Application of Neural Networks for Prediction of Authorization to Organ Donation

Pradeep Bagavan

December 21, 2016

Objective

Predict whether the family of a organ donor will say Yes/No to Organ Donation Methodology: Neural Network (Multilayer Perceptron)

In this project, no hidden layers were used. Linear basis function and sigmoidal activation function were used in the MLP

Specify the Input and Output Layer Size

```r
library(ggplot2)
input_layer_size <- 8 # Excluding bias
output_layer_size <- 1
```

Loading data

```r
r <- read.csv("C:/Users/pbagavan/SSIE519/mlclass-ex4/data.csv")

# include bias unit
bias <- rep(1, times=nrow(r))
r <- cbind(bias,r)

# Standardize age (AG) and Admission to Referral Time (AR)
r$AG <- ((r$AG*0.8)/95)+0.1
r$AR <- ((r$AR*0.08)/644.89)+0.1
r <- as.matrix(r)
```

Data Slicing

```r
set.seed(1050)
index <- sample(1:nrow(r),round(0.70*nrow(r)))
train <- r[index,]
test <- r[-index,]
```

Training data

Preview table

```
head(round(train[1:5,],4))
```

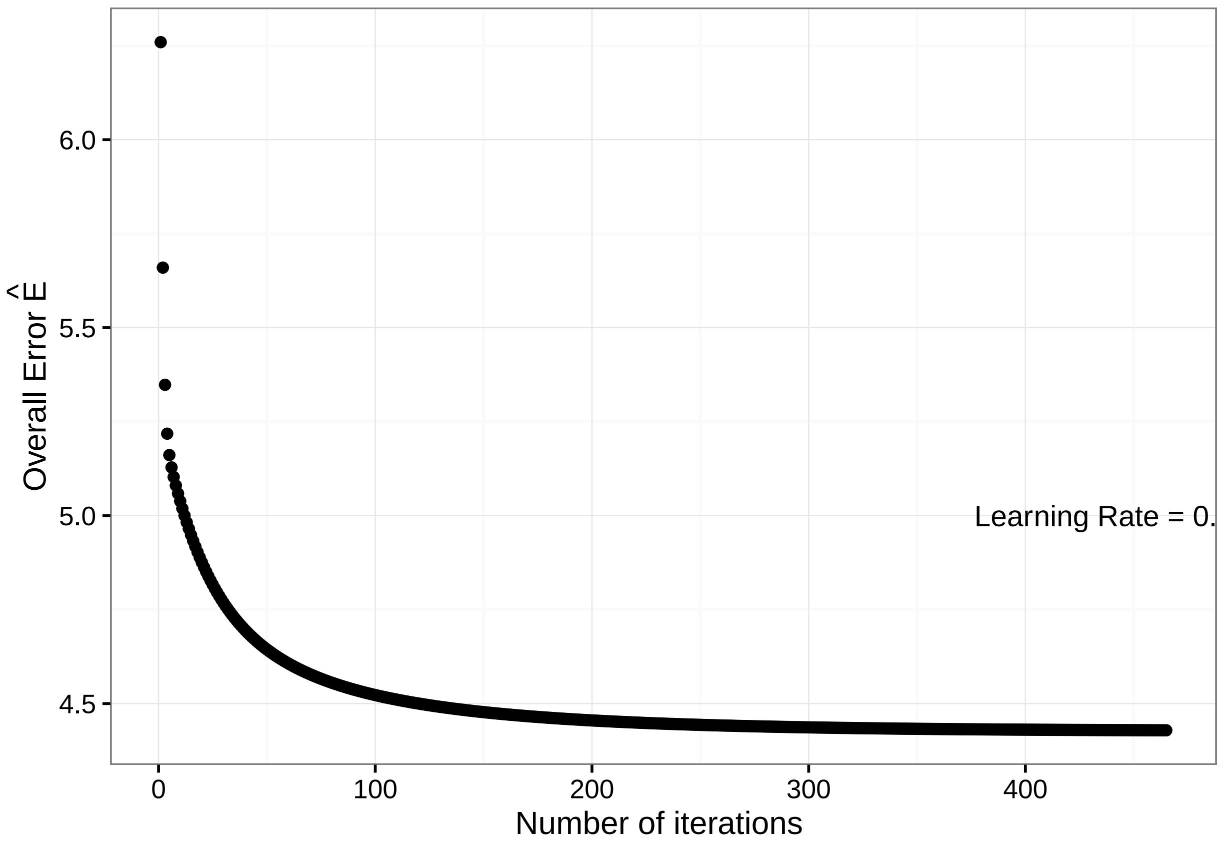
```
##      bias     AR     AG Atele DSA_Def ODC RaceAfAm RaceAsian Timely Auth
## [1,]    1 0.1327 0.5716   0.9     0.1 0.1      0.1       0.1    0.1  0.9
## [2,]    1 0.1025 0.5042   0.1     0.1 0.1      0.9       0.1    0.1  0.9
## [3,]    1 0.1208 0.5716   0.1     0.1 0.1      0.1       0.1    0.1  0.9
## [4,]    1 0.1000 0.7147   0.1     0.9 0.9      0.1       0.1    0.1  0.1
## [5,]    1 0.1195 0.1000   0.1     0.1 0.9      0.1       0.1    0.1  0.1
```

```
# Storing the error in Epsilon_Store
Epsilon_Store <- matrix(data=0,nrow=nrow(train),ncol=output_layer_size)
Overall_Error <- 0


# Input and output variables
train_x <- train[,1:(input_layer_size+1)]
train_Y <- train[,-(1:(input_layer_size+1))]


# Initialize weights
set.seed(500)
w1 <- matrix(data=round((4*runif(input_layer_size+1,0,1)-2)/3,4),
             nrow=output_layer_size,ncol=input_layer_size+1,
             byrow = T)

#t=1{1st Epoch, first step in epoch: zero out the c1 matrix}
t=1
for(t in 1:465){
c1 <- matrix(data=0,nrow=output_layer_size,ncol=input_layer_size+1,
             byrow = T)
 #i=1 {1st training example. First read in values for input layer activations}
  i <- 1
  for(i in 1:nrow(train)){
  # Forward propagate
  s <- train_x[i,] %*% t(w1)
  a <- 1/(1+exp(-s))

  # Backward propagate
  Epsilon <- a-train_Y[i]
  Epsilon_Store[i] <- Epsilon
  d <- Epsilon*a*(1-a)
  # Update c matrix
  c1 <- c1+t(train_x[i,])*d[1,1]
  }

 #End of first epoch. Update weights
 w1 <- w1 - 0.01*c1
 Overall_Error[t] <- 0.5*sum(Epsilon_Store^2)/input_layer_size
}

# Input and out test variables
test_x <- test[,1:(input_layer_size+1)]
test_Y <- test[,-(1:(input_layer_size+1))]
```

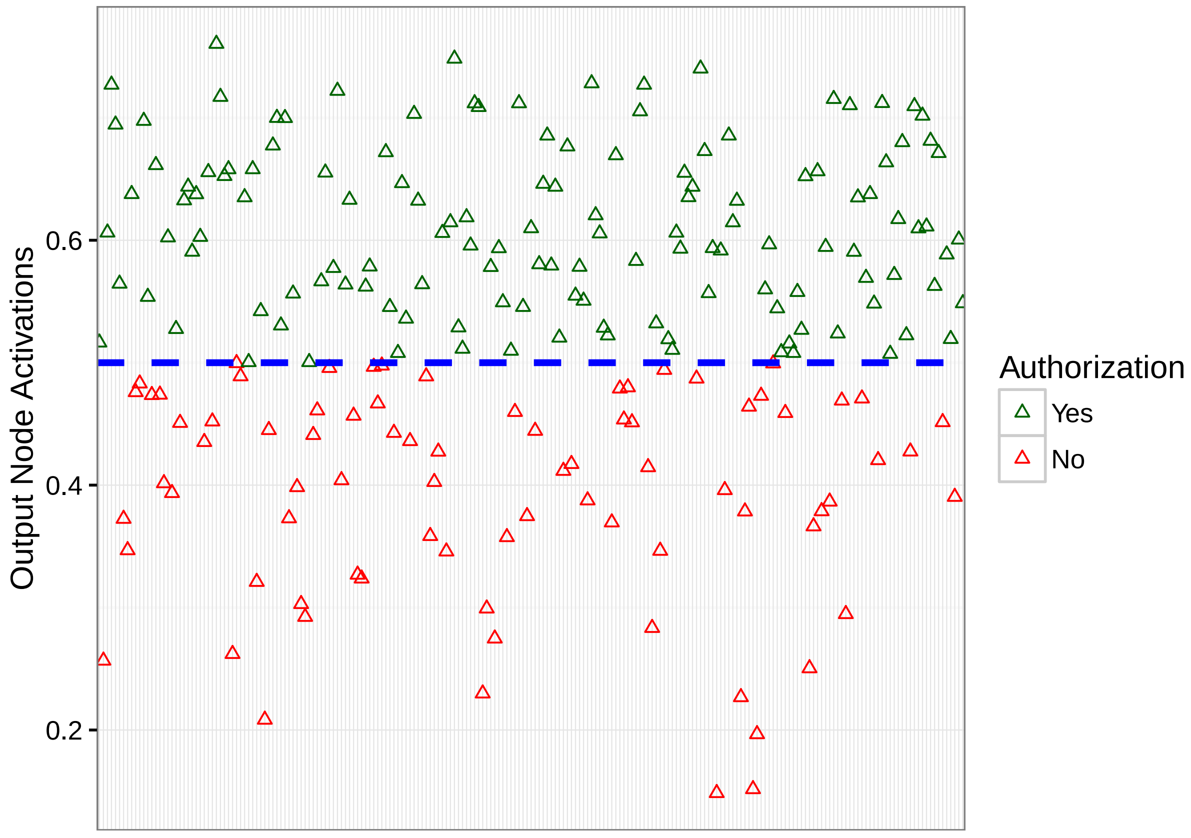
```r
# Save activations of output node
test_a <- rep(0, times=nrow(test))
i <- 1
for(i in 1:nrow(test)){
  # Forward propagate
  s <- test_x[i,] %*% t(w1)
  test_a[i] <- 1/(1+exp(-s))
}
```

Plotting activations

```r
SampleNo <- 1:nrow(test_x)
OpActivations <- test_a
Authorization <- ifelse(OpActivations>=0.5,"darkgreen","red")
plot_data <- as.data.frame(cbind(SampleNo,round(OpActivations,2),Authorization))

#Overall_Error
ggplot() + geom_point(aes(x=1:t,y=Overall_Error)) + theme_bw() +
  xlab("Number of iterations ")+ylab(expression(paste("Overall Error ",hat(E))))+
  annotate("text",x=440,y=5, label= "Learning Rate = 0.01")
```



```r
ggplot(plot_data, aes(x=SampleNo,y=OpActivations)) +
  geom_point(aes(group=Authorization,color=Authorization),shape=2) +
  ylab("Output Node Activations")+
  geom_hline(aes(yintercept=0.5), lty=2,color="blue",lwd=1.2)+
    scale_color_manual(values = c("darkgreen", "red"), labels = c("Yes", "No"))+
  theme_bw()+
  theme(axis.title.x=element_blank(),axis.text.x=element_blank(),
        axis.ticks.x=element_blank())
```



Converting Auth from continuous to categorical variable

```r
plot_data$test_Y_cat <- ifelse(test_Y ==0.1,"No","Yes")
plot_data$test_Y_cat <- factor(plot_data$test_Y_cat, levels = c("Yes","No"))
# Converting Predicted varible into categorical variable
plot_data$pred_Y_cat <- ifelse(OpActivations>=0.5,"Yes","No")
plot_data$pred_Y_cat <- factor(plot_data$pred_Y_cat, levels = c("Yes","No"))
plot_data$category <- ifelse(plot_data$test_Y_cat=="Yes" & plot_data$pred_Y_cat=="Yes","A",
                      ifelse(plot_data$test_Y_cat=="No" & plot_data$pred_Y_cat=="Yes","B",
                      ifelse(plot_data$test_Y_cat=="Yes" & plot_data$pred_Y_cat=="No","C","D")))

#Classification Accuracy
class.accuracy <- round((sum(plot_data$category=="A"|plot_data$category=="D")/nrow(plot_data))*100,2)
class.accuracy
```

```
## [1] 66.05
```

```
# Misclassification Rate
misclass <- round((sum(plot_data$category=="B"|plot_data$category=="C")/nrow(plot_data))*100,2)
misclass
```

```
## [1] 33.95
```

Threshold Distribution

```
df <- as.data.frame(cbind(test_a,test_Y))

plot_distribution <- function(df,threshold){
  v <- rep(NA, nrow(df))
  v <- ifelse(df$test_a >= threshold & df$test_Y == 0.9, "TP", v)
  v <- ifelse(df$test_a >= threshold & df$test_Y == 0.1, "FP", v)
  v <- ifelse(df$test_a < threshold & df$test_Y == 0.9, "FN", v)
  v <- ifelse(df$test_a < threshold & df$test_Y == 0.1, "TN", v)

  df$pred_type <- v

  ggplot(df, aes(x=test_Y, y=test_a))+
  geom_violin(fill=rgb(1,1,1,alpha = 0.6), color=NA)+
  geom_jitter(aes(color=pred_type),alpha=0.6)+
  geom_hline(yintercept=threshold,color="red",alpha=0.6)+
  scale_color_discrete(name=" ")+
  labs(title=sprintf("Threshold = %.2f", threshold))+
  xlab("Actual Value")+ylab("Predicted Value")
}
```

ROC Calculation

```
calculate_roc <- function(df, cost_of_fp, cost_of_fn, n=100) {
  tpr <- function(df, threshold) {
    sum(df$test_a >= threshold & df$test_Y == 0.9) / sum(df$test_Y == 0.9)
  }

  fpr <- function(df, threshold) {
    sum(df$test_a >= threshold & df$test_Y == 0.1) / sum(df$test_Y == 0.1)
  }

  cost <- function(df, threshold, cost_of_fp, cost_of_fn) {
    sum(df$test_a >= threshold & df$test_Y == 0.1) * cost_of_fp +
      sum(df$test_a < threshold & df$test_Y == 0.9) * cost_of_fn
  }

  roc <- data.frame(threshold = seq(0,1,length.out=n), tpr=NA, fpr=NA)
  roc$tpr <- sapply(roc$threshold, function(th) tpr(df, th))
  roc$fpr <- sapply(roc$threshold, function(th) fpr(df, th))
  roc$cost <- sapply(roc$threshold, function(th) cost(df, th, cost_of_fp, cost_of_fn))
```

```r
  return(roc)
}

# ROC plotting
plot_roc <- function(roc, threshold, cost_of_fp, cost_of_fn) {
  library(gridExtra)
  library(grid)
  norm_vec <- function(v) (v - min(v))/diff(range(v))

  idx_threshold = which.min(abs(roc$threshold-threshold))

  col_ramp <- colorRampPalette(c("green","orange","red","black"))(100)
  col_by_cost <- col_ramp[ceiling(norm_vec(roc$cost)*99)+1]
  p_roc <- ggplot(roc, aes(fpr,tpr)) +
    geom_line(color=rgb(0,0,1,alpha=0.3)) +
    geom_point(color=col_by_cost, size=4, alpha=0.5) +
    coord_fixed() +
    geom_line(aes(threshold,threshold), color=rgb(0,0,1,alpha=0.5)) +
    labs(title = sprintf(" ")) + xlab("1-Specificity") + ylab("Sensitivity") +
    geom_hline(yintercept=roc[idx_threshold,"tpr"], alpha=0.5, linetype="dashed") +
    geom_vline(xintercept=roc[idx_threshold,"fpr"], alpha=0.5, linetype="dashed")

  p_cost <- ggplot(roc, aes(threshold, cost)) +
    geom_line(color=rgb(0,0,1,alpha=0.3)) +
    geom_point(color=col_by_cost, size=4, alpha=0.5) +
    labs(title = sprintf(" ")) +
    geom_vline(xintercept=threshold, alpha=0.5, linetype="dashed")+
    xlab("Threshold")+ylab("Cost")
  sub_title <- sprintf("      threshold at %.2f - cost of FP = %d, cost of FN = %d", threshold, cost_of

  grid.arrange(p_roc, p_cost, ncol=2, sub=textGrob(sub_title, gp=gpar(cex=1), just="bottom"))
}
```

Cost-Specific Threshold Values

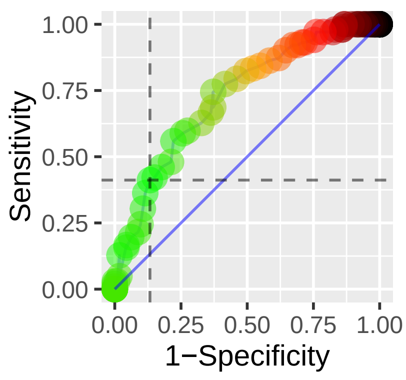
False Positives (FP): The model predicting incorrectly that the family will authorize when it actually does not

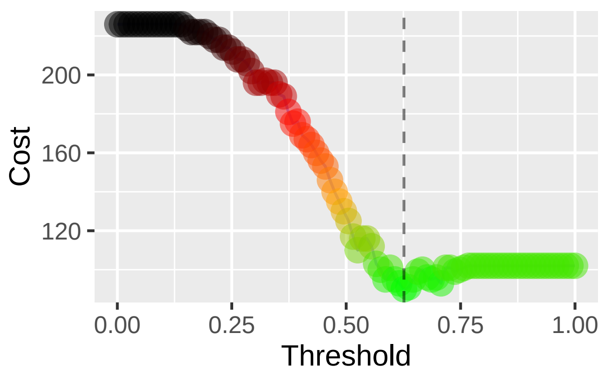
If I assume cost of 2 for FP and 1 for False Negatives (FN)

```r
roc <- calculate_roc(df, 2,1,n=100)
Best_Threshold <- roc$threshold[which.min(roc$cost)]
Best_Threshold
```

```
## [1] 0.6262626
```

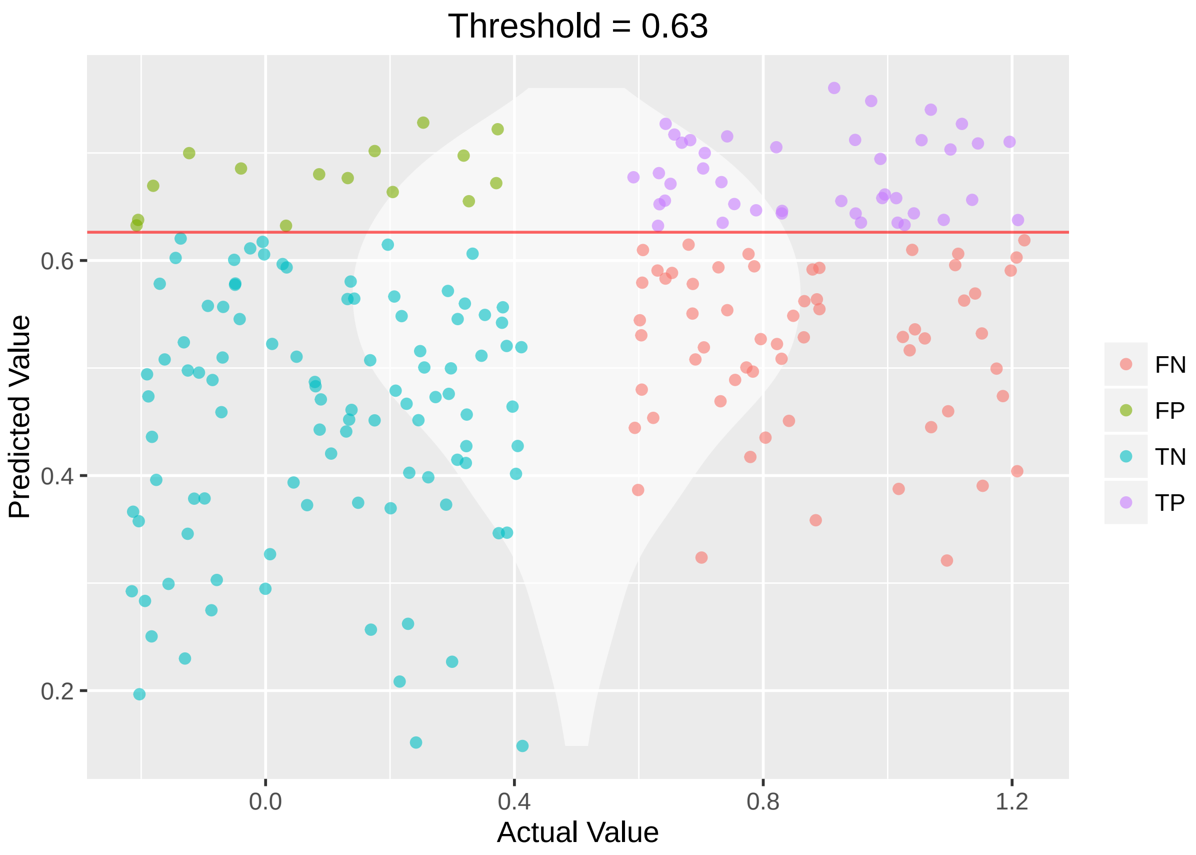
```r
plot_roc(roc, Best_Threshold, 2, 1)
```





hreshold at 0.63 − cost of FP = 2, cost of FN = 1

```
plot_distribution(df, Best_Threshold)
```


Threshold = 0.63
Predicted Value
0.2
0.4
0.6
Actual Value
0.0
0.4
0.8
1.2
FN
FP
TN
TP