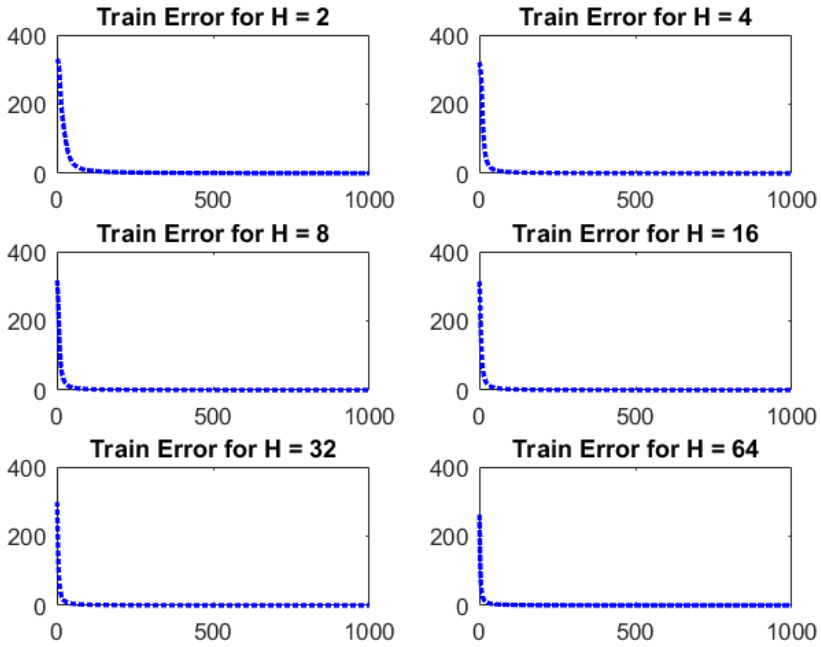
Lab 3: Machine Learning

Naman Goyal (2015CSB1021)

Question 1) 2-dimensional 3 class classification problem

Part 1: Training error against number of epochs for the different choices of hidden layer units

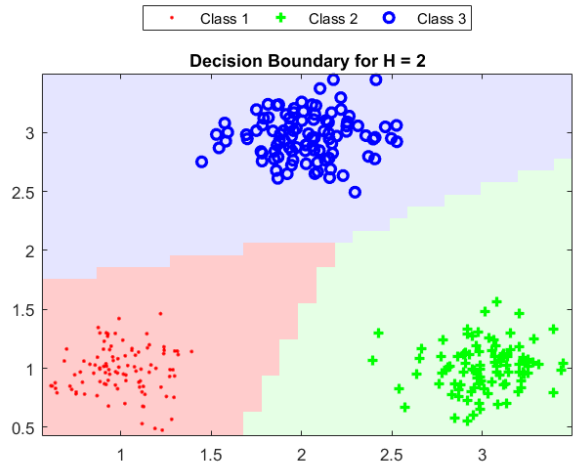


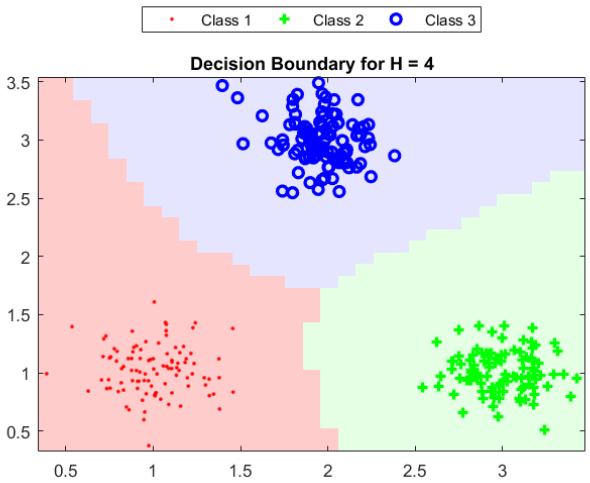
Observations

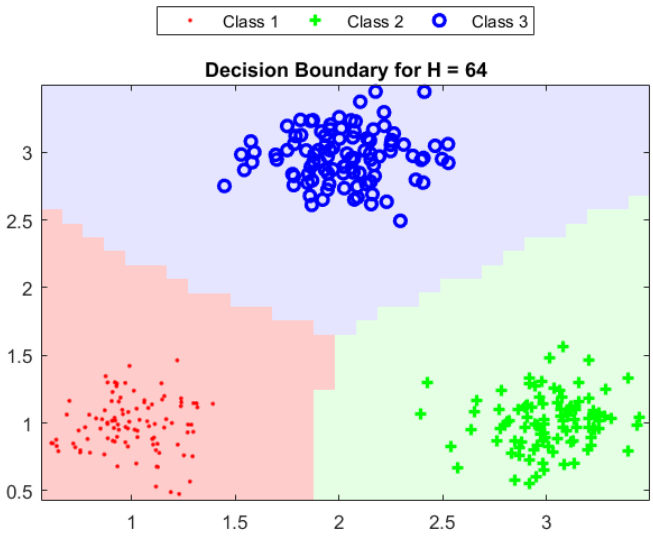
- As the hidden layer units increase the convergence is much faster i.e. training error decreases rapidly as the hidden layer units are increased.

  Explanation – Because of higher number of neurons; different neurons get activated at different points and error is propagated accordingly; hence training error reduces rapidly as Hidden units are increased.

Part 2: Decision boundaries for three choices of hidden layer units - 2, 4 and 64







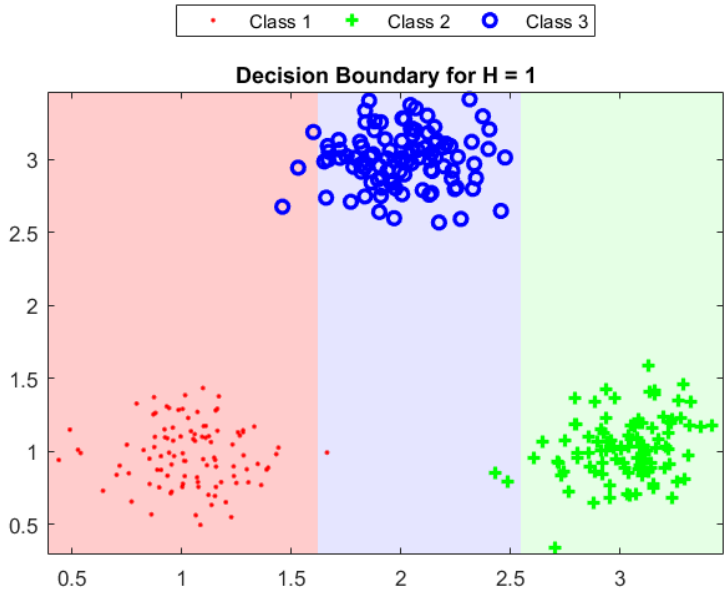
H stands for number of hidden layer units

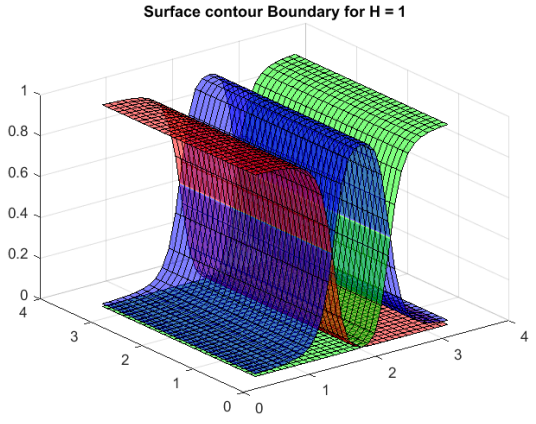
Observations

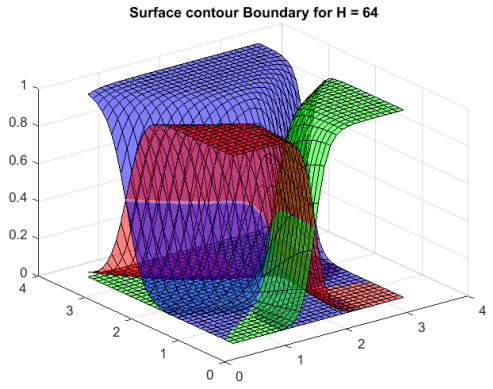
- As the hidden layer units increase the boundary is able to classify the instances more effectively – points distributed uniformly near decision boundary.
- Decision boundary is less skewed as hidden layers are increased.

Explanation – Higher number hidden units are able to learn a more complex decision surface. Since it is a classification task a more complex decision surface.

It is more evident from fact that with only one hidden unit; it still able to classify; as only cluster centers can be nearly classified with one dimension only; but training error is high as examples are misclassified.







Question 2) To predict the steering angle the road image for a self-driving car application

Part 1: Using given architecture and model

Points to note –

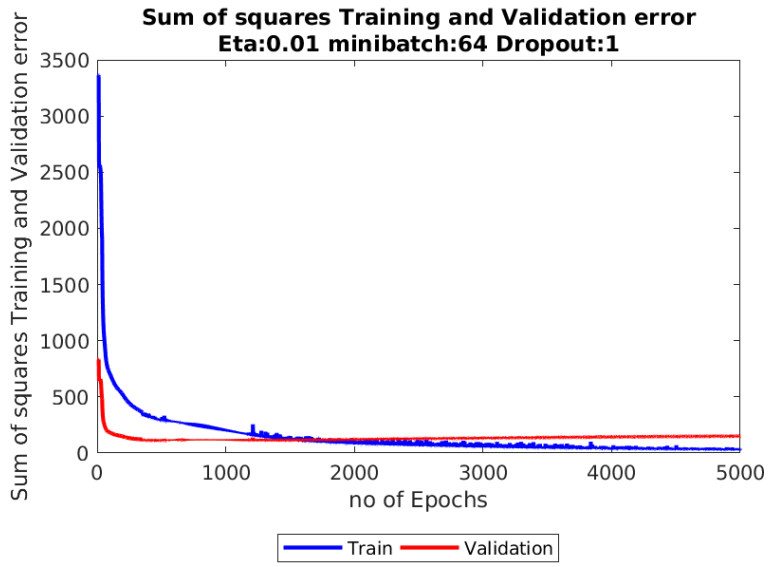
- Error for the neural network used is sum of squares error; hence during weight update the learning rate was NOT divided by batch size.

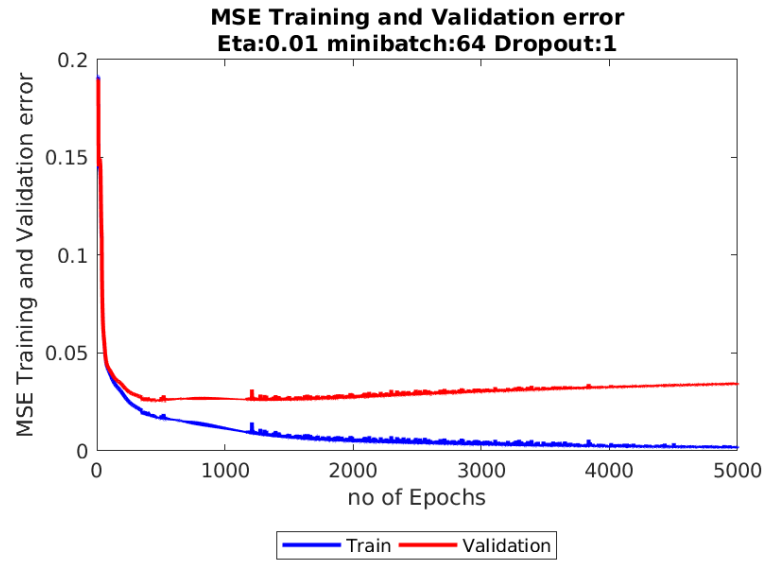
- MSE is mean squared error.

- Both sum of square error and followed by mean square error are plotted for easier understanding but all comparisons are on MSE since size of training and validation set is disproportionate.

- Dropout 1 implies all nodes were kept i.e. it the probability of keeping the nodes.

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





Observations

- The training error and validation error decrease rapidly during first few epochs and then convergence is slow.

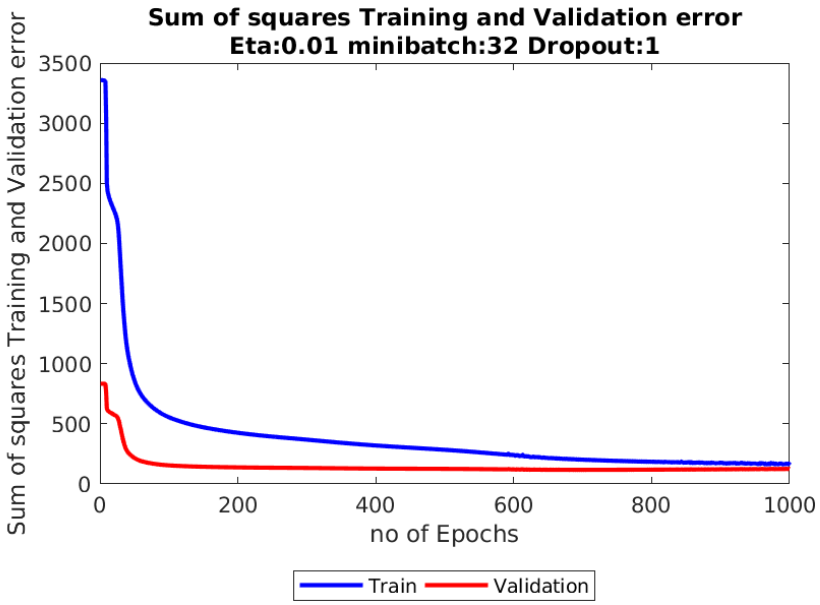
  Explanation: Initially the model error is high as weights were randomly initialized. Hence gradient was higher; the weights updates were larger and error on both training and validation decreases rapidly. But as models is trained; since **learning rate is fixed since the magnitude of gradient decreases over the time; hence smaller weight update.**

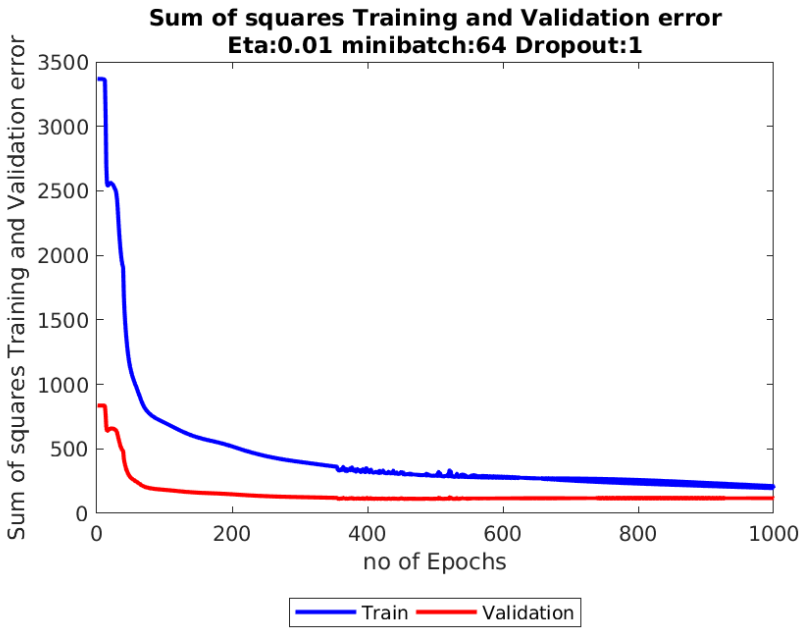
- Initially the both training and validation error decrease; but after nearly 1000 epochs only the training error decreases and validation error increase.

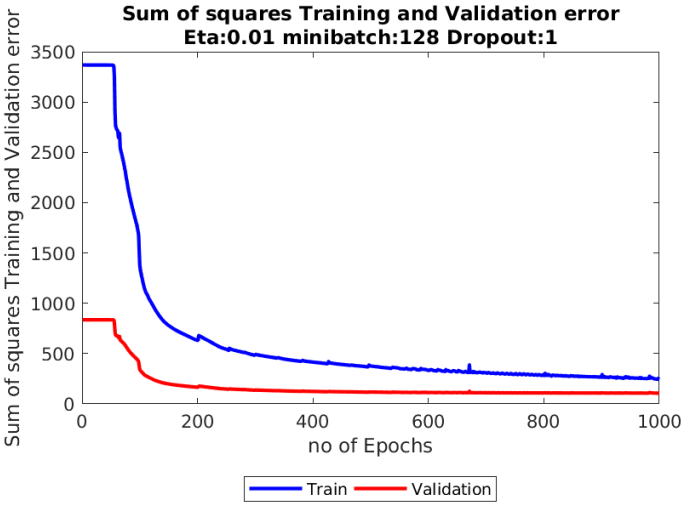
  Explanation: Initially the model learns generalized unbiased estimate, but as epochs increase there is clearly **overfitting** on the training data; hence the validation error is increase.

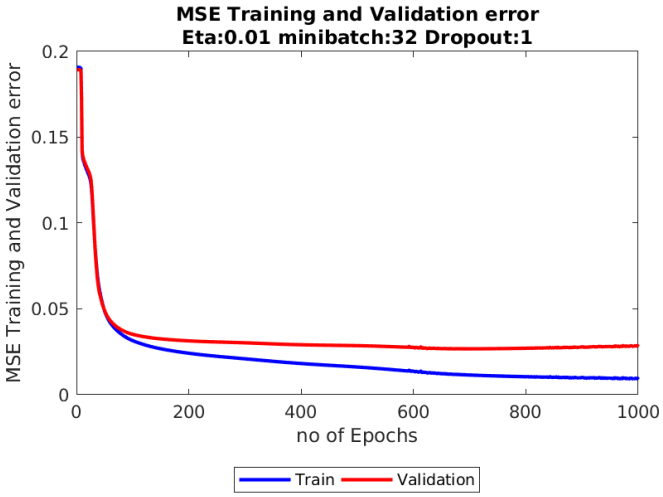
  Hence training decreases to nearly 0 while validation error (MSE) is nearly 0.05 (higher).

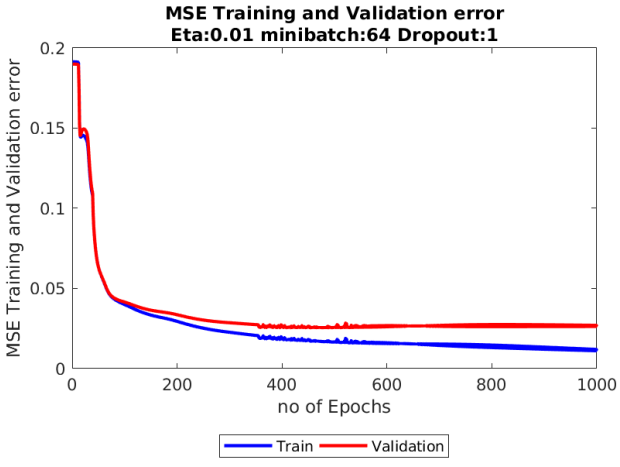
Plot ii. A plot of sum of squares error on the training and validation set as a function of training iterations (for 1000 epochs) with a fixed learning rate of 0.01 for three minibatch sizes – 32, 64, 128

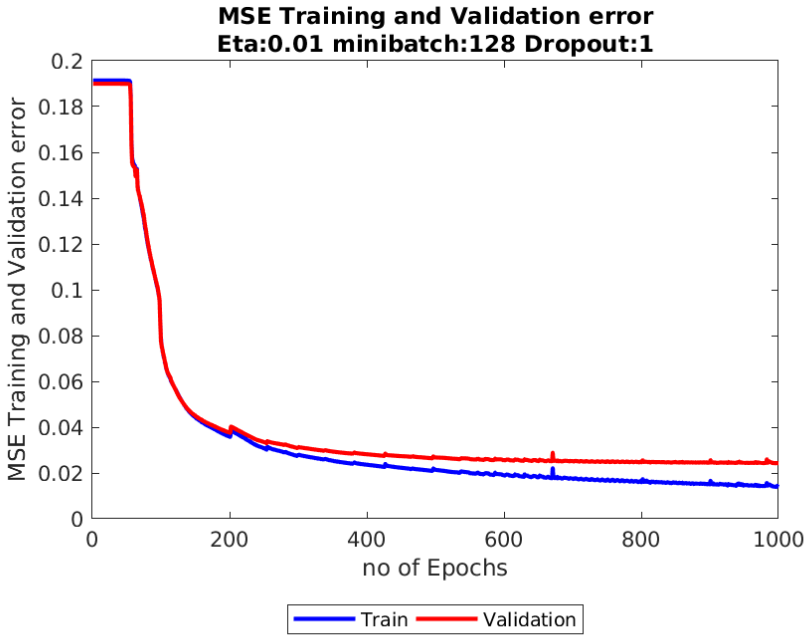





Sum of squares Training and Validation error
Eta:0.01 minibatch:128 Dropout:1
Sum of squares Training and Validation error
3500
3000
2500
2000
1500
1000
500
0
0
200
400
600
800
1000
no of Epochs
Train
Validation


MSE Training and Validation error
Eta:0.01 minibatch:32 Dropout:1
MSE Training and Validation error
0.2
0.15
0.1
0.05
0
0
200
400
600
800
1000
no of Epochs
Train
Validation


MSE Training and Validation error
Eta:0.01 minibatch:64 Dropout:1
MSE Training and Validation error
0.2
0.15
0.1
0.05
0
0
200
400
600
800
1000
no of Epochs
Train
Validation



Observations

- There is an initial lag in larger batch size for decrease of error

  Explanation: For larger batch size; the sum of gradients is much larger initially; hence weights update is much larger, there are some initial lag.

- There are more peaks oscillations (sudden increase) of larger batch size during later parts of epochs.

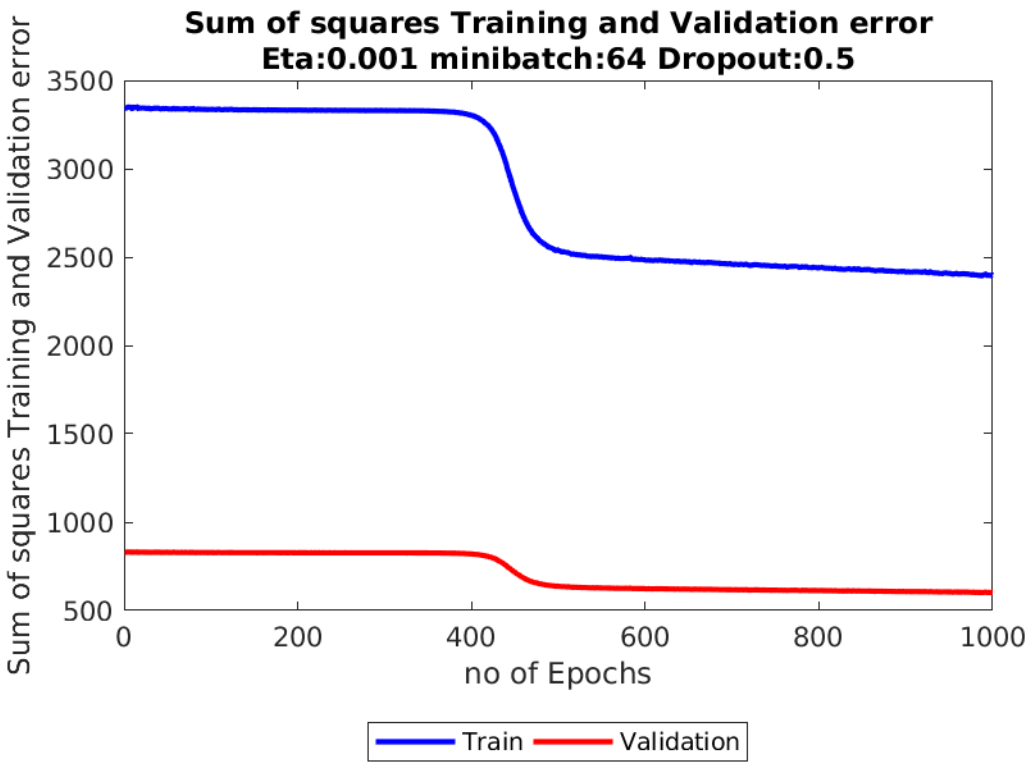
  Explanation: During larger phases the of training, the model tends to overfit a particular sample; but when this happens the gradient for larger batch size overshoot which cause the error to overshoot.

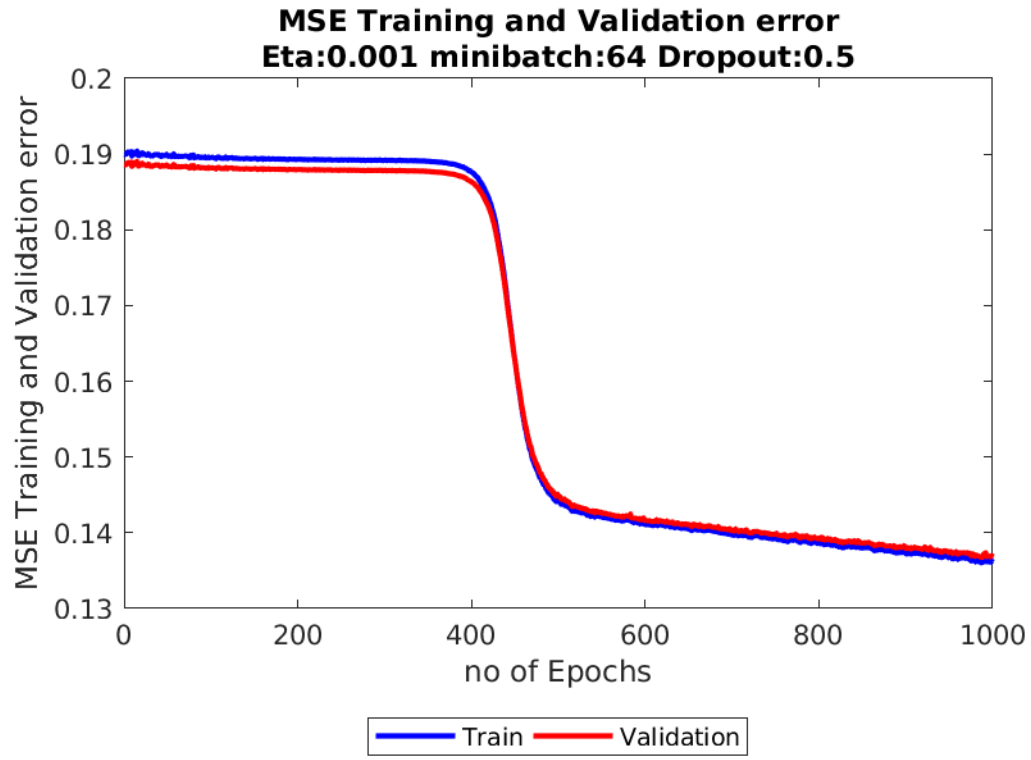
- An optimal value of batch size **prevents** oscillations and reaching a local minimum.

  Explanation: The optimal in this case seems to be batch size 64. It combines benefit of stochastic (less oscillations) and batch mode gradient descent (prevent overfitting).

  This is evident that validation error is less for larger batch size and at same time oscillations are less for smaller batch size.

Plot iii. 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.





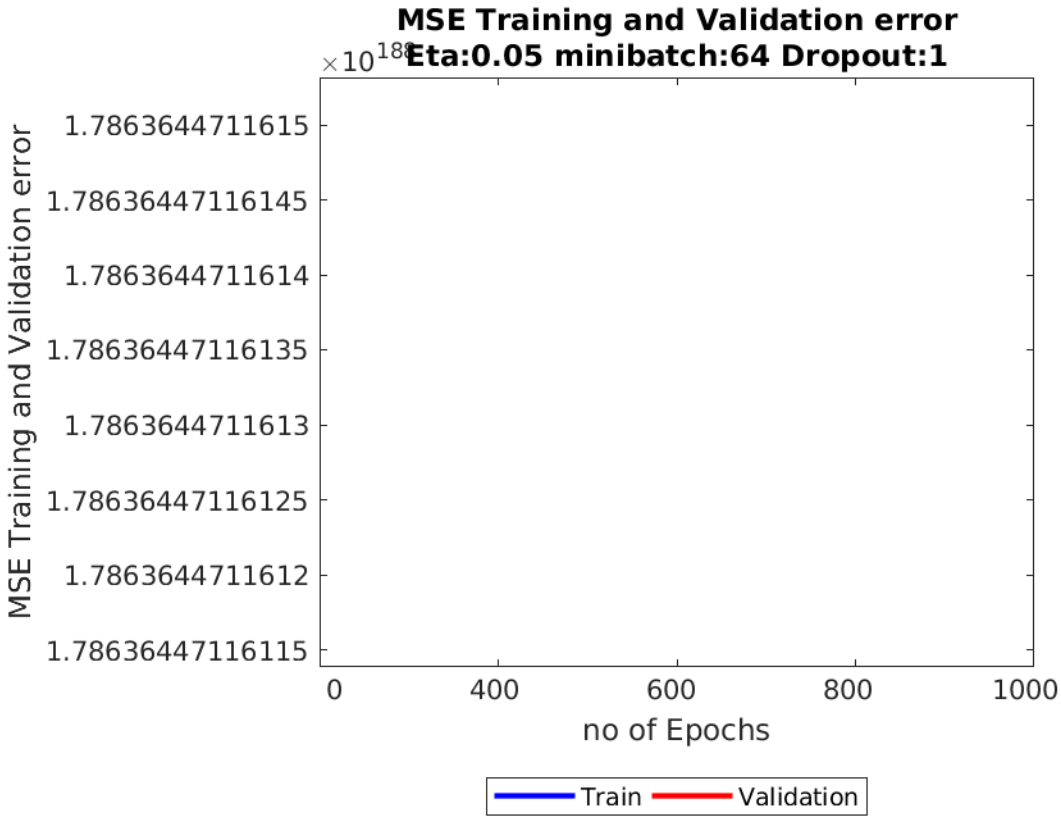
Batch size take is 64

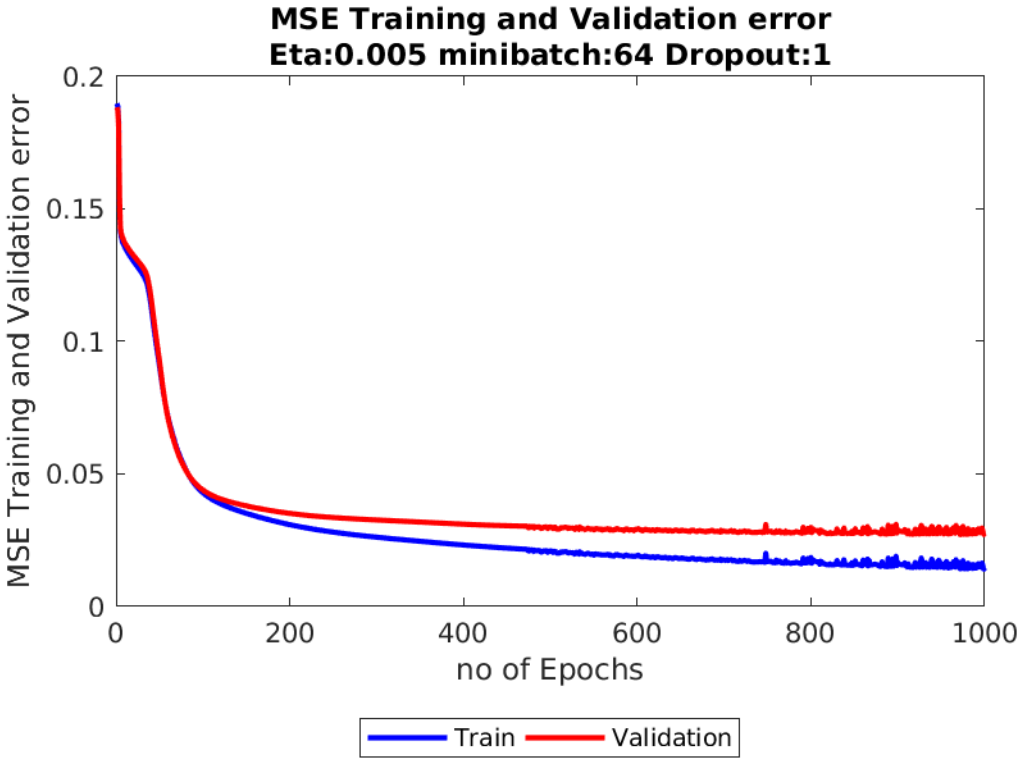
Observations

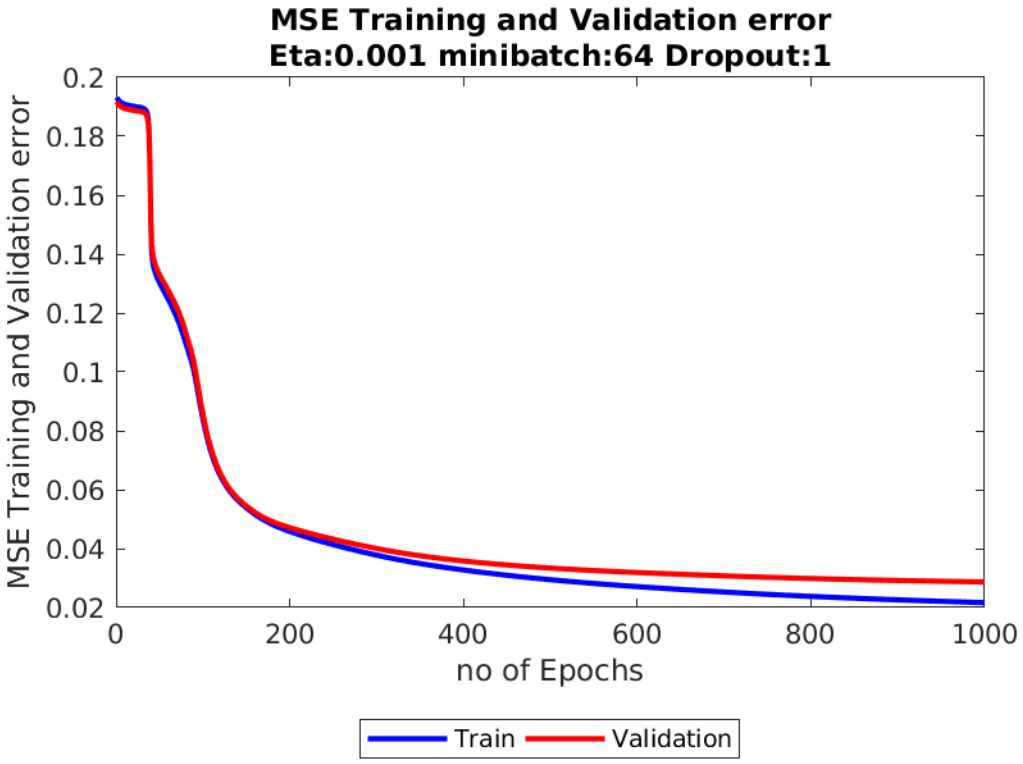
- Overfitting is significantly less. The graph MSE of Training and Validation error nearly overlap.

  Explanation: Dropout is like learning many crippled networks with a single network and hence reduces overfitting. Dropout penalizes the convergence rate. Evident by very slow convergence; as intuitively many networks are being learned simultaneously.

Plot iv. 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)







Observations

- The learning rate 0.05 cause the error to overshoot.

- The learning rate 0.005 caused faster convergence with oscillations in end.

- The learning rate 0.01 caused slower convergence with no oscillations.

Explanation: It is as expected, higher learning rate cause faster convergence with possible oscillations. And lower learning rate, cause no oscillations but very slow convergence.

Part 2) Improved network (approach for competition)

Final model

- It consists of 5 fully layers, with sigmoid activation.

- Network parameter (No of nodes at each layer) [1024 512 64 32 1]

- Adam (Adaptive Moment Estimation) optimiser
  It combines the first and second order moment and decayed over time.

  Separate optimiser for bias and weights were added.

- The initial weights were taken from
  2/sqrt(n(i)+n(j))*randomnormal(0,1)
  where n(i) and n(j) are no of nodes between two nodes.

Techniques that were also implemented

- The dropout was replaced with inverted dropout to separate training and testing phase. Also, inverted dropout was implement to only hidden layers and not on biases. It allowed reducing dropout probability over iterations.
  Reason for not using: Dropout still causes oscillations and very slow convergence.

- RMS Prop: It is a technique to update learning rate over second over moment.
  Reason for not using: Adam proved faster convergence as tis combines benefits of RMS prop and AdaDelta.

Tuning the hyper parameters
A cross validation approach was used. The training data was split in validations and training set. And model tuned on validation set.

References

http://cs231n.github.io/convolutional-networks/
https://medium.com/towards-data-science/types-of-optimization-algorithms-used-in-neural-networks-and-ways-to-optimize-gradient-95ae5d39529f
https://pgaleone.eu/deep-learning/regularization/2017/01/10/anaysis-of-dropout/http://courses.cs.tau.ac.il/Caffe_workshop/Bootcamp/pdf_lectures/Lecture%203%20CNN%20-%20backpropagation.pdf