# Neural Networks

## Brian Fortier

### Module 5 Assignment 1

Libraries:

#install.packages("tidyverse")  
#install.packages("caret")  
#install.packages("nnet")  
library(tidyverse)

## -- Attaching packages ---------------------------------------------------------------------------------------------- tidyverse 1.2.1 --

## v ggplot2 3.1.0 v purrr 0.2.5  
## v tibble 1.4.2 v dplyr 0.7.7  
## v tidyr 0.8.2 v stringr 1.3.1  
## v readr 1.1.1 v forcats 0.3.0

## -- Conflicts ------------------------------------------------------------------------------------------------- tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

library(caret)

## Loading required package: lattice

##   
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':  
##   
## lift

library(nnet)

Loading Parole Data and Converting Variables:

parole <- read.csv("parole.csv")  
parole = parole %>% mutate(male = as\_factor(as.character(male))) %>%   
 mutate(male = fct\_recode(male,  
 "male" = "1",  
 "female" = "0"))  
parole = parole %>% mutate(race = as\_factor(as.character(race))) %>%  
 mutate(race = fct\_recode(race,  
 "white" = "1",  
 "otherwise" = "2"))  
parole = parole %>% mutate(state = as\_factor(as.character(state))) %>%  
 mutate(state = fct\_recode(state,  
 "Kentucky" = "2",  
 "Louisiana" = "3",  
 "Virginia" = "4",  
 "Other" = "1"))  
parole = parole %>% mutate(crime = as\_factor(as.character(crime))) %>%  
 mutate(crime = fct\_recode(crime,  
 "larceny" = "2",  
 "drug-related" = "3",  
 "driving-related" = "4",  
 "other" = "1"))  
parole = parole %>% mutate(multiple.offenses = as\_factor(as.character(multiple.offenses))) %>%  
 mutate(multiple.offenses = fct\_recode(multiple.offenses,  
 "multiple" = "1",  
 "other" = "0"))  
parole = parole %>% mutate(violator = as\_factor(as.character(violator))) %>%  
 mutate(violator = fct\_recode(violator,  
 "violator" = "1",  
 "other" = "0"))  
parole = parole %>% drop\_na()  
glimpse(parole)

## Observations: 675  
## Variables: 9  
## $ male <fct> male, female, male, male, male, male, male, ...  
## $ race <fct> white, white, otherwise, white, otherwise, o...  
## $ age <dbl> 33.2, 39.7, 29.5, 22.4, 21.6, 46.7, 31.0, 24...  
## $ state <fct> Other, Other, Other, Other, Other, Other, Ot...  
## $ time.served <dbl> 5.5, 5.4, 5.6, 5.7, 5.4, 6.0, 6.0, 4.8, 4.5,...  
## $ max.sentence <int> 18, 12, 12, 18, 12, 18, 18, 12, 13, 12, 12, ...  
## $ multiple.offenses <fct> other, other, other, other, other, other, ot...  
## $ crime <fct> driving-related, drug-related, drug-related,...  
## $ violator <fct> other, other, other, other, other, other, ot...

Splitting into Train/Test Sets:

set.seed(12345)  
train.rows = createDataPartition(y = parole$violator, p=0.7, list=FALSE)  
train = parole[train.rows,]  
test = parole[-train.rows,]

Neural Network 1:

start\_time = Sys.time()  
fitControl = trainControl(method = "cv",  
 number = 10)  
  
nnetGrid <- expand.grid(size = 12, decay = 0.1)  
  
set.seed(1234)  
nnetBasic = train(violator ~.,  
 parole,  
 method = "nnet",  
 tuneGrid = nnetGrid,  
 trControl = fitControl,  
 verbose = FALSE,  
 trace = FALSE)  
  
end\_time = Sys.time()  
end\_time-start\_time

## Time difference of 2.461436 secs

Training Set Prediction and Confusion Matrix (size 12 decay 0.1):

predNetBasic = predict(nnetBasic, train)  
confusionMatrix(predNetBasic, train$violator, positive = "violator")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction other violator  
## other 409 24  
## violator 9 31  
##   
## Accuracy : 0.9302   
## 95% CI : (0.9034, 0.9515)  
## No Information Rate : 0.8837   
## P-Value [Acc > NIR] : 0.0005254   
##   
## Kappa : 0.6149   
## Mcnemar's Test P-Value : 0.0148061   
##   
## Sensitivity : 0.56364   
## Specificity : 0.97847   
## Pos Pred Value : 0.77500   
## Neg Pred Value : 0.94457   
## Prevalence : 0.11628   
## Detection Rate : 0.06554   
## Detection Prevalence : 0.08457   
## Balanced Accuracy : 0.77105   
##   
## 'Positive' Class : violator   
##

We have an accuracy of 93% which is very good for this model. Out of all the observations, there were only 33 that were misclassified. We also see our sensitivity and specificity to be 0.5636 and 0.98, respectively. Also, we can observe our p-value to be less than 0.05 which is good.

Neural Network 2:

start\_time = Sys.time()  
fitControl = trainControl(method = "cv",  
 number = 10)  
  
nnetGrid <- expand.grid(size = seq(from = 1, to = 12, by = 1),  
 decay = seq(from = 0.1, to = 0.5, by = 0.1))  
  
set.seed(1234)  
nnetFit = train(violator ~.,  
 parole,  
 method = "nnet",  
 tuneGrid = nnetGrid,  
 trControl = fitControl,  
 verbose = FALSE,  
 trace = FALSE)  
  
end\_time = Sys.time()  
end\_time-start\_time

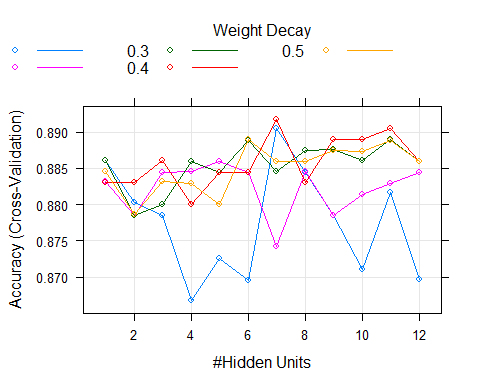
## Time difference of 55.46159 secs

Optimal Model:

nnetFit

## Neural Network   
##   
## 675 samples  
## 8 predictor  
## 2 classes: 'other', 'violator'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 608, 607, 609, 607, 607, 607, ...   
## Resampling results across tuning parameters:  
##   
## size decay Accuracy Kappa   
## 1 0.1 0.8860996 0.27539625  
## 1 0.2 0.8831358 0.22036659  
## 1 0.3 0.8860770 0.22879320  
## 1 0.4 0.8831132 0.15212746  
## 1 0.5 0.8845166 0.09792188  
## 2 0.1 0.8802851 0.27447784  
## 2 0.2 0.8785245 0.22152192  
## 2 0.3 0.8785464 0.19928703  
## 2 0.4 0.8830686 0.19772626  
## 2 0.5 0.8786130 0.12512155  
## 3 0.1 0.8784799 0.29081727  
## 3 0.2 0.8844507 0.25013234  
## 3 0.3 0.8799725 0.19589131  
## 3 0.4 0.8860996 0.26138337  
## 3 0.5 0.8831358 0.22640360  
## 4 0.1 0.8667591 0.24486223  
## 4 0.2 0.8845399 0.28477941  
## 4 0.3 0.8859433 0.25528365  
## 4 0.4 0.8800609 0.17015102  
## 4 0.5 0.8829575 0.20886228  
## 5 0.1 0.8725523 0.28022760  
## 5 0.2 0.8859439 0.29679295  
## 5 0.3 0.8844953 0.23929651  
## 5 0.4 0.8844281 0.21982147  
## 5 0.5 0.8800164 0.18428469  
## 6 0.1 0.8695892 0.24255063  
## 6 0.2 0.8844727 0.31579548  
## 6 0.3 0.8888838 0.24521616  
## 6 0.4 0.8844288 0.22369549  
## 6 0.5 0.8889729 0.24899322  
## 7 0.1 0.8904435 0.35813188  
## 7 0.2 0.8741566 0.25430037  
## 7 0.3 0.8845173 0.25597072  
## 7 0.4 0.8918030 0.23402163  
## 7 0.5 0.8860098 0.22798699  
## 8 0.1 0.8844501 0.34281531  
## 8 0.2 0.8845618 0.33902028  
## 8 0.3 0.8874804 0.27575046  
## 8 0.4 0.8830686 0.19983020  
## 8 0.5 0.8859878 0.24128266  
## 9 0.1 0.8784793 0.30099362  
## 9 0.2 0.8784566 0.28616746  
## 9 0.3 0.8875695 0.30428209  
## 9 0.4 0.8889510 0.23204690  
## 9 0.5 0.8874804 0.23375539  
## 10 0.1 0.8710824 0.27885801  
## 10 0.2 0.8814198 0.32244321  
## 10 0.3 0.8860544 0.26395559  
## 10 0.4 0.8889290 0.26397964  
## 10 0.5 0.8873913 0.24020504  
## 11 0.1 0.8816200 0.35642251  
## 11 0.2 0.8828677 0.28647690  
## 11 0.3 0.8889510 0.30462243  
## 11 0.4 0.8904442 0.28791900  
## 11 0.5 0.8889064 0.22917786  
## 12 0.1 0.8696112 0.28600455  
## 12 0.2 0.8844946 0.31973627  
## 12 0.3 0.8860098 0.27921784  
## 12 0.4 0.8859878 0.24050575  
## 12 0.5 0.8859213 0.22716343  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were size = 7 and decay = 0.4.

plot(nnetFit)



Training Set Prediction and Confusion Matrix (grid):

predNet = predict(nnetFit, train)  
confusionMatrix(predNet, train$violator, positive = "violator")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction other violator  
## other 411 38  
## violator 7 17  
##   
## Accuracy : 0.9049   
## 95% CI : (0.8748, 0.9298)  
## No Information Rate : 0.8837   
## P-Value [Acc > NIR] : 0.08373   
##   
## Kappa : 0.3871   
## Mcnemar's Test P-Value : 7.744e-06   
##   
## Sensitivity : 0.30909   
## Specificity : 0.98325   
## Pos Pred Value : 0.70833   
## Neg Pred Value : 0.91537   
## Prevalence : 0.11628   
## Detection Rate : 0.03594   
## Detection Prevalence : 0.05074   
## Balanced Accuracy : 0.64617   
##   
## 'Positive' Class : violator   
##

Here we see a little smaller accuracy at basically 90.5%. From our plot, we can see our best/optimal model at size = 7 and decay of 0.4.

Testing Set Prediction and Confusion Matrix (size 12 decay 0.1):

predNetBasic2 = predict(nnetBasic, test)  
confusionMatrix(predNetBasic2, test$violator, positive = "violator")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction other violator  
## other 178 16  
## violator 1 7  
##   
## Accuracy : 0.9158   
## 95% CI : (0.8687, 0.9502)  
## No Information Rate : 0.8861   
## P-Value [Acc > NIR] : 0.108358   
##   
## Kappa : 0.4174   
## Mcnemar's Test P-Value : 0.000685   
##   
## Sensitivity : 0.30435   
## Specificity : 0.99441   
## Pos Pred Value : 0.87500   
## Neg Pred Value : 0.91753   
## Prevalence : 0.11386   
## Detection Rate : 0.03465   
## Detection Prevalence : 0.03960   
## Balanced Accuracy : 0.64938   
##   
## 'Positive' Class : violator   
##

The prediction on the testing set of size 12 and decay rate of 0.1, we see an accuracy of 91.58% but with a higher p-value. Specificity almost equals 1 and the sensitivity is at 0.3.

Testing Set Prediction and Confusion Matrix (grid):

predNet2 = predict(nnetFit, test)  
confusionMatrix(predNet2, test$violator, positive = "violator")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction other violator  
## other 176 20  
## violator 3 3  
##   
## Accuracy : 0.8861   
## 95% CI : (0.8341, 0.9264)  
## No Information Rate : 0.8861   
## P-Value [Acc > NIR] : 0.5552509   
##   
## Kappa : 0.1677   
## Mcnemar's Test P-Value : 0.0008492   
##   
## Sensitivity : 0.13043   
## Specificity : 0.98324   
## Pos Pred Value : 0.50000   
## Neg Pred Value : 0.89796   
## Prevalence : 0.11386   
## Detection Rate : 0.01485   
## Detection Prevalence : 0.02970   
## Balanced Accuracy : 0.55684   
##   
## 'Positive' Class : violator   
##

The prediction on the testing set of our grid model gives us an accuracy of 88.6% but a very high p-value at 0.55. There is a similar disparity in the sensitivity and specificity as the other predictions with these values at 0.13 and 0.98, respectively.

We see a fairly high accuracy across all models with each one coming in above 88%. This would lead me to believe there is not overfitting that is occurring between our models since our testing and training sets seem to be performing very similarly.