Part B- California Housing Dataset

# Introduction:

The California Housing Prices Dataset consists of 8 different attributes of housing complexes in California like location, number of bedrooms, number of bathrooms etc. The data set also contains prices of the houses. The objective of this project is to predict housing prices by training the neural networks with the given attributes. The following experiments give an insight into choosing parameters like the right learning rate, number of hidden neurons, number of hidden layers for obtaining the best model to predict the housing prices.

## Train data and Test data:

The California housing price dataset has been divided into subsets for training and testing in the ratio of 70:30 respectively

## Five-Fold Cross Validation:

The training data has been further divided into a training subset and the validation subset in the ratio of 4:1. This is called as the five-fold cross validation method where 1/5 of the data is reserved for validation. The 5-fold cross validation method will train the neural network over the whole training dataset and also avoid overfitting.

## Epochs:

A 1000 epochs are used for the experiments.

## Batch-Size:

Stochastic gradient descent was initially used but it took a lot of time to converge, so we used gradient descent algorithm for the training of the neural network

## Activation Function:

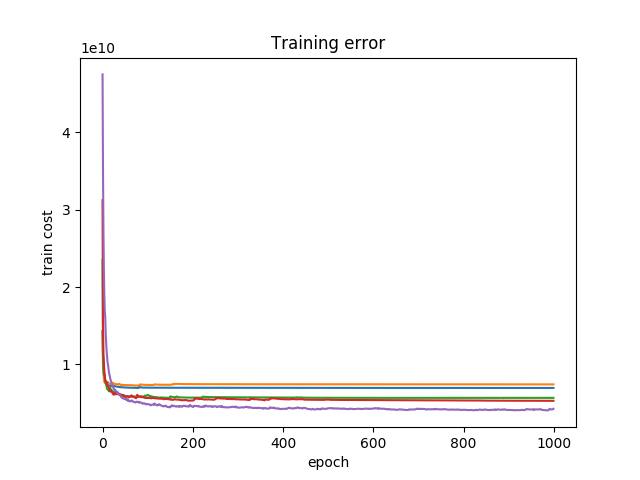
Sigmoidal activation function was used for the hidden layers ad a linear layer was used as the output layer

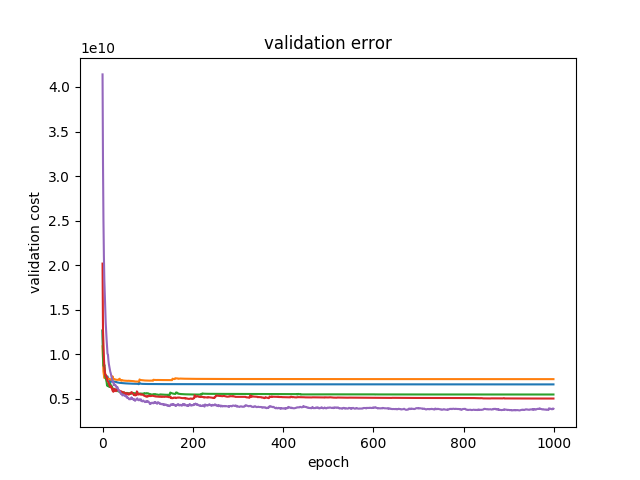
# 3- Layer

We build a basic 3-layer network and experimented with different learning rates and number of hidden neurons in the hidden layer. The result was as documented below

# Different Learning rates:

The 3-layer network designed in the stage above was trained with different learning rates using 5-fold cross validation in order to find the optimal learning rates. The different learning rates that were tried were [0.001,0.5\*0.001,0.0001,0.5\*0.0001,0.00001].





## Index:

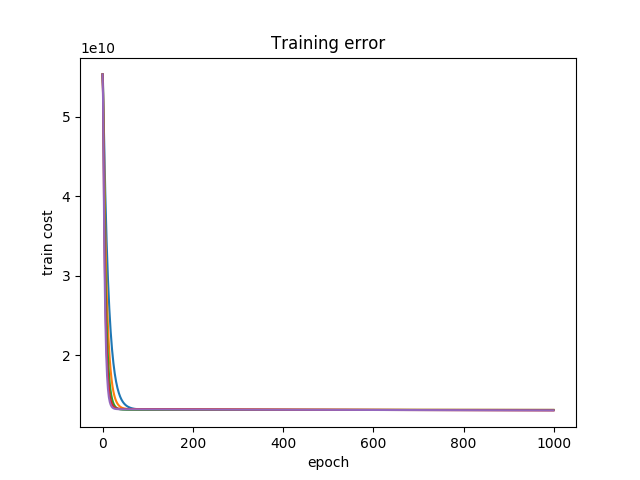
|  |  |
| --- | --- |
| 0.001 | Blue |
| 0.0005 | Orange |
| 0.0001 | Green |
| 0.00005 | Red |
| 0.00001 | Purple |

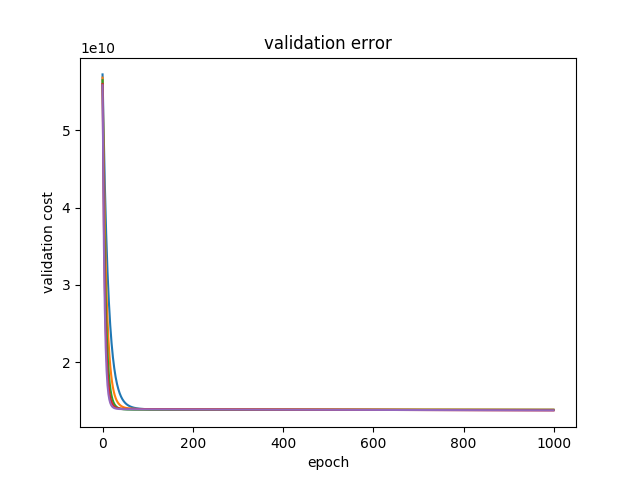
As we can see that alpha= 0.00001 converges faster than the other rates and hence that should be chosen as the optimal learning rate = 0.00001 (1e-5) or 10 raised to -5

We trained the network again using the optimal training rate and plotted the predictions against the outcomes.

# Different number of hidden neurons:

The 3-layer network designed in the stage above was trained with different neurons in the hidden layer using 5-fold cross validation in order to find the optimal hidden neurons. The different number of neurons that were tried were [20,30,40,50,60].





## Index:

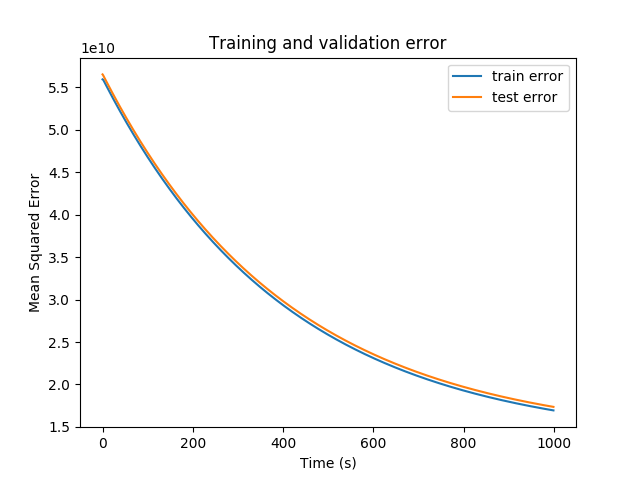
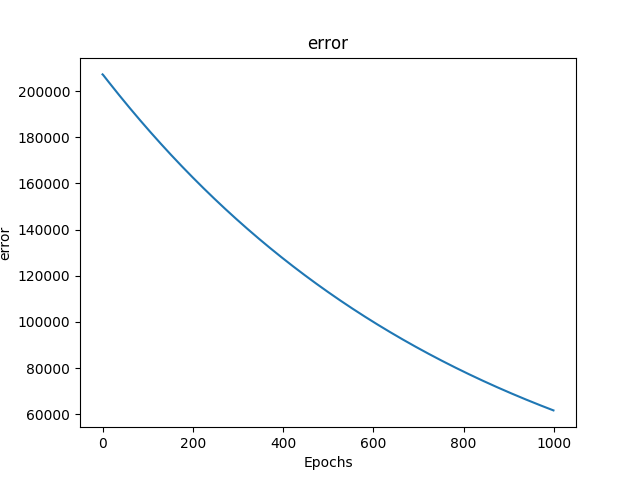
|  |  |
| --- | --- |
| 20 | Blue |
| 30 | Orange |
| 40 | Green |
| 50 | Red |
| 60 | Purple |

As we can see the training converges faster when there are more number of neurons in the hidden layer. Hence n=60 should be chosen as the optimum amount of neurons.

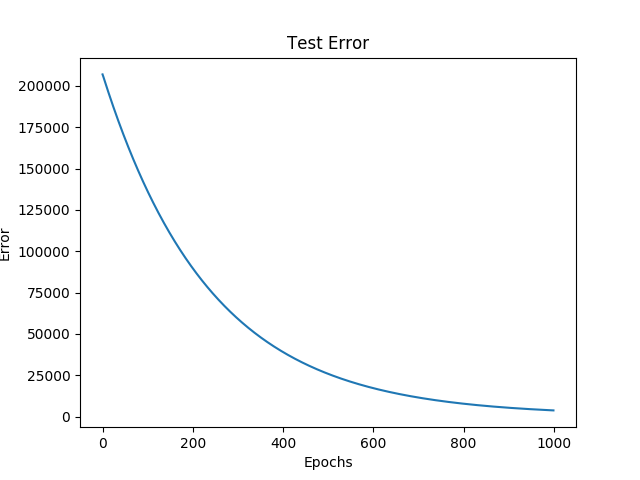
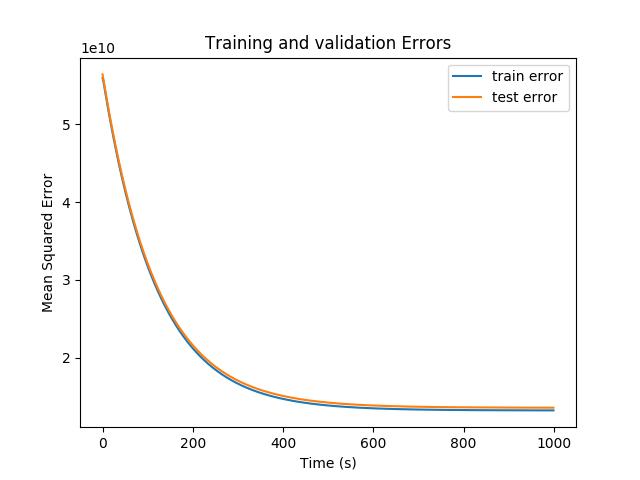
# Different number of layers:

We experimented with four and five layer neurons with optimal neurons found in the first layer i.e. 60 neurons and other layers having 20 neurons each. We used a learning rate of 10 raised to -4. We compared the three four and five layer networks to find the best model. We plotted the errors against the number of iterations

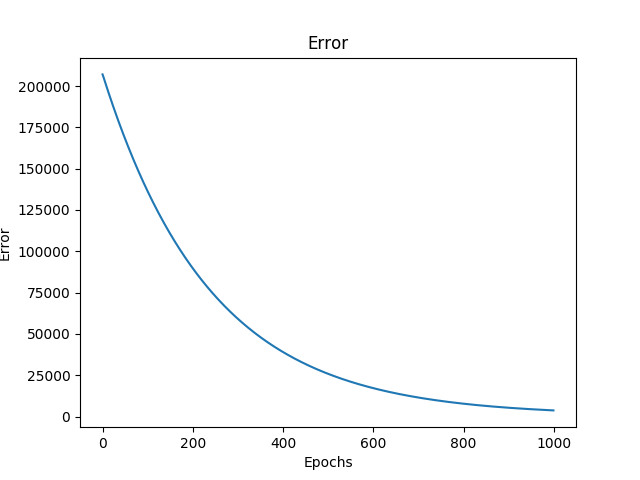
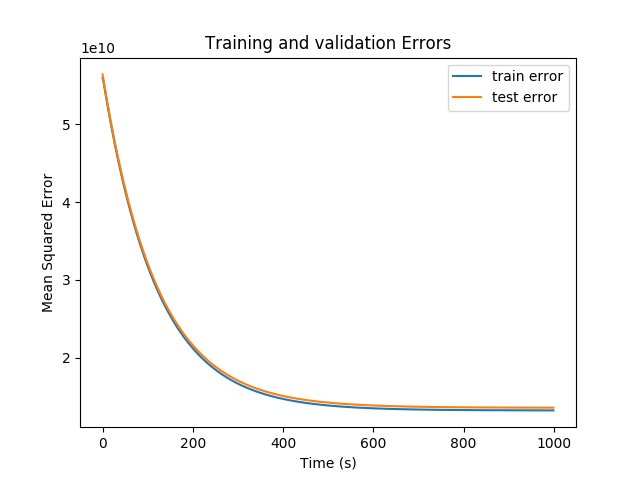
## 3-Layer: Best Model Chosen from previous experiments:



## 4-Layer: Training, validation and test errors versus epochs



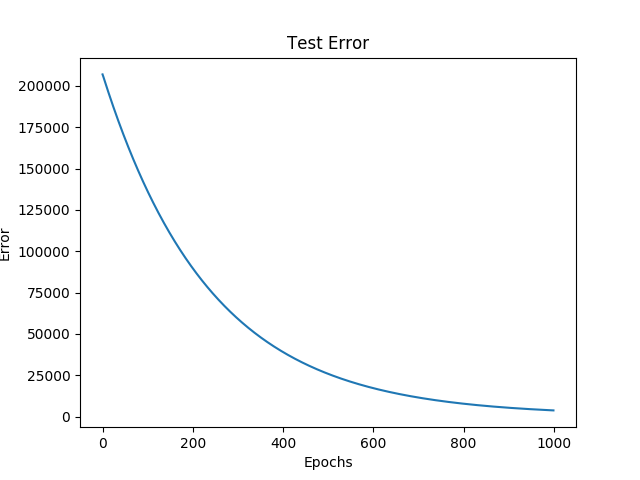
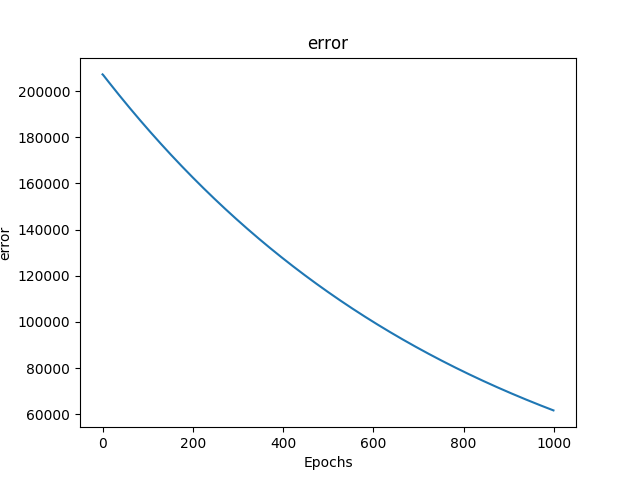
## 5-Layer: Training, validation and test errors versus epochs



## Comparison:

A large difference could be seen in the error function of a 3 layer and a 4-layer neural network but not between a 4 layer and a 5-layer neural network

So, according to the results, a 4 would be the optimum number of layer. Comparison between 3 layers and 4 layer network.



# Final model:

**Learning rate** = 1e-4

**Number of layers**= 4

**Number of neurons in the first hidden layer**= 60

**Number of neurons in the second hidden layer**=20

**Activation Function**: Sigmoidal activation function in the hidden layers and linear function in the output function.