LAB – 3 REPORT

MULTI-LAYER PERCEPTRON

Introduction :

The aim of this experiment is to predict the correct steering angle of a vehicle by using multi-layer perceptron. The training set consists of 22000 gray scale images. Each image is of 32x32 dimension.

A **multilayer perceptron** (MLP) is a class of feedforward artificial neural network. MLP utilizes a supervised learning technique called **backpropagation** of training its multiple layers and non-linear activation distinguish MLP from a linear perceptron. It can distinguish data that is not linearly separable.

Perceptron Structure :

1. The network architecture is 1024-512-64-1.

2. All the hidden layers have employeed sigmoid activation function, while the output layer does not have any activation function.

3. Sum of squares error is used as loss function at the output layer.

4. The mini-batches shuffle sequentially through data. The data is shuffled initially before the training.

5. Dropout is implemented for all layers. The implementation has ability to tune dropout percentage.

6. The weights of each layer are initialized by uniformly spacing over the range [-0.01, 0.01]. The bias term is initialized to 0.

7. The dataset provided is splitted into training/validation according to 80:20 ratio.

Involved Mathematics :

- Intermediate hidden layers :

Z1 = sigmoid (X * transpose(W1))

Z2 = sigmoid (Z1 * transpose(W2))

O = (Z2 * V)

- The weight update equations are as follows :

d = O – Y

V = V – (n * transpose(Z2) * d))/ batch size

b = d * transpose(V) * Z2 * (1-Z2)

W2 = W2 – (n * transpose(Z1) * b))/ batch size

c = b * transpose(W2) * Z1 * (1-Z1)

W1 = W1 - (n * transpose(X) * c))/ batch size

- Activation functions used :

1. sigmoid function

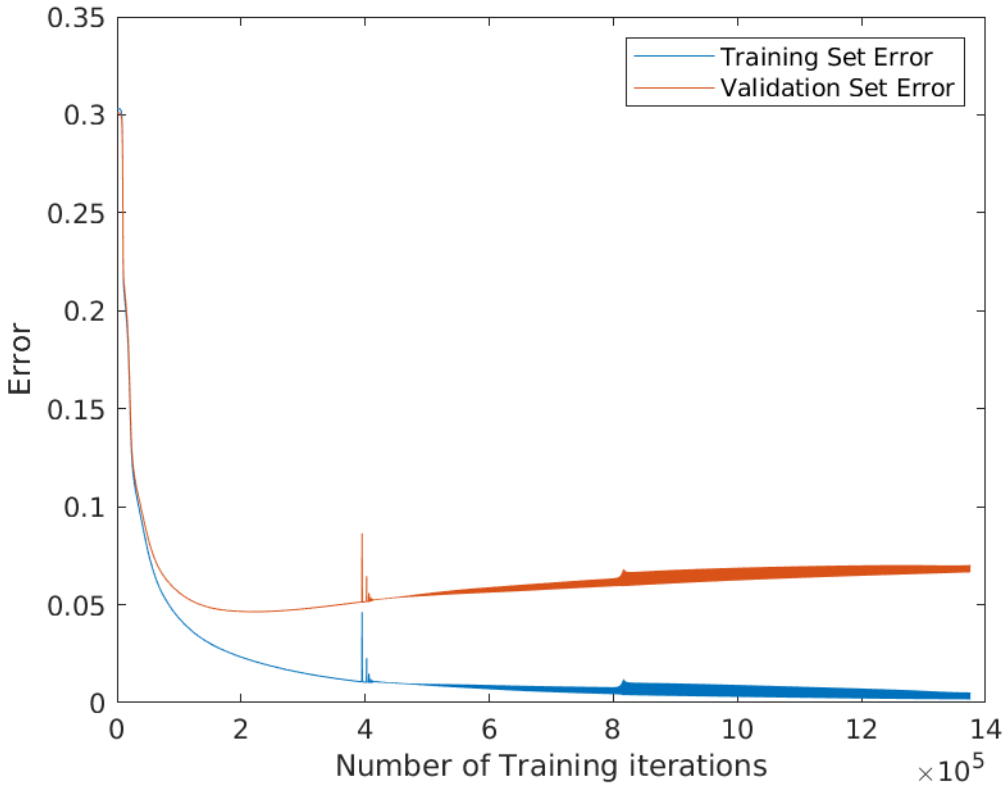
2. Tanh function

Observations:

We have studied different learning rates , mini batch size, plots of training error and validation error for these varying hyperparameters.

Plot with learning rate 0.01 & batch size = 64 :

(5000 epochs)



We observe from plot that, training error always decreases as a function of training iterations (epochs). This happens due to more training will always reduce training error, by updating the weights where training error is minimum.

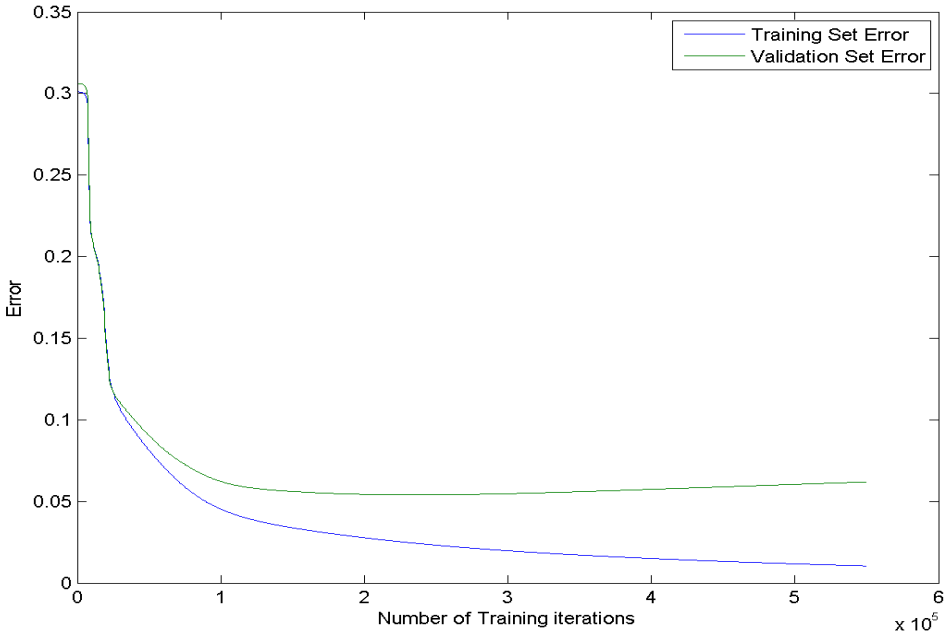
But, validation error first decreases and then increases. That is, it overfits the data. This is because model has high training accuracy but not so much testing accuracy. This model is poor in generalized performance. Generalization means how well are concepts learned by model on unseen data.

Thus, after around 10000 iterations == (800 epochs) , validation error starts increasing and from that it **starts overfitting**.

2. Plots with varying mini batch size :

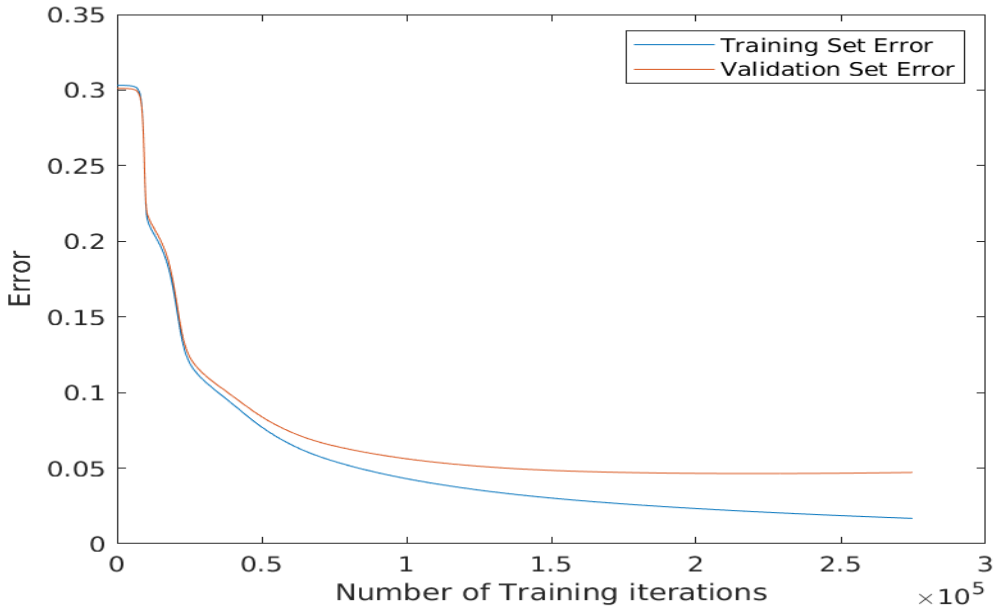
Plot with learning rate 0.01 & batch size = 32 :

(1000 epochs)



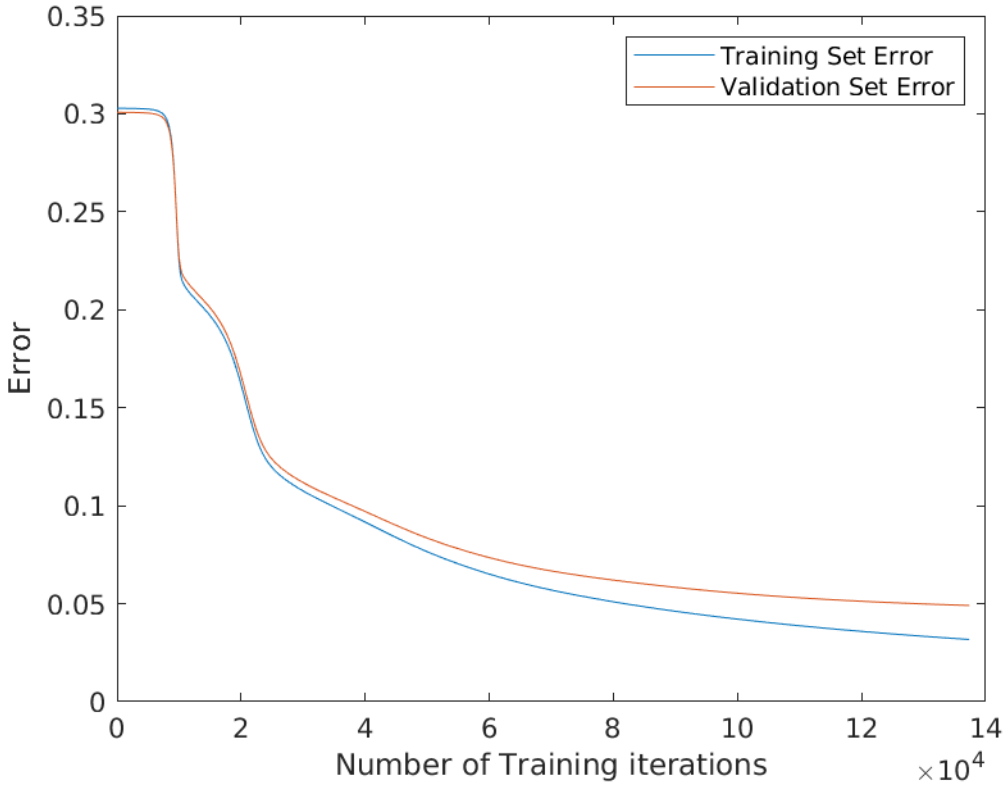
Plot with learning rate 0.01 & batch size = 64 :

(1000 epochs)



Plot with learning rate 0.01 & batch size = 128 :

(1000 epochs)



We can observe from these plots, as batch size is increasing more epochs are needed to reach same level of error. Batch size mainly improves the convergence over stochastic gradient descent. **one update** with a big minibatch is "better" (in terms of accuracy) than **one update** with a small minibatch.

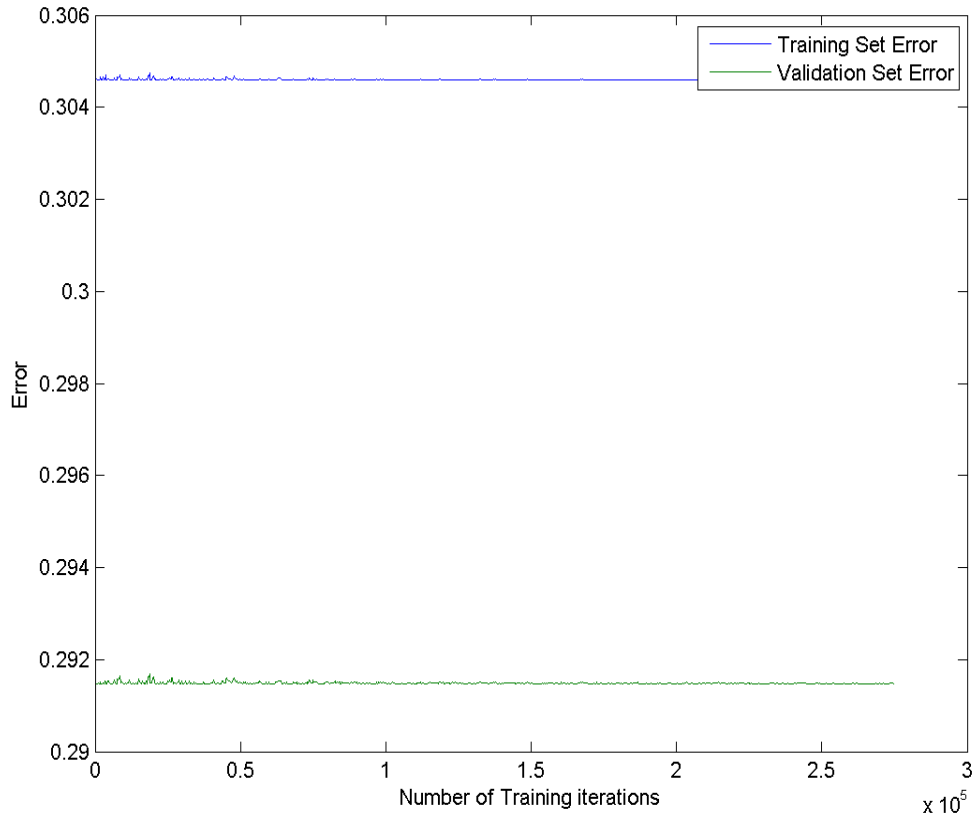
But, here we are comparing with same learning rate, hence large batch size requires more epochs to reach the same error as it needs to visit more examples. This is because there are less updates per epoch. This is evident from these plots.

In reality, batch size only improves training time and not much effect test performance.

3. Plot with dropout probability :

Plot with learning rate 0.001 & dropout = 0.5 :

(1000 epochs)



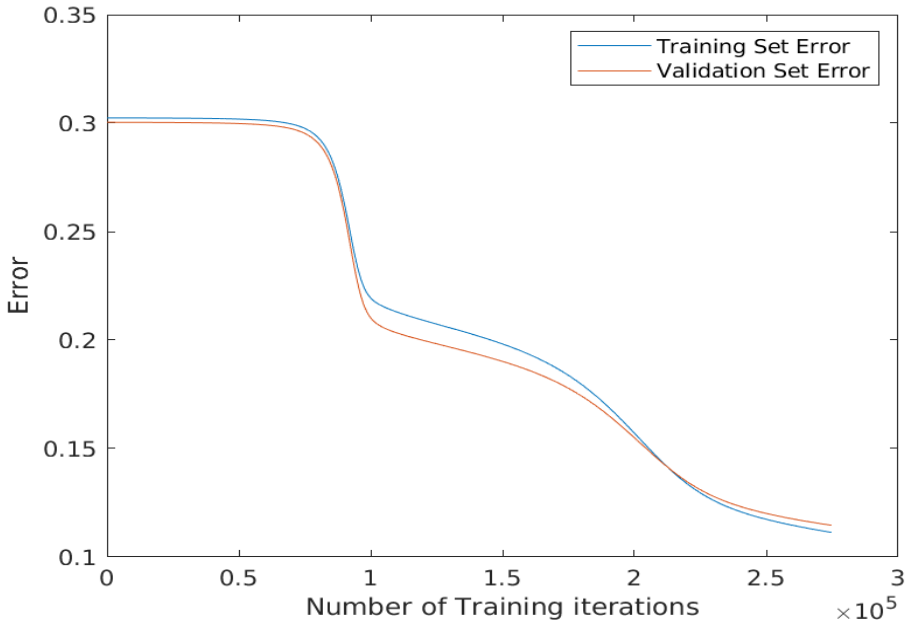
Dropout is a regularization technique for reducing overfitting in neural networks by preventing complex co-adaptations on training data. It is a very efficient way of performing model averaging with neural networks. The term "dropout" refers to dropping out units (both hidden and visible) in a neural network.

But, if we increase the dropout probability to very high , it causes the network to learn less & slowly, but causing extremely slow overfitting. We can see that error is not decreasing , this is due to large dropout probability but validation error is not increasing as the number of epochs increase. Also difference between validation error and training error is very less. Thus, no overfitting.

4. Plots with varying learning rates :

Plot with learning rate 0.001 & Batch size = 64 :

(1000 epochs)



Plot with learning rate 0.005 & Batch size = 64 :

(1000 epochs)