Analysis of Logistic Regression vs Neural Network

By: Pranav Vinod

Supervisor: Professor Donghui Yan

MTH 522: Advanced Mathematical Statistics

University of Massachusetts Dartmouth

05/5/2022

Contents

Abstract

In this homework, we have analyzed the performance of the logistic regression and neural network models on 2 different datasets. The metric chosen for comparison is the accuracy of the model. It was found the depending on the dataset, both the models had a chance of performing better than the other one.

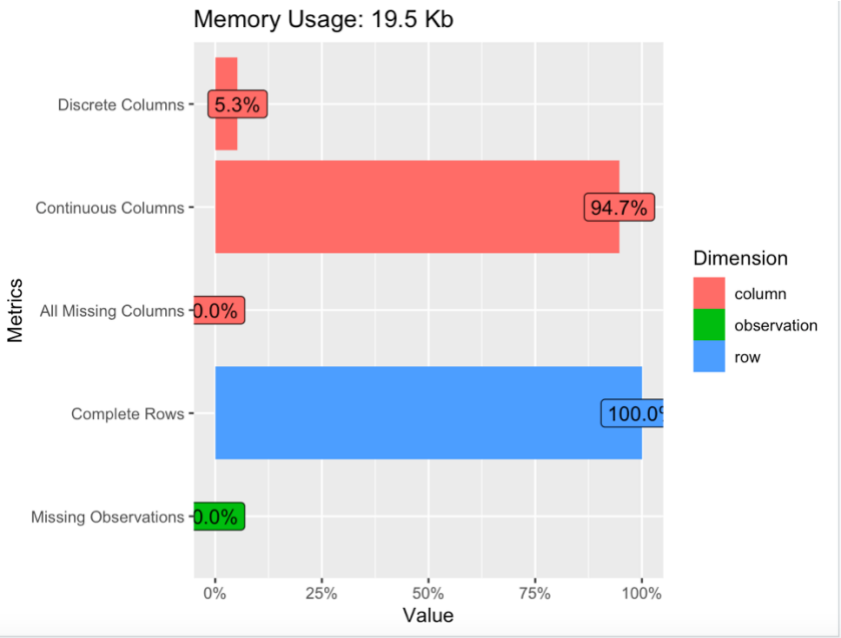
Data Description

For the first dataset, we have chosen a very small dataset about lung cancer that has only 30 observations of 57 variables. The target variable in our case is the variable X1 which is a multinomial categorical variable with 3 classes. The source of the dataset is as follows:

https://archive.ics.uci.edu/ml/datasets/Lung+Cancer

There isn't much information available about the dataset, so we must try to gather some on our own.

Other than the target variable, all other variables are numerical with range between 0 and 3.



There were found to be no missing values in the dataset.

Since the dataset itself is small, there are not many values in each class. There are the most observations in class 2

```
> table(d$X1)

 1  2  3
 8 13  9
```

Now, the dataset is split into training and testing set, in the ratio of 80:20.

```
##To split the data into training and test set
set.seed(420)
ind <- sample(2, nrow(d), replace = T, prob = c(0.8, 0.2))
train <- d[ind == 1,]
test <- d[ind == 2,]
```

Logistic Regression

Now, Logistic Regression is implemented on the dataset.

```
# Logistic Regression

mod <- glmnet(train[,-c(1)],train$X1,family = 'multinomial')

##Apply the trained model to the test set
newX <- model.matrix(~.-X1,data=test)
newX <- newX[,-c(1)]
mypred4<-predict(mod,newx=newX,type="response",s=0.01);
posteriprob<-mypred4[,,1];
yhat<-matrix(1,nrow(test),1);
for(i in 1:nrow(test))
{
  yhat[i]<-which.max(posteriprob[i,]);
}
acc<-sum(yhat==test[,c(1)])/nrow(test);
cat("Accuracy on the test set is", acc, "\n");
```

Based on the applied model, we found that the accuracy of this model was 0.5. Such a low score is acceptable because of the small dataset.

```
> cat("Accuracy on the test set is", acc, "\n");
Accuracy on the test set is 0.5
```

Now we will look to compare the performance of a neural network on the same dataset.

Neural Network

Now a neural network is applied on the same dataset.

```
train$X. <- as.numeric(train$X.)
train<- na.omit(train)
labels <- class.ind(train[,1])
l <- labels[ind==1]
myiris<-nnet(train[,-c(1)], labels,
             size=2, rang=0.1,
             decay=5e-4, maxit=200)
summary(myiris)
```

The architecture of the resulting neural network were that it had 1 hidden layer with 2 nodes. It had 56 input nodes and 3 output nodes for each of the 3 classes.

```
a 56-2-3 network with 123 weights
options were - decay=5e-04
```

Now, to compute the performance of this neural network, we calculated the accuracy of this network

```
test.cl <- function(true, pred) {
  true <- max.col(true)
  cres <- max.col(pred)
  table(true, cres)
}
labels2 <- class.ind(test[,1])
conf<-test.cl(labels2, predict(myiris, test))
acc<-sum(diag(conf))/sum(conf)
cat("The accuracy on the test set is", acc,"\n")
```

The accuracy of the network was found to be 0.333

```
> cat("The accuracy on the test set is", acc,"\n")
The accuracy on the test set is 0.3333333
```

Conclusion

Comparing the performance based on accuracy of both these models, we have found that the logistic regression model outperforms neural network on this dataset. This can be attributed to the fact that the dataset has many more attributes as compared to observations. Neural Networks generally perform better on a large dataset and have poor performance on small datasets as exhibited by this one.

Dataset 2

Data Description

The second dataset used in this homework is Banknote Authentication dataset. The following link is the source for this dataset:

https://www.kaggle.com/code/marwa01/exploratory-data-analysis-eda-model-comparison/data

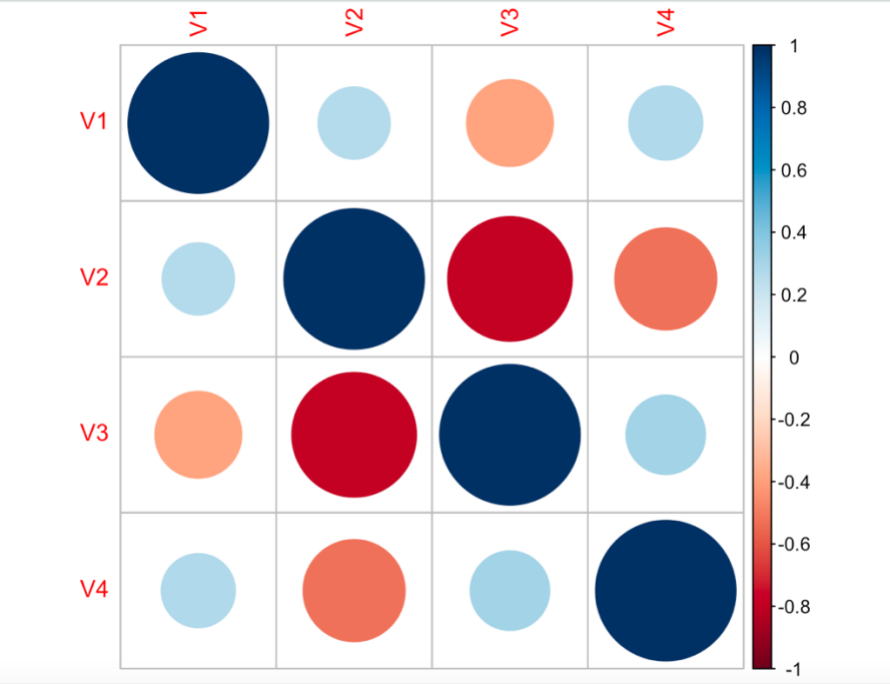
This dataset was extracted from UCI website and was taken from images of authentic and fake banknotes. The aim of this project is to classify bank notes as either authentic or forged based on the 4 given parameters.

```
> summary(d2)
       V1                V2                V3                V4          V5
 Min.   :-7.0421   Min.   :-13.773   Min.   :-5.2861   Min.   :-8.5482   0:762
 1st Qu.:-1.7730   1st Qu.: -1.708   1st Qu.:-1.5750   1st Qu.:-2.4135   1:610
 Median : 0.4962   Median :  2.320   Median : 0.6166   Median :-0.5867
 Mean   : 0.4337   Mean   :  1.922   Mean   : 1.3976   Mean   :-1.1917
 3rd Qu.: 2.8215   3rd Qu.:  6.815   3rd Qu.: 3.1793   3rd Qu.: 0.3948
 Max.   : 6.8248   Max.   : 12.952   Max.   :17.9274   Max.   : 2.4495
```

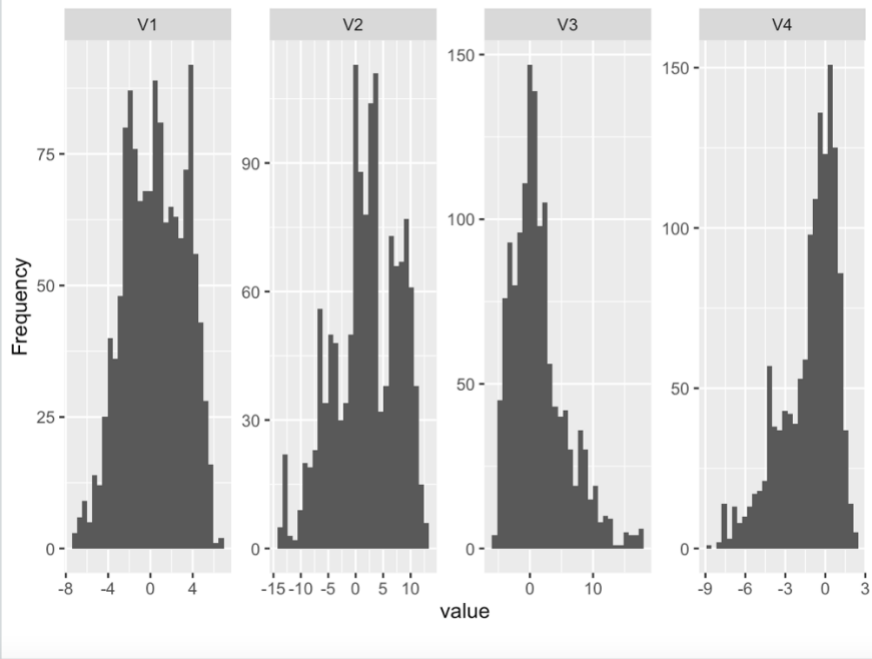
The dataset consists of 5 attributes, out of which V5 is the target categorical attribute. It is almost evenly divided between forged and authentic banknotes.

V1,V2, V3 and V4 are the variance, skewness, kurtosis and entropy of the image from which these banknotes were extracted.

On plotting the correlation plot of the numerical variables, we found that none of them were strongly correlated to each other, so we could go ahead with further analysis.



On plotting the 4 variables as histograms, we found that there was a slight skewness in those variables.



Having done some data analysis, we split the dataset into train and test for further analysis.

```r
#To split the data into training and test set
set.seed(420)
ind <- sample(2, nrow(d2), replace = T, prob = c(0.8, 0.2))
train <- d2[ind == 1,]
test <- d2[ind == 2,]
```

Logistic Regression

Now logistic regression was implemented on the dataset.

```r
# Logistic Regression
mod <- glm(V5~.,data = train,family = 'binomial')

##Apply the trained model to the test set
p <- predict(mod, test, type = 'response')
p <- as.factor(ifelse(p >0.5,1,0))
confusionMatrix(p, test$V5)
```

Based on the model, we created the confusion matrix using the test set and tried to find the accuracy of the logistic regression model on this dataset.

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 154   0
         1   2 117

               Accuracy : 0.9927
                 95% CI : (0.9738, 0.9991)
    No Information Rate : 0.5714
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9851

 Mcnemar's Test P-Value : 0.4795

            Sensitivity : 0.9872
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.9832
             Prevalence : 0.5714
         Detection Rate : 0.5641
   Detection Prevalence : 0.5641
      Balanced Accuracy : 0.9936

       'Positive' Class : 0
```

This model has a very high accuracy of 0.992 on this dataset. This is almost perfect.

But we will implement the neural network and see if there is any difference in its performance.

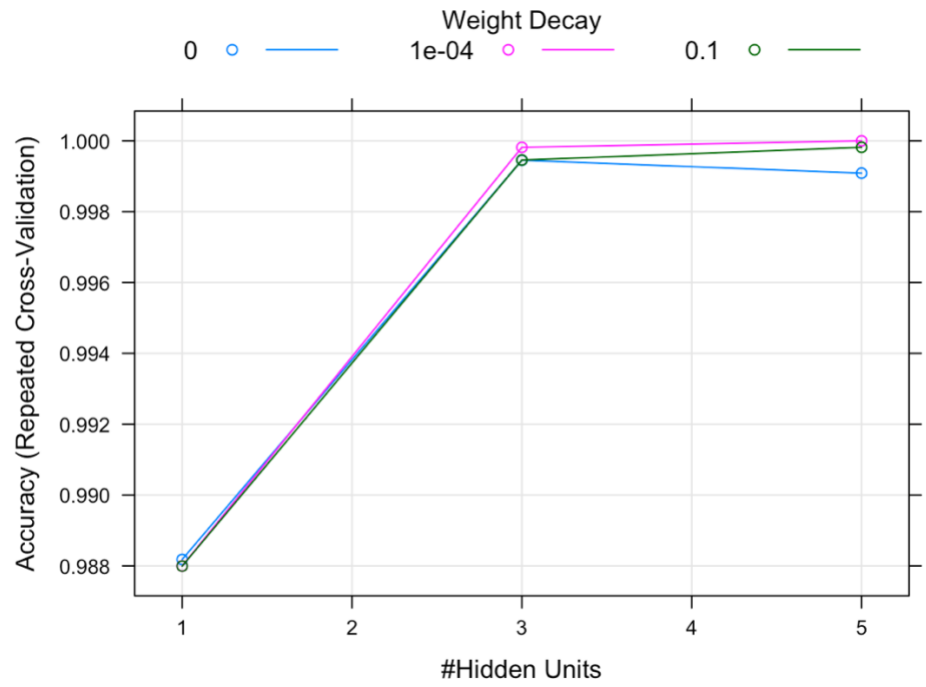
Neural Network

Now a neural network is implemented on this dataset

```
for (x in 1:100) {
  Model.nn <- train(V5 ~., data=train,
                    method="nnet",
                    trControl=cvcontrol,
                    preProcess=c("center","scale"),
                    tunelength = 5,
                    maxit = 100,
                    metric="Accuracy")

  # plotnet(Model.nn$finalModel)

  testpred.nn <- predict(Model.nn, test)
   k <- confusionMatrix(test$V5, testpred.nn, mode='everything')   #1

  a <- as.numeric(k[["overall"]][1])
  b <- b+a
  print(x)

}
```

On averaging the result over 100 runs of the model, we found that the accuracy of the neural network was 0.9998~1. That means it perfectly predicted the class of each banknote based on its attributes.

Further, to anaylse the performance of the network over different values of hidden layers an weights , I plotted the following two plots

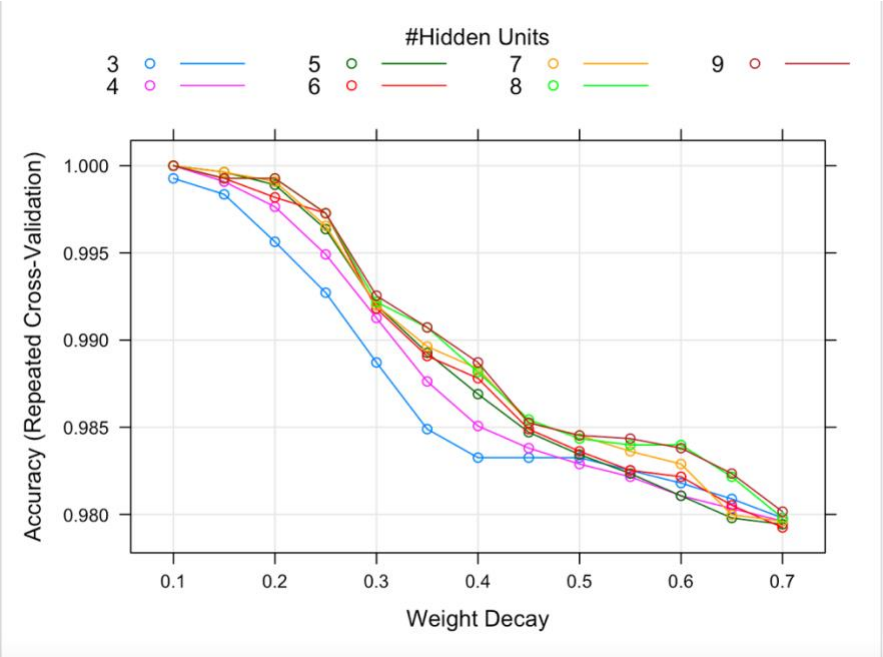


This is a plot of the model and its variation in accuracy over 5 different hidden units for 3 different weight decays. We find that there is not much variation in the performance of the model after 3 hidden units and that it reached a optimum level by then.

To look into it further, we created a grid of values to create a network of and check its performance

```
g <- expand.grid(size = seq(from=3, to=9, by=1) ,
                 decay = seq(from=0.1, to=0.7, by=0.05))

Model.nn <- train(V5 ~., data=train,
                  method="nnet",
                  trControl=cvcontrol,
                  preProcess=c("center","scale"),
                  tuneGrid = g,
                  maxit = 100,
                  metric="Accuracy")

plot(Model.nn)
```

The result of this model was as follows:



In this model we can see that as the weight decay value increases, the performance of the model decreases irrespective of the number of nodes. For small values of the weight decay, all the models have a high accuracy. But as the value of weight decay increases the accuracy of the model decreases.

Conclusion

For this dataset, that has many observations as compared to attributes, both logistic regression and neural network performed well. But even then, the neural network was able to perfectly predict the class for each banknote. This goes to show that the size of the dataset mattes when it comes to the performance of neural network. It performs better for a large dataset as compared to a small dataset.

References

1. Dataset 1
2. Dataset 2

Appendix

```
library(mlbench)
library(caret)
library(e1071)
library(lime)
library(DAAG)
library(party)
library(rpart)
library(rpart.plot)
library(pROC)
library(nnet)
library(dplyr)
library(corrplot)
library(DataExplorer)
library(glmnet)

d <- read.csv(file.choose(), header = T)
d <- na.omit(d)
d <- d[-25,]
summary(d)
str(d)
d$X1 <- as.factor(d$X1)
table(d$X1)

plot_intro(d)
##To split the data into training and test set
set.seed(420)
ind <- sample(2, nrow(d), replace = T, prob = c(0.8, 0.2))
train <- d[ind == 1,]
test <- d[ind == 2,]


# Logistic Regression

mod <- glmnet(train[,-c(1)],train$X1,family = 'multinomial')

##Apply the trained model to the test set
newX <- model.matrix(~.-X1,data=test)
newX <- newX[,-c(1)]
mypred4<-predict(mod,newx=newX,type="response",s=0.01);
posteriprob<-mypred4[,,1];
yhat<-matrix(1,nrow(test),1);
for(i in 1:nrow(test))
{
  yhat[i]<-which.max(posteriprob[i,]);
}
acc<-sum(yhat==test[,c(1)])/nrow(test);
```

```r
cat("Accuracy on the test set is", acc, "\n");

################################################################################
##############
# Neural Network
train$X. <- as.numeric(train$X.)
train<- na.omit(train)
labels <- class.ind(train[,1])
l <- labels[ind==1]
myiris<-nnet(train[,-c(1)], labels,
             size=2, rang=0.1,
             decay=5e-4, maxit=200)
summary(myiris)

test.cl <- function(true, pred) {
  true <- max.col(true)
  cres <- max.col(pred)
  table(true, cres)
}
labels2 <- class.ind(test[,1])
conf<-test.cl(labels2, predict(myiris, test))
acc<-sum(diag(conf))/sum(conf)
cat("The accuracy on the test set is", acc,"\n")


################################################################################
##############

# Banknotes dataset
d2 <- read.csv(file.choose(), header = F)

summary(d2)
d2$V5 <- as.factor(d2$V5)
str(d2)

# Corr Plot
corr <- cor(d2[,-c(5)])
corrplot(corr)

# Histogram
plot_histogram(d2[,-c(5)])

# Distribution of target variable
table(d2$V5)   # ALmost evenly distributed between the 2 classes

#To split the data into training and test set
set.seed(420)
ind <- sample(2, nrow(d2), replace = T, prob = c(0.8, 0.2))
train <- d2[ind == 1,]
```

```r
test <- d2[ind == 2,]

##############################################################################
##################s
# Logistic Regression
mod <- glm(V5~.,data = train,family = 'binomial')

##Apply the trained model to the test set
p <- predict(mod, test, type = 'response')
p <- as.factor(ifelse(p >0.5,1,0))
confusionMatrix(p, test$V5)

##############################################################################
#############

# Neural Network
set.seed(123)
cvcontrol <- trainControl(method="repeatedcv",
                          number = 10,
                          repeats = 5,
                          allowParallel=TRUE,
                          savePredictions = T)

b <- 0


for (x in 1:100) {
  Model.nn <- train(V5 ~., data=train,
                    method="nnet",
                    trControl=cvcontrol,
                    preProcess=c("center","scale"),
                    tunelength = 5,
                    maxit = 100,
                    metric="Accuracy")

  # plotnet(Model.nn$finalModel)

  testpred.nn <- predict(Model.nn, test)
  k <- confusionMatrix(test$V5, testpred.nn, mode='everything')   #1

  a <- as.numeric(k[["overall"]][1])
  b <- b+a
  print(x)

}

accuracy <- (b)/100

g <- expand.grid(size = seq(from=3, to=9, by=1) ,
```

```
                decay = seq(from=0.1, to=0.7, by=0.05))

Model.nn <- train(V5 ~., data=train,
                  method="nnet",
                  trControl=cvcontrol,
                  preProcess=c("center","scale"),
                  tuneGrid = g,
                  maxit = 100,
                  metric="Accuracy")

plot(Model.nn)

accuracy <- (b)/100
```