Samet Yazak
1604262

UNIVERSITY OF BAHÇEŞEHİR

Department of Computer Engineering

CMP5133 Artficial Neural Networks – Assignment 1

1. Abstract

The subject of the assigment is to implement multi layer perceptron with Forest Fires data from UCI Repository. It is a regression problem.

2. Implementation

I implemented the multi layer percepton using R Studio. You can find the source code (Section 5) at the end of the document. To run the code successfully, you need to create two files from source excel data in csv format. Names of the files should be "forestfires_data_tr.csv" for training, forestfires_data_val.csv for validation (or you can simply change the names in code as you want). I split the data to avoid additional library to load data in excel format.

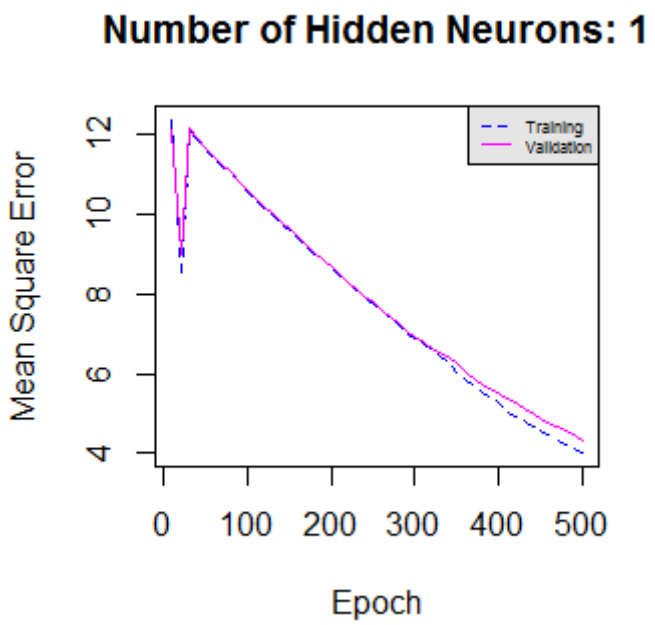
Before implementing the algorithm, I first scaled numeric data and apply 1-of-C coding (applyCCoding function in code) for categorical attributes, month and day. I did not apply c coding for X and Y attribute. I assumed they are numeric. I added bias unit as value of 1 to the set and dropped area (target class) from data set and kept it seperately to make the matrix calculations better.

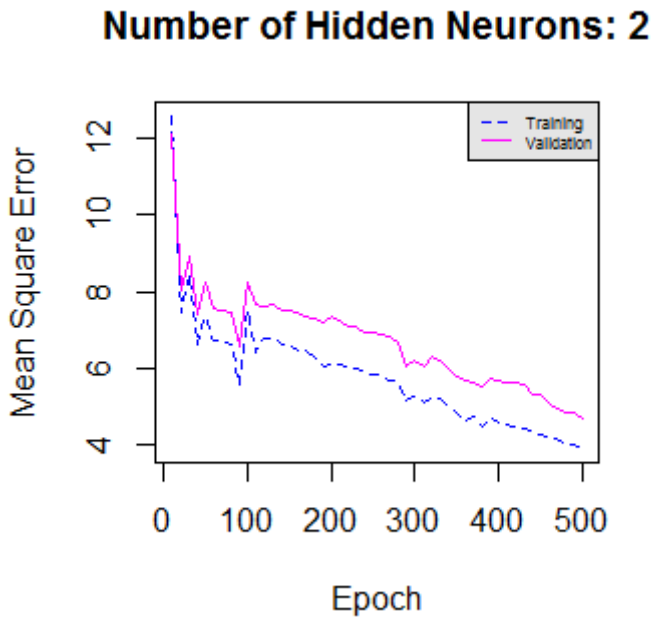
For the hyper parameters, I used learning rate as 0.2 and multiply it by 0.998 after each epoch. For each hidden unit (from 1 to 10), I trained the data with 500 epochs and calculated training and validation error between 10 epochs to reduce running time of the algorithm. I initialized weights randomly between -0.01 and 0.01.

To plot error, I kept all error evaluations in one set and plotted them at the end of the program. You can find the plots in section 3. In plots, there are 3 parameters; epochs, training and validation error in mean square error.

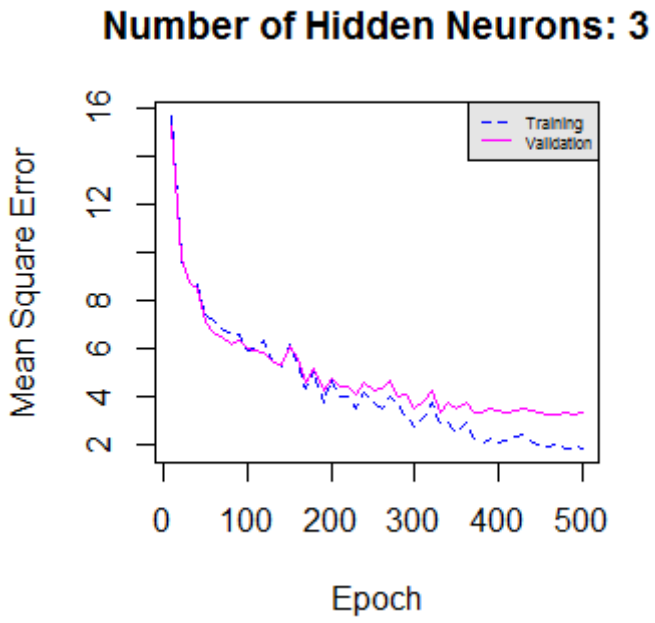
Samet Yazak
1604262

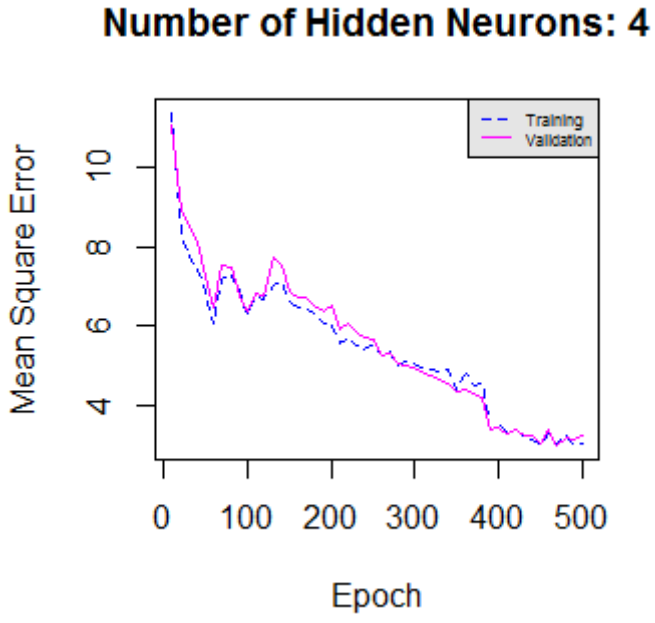
3. Error Plots



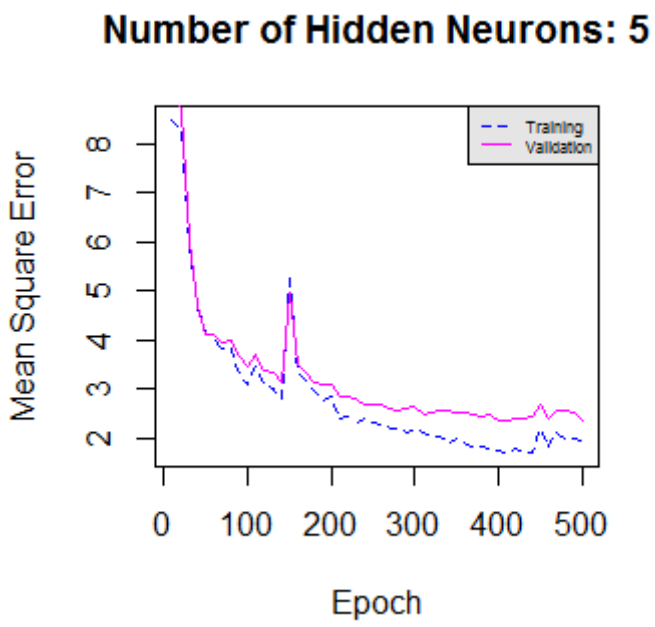


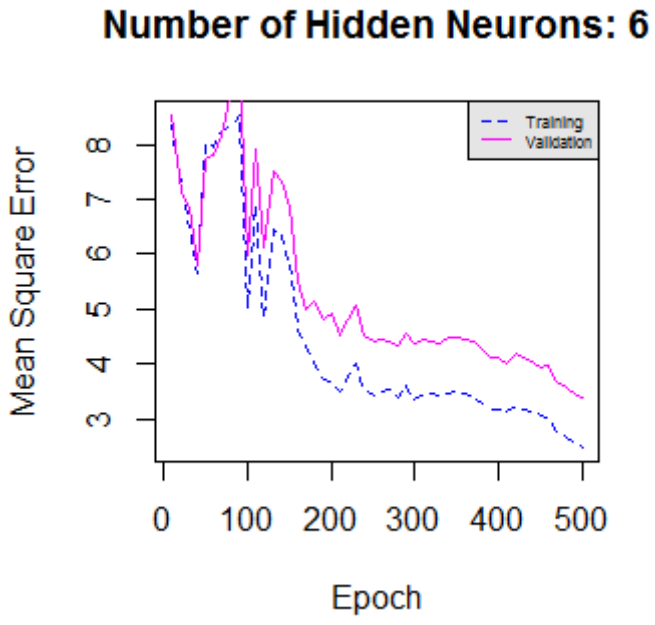
Samet Yazak
1604262



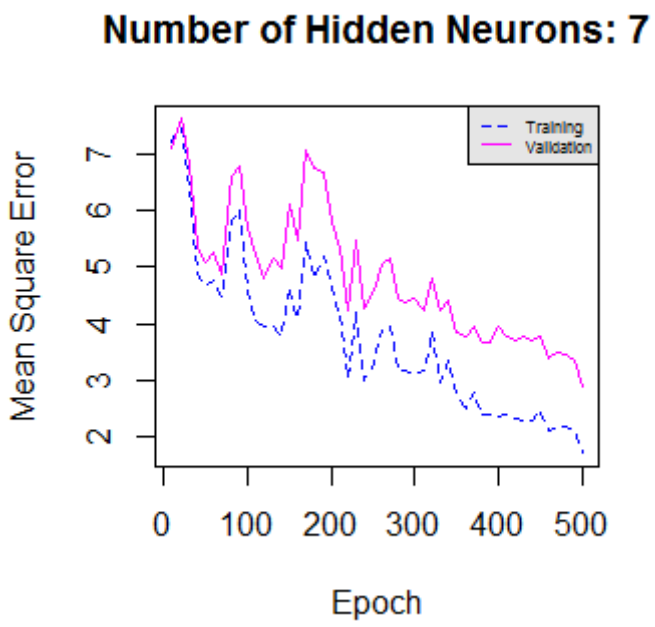


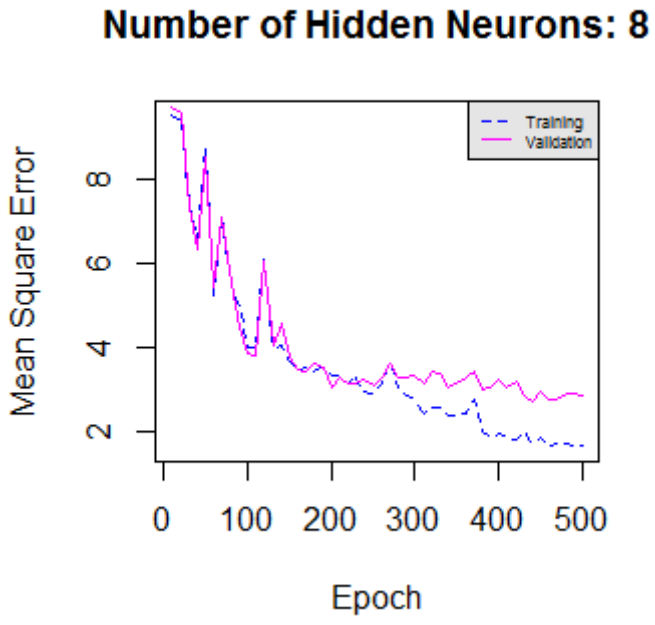
Samet Yazak
1604262



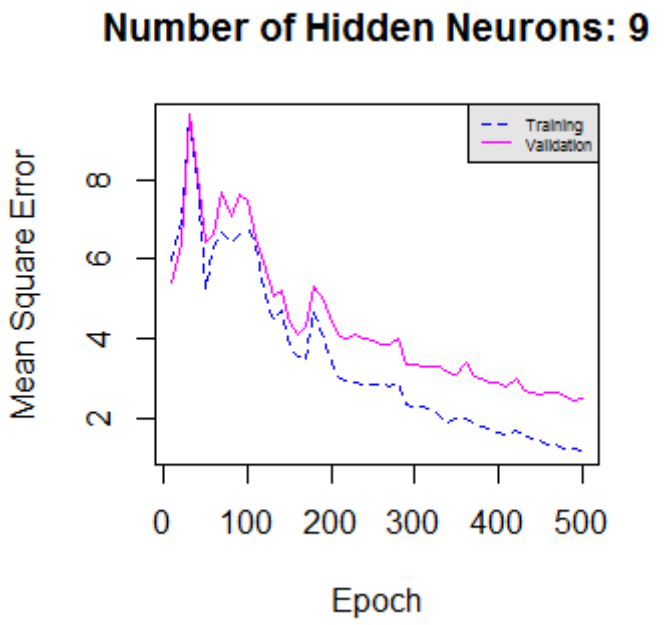


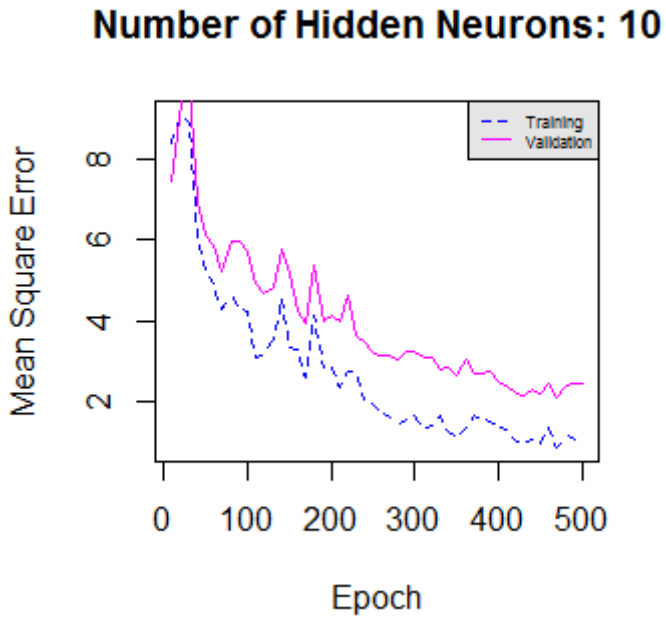
Samet Yazak
1604262





Samet Yazak
1604262





4. Conclusion

As a result of the implementation of multi layer perceptron using Forest Fires data, we see that error decreases both when epoch or number of hidden neurons (from 1 to 10) increases. It seems there is no overfitting in samples in 500 epochs (there is a slight probability for overfitting in H=8 if we increase epoch over 500).

Samet Yazak
1604262

Algorithm completes one epoch for 1 hidden neuron in 3 seconds, and it takes 30 seconds for 10 hidden neurons. It takes 15 seconds for avarage epoch for 1-10 hidden neurons. Considering 500 epochs, it approximately takes 2 hours (avarage) per execution.

5. Source Code

You can find the copied source code below. Source code and other material (csv data, error plots, report) for this assignment is also available on github; https://github.com/yazaksamet/MLP.

```
#### function declerations - start

print(paste0("Start: ", Sys.time()))
standardize <- function(field, value)
{
 return ((value - trainingDataMean[field]) / trainingDataSd[field])
}

destandardize <- function(field, value)
{
 return ((value * trainingDataSd[field]) + trainingDataMean[field])
}

sigmoid <- function(n)
{
 sig <- 1 / (1 + exp(-n))
 return (sig)
}

applyCCoding <- function(dataSet, numericSet)
{
 monthData = data.frame(dataSet$month)
 dayData = data.frame(dataSet$day)

 monthCoded = model.matrix(~ . + 0, data=monthData, contrasts.arg = lapply(monthData, contrasts, contrasts=FALSE))
 dayCoded = model.matrix(~ . + 0, data=dayData, contrasts.arg = lapply(dayData, contrasts, contrasts=FALSE))

 monthCoded = data.frame(monthCoded)
 dayCoded = data.frame(dayCoded)
```

Samet Yazak
1604262

```
 dataSet = data.frame(numericSet,
                      monthCoded$dataSet.monthjan, monthCoded$dataSet.monthfeb,
                      monthCoded$dataSet.monthmar, monthCoded$dataSet.monthapr,
                      monthCoded$dataSet.monthmay, monthCoded$dataSet.monthjun,
                      monthCoded$dataSet.monthjul, monthCoded$dataSet.monthaug,
                      monthCoded$dataSet.monthsep, monthCoded$dataSet.monthoct,
                      monthCoded$dataSet.monthdec,
                      dayCoded$dataSet.daymon, dayCoded$dataSet.daytue,
                      dayCoded$dataSet.daywed, dayCoded$dataSet.daythu,
                      dayCoded$dataSet.dayfri, dayCoded$dataSet.daysat,
                      dayCoded$dataSet.daysun)

 colnames(dataSet) <- c("X","Y", "FFMC", "DMC", "DC", "ISI", "temp", "RH", "wind", "rain", "area",
                        "jan", "feb", "mar", "apr", "may", "jun", "jul", "aug", "sep", "oct", "dec",
                        "mon", "tue", "wed", "thu", "fri", "sat", "sun")
 return (dataSet)
}



#### read data

trainingData = read.csv(file="forestfires_data_tr.csv", header=TRUE, sep=";")
validationData = read.csv(file="forestfires_data_val.csv", header=TRUE, sep=";")

## prepare data

trainingDataNumeric = data.frame(trainingData[c('X','Y','FFMC','DMC','DC','ISI','temp','RH','wind','rain','area')])
validationDataNumeric =
data.frame(validationData[c('X','Y','FFMC','DMC','DC','ISI','temp','RH','wind','rain','area')])

trainingDataNumeric <- data.frame(lapply(trainingDataNumeric, function(x) {
 as.double(sub(",", ".", x, fixed = TRUE))
}))

validationDataNumeric <- data.frame(lapply(validationDataNumeric, function(x) {
```

```
  as.double(sub(",", ".", x, fixed = TRUE))
})))

trainingDataMean = colMeans(trainingDataNumeric)
validationDataMean = colMeans(validationDataNumeric)

trainingDataSd = sapply(trainingDataNumeric, sd)
validationDataSd = sapply(validationDataNumeric, sd)

trainingDataNumeric = data.frame(scale(trainingDataNumeric))
validationDataNumeric = data.frame(scale(validationDataNumeric))

## make data suitable for multi layer perceptron with 1-of-C coding and standardization

trainingSet = applyCCoding(trainingData, trainingDataNumeric)
validationSet = applyCCoding(validationData, validationDataNumeric)

trainingSet = cbind(bias = 1, trainingSet)
validationSet = cbind(bias = 1, validationSet)

trainingOutputSet = data.frame(trainingSet$area)
validationOutputSet = data.frame(validationSet$area)

drops <- c("area")
trainingSet = trainingSet[ , !(names(trainingSet) %in% drops)]
validationSet = validationSet[ , !(names(validationSet) %in% drops)]

## hyper parameters
numberOfOutputNodes = 1

minHiddenNeurons = 1
maxHiddenNeurons = 10

epoch = 500
lerningRateDefault = 0.2
```

Samet Yazak
1604262

```
epochLogNumber = 10



errorMatrix = matrix(0, ncol = 4, nrow = ((epoch / epochLogNumber) * (maxHiddenNeurons - 
minHiddenNeurons + 1)))

errorSet = data.frame(errorMatrix)

colnames(errorSet)<-c("H","epoch","trainingMSE", "validationSME")


for (hiddenNeuronNumber in minHiddenNeurons:maxHiddenNeurons) {

 learningRate = lerningRateDefault

 numberOfHiddenUnits = hiddenNeuronNumber;


 ## weight initialization


 inputWeight = matrix(sample(c(-100:100), numberOfHiddenUnits * (dim(trainingSet)[2]), TRUE)/10000,

                numberOfHiddenUnits, (dim(trainingSet)[2]))


 hiddenWeight = matrix(sample(c(-100:100), numberOfOutputNodes * (numberOfHiddenUnits + 1), 
TRUE)/10000,

                 numberOfOutputNodes, (numberOfHiddenUnits + 1))


 hiddenValues =  matrix(sample(c(0:0), 1 * (numberOfHiddenUnits + 1), TRUE),

                  1, (numberOfHiddenUnits + 1))


 hiddenValues[,1] = 1 ## hidden bias unit


 print(paste0("Start - Epoch: ", Sys.time()))



 ## run epochs
 for (e in 1:epoch){
  for(x in 1:dim(trainingSet)[1])  # train set for each row
  {
      for (h in 2:dim(hiddenValues)[2]) {
         hiddenSum = sum(inputWeight[h-1,] * trainingSet[x,])
```

```
      zh = sigmoid(hiddenSum)
      hiddenValues[1,h] = zh
    }

    y = sum(hiddenWeight[1,] * hiddenValues[1,])
    inputError = trainingOutputSet[x,1] - y
    deltaHidden = learningRate*inputError*hiddenValues

    for (h in 2:dim(hiddenValues)[2]) {
     for (j in 1:dim(trainingSet)[2]) {
       deltaInput = inputError * hiddenWeight[1,h] * hiddenValues[1,h] * (1 - hiddenValues[1,h]) *
trainingSet[x,j]
       inputWeight[h-1,j] = inputWeight[h-1,j] + deltaInput
     }
    }

    hiddenWeight = hiddenWeight + deltaHidden
  }

  if (e %% epochLogNumber == 0) {
   hiddenValuesTr =  matrix(sample(c(0:0), 1 * (numberOfHiddenUnits + 1), TRUE),
                    1, (numberOfHiddenUnits + 1))

   hiddenValuesTr[,1] = 1 ## hidden bias unit

   trainingErrorSum = 0

   # calculate training error
   for(x in 1:dim(trainingSet)[1])
   {
     for (h in 2:dim(hiddenValuesTr)[2]) {
      hiddenSum = sum(inputWeight[h-1,] * trainingSet[x,])
      zh = sigmoid(hiddenSum)
      hiddenValuesTr[1,h] = zh
     }
```

```
    y = sum(hiddenWeight[1,] * hiddenValuesTr[1,])
    trainingErrorSum = trainingErrorSum + ((trainingOutputSet[x,1] - y)^2)
  }

  # training error - mean square error
  trainingMSE = trainingErrorSum / dim(trainingSet)[1]

  hiddenValuesVal =  matrix(sample(c(0:0), 1 * (numberOfHiddenUnits + 1), TRUE),
                    1, (numberOfHiddenUnits + 1))

  hiddenValuesVal[,1] = 1 ## hidden bias unit

  validationErrorSum = 0

  # calculate validation error
  for(x in 1:dim(validationSet)[1])
  {
    for (h in 2:dim(hiddenValuesVal)[2]) {
      hiddenSum = sum(inputWeight[h-1,] * validationSet[x,])
      zh = sigmoid(hiddenSum)
      hiddenValuesVal[1,h] = zh
    }

    y = sum(hiddenWeight[1,] * hiddenValuesVal[1,])
    validationErrorSum = validationErrorSum + ((validationOutputSet[x,1] - y)^2)
  }

  # validation error - mean square error
  validationMSE = validationErrorSum / dim(validationSet)[1]

  epochLogIndex = ((epoch / epochLogNumber) * (hiddenNeuronNumber - 1)) + (e / epochLogNumber)
  errorSet[epochLogIndex,1] = hiddenNeuronNumber
  errorSet[epochLogIndex,2] = e
  errorSet[epochLogIndex,3] = trainingMSE
```

```
    errorSet[epochLogIndex,4] = validationMSE
   }
   learningRate = learningRate * 0.998
   print(paste0("H: ", hiddenNeuronNumber, ", epoch:", e))
 }
}

paste0("ANN Finish: ", Sys.time())

paste0("Plot Start: ", Sys.time())

for (p in minHiddenNeurons:maxHiddenNeurons) {
 subErrorSet = errorSet[errorSet$H == p, ]
 subTrainingError = data.frame(subErrorSet$epoch, subErrorSet$trainingMSE)
 subValidationError = data.frame(subErrorSet$epoch, subErrorSet$validationSME)

 plot(subTrainingError,type="l",col=4, lty = 2, xlab="Epoch", ylab="Mean Square Error")
 #axis(1, xaxp=c(epochLogNumber*5, epoch, (epoch/(epochLogNumber*5)) - 1), las=2)
 lines(subValidationError,col=6,lty = 1, pch = 4)
 title(paste0("Number of Hidden Neurons: ", p))
 legend("topright", c("Training", "Validation"), col = c(4, 6),
       text.col = "black", lty = c(2,1),
       bg = "gray90", cex = 0.5)

}
```