Neural Networks in R

December 31, 2016

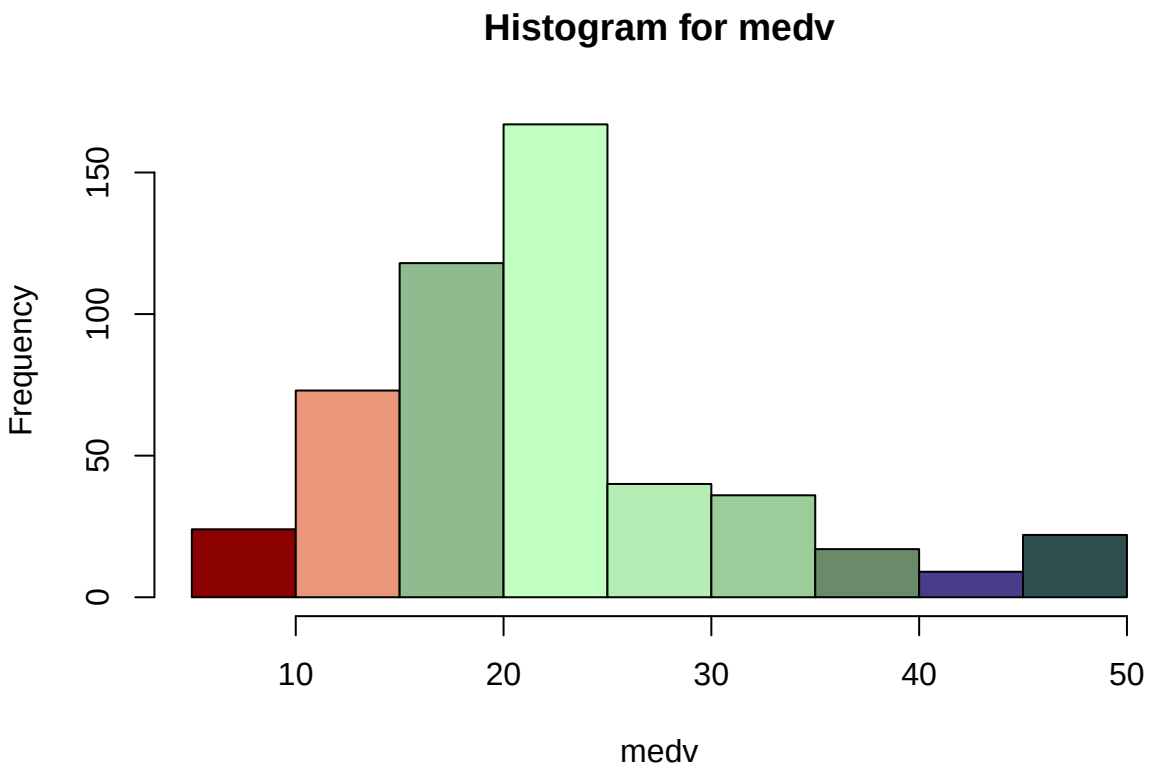
```
#### importing the library MASS for "Boston" dataset
library(MASS)
library(neuralnet)
```

```
## Loading required package: grid
```

```
####  Setting the seed so that we get same results each time
####  we run the  neural nets again
set.seed(123)

#### Storing the data set named "Boston" into DataFrame
DataFrame <- Boston

#### To get the Help on Boston data set uncomment the below line
####  help("Boston")

####  For looking at Structure of Boston data
str(DataFrame)
```

```
## 'data.frame':    506 obs. of  14 variables:
##  $ crim   : num  0.00632 0.02731 0.02729 0.03237 0.06905 ...
##  $ zn     : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
##  $ indus  : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...
##  $ chas   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ nox    : num  0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 ...
##  $ rm     : num  6.58 6.42 7.18 7 7.15 ...
##  $ age    : num  65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...
##  $ dis    : num  4.09 4.97 4.97 6.06 6.06 ...
##  $ rad    : int  1 2 2 3 3 3 5 5 5 5 ...
##  $ tax    : num  296 242 242 222 222 222 311 311 311 311 ...
##  $ ptratio: num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...
##  $ black  : num  397 397 393 395 397 ...
##  $ lstat  : num  4.98 9.14 4.03 2.94 5.33 ...
##  $ medv   : num  24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...
```

```
#### Look at the histogram of the target or outcome variable named medv
hist(DataFrame$medv,col=colors()[100:110],breaks=10,
     main = "Histogram for medv",
     xlab = "medv")
```




```
####  Check the dimension of this data frame
dim(DataFrame)
```

```
## [1] 506  14
```

```
####  Check first 3 rows
head(DataFrame,3)
```

```
##      crim zn indus chas   nox    rm  age    dis rad tax ptratio  black
## 1 0.00632 18  2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3 396.90
## 2 0.02731  0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8 396.90
## 3 0.02729  0  7.07    0 0.469 7.185 61.1 4.9671   2 242    17.8 392.83
##   lstat medv
## 1  4.98 24.0
## 2  9.14 21.6
## 3  4.03 34.7
```

```
####  Check the summary of each variable
####  This will give min and max value for each of the variable
apply(DataFrame,2,range)
```

```
##          crim  zn indus chas   nox    rm   age     dis rad tax ptratio
## [1,]  0.00632   0  0.46    0 0.385 3.561   2.9  1.1296   1 187    12.6
## [2,] 88.97620 100 27.74    1 0.871 8.780 100.0 12.1265  24 711    22.0
```

```
##       black lstat medv
## [1,]   0.32  1.73    5
## [2,] 396.90 37.97   50
```

```
####  Seems like scale of each variable is not same

### We need to Normalize the data in interval [0,1]
### Normalization  is necessay so that each variable is scaled properly
### and none of the variables overdominates in the model
### scale function will give mean =0 and standard deviation=1 for each variable

maxValue <- apply(DataFrame, 2, max)
minValue <- apply(DataFrame, 2, min)
DataFrame<-as.data.frame(scale(DataFrame,center = minValue,
                               scale =maxValue-minValue))


#### Lets partition the dataset  into train and test data set
ind<-sample(1:nrow(DataFrame),400)
trainDF<-DataFrame[ind,]
testDF<-DataFrame[-ind,]


####  Lets take some configuration for neural network
### say 13-4-2-1
### So number of hidden layes=2
### input layer has 13 units=number of predictor variables
### No. of units in first hidden layer =4
### No. of units in second hidden layer =2
### No. of units in output layer=1 as there is only target variable which we want ### to predict.Here ta

### We need this  formula like below in order to use neuralnet
### medv ~ crim + zn + indus + chas + nox + rm + age + dis + rad +
### tax + ptratio + black + lstat
### Below is the code for doing the same without writing the names individually

allVars<-colnames(DataFrame)
predictorVars<-allVars[!allVars%in%"medv"]
predictorVars<-paste(predictorVars,collapse = "+")
form=as.formula(paste("medv~",predictorVars,collapse = "+"))


#### Let's fit the model now
neuralModel<-neuralnet(formula =form,hidden = c(4,2),linear.output = T,
                       data =trainDF)


#### Let's Plot the neural net
plot(neuralModel)


#### Let's Predict for test data set
predictions <- compute(neuralModel,testDF[,1:13])
```

```
#### Let's check the structure of predictions.It is a dataframe.
str(predictions)
```

```
## List of 2
##  $ neurons   :List of 3
##   ..$ : num [1:106, 1:14] 1 1 1 1 1 1 1 1 1 1 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:106] "6" "8" "9" "10" ...
##   .. .. ..$ : chr [1:14] "1" "crim" "zn" "indus" ...
##   ..$ : num [1:106, 1:5] 1 1 1 1 1 1 1 1 1 1 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:106] "6" "8" "9" "10" ...
##   .. .. ..$ : NULL
##   ..$ : num [1:106, 1:3] 1 1 1 1 1 1 1 1 1 1 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:106] "6" "8" "9" "10" ...
##   .. .. ..$ : NULL
##  $ net.result: num [1:106, 1] 0.443 0.292 0.244 0.291 0.293 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:106] "6" "8" "9" "10" ...
##   .. ..$ : NULL
```

```
#### Let's unscale the predictions in order to evaluate the mean squared ####error(MSE)
predictions <- predictions$net.result*(max(testDF$medv)-min(testDF$medv))+                    min(t
actualValues <- (testDF$medv)*(max(testDF$medv)-min(testDF$medv))+min(testDF$medv)


#### Let's calculate the MSE
MSE <- sum((predictions - actualValues)^2)/nrow(testDF)
MSE
```

```
## [1] 0.009414517716
```

```
#### Let's plot the actual and predicted values
plot(testDF$medv,predictions,col='blue',main='Real vs Predicted',pch=1,cex=0.9,type = "p",xlab = "Actua

#### A line with 45 degree slope is showing that predictions and actual values
#### of unseen data i.e test data set are almost the same


####For more better training of model for any dirty data
####preprocessing and cleaning of data is must
####crossvalidation is also must
```