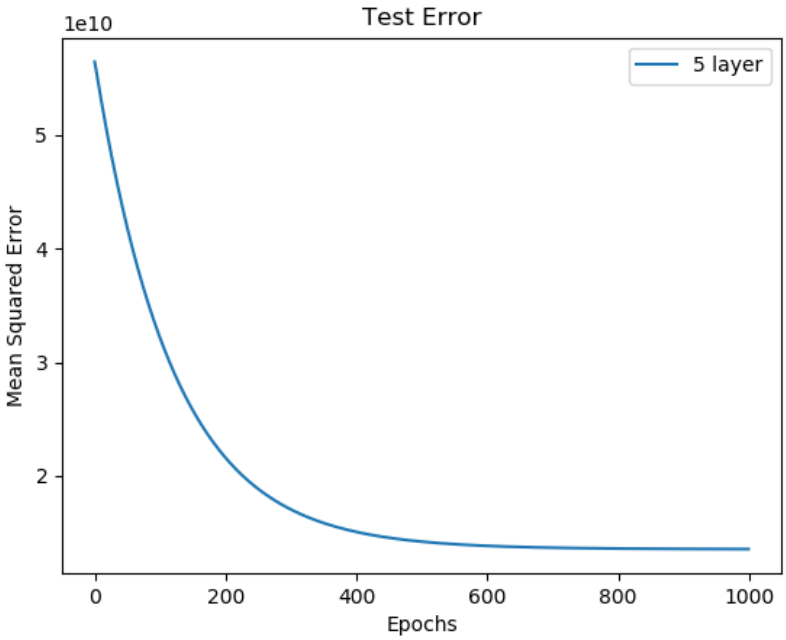
Experiment 4: The 4-layer and 5-layer neural network (Question 4)

The 4- and 5-layer networks are implemented in `nn_4_layer()` and `nn_5_layer()`, accordingly.

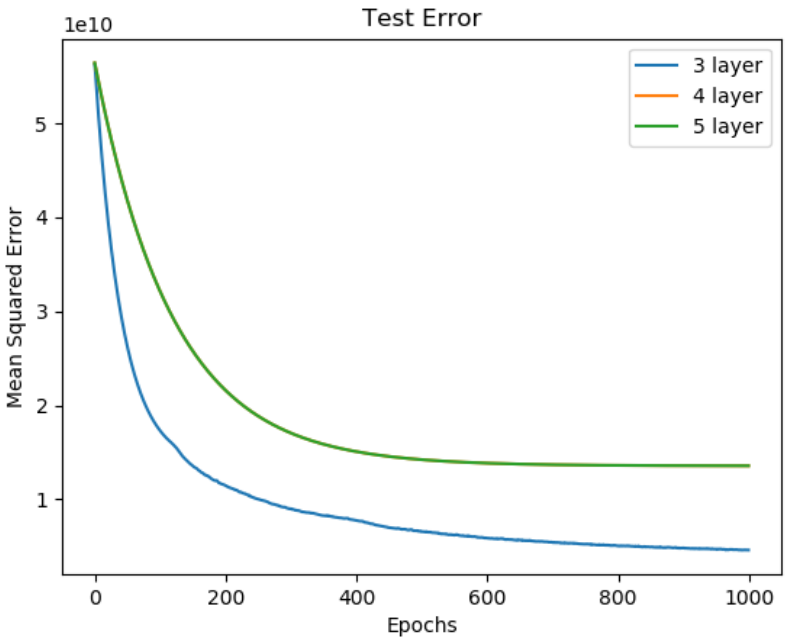
Plotting the test errors against the number of epochs for the 4-layer network:



Plotting the test errors against the number of epochs for the 5-layer network:



Comparing the performance of the 4 and 5-layer networks with the 3-layer network:



Note that the plots for the 4-layer and 5-layer networks overlaps on each other.

The 3-layer network achieves a higher accuracy compared to the 4-layer and 5-layer networks. This shows a similar result to the classification task, where more layers do not neccesarily lead to higher performance.