To Investigate Airline Arrival Delays

By Pema R. Lama

The purpose of this analysis is to study factors that may influence arrival time delays of commercial airline flights in the USA and to find an appropriate model for future use. We will utilize the Artificial Neural Networks (ANN) theory (one of the Machine Learning algorithms) with R software packages. All necessary R-code will be shown in the Appendix.

We will use publicly available data from the ASA Statistical Computing and Statistical Graphics Data expo 2009. This site contains detailed information on every commercial flight in the United States, from 1987 to 2008. We are going to use only the 1989 data from this large dataset. This 1989 dataset contains 5,041,200 observations with 29 features for each.

It is neither necessary nor feasible (given the limitation of computer memory) to use all 5,000,000 observations for the purpose of this study; hence, we will use only 5,000 observations. We will use r-command, sample(), to select a random sample of size 5,000 from the population of 5,041,200 observations, so that each of these observations is equally likely to be selected.

Of the twenty-nine features, only a few are relevant to our study. The features of interest are: arrival delay, departure delay, flight distance, and flight time delay. The arrival delay is the response varible and the remaining features are predictor variables.

Descriptions of each features:

1. ArrDelay represents: Arrival delay, in minutes. Difference in minutes between scheduled and actual arrival time. Early arrivals show negative numbers.
2. DepDelay represents: Departure delay, in minutes. Difference in minutes between scheduled and actual departure time. Early departures show negative numbers.
3. Distance: Distance between airports (in miles).
4. FlightTimeDelay: Difference in actual flight time and scheduled flight time (hhmm).

The flight time delay feature is not directly listed in this dataset, but it can be easily calculated. The flight time delay is the difference between the actual flight time and the scheduled flight time, both of which are listed in the dataset. See the detailed structure of the dataset in the Appendix.

Because of the wide variations in the ranges of the variables in this dataset, we need to rescale them into the unit interval in order to perform the Neural Network analysis. To do this we will use the normalize() function; see Appendix for details.

To illustrate, for example, the normalized arrival delay versus the original arrival delay, we use the summary() function:

summary(ArrDelay\_norm$ArrDelay)

Minimum 1st Quarter Median Mean 3rd Quarter Maximum  
 0.0000 0.1140 0.1368 0.1538 0.1694 1.0000

summary(r6$ArrDelay)

Minimum 1st Quarter Median Mean 3rd Quarter Maximum  
 -39.000 -4.000 3.000 8.208 13.000 268.000

Note that the normalized arrival time minimum and maximum values are 0 and 1, compared with the original minimum and maximum values, which are -39 and 268. This indicates that the normalization worked. However, the 50 percent of the normalized data lying between the first quarter and the third-quarter is squashed between 0.114 and 0.169 near the left end of the unit interval. That is caused by skewedness of the original data. Moreover, the mean is greater than the median, so we know that we have right-skewed observations. The histograms in the Appendix visually display the skewedness. Also, the minimum arrival delay time of -39 indicates that flight arrived earlier than scheduled and the maximum value of 268 indicates that flight arrived later than it was expected; both of these extreme values seem outliers.

The purpose of the normalization transformation is to be able to perform neural network analysis; so at the end we may need to convert any transformed data back to the original scale.

We will partition our dataset into two batches, a training dataset and a testing dataset. The training dataset is used to build the neural network and the testing dataset is used to evaluate how well the model works. Experience has shown that the training dataset should be three to four times as large as the testing dataset.

Therefore, we will use eighty percent of the dataset to build the neural network model and the remaining twenty percent for the purpose of testing the model.

To model the relationship between the predictor features and the arrival delay, we will use a multilayer feedforward neural network. To train the simplest multilayer feedforward network, we will use the neuralnet() function with a single hidden node. See the Appendix for detail. We can use the plot() function to visualize this network topology. See the Appendix for detail.

The only question we can ask is how much weight should be given to each of the three predictor features. We will start out by randomly assigning three weights for the three predictor features; then we will examine the results of these weights to see how good they are.

We will do this by processing the 4,000 flights that are in the training dataset, and the remaining 1000 observations will be left for the testing phase. Note that each of the 4000 observations has three predictor features. For each observation, we weight each of the three predicted variables by the randomly chosen weights. Then we add them up for the purpose of evaluating the activation function, which here is chosen to be the sigmoid function where x is the weighted sum. This will be our temporary predicted arrival delay for each observation. We determine a predicted arrival delay for each of the 4,000 observations by repeating the above procedure. Since we know the actual arrival delay for each of the 4,000 observations, we can calculate the residuals for the each of the 4,000 observations. To obtain the sum of squared errors, SSE, we need to square each of the residuals and add them up. Fortunately, the machine learning algorithm neuralnet() function does all of the tedious calculations for us, so none of these arduous calculations need to be done by hand.

We wish to have small values of SSE, because the smaller the total error the better the prediction model. To reduce the error, we need to choose better weights. The process of choosing better weights is called backpropagation. This weight modification process involves a technique called gradient descent, and is done by the machine learning algorithm for us. We need not dwell on the details. After each modification we need to perform the whole process again, presumably finding that the total error has been reduced. Each of these identical repetitions is called a step. When no further improvements is possible the process stops.

The result of applying the above procedure requires 46,064 steps and reduces the sum of the squared errors to 0.002137. The final weights are 1.31804 for the departure delay, -0.00046 for the distance, and 0.83527 for the flight time delay. See the Appendix.

We can see from plot one in the Appendix that none of the predictor variables' weights are zero. This indicates that all three independent features have some effect on the arrival time delay. Moreover, the weights of the features departure delay and flight time delay are positive values, which indicates that they have a positive impact on the arrival delay; that is, as these features increase so does the predicted arrival time delay. However, the weight of the distance feature is negative, implying that longer the distance the less is the arrival delay. But note that the weight is a very small number, so the effect is minimal.

Up to this point we have established our model exclusively based on the training dataset. Now we will evaluate our model using the test dataset, which has 1,000 observations. We do this evaluation with the testing dataset since the training dataset, having been used to create the model, would give us an exaggerated measure of the effectiveness of the model; we must choose an independent set of data.

We will compute the strength of the linear association between the predicted arrival delay and the true value using cor() function. To do this we need to create the compute() function to predicted arrival delay. See the details in the Appendix.

The correlation of about 0.7192 indicates a fairly strong linear association between the predicted and the true arrival delay. This is an indication that the model is doing fairly well. See details in Appendix.

The calculation of a weighted sum at which to evaluate the activation function is conceptually thought of as a "hidden node". It is possible to enrich the model by adding more hidden nodes, which is supposed to improve model performance by increasing the correlation coefficient. We tried using three hidden nodes, this did not increase the value of the correlation coefficient significantly, but it did reduce the SSE terms slightly, from 0.002513 to 0.002424. Thus, we will stick with the improved model.

The value of the correlation coefficient, 0.7233, indicates the three independent factors have a quite strong influence on the response feature arrival time delay of commercial airline flights in the USA. This means that all of these features, departure delay, flight distance, and flight time delay, matter for the arrival delay. Therefore, we can conclude that our model for analysis is fairly good.

Reference:

<http://vk.com/doc168073_317718618?hash=90cd2a2976f079b1e7&dl=43db8b80daa2831cc1>

A1989 <- read.table("C:/Users/Owner/Desktop/proj1/data/1989.csv",  
 sep = ",", header = T)  
str(A1989)

## 'data.frame': 5041200 obs. of 29 variables:  
## $ Year : int 1989 1989 1989 1989 1989 1989 1989 1989 1989 1989 ...  
## $ Month : int 1 1 1 1 1 1 1 1 1 1 ...  
## $ DayofMonth : int 23 24 25 26 27 28 29 30 31 2 ...  
## $ DayOfWeek : int 1 2 3 4 5 6 7 1 2 1 ...  
## $ DepTime : int 1419 1255 1230 1230 1232 1228 1639 1231 1405 1057 ...  
## $ CRSDepTime : int 1230 1230 1230 1230 1230 1230 1230 1230 1230 1045 ...  
## $ ArrTime : int 1742 1612 1533 1523 1513 1550 1942 1531 1827 1537 ...  
## $ CRSArrTime : int 1552 1552 1552 1552 1552 1552 1552 1552 1552 1554 ...  
## $ UniqueCarrier : Factor w/ 13 levels "AA","AS","CO",..: 11 11 11 11 11 11 11 11 11 11 ...  
## $ FlightNum : int 183 183 183 183 183 183 183 183 183 184 ...  
## $ TailNum : logi NA NA NA NA NA NA ...  
## $ ActualElapsedTime: int 323 317 303 293 281 322 303 300 382 160 ...  
## $ CRSElapsedTime : int 322 322 322 322 322 322 322 322 322 189 ...  
## $ AirTime : logi NA NA NA NA NA NA ...  
## $ ArrDelay : int 110 20 -19 -29 -39 -2 230 -21 155 -17 ...  
## $ DepDelay : int 109 25 0 0 2 -2 249 1 95 12 ...  
## $ Origin : Factor w/ 237 levels "ABE","ABI","ABQ",..: 205 205 205 205 205 205 205 205 205 57 ...  
## $ Dest : Factor w/ 237 levels "ABE","ABQ","ACY",..: 98 98 98 98 98 98 98 98 98 104 ...  
## $ Distance : int 2398 2398 2398 2398 2398 2398 2398 2398 2398 1452 ...  
## $ TaxiIn : logi NA NA NA NA NA NA ...  
## $ TaxiOut : logi NA NA NA NA NA NA ...  
## $ Cancelled : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ CancellationCode : logi NA NA NA NA NA NA ...  
## $ Diverted : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ CarrierDelay : logi NA NA NA NA NA NA ...  
## $ WeatherDelay : logi NA NA NA NA NA NA ...  
## $ NASDelay : logi NA NA NA NA NA NA ...  
## $ SecurityDelay : logi NA NA NA NA NA NA ...  
## $ LateAircraftDelay: logi NA NA NA NA NA NA ...

r1 <- A1989[, -c(11,14, 20,21,23, 25:29), drop = NA]  
r2 <- r1[, -c(1, 9, 15, 16, 18, 19)]  
  
r3 = subset(r2, c(DepTime != "NA"))  
r4 = subset(r3, c(ActualElapsedTime != "NA"))  
r5 = subset(r4, c(Distance != "NA"))

r5$FlightTimeDelay <- r5$ActualElapsedTime - r5$CRSElapsedTime  
  
set.seed(6620)  
r6 <- r5[sample(1:nrow(r5), 5000, replace = FALSE),]  
head(r6)

## Month DayofMonth DayOfWeek DepTime CRSDepTime ArrTime CRSArrTime  
## 1986475 5 15 1 1030 1030 1357 1409  
## 4210355 11 22 3 1500 1430 1544 1509  
## 299255 1 8 7 1957 1957 2122 2133  
## 3938931 10 10 2 1850 1800 2159 2106  
## 1890750 5 7 7 635 635 906 906  
## 279823 1 27 5 1945 1945 2054 2105  
## FlightNum ActualElapsedTime CRSElapsedTime ArrDelay DepDelay  
## 1986475 632 147 159 -12 0  
## 4210355 368 44 39 35 30  
## 299255 684 85 96 -11 0  
## 3938931 390 129 126 53 50  
## 1890750 225 151 151 0 0  
## 279823 1938 69 80 -11 0  
## Distance FlightTimeDelay  
## 1986475 963 -12  
## 4210355 140 5  
## 299255 528 -11  
## 3938931 819 3  
## 1890750 946 0  
## 279823 390 -11

normalize <- function(x)   
{  
 return((x - min(x)) / (max(x) - min(x)))  
}

ArrDelay\_norm <- as.data.frame(lapply(r6, normalize))  
head(ArrDelay\_norm)

## Month DayofMonth DayOfWeek DepTime CRSDepTime ArrTime CRSArrTime  
## 1 0.3636364 0.4666667 0.0000000 0.4363868 0.4363868 0.5652355 0.5966030  
## 2 0.9090909 0.7000000 0.3333333 0.6357082 0.6060221 0.6431847 0.6390658  
## 3 0.0000000 0.2333333 1.0000000 0.8295165 0.8295165 0.8841184 0.9040340  
## 4 0.8181818 0.3000000 0.1666667 0.7841391 0.7629347 0.8995415 0.8925690  
## 5 0.3636364 0.2000000 1.0000000 0.2688719 0.2688719 0.3772405 0.3830149  
## 6 0.0000000 0.8666667 0.6666667 0.8244275 0.8244275 0.8557732 0.8921444  
## FlightNum ActualElapsedTime CRSElapsedTime ArrDelay DepDelay  
## 1 0.13950918 0.24253731 0.26666667 0.08794788 0.09621993  
## 2 0.08114084 0.05037313 0.04444444 0.24104235 0.19931271  
## 3 0.15100597 0.12686567 0.15000000 0.09120521 0.09621993  
## 4 0.08600486 0.20895522 0.20555556 0.29967427 0.26804124  
## 5 0.04952465 0.25000000 0.25185185 0.12703583 0.09621993  
## 6 0.42825558 0.09701493 0.12037037 0.09120521 0.09621993  
## Distance FlightTimeDelay  
## 1 0.21022093 0.1729730  
## 2 0.02655657 0.2648649  
## 3 0.11314439 0.1783784  
## 4 0.17808525 0.2540541  
## 5 0.20642714 0.2378378  
## 6 0.08234769 0.1783784

summary(ArrDelay\_norm$ArrDelay)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.0000 0.1140 0.1368 0.1538 0.1694 1.0000

summary(r6$ArrDelay)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## -39.000 -4.000 3.000 8.208 13.000 268.000

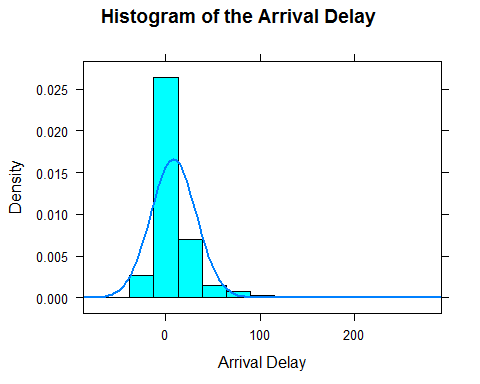
library(car)  
library(stats)  
library(dplyr)

##   
## Attaching package: 'dplyr'  
##   
## The following object is masked from 'package:stats':  
##   
## filter  
##   
## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

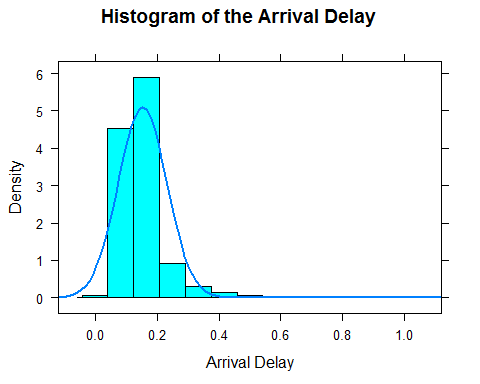
library(lattice)  
library(ggplot2)  
library(mosaic)

##   
## Attaching package: 'mosaic'  
##   
## The following objects are masked from 'package:dplyr':  
##   
## count, do, tally  
##   
## The following object is masked from 'package:car':  
##   
## logit  
##   
## The following objects are masked from 'package:stats':  
##   
## binom.test, cor, cov, D, fivenum, IQR, median, prop.test,  
## quantile, sd, t.test, var  
##   
## The following objects are masked from 'package:base':  
##   
## max, mean, min, prod, range, sample, sum

histogram(~ArrDelay, fit="normal", data=filter(r6),  
xlab = "Arrival Delay", main = "Histogram of the Arrival Delay")



histogram(~ArrDelay, fit="normal", data=filter(ArrDelay\_norm),  
xlab = "Arrival Delay", main = "Histogram of the Arrival Delay")



ArrDelay\_train <- ArrDelay\_norm[1:4000, ]  
ArrDelay\_test <- ArrDelay\_norm[4001:5000, ]

ArrDelay\_train1 <- ArrDelay\_train[, c(11, 12, 13, 14)]  
ArrDelay\_test1 <- ArrDelay\_norm[, c(11, 12, 13, 14)]

library(grid)  
library(MASS)

##   
## Attaching package: 'MASS'  
##   
## The following object is masked from 'package:dplyr':  
##   
## select

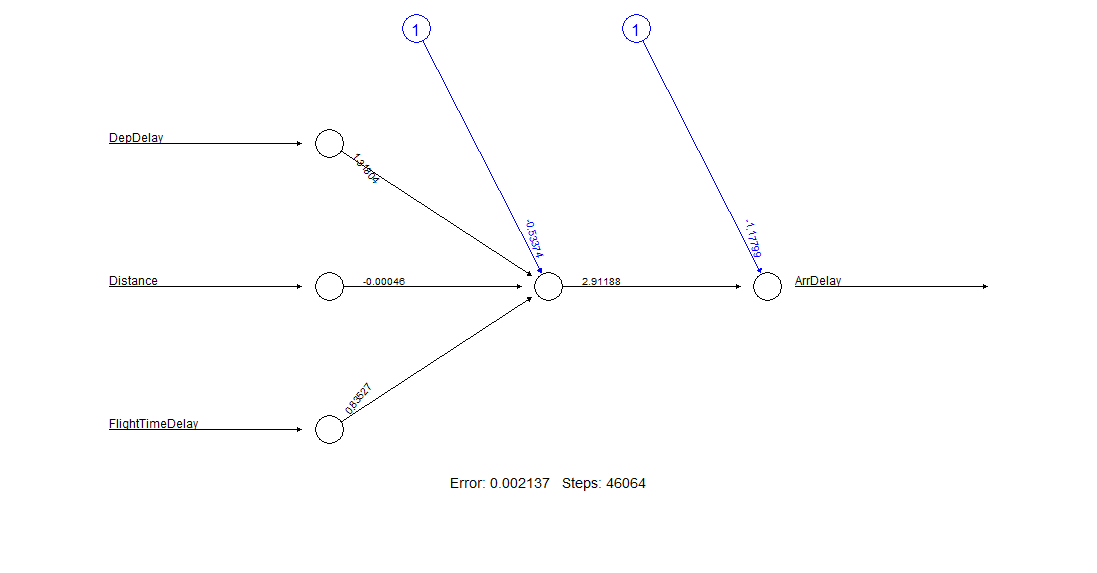
library(neuralnet)

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

ArrDelay\_model <- neuralnet(ArrDelay ~ DepDelay   
 + Distance + FlightTimeDelay,  
 data = ArrDelay\_train1)  
ArrDelay\_model

## Call: neuralnet(formula = ArrDelay ~ DepDelay + Distance + FlightTimeDelay, data = ArrDelay\_train1)  
##   
## 1 repetition was calculated.  
##   
## Error Reached Threshold Steps  
## 1 0.002512604965 0.009517613838 7477

plot(ArrDelay\_model)



model\_results <- compute(ArrDelay\_model, ArrDelay\_test1[1:3])  
predicted\_ArrDelay <- model\_results$net.result

cor(predicted\_ArrDelay, ArrDelay\_test1$ArrDelay)

## [,1]  
## [1,] 0.7192305467

ArrDelay\_model2 <- neuralnet(ArrDelay ~ DepDelay   
 + Distance + FlightTimeDelay,  
 data = ArrDelay\_train1, hidden = 3)  
ArrDelay\_model2

## Call: neuralnet(formula = ArrDelay ~ DepDelay + Distance + FlightTimeDelay, data = ArrDelay\_train1, hidden = 3)  
##   
## 1 repetition was calculated.  
##   
## Error Reached Threshold Steps  
## 1 0.002423885434 0.00963380019 11955

model\_results2 <- compute(ArrDelay\_model2, ArrDelay\_test1[1:3])  
predicted\_ArrDelay2 <- model\_results2$net.result

cor(predicted\_ArrDelay2, ArrDelay\_test1$ArrDelay)

## [,1]  
## [1,] 0.7232540987