Hw4

PHAN THI THU TRANG

2022-05-24

# Library

library(MASS)

## Warning: package 'MASS' was built under R version 4.1.3

library(plyr)

## Warning: package 'plyr' was built under R version 4.1.3

library(neuralnet)

## Warning: package 'neuralnet' was built under R version 4.1.3

library(ggplot2)  
library(boot)

set.seed(500)  
data=Boston

# Check if there is missing data

apply(data,2,function(x) sum(is.na(x)))

## crim zn indus chas nox rm age dis rad tax   
## 0 0 0 0 0 0 0 0 0 0   
## ptratio black lstat medv   
## 0 0 0 0

No missing data

# Randomly split the data into train (75%) and test (25%) data

index <- sample(1:nrow(data),round(0.75\*nrow(data)))  
train <- data[index,]  
test <- data[-index,]

# Scale and split the data.

Use the min-max method to normalize the data and scale the data in the interval [0,1]

maxs <- apply(data, 2, max)   
mins <- apply(data, 2, min)  
scaled <- as.data.frame(scale(data, center = mins, scale = maxs - mins))  
train\_ <- scaled[index,]  
test\_ <- scaled[-index,]

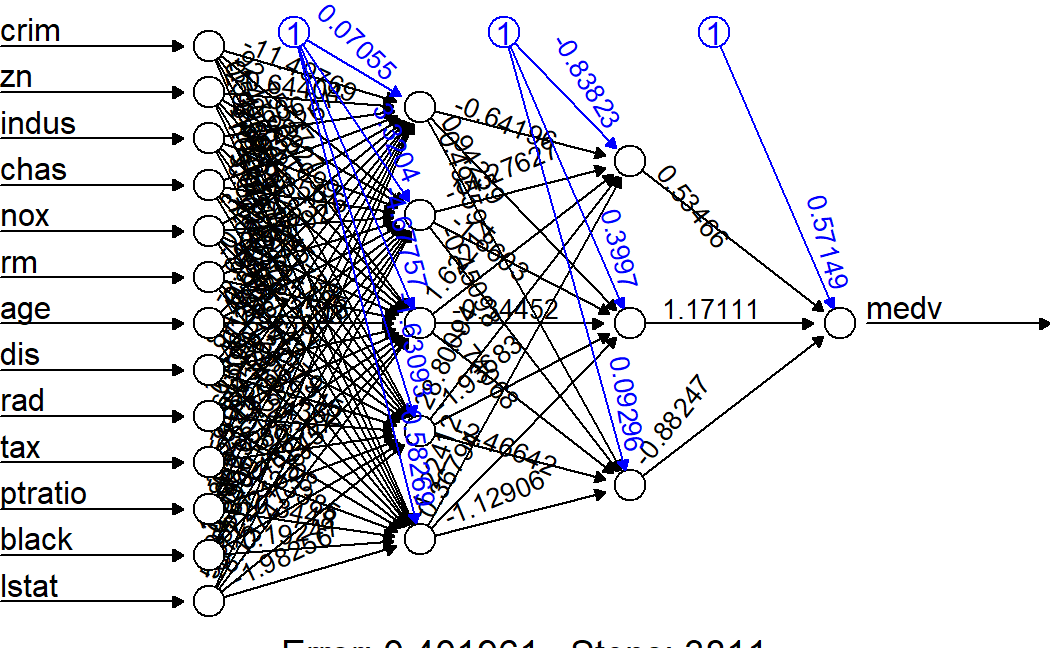
# Parameters

The input layer has 13 inputs, the two hidden layers have 5 and 3 neurons and a single output, which is medv: median value of owner-occupied homes in $1000s

n <- names(train\_)  
f <- as.formula(paste("medv ~", paste(n[!n %in% "medv"], collapse = " + ")))  
nn <- neuralnet(f,data=train\_,hidden=c(5,3),linear.output=T)

# Plot the graphics of the fitted model.

plot(nn)



# Predict medv using the neural network model

pr.nn <- compute(nn,test\_[,1:13])  
pr.nn\_ <- pr.nn$net.result\*(max(data$medv)-min(data$medv))+min(data$medv)  
test.r <- (test\_$medv)\*(max(data$medv)-min(data$medv))+min(data$medv)  
MSE.nn <- sum((test.r - pr.nn\_)^2)/nrow(test\_)

# Fit a linear model to the Boston housing price

lm.fit <- glm(medv~., data=train)  
summary(lm.fit)

##   
## Call:  
## glm(formula = medv ~ ., data = train)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -15.2113 -2.5587 -0.6552 1.8275 29.7110   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 31.111702 5.459811 5.698 2.49e-08 \*\*\*  
## crim -0.111372 0.033256 -3.349 0.000895 \*\*\*  
## zn 0.042633 0.014307 2.980 0.003077 \*\*   
## indus 0.001483 0.067455 0.022 0.982473   
## chas 1.756844 0.981087 1.791 0.074166 .   
## nox -18.184847 4.471572 -4.067 5.84e-05 \*\*\*  
## rm 4.760341 0.480472 9.908 < 2e-16 \*\*\*  
## age -0.013439 0.014101 -0.953 0.341190   
## dis -1.553748 0.218929 -7.097 6.65e-12 \*\*\*  
## rad 0.288181 0.072017 4.002 7.62e-05 \*\*\*  
## tax -0.013739 0.004060 -3.384 0.000791 \*\*\*  
## ptratio -0.947549 0.140120 -6.762 5.38e-11 \*\*\*  
## black 0.009502 0.002901 3.276 0.001154 \*\*   
## lstat -0.388902 0.059733 -6.511 2.47e-10 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for gaussian family taken to be 20.23806)  
##   
## Null deviance: 32463.5 on 379 degrees of freedom  
## Residual deviance: 7407.1 on 366 degrees of freedom  
## AIC: 2237  
##   
## Number of Fisher Scoring iterations: 2

pr.lm <- predict(lm.fit,test)  
MSE.lm <- sum((pr.lm - test$medv)^2)/nrow(test)

# Compare the results of linear regression over its of neural network

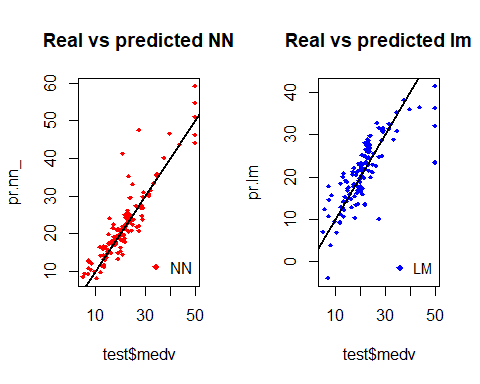
print(paste(MSE.lm,MSE.nn))

## [1] "31.2630222372615 16.4595537665717"

The net is doing a better work than the linear model at predicting medv

# Plot network and the linear model on the test set

par(mfrow=c(1,2))  
plot(test$medv,pr.nn\_,col='red',main='Real vs predicted NN',pch=18,cex=0.7)  
abline(0,1,lwd=2)  
legend('bottomright',legend='NN',pch=18,col='red', bty='n')  
plot(test$medv,pr.lm,col='blue',main='Real vs predicted lm',pch=18, cex=0.7)  
abline(0,1,lwd=2)  
legend('bottomright',legend='LM',pch=18,col='blue', bty='n', cex=.95)



The predictions made by the neural network are (in general) more concetrated around the line than those made by the linear model.

# Preform a 10-fold cross validation for the neural network model.

Here is the 10 fold cross-validated MSE for the linear model:

library(boot)  
set.seed(200)  
lm.fit <- glm(medv~.,data=data)  
cv.glm(data,lm.fit,K=10)$delta[1]

## [1] 23.17094

# Initializing a progress bar

set.seed(450)  
cv.error <- NULL  
k <- 10  
library(plyr)   
pbar <- create\_progress\_bar('text')  
pbar$init(k)

## | | | 0%

for(i in 1:k){  
 index <- sample(1:nrow(data),round(0.9\*nrow(data)))  
 train.cv <- scaled[index,]  
 test.cv <- scaled[-index,]  
 nn <- neuralnet(f,data=train.cv,hidden=c(5,2),linear.output=T)   
 pr.nn <- compute(nn,test.cv[,1:13])  
 pr.nn <- pr.nn$net.result\*(max(data$medv)-min(data$medv))+min(data$medv)   
 test.cv.r <- (test.cv$medv)\*(max(data$medv)-min(data$medv))+min(data$medv)   
 cv.error[i] <- sum((test.cv.r - pr.nn)^2)/nrow(test.cv)   
 pbar$step()  
}

## | |======= | 10% | |============== | 20% | |===================== | 30% | |============================ | 40% | |=================================== | 50% | |========================================== | 60% | |================================================= | 70% | |======================================================== | 80% | |=============================================================== | 90% | |======================================================================| 100%

# Calculate the average MSE

mean(cv.error)

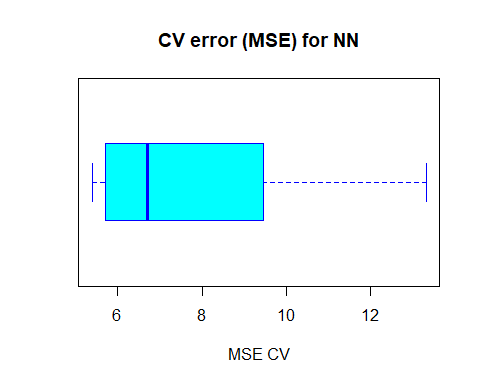
## [1] 7.641292

cv.error

## [1] 13.331937 7.099840 6.580337 5.697609 6.841745 5.771481 10.751406  
## [8] 5.384253 9.452109 5.502201

# Plot the boxplot for the cross-validation error.

boxplot(cv.error,xlab='MSE CV',col='cyan',  
 border='blue',names='CV error (MSE)',  
 main='CV error (MSE) for NN',horizontal=TRUE)



The average MSE for the neural network (13.33) is lower than the one of the linear model although there seems to be a certain degree of variation in the MSEs of the cross validation.