Asg3-2.R

Leung Cheuk Wai Dominic 1155093086

Mon Nov 26 15:49:39 2018

```
#RMSC4002 HW3 2
#Leung Cheuk Wai Dominic 1155093086
#setwd("~/QFRM/RMSC4002/Assignment3") #For my own use only

#from 1a
d <- read.csv("credit.csv")
set.seed(980209) #set seed, my birth date is 9th Feb 1998
n <- nrow(d) #Get the length of the dataset
n #display length

## [1] 690

id <- sample(1:n,size=580)  #get the 580 random index for trianing data set
head(id) #display id

## [1] 178 689 371  12 100 653

d1 <- d[id,] #Save 580 data into training dataset d
dim(d1) #check dimension

## [1] 580   7

d2 <- d[-id,] #Save 110 data into testing dataset d1
dim(d2) #check dimension

## [1] 110   7

#2a
library(nnet) #Import library nnet
#set up the ann function for repeating ANN to get the best solution
ann <- function(x,y,size,maxit,linout,try){ #Define parameter for this functi
on
  ann1 <- nnet(y~.,data=x,size=size,maxit=maxit,linout=linout) #the first nne
t result

  v1 <- ann1$value # v1 stores error the first nnet
  for (i in 2:try) { #start a for loop
    ann2 <- nnet(y~., data=x,size=size,maxit=maxit,linout=linout)# get anothe
r nnet result
    if (ann2$value <v1){ #compare for a smaller error
      v1<-ann2$value #replace if the error is smaller
      ann1<-ann2 #replace the final one with new neural network
    }
  }
  ann1# return the result
}
head(d1) #show data

##       Age Address Employ Bank House Save Result
## 178 26.08   8.665  1.415    0   160  150      1
## 689 17.92   0.205  0.040    0   280  750      0
## 371 33.00   2.500  7.000    0   280    0      0
## 12  29.92   1.835  4.335    0   260  200      1
## 100 28.50   1.000  1.000    2   167  500      0
## 653 22.50   0.415  0.335    0   144    0      0

ann7 <- ann(d1[,1:6],d1[,7],size=7,maxit=500,linout=T,try=25) #run size=7 ann

## # weights:  57
## initial  value 291.057713
## iter  10 value 123.550329
## iter  20 value 118.996476
## iter  30 value 109.912828
## iter  40 value 101.642358
## iter  50 value 96.834407
## iter  60 value 94.199875
## iter  70 value 92.985002
## iter  80 value 92.289965
## iter  90 value 91.339337
## iter 100 value 91.126789
## iter 110 value 90.724364
## iter 120 value 89.676083
## iter 130 value 89.020248
## iter 140 value 88.837383
## iter 150 value 88.247620
## iter 160 value 88.160093
## iter 170 value 88.111525
## iter 180 value 88.082831
## iter 190 value 88.081037
## iter 200 value 88.052338
## iter 210 value 87.899144
## iter 220 value 87.773206
## iter 230 value 87.691676
## iter 240 value 87.653825
## iter 250 value 87.647709
## iter 260 value 87.601997
## iter 270 value 87.579633
## iter 280 value 87.383092
## iter 290 value 87.197833
## iter 300 value 87.180918
## iter 310 value 87.178141
## iter 320 value 87.177839
## iter 330 value 87.177789
## final  value 87.177784
## converged
```

Skipped displaying the remaining trial

```
#2b
ann8 <- ann(d1[,1:6],d1[,7],size=8,maxit=500,linout=T,try=25) #run size=8 ann
```

```
## # weights:  65
## initial  value 855.744251
## iter  10 value 133.623072
## iter  20 value 123.741566
## iter  30 value 116.869226
## iter  40 value 114.708241
## iter  50 value 113.772354
## iter  60 value 112.417914
## iter  70 value 112.097727
## iter  80 value 111.844606
## iter  90 value 111.523131
## iter 100 value 111.513574
## iter 110 value 111.479875
## iter 120 value 111.478314
## final  value 111.478252
## converged
```

Skipped displaying the remaining trial

```
ann9 <- ann(d1[,1:6],d1[,7],size=9,maxit=500,linout=T,try=25) #run size=9 ann
```

```
## # weights:  73
## initial  value 619.691337
## iter  10 value 124.292022
## iter  20 value 117.903591
## iter  30 value 110.048055
## iter  40 value 102.230361
## iter  50 value 99.228764
## iter  60 value 95.627053
## iter  70 value 95.144420
## iter  80 value 94.616856
## iter  90 value 94.154749
## iter 100 value 93.107461
## iter 110 value 92.686446
## iter 120 value 92.626553
## iter 130 value 92.549206
## iter 140 value 92.319025
## iter 150 value 91.877068
## iter 160 value 91.137189
## iter 170 value 90.949099
## iter 180 value 90.864100
## iter 190 value 90.844271
## iter 200 value 90.836520
## iter 210 value 90.835670
## final  value 90.835647
## converged
```

Skipped displaying the remaining trial

```
ann10 <- ann(d1[,1:6],d1[,7],size=10,maxit=500,linout=T,try=25) #run size=10
ann
```

```
## # weights:  81
## initial  value 234.644641
## iter  10 value 124.441071
## iter  20 value 118.122078
## iter  30 value 96.761566
## iter  40 value 92.932095
## iter  50 value 89.173561
## iter  60 value 85.184698
## iter  70 value 81.195334
## iter  80 value 79.268105
## iter  90 value 78.449100
## iter 100 value 77.870789
## iter 110 value 77.814992
## iter 120 value 77.809193
## iter 130 value 77.721519
## iter 140 value 77.563465
## iter 150 value 77.420502
## iter 160 value 77.376645
## iter 170 value 77.357954
## iter 180 value 77.292432
## iter 190 value 77.018316
## iter 200 value 76.810182
## iter 210 value 76.769208
## iter 220 value 76.761078
## iter 230 value 76.757237
## iter 240 value 76.750384
## iter 250 value 76.732623
## iter 260 value 76.717982
## iter 270 value 76.717143
## final  value 76.715198
## converged
```

Skipped displaying the remaining trial

```
#2c
ann7$value #display the error
```

```
## [1] 75.29585
```

```
ann8$value #display the error
```

```
## [1] 70.74597

ann9$value #display the error

## [1] 75.26272

ann10$value #display the error

## [1] 70.84947

#ann8 has the smallest error, ann8 is chosen for this model
pred <- 1*(ann8$fit>1/2) #round those fitted value, should be 0 or 1
table(pred,d1$Result) #display the table

##
## pred   0   1
##    0 283  60
##    1  39 198

#The error rate = (60+39)/580=17.06%

#2d
pred2 <- predict(ann8,d2) #use predict() the use the weight of ann8 to comput
e the fitted value for d2
pred2 <- 1*(pred2>1/2) #round those fitted value, should be 0 or 1
table(pred2,d2$Result) #Display the result

##
## pred2  0  1
##     0 53 13
##     1  8 36

#The error rate = (13+8)/110=19.09%
#Precision = 36/(36+8)=81.82%
#Recall = 36/(36+13)=73.47%
#F1 Score = 2/(1/Precision+1/Recall)=77.42%

#2e
# Result for testing data set
# ctree: Error rate = 22.73%. F1 score=69.13%
# ann:   Error rate = 19.09%. F1 score = 77.42%
# Using ANN can obtain a higher F1 score and smaller error, which means we ca
n have a more accurate classification.
# It agress with our intuition that ANN is better, because it has hidden laye
r to recognize and "memorize" more complex pattern.
```