Neural-Network.R

patriciamaya

2020-11-03

```
#Neural Network Model
# we are predicting sales of child car seats at 400 different stores.

library(ISLR)
```

```
## Warning: package 'ISLR' was built under R version 4.0.2
```

```
data("Carseats")
attach(Carseats)
data<- Carseats
#?Carseats

# ** NORMALIZE DATASET
#we can only normalize numerical variables
num_cols <- unlist(lapply(data, is.numeric))
num_cols
```

```
##       Sales   CompPrice      Income Advertising  Population       Price
##        TRUE        TRUE        TRUE        TRUE        TRUE        TRUE
##   ShelveLoc         Age   Education       Urban          US
##       FALSE        TRUE        TRUE       FALSE       FALSE
```

```
dataNum <- data[,num_cols]  #only numerical variables
str(dataNum)
```

```
## 'data.frame':    400 obs. of  8 variables:
##  $ Sales      : num  9.5 11.22 10.06 7.4 4.15 ...
##  $ CompPrice  : num  138 111 113 117 141 124 115 136 132 132 ...
##  $ Income     : num  73 48 35 100 64 113 105 81 110 113 ...
##  $ Advertising: num  11 16 10 4 3 13 0 15 0 0 ...
##  $ Population : num  276 260 269 466 340 501 45 425 108 131 ...
##  $ Price      : num  120 83 80 97 128 72 108 120 124 124 ...
##  $ Age        : num  42 65 59 55 38 78 71 67 76 76 ...
##  $ Education  : num  17 10 12 14 13 16 15 10 10 17 ...
```

```
#to normalize numerical variables we will use min - max transformation
#x in [min, max]
#x'= x-min / max -min, so x' in [0,1]

#min & max of all columns
mins<- apply(dataNum, 2, min) #1-rows, 2 -columns
maxs<- apply(dataNum, 2, max)

scaled.data<-as.data.frame(scale(dataNum, center = mins, scale= maxs - mins))
summary(scaled.data) #we can see now min is 0 and max is 1.
```

```
##      Sales          CompPrice          Income        Advertising
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.3313   1st Qu.:0.3878   1st Qu.:0.2197   1st Qu.:0.0000
##  Median :0.4604   Median :0.4898   Median :0.4848   Median :0.1724
##  Mean   :0.4607   Mean   :0.4895   Mean   :0.4814   Mean   :0.2288
##  3rd Qu.:0.5728   3rd Qu.:0.5918   3rd Qu.:0.7071   3rd Qu.:0.4138
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
##    Population         Price             Age           Education
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.2585   1st Qu.:0.4551   1st Qu.:0.2682   1st Qu.:0.2500
##  Median :0.5251   Median :0.5569   Median :0.5364   Median :0.5000
##  Mean   :0.5107   Mean   :0.5497   Mean   :0.5150   Mean   :0.4875
##  3rd Qu.:0.7786   3rd Qu.:0.6407   3rd Qu.:0.7455   3rd Qu.:0.7500
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
```

```r
#add to scaled df the variables that were factors (categorical)
data <- data.frame(scaled.data, data[!num_cols] )
str(data)
```

```
## 'data.frame':    400 obs. of  11 variables:
##  $ Sales      : num  0.584 0.69 0.618 0.455 0.255 ...
##  $ CompPrice  : num  0.622 0.347 0.367 0.408 0.653 ...
##  $ Income     : num  0.525 0.273 0.141 0.798 0.434 ...
##  $ Advertising: num  0.379 0.552 0.345 0.138 0.103 ...
##  $ Population : num  0.533 0.501 0.519 0.914 0.661 ...
##  $ Price      : num  0.575 0.353 0.335 0.437 0.623 ...
##  $ Age        : num  0.309 0.727 0.618 0.545 0.236 ...
##  $ Education  : num  0.875 0 0.25 0.5 0.375 0.75 0.625 0 0 0.875 ...
##  $ ShelveLoc  : Factor w/ 3 levels "Bad","Good","Medium": 1 2 3 3 1 1 3 2 3 3 ...
##  $ Urban      : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 1 2 2 1 1 ...
##  $ US         : Factor w/ 2 levels "No","Yes": 2 2 2 2 1 2 1 2 1 2 ...
```

```r
# ** CONSTRUCT NN MODEL
set.seed(123)
indx<- sample(2, nrow(data), replace=T, prob=c(0.8,0.2))
train <- data[indx==1, ]
test <- data[indx==2, ]

#library(neuralnet)
library(nnet)
nn <- nnet(Sales ~ . , data= train, linout= T , size= 10, decay=0.01)
```

```
## # weights:  131
## initial  value 118.897428
## iter  10 value 2.787889
## iter  20 value 1.469418
## iter  30 value 1.355768
## iter  40 value 1.301456
## iter  50 value 1.280936
## iter  60 value 1.268716
## iter  70 value 1.254081
## iter  80 value 1.240087
## iter  90 value 1.234015
## iter 100 value 1.230016
## final  value 1.230016
## stopped after 100 iterations
```

```
#linout -- stands for linear out-put and is used to determine whether the target variable is continuous
#linout = T if numerical varible, F if categorical variables.
#size = X , sets the num of neurons in hidden layer
#decay = ,  regularization term [0,1] -> if small: large network-may overfit, if large: smaller network
summary(nn)
```

```
## a 11-10-1 network with 131 weights
## options were - linear output units  decay=0.01
##   b->h1  i1->h1  i2->h1  i3->h1  i4->h1  i5->h1  i6->h1  i7->h1  i8->h1  i9->h1
##   -0.16   -0.61    0.29    0.07   -0.01    0.66   -0.05    0.06    0.38    0.14
## i10->h1 i11->h1
##    0.14    0.85
##   b->h2  i1->h2  i2->h2  i3->h2  i4->h2  i5->h2  i6->h2  i7->h2  i8->h2  i9->h2
##   -0.06    0.05    0.12   -0.04   -0.07   -0.45   -0.10    0.00   -0.17    0.17
## i10->h2 i11->h2
##   -0.07   -0.19
##   b->h3  i1->h3  i2->h3  i3->h3  i4->h3  i5->h3  i6->h3  i7->h3  i8->h3  i9->h3
##    0.27    0.30   -0.96   -0.80   -0.25    1.04    0.26    0.37   -0.44    0.25
## i10->h3 i11->h3
##   -0.03   -0.32
##   b->h4  i1->h4  i2->h4  i3->h4  i4->h4  i5->h4  i6->h4  i7->h4  i8->h4  i9->h4
##    0.04    0.03    0.04    0.01    0.00    0.00    0.02    0.03    0.01    0.03
## i10->h4 i11->h4
##    0.03    0.03
##   b->h5  i1->h5  i2->h5  i3->h5  i4->h5  i5->h5  i6->h5  i7->h5  i8->h5  i9->h5
##   -0.03    0.88   -0.70   -0.36   -0.96   -0.64   -0.57    0.05    0.66    0.21
## i10->h5 i11->h5
##    0.41    0.38
##   b->h6  i1->h6  i2->h6  i3->h6  i4->h6  i5->h6  i6->h6  i7->h6  i8->h6  i9->h6
##    0.12    0.39   -0.20    0.22    0.71   -0.58   -0.19    0.43    0.22    0.67
## i10->h6 i11->h6
##   -0.04   -0.05
##   b->h7  i1->h7  i2->h7  i3->h7  i4->h7  i5->h7  i6->h7  i7->h7  i8->h7  i9->h7
##   -0.06    0.01   -0.03   -0.04   -0.03   -0.11    0.00   -0.04   -0.06    0.01
## i10->h7 i11->h7
##   -0.05   -0.13
##   b->h8  i1->h8  i2->h8  i3->h8  i4->h8  i5->h8  i6->h8  i7->h8  i8->h8  i9->h8
##    0.20    0.05    0.07    0.12    0.05    0.26    0.17    0.12    0.10   -0.02
## i10->h8 i11->h8
##    0.30    0.33
##   b->h9  i1->h9  i2->h9  i3->h9  i4->h9  i5->h9  i6->h9  i7->h9  i8->h9  i9->h9
##   -0.22   -0.43   -0.61    0.03    0.18    0.09   -0.01   -0.40   -0.15   -0.08
## i10->h9 i11->h9
##    0.13    0.27
##   b->h10  i1->h10  i2->h10  i3->h10  i4->h10  i5->h10  i6->h10  i7->h10
##     0.37     0.43     0.23     0.31     0.10    -0.66     0.64    -0.41
##  i8->h10  i9->h10 i10->h10 i11->h10
##     0.11    -0.28    -0.07     0.56
##   b->o  h1->o  h2->o  h3->o  h4->o  h5->o  h6->o  h7->o  h8->o  h9->o h10->o
##   0.24  -1.19   0.47  -1.23   0.01   1.14   1.30   0.01  -0.24  -0.62   0.90
```

```
#b- bias, i-input,  h1-first neuron in hidden layer, o-output layer

#VISUALIZE FUNCTION
```

```r
#To plot the neural network using nnet we need to use devtools
library(devtools)
```

```
## Warning: package 'devtools' was built under R version 4.0.2
## Loading required package: usethis
## Warning: package 'usethis' was built under R version 4.0.2
```

```r
source_url('https://gist.githubusercontent.com/fawda123/7471137/raw/466c1474d0a505ff044412703516c34f1a4
```

```
## SHA-1 hash of file is 74c80bd5ddbc17ab3ae5ece9c0ed9beb612e87ef
```

```r
plot.nnet(nn)
```

```
## Loading required package: scales
## Loading required package: reshape
## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'
## Loading required package: reshape
## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'
## Loading required package: reshape
## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'
## Loading required package: reshape
## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'
## Loading required package: reshape
## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'
## Loading required package: reshape
## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'
## Loading required package: reshape
## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'
## Loading required package: reshape
## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'
## Loading required package: reshape
## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'
## Loading required package: reshape
## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'
```

```
## Loading required package: reshape
## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'
## Loading required package: reshape
## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'
## Loading required package: reshape
## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'
## Loading required package: reshape
## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'
## Loading required package: reshape
## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'
## Loading required package: reshape
## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'
## Loading required package: reshape
## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'
## Loading required package: reshape
## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'
## Loading required package: reshape
## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'
## Loading required package: reshape
## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'
## Loading required package: reshape
## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'
## Loading required package: reshape
## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'
## Loading required package: reshape
## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'
## Loading required package: reshape
```

```
## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'
```

```
## Loading required package: reshape
```

```
## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'
```

```
## Loading required package: reshape
```

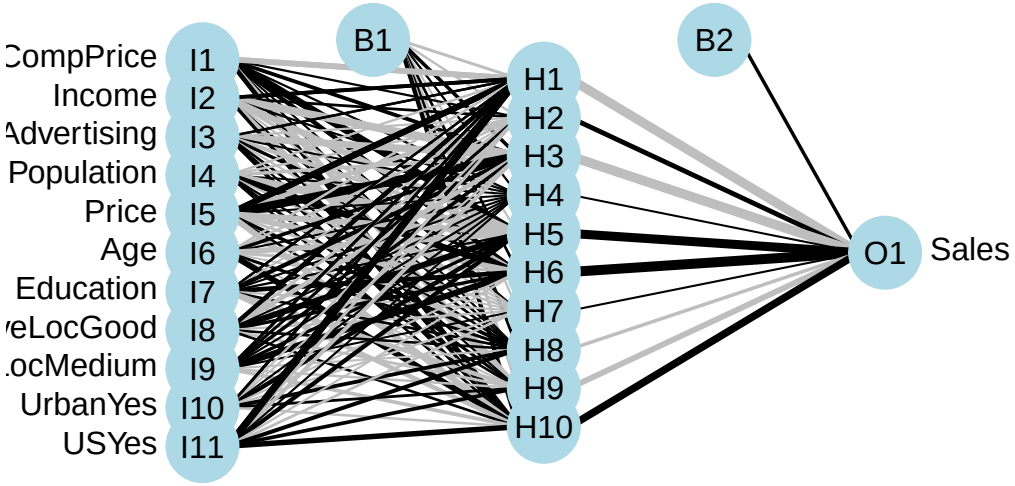
```
## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'
```

```
## Loading required package: reshape
```

```
## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'
```

```
## Loading required package: reshape
```

```
## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'reshape'
```




```
#The darker lines are associated to the higher weights and gray lines are for small weights
#remember data was normalized

#nn$wts              #weights
#nn$fitted.values    #output for the training examples
#nn$residuals         #residuals

#TEST
nn.preds<- predict(nn, test)
nn.preds   #predicted values on test data
```

```
##           [,1]
## 4   0.52088406
## 5   0.37645118
## 8   0.69594851
## 11  0.49095201
```

```
## 16  0.36510071
## 20  0.47949243
## 21  0.37672243
## 24  0.33998799
## 31  0.85072100
## 32  0.49142524
## 50  0.62531793
## 59  0.30645896
## 65  0.45070825
## 67  0.52876758
## 68  0.52758249
## 87  0.47567004
## 88  0.58881321
## 89  0.38807777
## 104 0.32470792
## 106 0.48077012
## 107 0.07396386
## 111 0.50008265
## 114 0.35281685
## 118 0.50693334
## 126 0.61310370
## 132 0.38617805
## 137 0.33489335
## 139 0.61057279
## 145 0.62355666
## 151 0.61260162
## 173 0.72499878
## 179 0.66872340
## 181 0.29475259
## 189 0.41005701
## 190 0.67722007
## 193 0.38196235
## 195 0.47328812
## 202 0.37110482
## 206 0.27318504
## 219 0.62010379
## 220 0.69399831
## 222 0.37954722
## 230 0.56275418
## 238 0.45319213
## 240 0.25188148
## 248 0.14627399
## 249 0.37547949
## 260 0.31928753
## 261 0.55768443
## 262 0.36999990
## 264 0.39162290
## 271 0.76358756
## 277 0.41105809
## 294 0.73626270
## 296 0.21634086
## 297 0.61138054
## 316 0.38732564
## 317 0.90016385
```

```
## 320 0.41363693
## 321 0.36277931
## 327 0.41305825
## 330 0.69933616
## 334 0.36472243
## 340 0.61921694
## 347 0.45716790
## 352 0.59472339
## 356 0.49085516
## 360 0.25779825
## 363 0.22656155
## 373 0.45003937
## 376 0.36902141
## 380 0.40198550
## 386 0.38874874
## 391 0.37259571
## 400 0.57126469
```

```r
#EVALUATE
#MEAN SQUARED ERROR
mse <- mean((nn.preds - test$Sales)^2)
```