Brauchle_HW_9

Natascha Brauchle

4/21/2019

```
library(ISLR)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(neuralnet)
```

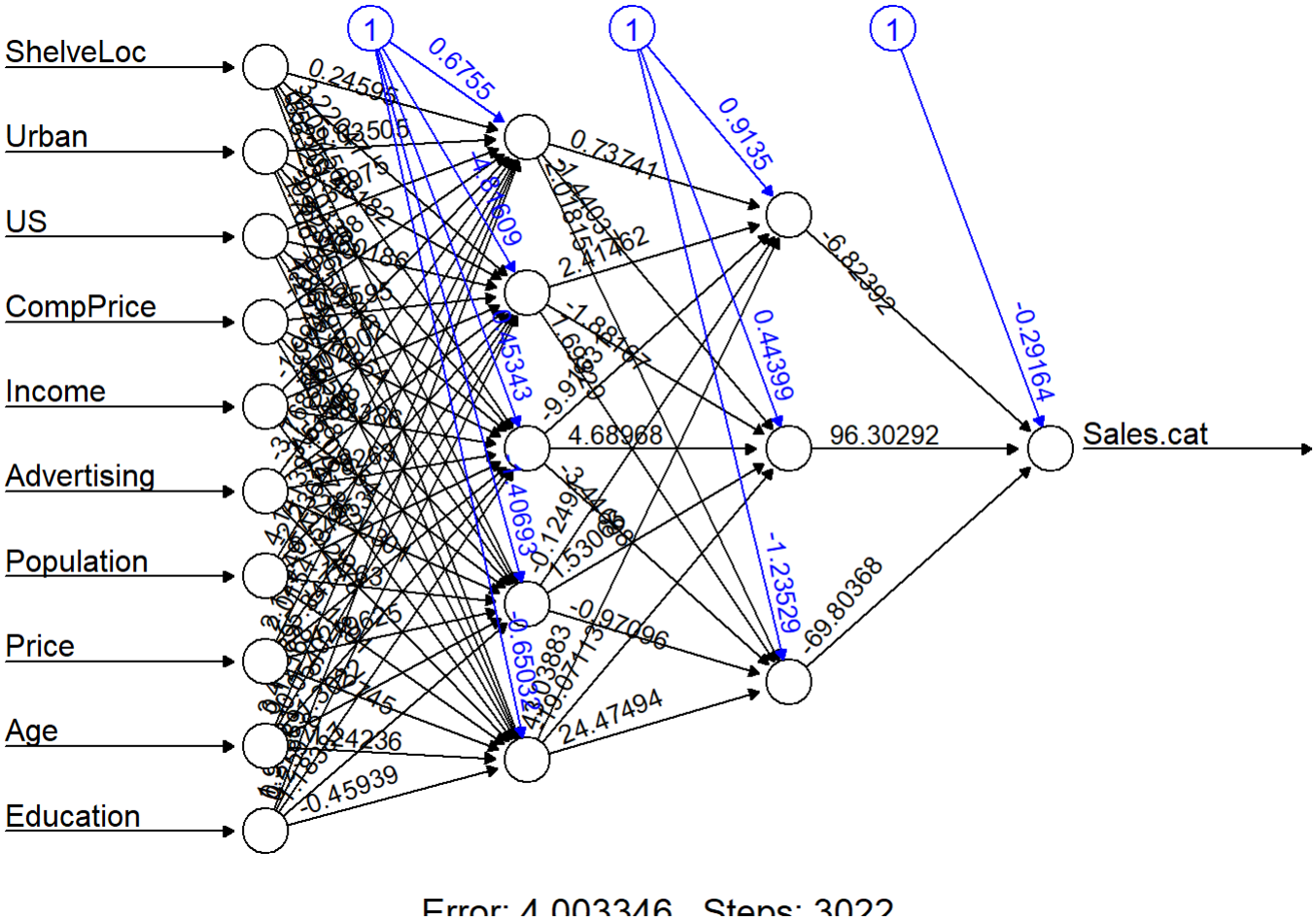
```
##
## Attaching package: 'neuralnet'
```

```
## The following object is masked from 'package:dplyr':
##
##     compute
```

```
set.seed(1)
#categorize Sales
carseats <- Carseats %>% mutate(Sales.cat = ifelse(Sales>8,1,0))
# make categorical variables numeric to use in neural net
carseats[,c(7,10,11)] <- lapply(carseats[,c(7,10,11)], as.numeric)

# scale numeric variables
numeric_carseats <- carseats %>% dplyr::select_if(is.numeric) %>% dplyr::select(-Sales, -Sales.c
at, -7, -10, -11)

maxs <- apply(numeric_carseats, 2, max)
mins <- apply(numeric_carseats, 2, min)

# Use scale() and convert the resulting matrix to a data frame
scaled.data <- as.data.frame(scale(numeric_carseats, center=mins, scale=maxs-mins))
carseats_scaled <- cbind(carseats[,c(7,10,11, 12)], scaled.data)


#create train and test set
subset <- sample(1:nrow(carseats_scaled),round(0.75*nrow(carseats_scaled)))
train <- carseats_scaled[subset,]
test <- carseats_scaled[-subset,]


#make formula of names to plug into neural net
feats <- names(dplyr::select(carseats_scaled, -Sales.cat))
# Concatenate strings
f <- paste(feats,collapse=' + ')
f <- paste('Sales.cat ~',f)
# Convert to formula
f <- as.formula(f)
f
```

```
## Sales.cat ~ ShelveLoc + Urban + US + CompPrice + Income + Advertising +
##     Population + Price + Age + Education
```

```
nn <- neuralnet(f, train, hidden=c(5,3), linear.output=FALSE)
plot(nn, rep = "best")
```



This neural network has 3 layers and two hiiden layers. The first hidden layer has 5 nodes and the second has 3 nodes. This neural network predicts whether a location sold more than 8,000 carseats or not using various weights for each node.

```
library(plyr)
```

```
## ------------------------------------------------------------------------------
```

```
## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)
```

```
## ------------------------------------------------------------------------------
```

```
##
## Attaching package: 'plyr'
```

```
## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

```
#create data frame
best_cv <- data.frame(matrix(ncol = 3, nrow = 0))

#loop to get best number of layers and size of each layer
k <- c(1, 3:10)
for(i in k){
  for(j in 1:i){# j < i
    nn <- neuralnet(f,data=train,hidden=c(i,j),linear.output=F)
    print(i)
    predicted.nn.values <- neuralnet::compute(nn, test[,-c(4)])
    predicted.nn.values$net.result <- sapply(predicted.nn.values$net.result,round,digits=0)
    cv.error <- mean(predicted.nn.values$net.result==test$Sales.cat)
    best_cv <- rbind(best_cv, c(i, j, cv.error))
  }
}
```

```
## [1] 1
## [1] 3
## [1] 3
## [1] 3
## [1] 4
## [1] 4
## [1] 4
## [1] 4
## [1] 5
## [1] 5
## [1] 5
## [1] 5
## [1] 5
## [1] 6
## [1] 6
## [1] 6
## [1] 6
## [1] 6
## [1] 6
## [1] 7
## [1] 7
## [1] 7
## [1] 7
## [1] 7
## [1] 7
## [1] 7
## [1] 8
## [1] 8
## [1] 8
## [1] 8
## [1] 8
## [1] 8
## [1] 8
## [1] 8
## [1] 9
## [1] 9
## [1] 9
## [1] 9
## [1] 9
## [1] 9
## [1] 9
## [1] 9
## [1] 10
## [1] 10
## [1] 10
## [1] 10
## [1] 10
## [1] 10
## [1] 10
## [1] 10
## [1] 10
## [1] 10
```

```
x <- c("i_test", "j_test", "err")
colnames(best_cv) <- x

best_cv[which.max(best_cv$err),]
```

```
##    i_test j_test  err
## 23      7      4 0.91
```

The cross-validation shows that the best accuracy rate comes when using a neural network with 7 layers and a size of 4 nodes for each layer, with an accuracy of 91%.