CSL 603 - Machine Learning (Lab 3)

Multi-layer perceptrons and Self Driving Cars

Rishabh Singh - 2016CSB1054

October 30, 2018

Abstract

In this lab we implemented I implemented a basic neural network to predict the steering angle the road image for a self-driving car application that is inspired by Udacity's Behavior Cloning Project.

Introduction

The neural network contains 4 layers. The first layer consist of 1024 attributes, second layer consist of 512 attributes, third layer consist of 64 attributes and the last layer has only 1 output. Except the last layer all the layers has sigmoid activation. The input attributes were first divided into training and validation set of 80/20 ratio and then the training set was normalized. Using the mean and the standard deviation of training set, I normalised the validation set. After this preprocessing I applied the neural network on the input and the results obtained are as follows.

Experiment 1 :

A plot of sum of squares error on the training and validation set as a function of training iterations (for 5000 epochs) with a learning rate of 0.01. (no dropout, minibatch size of 64).

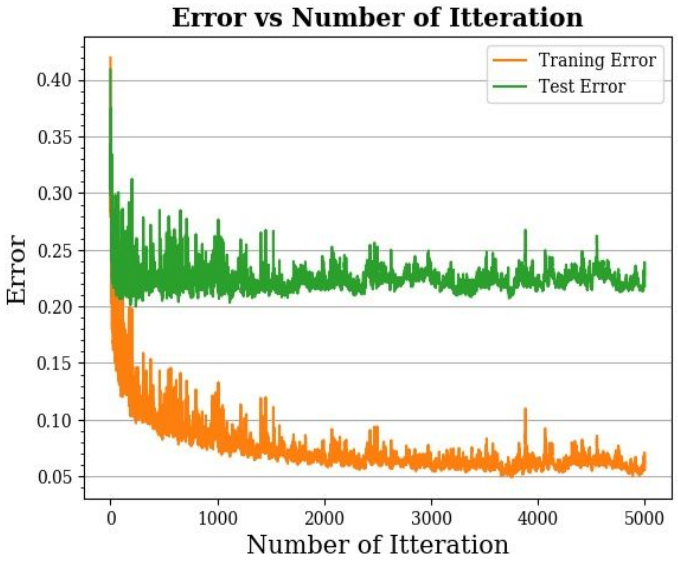

Fig1: LR = 0.01, batch_size = 64

From the graph it can be observed that initially as the number of iteration increases the validation and the training error decreases. After a point around 100 iteration the training error continues to decrease while the test error remains the same (increase a little bit). This observation clearly shows that the network tends to be overfit and hence decrease the performance. We can use dropout or other technique to improve performance.

Experiment 2 :

A plot of sum of squares error on the training and validation set as a function of training iterations (for 1000 epochs) with a fixed learning rate of 0.01 for three minibatch sizes – 32, 64, 128.

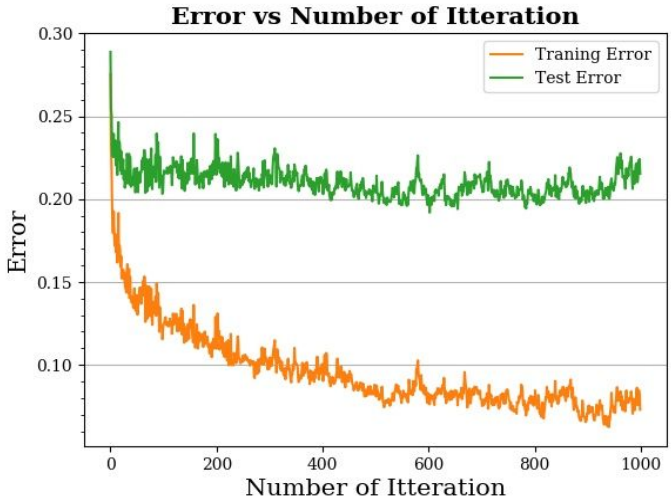

Fig 2: LR = 0.01, batch_size = 32

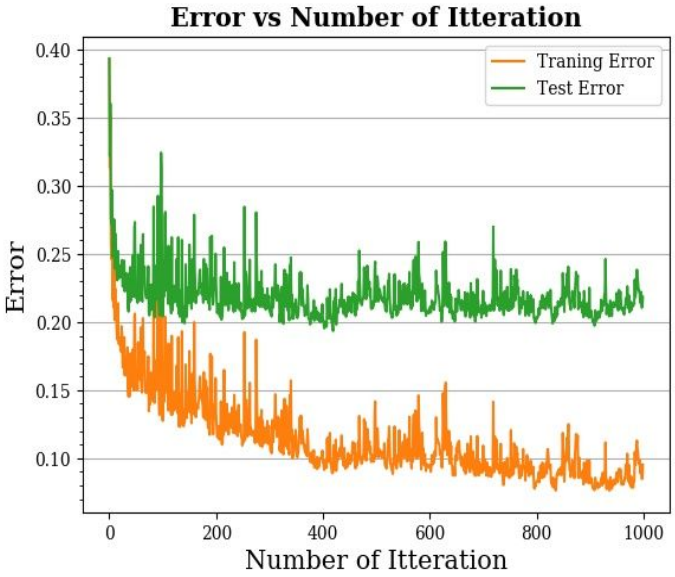

Fig 3: LR = 0.01, batch_size = 64

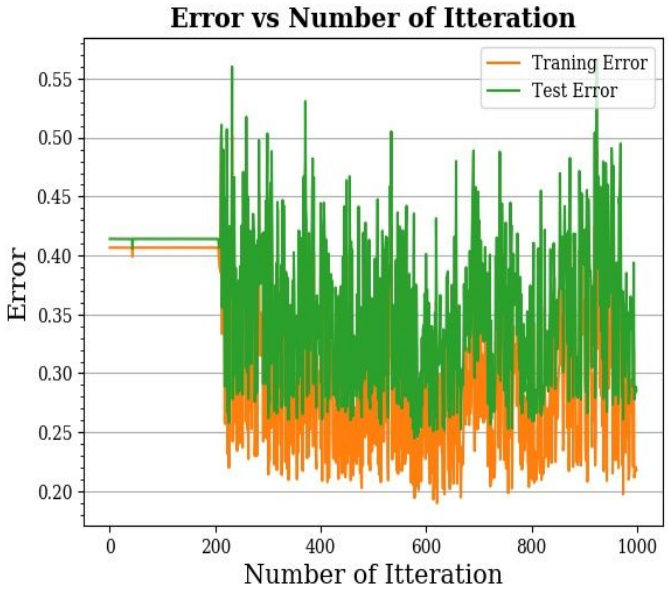

Fig 4: LR = 0.01, batch_size = 128

The above graphs is of the error vs the number of iteration. The Training error and the validation error are shown in the graph with varying batch size. From the graph we observe that training error keeps decreasing with the number of iterations. But this causes overfitting and hence test validation error starts increasing or moreover remains constant after some iteration. As we increase the batch size the number of addition for calculating gradient in each layer increases so the overall gradient increases which causes sharp changes in the error and thus shows a lot of fluctuation. We can also conclude that the 64 is the optimal size of the mini batch as it gives an acceptable training error and also less prone to overfitting than 32 mini batch size; It is also better than the 128 batch size as it gives a lot of fluctuation and convergence is really slow (After 1000 iteration error is 10 times than that of 64 batch size).

Experiment 3 :

A plot of sum of squares error on the training and validation set as a function of training iterations (for 1000 epochs) with a learning rate of 0.001, and dropout probability of 0.5 for the first, second and third layers.

Fig 5: LR = 0.001, batch_size = 64, dropout_frac = 0.5

From this experiment I observed that with the dropout of the random nodes, overfitting reduces and hence the training error decreases. Alternatively I can say that the difference between training error and the validation error is less than a model without dropout. This shows that the learning neural network model with dropout gives us a more generic model and is less prone to overfit.

Experiment 4 :

A plot of sum of squares error on the training and validation set as a function of training iterations (for 1000 epochs) with different learning rates – 0.05, 0.001, 0.005 (no drop out, minibatch size – 64).

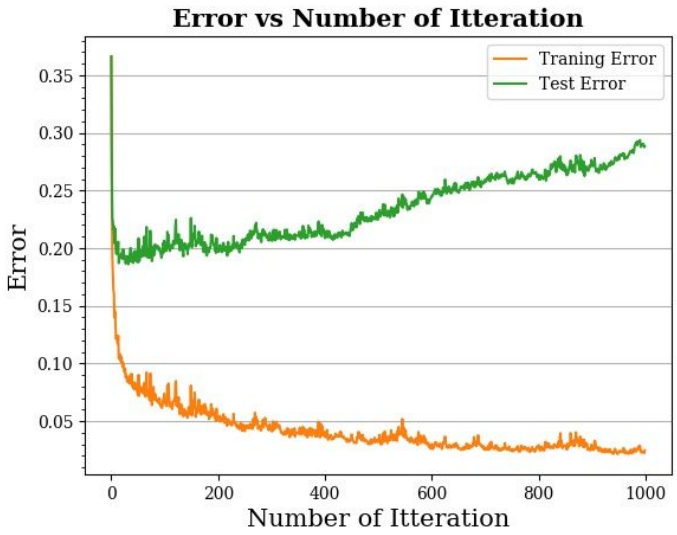

Fig 6: LR = 0.001, batch_size = 64

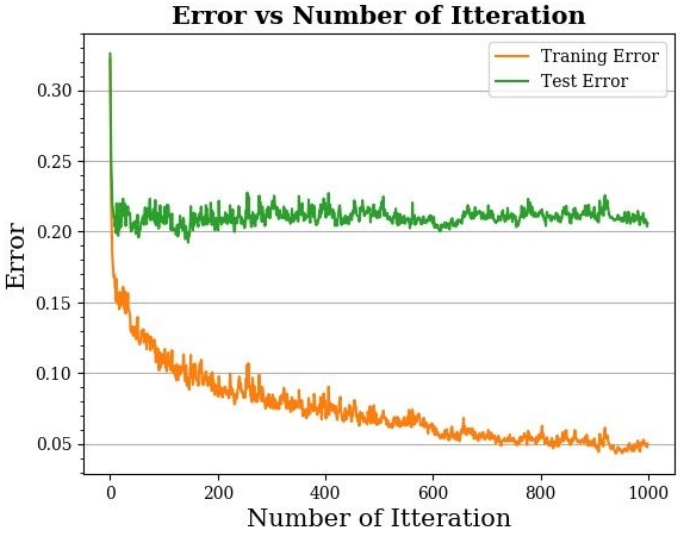

Fig 7: LR = 0.005, batch_size = 64

*Note ** With high learning rate i.e. for Lr = 0.05 my model was not converging so i was not able to draw the graph.*

From the above graphs I observed that with very low learning rate i.e. 0.005 the rate of descent is slow and the minimum error on the training set is observed after a lot of iteration. For error equals to 0.10 it takes around 175 iteration with LR = 0.005 while with LR = 0.001 it only take around 30 iteration. So, LR = 0.001 is good estimate of Learning rate for this model. A high value of Learning rate may diverge the error which is what happening with Lr = 0.05.