Lab8: Mortality Prediction Challenge, Solutions

Intro to Data Mathematics 2019

Manal Hajjam

Overview

Every day, doctors and nurses collect a lot of information about patients by asking questions and using adapted tools (stethoscope, syringe, sensors, etc.). This data is very useful to monitor health state, diagnose and choose treatments. It can also be used for statistical predictive analysis of how the patient will progress. Analysts at the GetUHealthy hospital are currently using the mean method to predict mortality. The first part of the lab introduces you to the mean method using the WBCD data. The second part involves you in an online challenge to improve a system for predicting mortality of patients in the Intesive Care Unit (ICU).

The Mean Method

In class you learned the following method to create the the normal w and threshold t of the classifier using the Mean Method.

1. Suppose `D` is the data matrix.
2. Let `Cplus` be rows of `D` having positive class and Cneg be the rows of D having negative class.
3. Set the `Mplus` to be the mean of `Cplus` and `Mneg` to be the mean of `Cneg`.
4. The the normal vector to the Mean Method's classifying hyperplane is
$$w = \frac{Mplus - Mneg}{||Mplus - Mneg||}$$
5. We need one more thing to fully specify the separating hyperplane: a point that lies on the hyperplane. An intuitive choice for this point is halfway between the mean of the positive and negative classes:
$$P = (Mplus + Mneg)/2$$
6. So the threshold for the equation of the classifying hyperplane is then
$$t = P \cdot w$$
7. The equation for the classifying hyperplane for the mean method is then
$$x \cdot w = t$$
8. In this lab we classify a new point, u, as Class 1 if $u \cdot w - t > 0$, and then Class Class -1 otherwise. Note this assumes that any scaling or centering has been accounted for when creating u.

*** Warning: Never use the Mean Method after this class. We just made it up for educational purposes. LDA and other methods should give you better results almost always. ***

Here, we perform the mean method for you using the data set from the prelab.

```r
# Prepare WBCD data
Q.df <- read.csv("~/MATP-4400/data/wdbc.csv")
Q<-as.matrix(Q.df[ ,3:32])
colnames(Q)<-sprintf('V%d',1:ncol(Q))
labels<-Q.df[,2]
# Make a vector y of class with 1 if M or -1 if B
y<-matrix(1,nrow=length(labels))
y[labels=='B']<- -1
# Make Cplus the matrix containing the positive data
Cplus=Q[y==1,]
# Make Cplus the matrix containing the positive data
Cneg=Q[y==-1,]

Mplus<-colMeans(Cplus)
Mneg<-colMeans(Cneg)
# Compute the normal to the hyperplane w
w<-(Mplus-Mneg)
w<-w/sqrt(sum(w^2)) # w/norm(w)
w<-matrix(w,nrow=length(w))
# Calculate threshold t
t<- ((Mplus+Mneg)%*%w)/2
t<-as.numeric(t) # make sure t is a scalar...not a matrix
```

Now we predict the training data.

```r
# Predict the training data in Q  by checking which side of the hyperplane the points are on using the
ypred <- sign(Q%*%w-t)
# Added this extra step to handle the case if the point is exactly on the hyperplane.  This happens ver
ypred[ypred==0]<-1
```

Now we calculate the accuracy.

```r
# Table command counts the actual versus the predicted labels for the training data. The result is stor
confusion.matrix<-table(y,ypred)
```

The predictions on the training data results in the following confusion matrix. The rows are the actual classes and the columns are the predicted classes.

```r
# kable formats a table to look nice in notebook
kable(confusion.matrix, type="html",digits = 2)
```

	-1	1
-1	353	4
1	58	153

We can see that 62 points are misclassified on the training data. For the training data, the Class 1 accuracy is 153/(58+153) or 72.5%, the Class -1 accuracy is 353/(353+4) or 98.9% and the balanced accuracy is 85.7%.

Mortality Prediction Challenge

The main question of this challenge is: “How to predict the survival of a patient in the ICU given his or her medical record?”" More specifically, you are asked to predict whether or not a patient will die during their stay at the hospital. Healthcare organizations use risk models one to help determine high risk patients that may need specialized care. Models like this one are being used at Mount Sinai hospitals in New York City to predict if COVID-19 patients are likely to need ICU care and ventilation. They are also used to help determine which COVID-19 patients may safely recuperate at home. They use techniques like this lab except except they are using more sophisticated deep learning models called neural networks.

Data

Data Description

The training dataset contains information 79,999 patients, represented by categorical, binary and numerical features (also named “variables”). Those features are for instance age, gender, ethnicity, marital status, as well as medical data such as blood pressure rate or glucose rate. There are a total of 342 variables.

The class (or label) to be predicted is a binary variable telling if the patient died or not during while in the hospital. Fortunately, for the patients, most of them don’t die. Unfortunately for us, that leads to a class imbalance.

Data Preparation

This section reads in the the training data and create the ‘internal’ training and validation set. A validation set acts like a test set. It has known class labels, and we use it to determine how good our classifier are. We call it a validation set to distinguish it from the ‘external’ test set. The big difference is that you don’t know the labels of the ‘external’ test set. You have to predict the labels of the ‘external’ test set to enter the challenge.

The first step is to prepare the data. For the basic entry provided here, we only use numeric features that contain age and various biochemical indicators. We do not scale. For your entry, you can decide to use more features or scale. Just remember that you may need to convert nonnumeric features to numbers first. Read the comments and the contest documentation to understand the files that are provided.

```r
# Read in mortality data
mortdata <- read.csv("~/MATP-4400/data/mimic_synthetic_train.data", sep=' ', header=F)[ ,-1] # exclude

# read in the labels of the training data
labels <- factor(read.csv("~/MATP-4400/data/mimic_synthetic_train.solution", header=F)[ ,1], levels=c(0

#read in the feature types
feat.types <- read.csv("~/MATP-4400/data/mimic_synthetic_feat.type", stringsAsFactors = F, header=F)[,1
# read in the feature names
cnames <- read.csv("~/MATP-4400/data/mimic_synthetic_feat.name", stringsAsFactors = F, header=F)[,1]

#Figure out which features are numerical
bool.feats <- feat.types=='Numerical'

mortdata <- mortdata[ ,bool.feats][ ,3:51] # only keep age and biochemical indicators
colnames(mortdata) <- cnames[bool.feats][3:51] # namet he columns
```

The data was randomly divided into two sets consisting of 90% training and 10% validation. We use the validation set to estimate how well the classifier may work on future data.

```r
# Split the data into training and testing sets
# train.size will be the number of data in the training set
n <- nrow(mortdata)
train.size <- ceiling(n*0.9)

#Set random seed to ensure reproducibility
set.seed(300)
train <- sample(n,train.size)

morttrain <- mortdata[train, ]  #The training data is just the training rows
mortval <- mortdata[-train, ]  # Using -train gives us all rows except the training rows.

trainclass <- labels[train]
testclass <- labels[-train]

#We leave off the last column (the class label) to get the matrices of features
trainmatrix <- as.matrix(morttrain)
valtmatrix <- as.matrix(mortval)
```

Exercise 1

Predict ICU Mortality using the Mean Method.

Construct a classifier on the training set using the mean method. This has been done for you.

```r
Q <- trainmatrix
labels <- trainclass
# Make a vector y of class with 1 if M or -1 if B
y <- matrix(1,nrow=length(labels))
y[labels=='lived',1] <- -1
# Make Cplus the matrix containing the positive data
Cplus <- Q[y==1,]
# Make Cplus the matrix containing the positive data
Cneg=Q[y==-1,]

Mplus<-colMeans(Cplus)
Mneg<-colMeans(Cneg)
w<-(Mplus-Mneg)
w<-w/sqrt(sum(w^2)) # w/norm(w)
# Calculate threshold t
t <- ((Mplus+Mneg)%*%w)/2
t<-as.numeric(t) # make sure t is a scalar...not a matrix
```

Predict the training set using the mean method. Calculate the Class 1 accuracy, the Class -1 accuracy, and the balanced accuracy on the training set.

```r
# Predict the training data in Q  by checking which side of the hyperplane the points are on using the
ypred <- sign(Q%*%w-t)
# Added this extra step to handle the case if the point is exactly on the hyperplane.  This happens ver
ypred[ypred==0]<-1

confusion.matrix<-table(y,ypred)

kable(confusion.matrix, type="html",digits = 2)
```

	-1	
-1	44568	
1	1153	
Predi	ct the v	alidation set using the mean method. Calculate the Class 1 accuracy, the Class -1 accuracy, and the bala

```r
# Split the data into training and testing sets
# train.size will be the number of data in the training set
n <- nrow(mortdata)
train.size <- ceiling(n*0.9)

#Set random seed to ensure reproducibility
set.seed(300)
train <- sample(n,train.size)

morttrain <- mortdata[train, ]  #The training data is just the training rows
mortval <- mortdata[-train, ]  # Using -train gives us all rows except the training rows.

trainclass <- labels[train]
testclass <- labels[-train]

#We leave off the last column (the class label) to get the matrices of features
trainmatrix <- as.matrix(morttrain)
valtmatrix <- as.matrix(mortval)
```

Exercise 2

Construct a classifier on the training set using the LDA method.

```r
# Prepare WBCD data
Q.df <- read.csv("~/MATP-4400/data/wdbc.csv")
# get the features and scale them on the entire dataset
Q<- as.data.frame(scale(Q.df[ ,3:32],center = TRUE,scale = FALSE))
d <- ncol(Q)
# We give the features names
colnames(Q)<-sprintf('V%d',1:d)
# The labels are M for malignant and B for benign
labels<-Q.df[,2]

#Split the data into training and testing sets
#ss will be the number of data in the training set
n <- nrow(Q)
ss<- ceiling(n*0.9)

#Set random seed to ensure reproducibility
set.seed(300)
train.perm <- sample(1:n,ss)

#The first column is just the sample label, we can discard it for statics (This is the -1 in the second
trainmatrix <- as.matrix(Q[train.perm , ])  #The training data is just the training rows
testmatrix <- as.matrix(Q[-train.perm, ]) # Using -train gives us all rows except the training rows.
#The last column is the class label, which is 1 or -1, we can split it off as the class
trainclass <- labels[train.perm]
testclass <- labels[-train.perm]
```

```r
# Make a train.df and test.df dataframes combining labels and features
train.df<-data.frame(trainmatrix, class=as.factor(trainclass))
test.df<-data.frame(testmatrix, class=as.factor(testclass))

lda.fit <- lda(class~., train.df,prior=c(1,1)/2)
#This returns the results in lda.fit. This contains lda.fit$scaling which is the vector normal to the s
w<-lda.fit$scaling
#Calculate the LDA threshold from the means and the normal vector.
thresh <- ((lda.fit$means[1,] +lda.fit$means[2,])/2)%*%lda.fit$scaling

#Compute the scalar projections of each class on the separating hyperplane.
proj <- trainmatrix%*%as.matrix(lda.fit$scaling)
pplus  <- proj[trainclass=='M'] #All the class 1 projections
pminus <- proj[trainclass=='B'] #All the class -1 projections
```

Predict the training set. Calculate the Class 1 accuracy, the Class -1 accuracy, and the balanced accuracy on the training set.

```r
# use the predict function for the classifier lda.fit to predict the train.df
train.pred <- predict(lda.fit,train.df)$class

# Table command counts the actual versus the predicted labels for the training data. The result is stor
confusion.matrix<-table(trainclass,train.pred)

kable(confusion.matrix, type="html",digits = 2)
```

	B	M
B	324	0
M	12	176

```r
# Calculate the accuracy of each class using the entries of the confusion matrix
accneg <-confusion.matrix[1,1]/(confusion.matrix[1,1]+confusion.matrix[1,2])
accplus <-confusion.matrix[2,2]/(confusion.matrix[2,1]+confusion.matrix[2,2])
# Calculate the balanced accuracy
accbal<-(accplus+accneg)/2
# note how the next text section display results in the text e.g. '`r 100*accplus`%
```

Predict the validation set using the LDA method. Calculate the Class 1 accuracy, the Class -1 accuracy, and the balanced accuracy on the validation set.

```r
# Split the data into training and testing sets
# train.size will be the number of data in the training set
n <- nrow(mortdata)
train.size <- ceiling(n*0.9)

#Set random seed to ensure reproducibility
set.seed(300)
train <- sample(n,train.size)

morttrain <- mortdata[train, ]  #The training data is just the training rows
mortval <- mortdata[-train, ]  # Using -train gives us all rows except the training rows.

trainclass <- labels[train]
```

```r
testclass <- labels[-train]

#We leave off the last column (the class label) to get the matrices of features
trainmatrix <- as.matrix(morttrain)
valtmatrix <- as.matrix(mortval)
```

Discuss how the LDA and Mean Method Classifiers are performing. Discuss the scores on the training and validation sets and compare. Which method is working better.

Exercise 3

Make you a model to predict mortality using R anyway you like. Use your creativity to make a better predictive model. Can you use more data? Can you use different features? Can you use a different modeling algorithm? Woud scaling help (don't forget to scale test and validation the same)?

You can get three points for making a different model. Earn up to 3 points extra credit for being creative and trying different things.

Predict the training set using your method. Calculate the Class 1 accuracy, the Class -1 accuracy, and the balanced accuracy on the training set.

```r
# A function to help plot faces
faceplot <- function(xx,width=64,midcolor="grey10",gcols=2,labels=row.names(xx)) {
# Note require(ggplot2); require(reshape2); require(gridExtra)
 if (is.vector(xx)) {
   xx <- matrix(xx,nrow=1)
}
  pl <- vector("list",nrow(xx))
  for(i in 1:nrow(xx)){
    face <- matrix(xx[i,], nrow=width)
    face.m <- melt(apply(face, 2, function(x) as.numeric(rev(x))))
    pl[[i]] <- ggplot(data=face.m) + geom_raster(aes(x=Var2, y=Var1, fill=value)) +
        theme(axis.text=element_blank(), axis.title=element_blank()) + guides(fill=FALSE) +
        scale_fill_gradient2(low="black", mid=midcolor, high="white") +ggtitle(labels[i])
  }
  grid.arrange(grobs=pl,ncol=gcols)
}







# Function to compute 2-norm
norm2 <- function(x) sqrt(as.vector(x)%*%as.vector(x))

# Function to find closest match
Matchimage <- function(image.matrix, refimage)
  # image.matrix contains the imaages
  # refimage contains the image
  # returns the index of the closest imaige in image.matrix
  {
  dist2myimage<-apply(image.matrix,1,function(x)norm2(x-refimage))
  q <- which.min(dist2myimage)
}
```

```
mortdata <- read.csv("~/MATP-4400/data/mimic_synthetic_train.data", sep=' ', header=F)[ ,-1] # exclude

# read in the labels of the training data
labels <- factor(read.csv("~/MATP-4400/data/mimic_synthetic_train.solution", header=F)[ ,1], levels=c(0

#read in the feature types
feat.types <- read.csv("~/MATP-4400/data/mimic_synthetic_feat.type", stringsAsFactors = F, header=F)[,1
# read in the feature names
cnames <- read.csv("~/MATP-4400/data/mimic_synthetic_feat.name", stringsAsFactors = F, header=F)[,1]

#Figure out which features are numerical
bool.feats <- feat.types=='Numerical'

mortdata <- mortdata[ ,bool.feats][ ,3:51] # only keep age and biochemical indicators
colnames(mortdata) <- cnames[bool.feats][3:51]
```

```
# Read in data
F.df <- mortdata

# Save first column as labels
labels <- as.numeric(F.df[,1])
F.matrix<-as.matrix(F.df[,-1])

# Assign the row names
row.names(F.matrix)<-1:(nrow(F.matrix))

# Scale the images to have unit length
rownorms<-apply(F.matrix, 1, function(x){sqrt(sum(x^2))})
Fbar.matrix<-diag(1/rownorms) %*% F.matrix

# Divide into trainface and  testface
# Since images occur in blocks of 10, work mod 10 (using %%10) and get  images 4-9 as train and 0-3 as
gg <- c(1:400)
h <- gg[ gg%%10 >3]

#h is the list of numbers of faces to extract
trainface.uncentered<- Fbar.matrix[h,]
trainfacelabels<-labels[h]

#Test face
gg <- c(1:400)
h <- gg[ gg%%10 <= 3  ]

#h is the list of numbers of faces to extract
```

```r
testface.uncentered<- Fbar.matrix[h,]
testfacelabels<-labels[h]



# Compute the train mean face and save in train.mean
train.mean <- colMeans(trainface.uncentered)

# Mean center the training data and save in trainface.
trainface.centered <- trainface.uncentered-matrix(1,ncol=1,nrow=nrow(trainface.uncentered)) %*% train.m




nimages<-nrow(testface.uncentered)
correctcount<-0
for (j in 1:nimages) {

  # find the closest image that isn't me
  match<-Matchimage(trainface.centered,testface.uncentered[j,])
  if (trainfacelabels[match]==testfacelabels[j])
  {
  correctcount<-correctcount+1
  }
}
# Report percentage correct
correctcount/nimages*100
```

```
## [1] 0.625
```

```r
# Insert your answer here
test.mean <- colMeans(testface.uncentered)
testface.centered <- testface.uncentered-matrix(1,ncol=1,nrow=nrow(testface.uncentered)) %*% test.mean
nimages<-nrow(testface.centered)
correctcount<-0
for (j in 1:nimages) {

  # find the closest image that isn't me
  match<-Matchimage(trainface.centered,testface.centered[j,])
  if (trainfacelabels[match]==testfacelabels[j])
  {
  correctcount<-correctcount+1
  }
}
# Report percentage correct
correctcount/nimages*100
```

```
## [1] 2.5
```

Predict the validation set using the your method. Calculate the Class 1 accuracy, the Class -1 accuracy, and the balanced accuracy on the validation set.

Exercise 4

- Compare performance of the Mean Method, LDA, and and your best shot model on the training and validation sets. Report how well the models do on the mortality data in terms of class 1 accuracy, class -1 accuracy, and balanced accuracy.

After using and applying all these models I have found that different models have different accuracys and some are fairly accurate while other aren't. Except for the mean method, the rest of the models worked well.

- Describe the predictive model you suggest for predicting ICU mortality on future patients. This could be any of the models that you have made in the previous sections. Explain why you selected this one. The model best for predicting ICU mortality for future patients would be the LDA fit. The LDA fit worked very well for calculing the class 1, class -1 data, and balanced accuracies. In addition, I found that mean method was the worst and least accurate method.

Exercise 5

For this project and the final project, you will being doing an online challenges.

The Lab8 challenge is the "To Be or Not To Be Mortality Challenge" at https://competitions.codalab.org/competitions/22873

The challenge asks you to predict survival of the patients in `~/MATP-4400/data/mimic_synthetic_test.data`

- The winner is the person who gets the highest score on this data.
- The label of this data are 0 for lived and 1 for died.
- Note this is a change from the -1, 1 format given above.
- The predictions must be in a file. The file **must** be called `mimic_synthetic_test.csv`.
- The submission is made by zipping this file, into a file which can be called by any name.

This exercise walks you through how to make an entry using the mean method. You should submit the mean method as your first entry. Everyones submissions will be the same, so you will know if you successfully submitted an entry. Then you can try entering again using your best shot method.

Prepare testing

First, we need to get the external testing data and prepare it just like we did the training data.

```r
# Read in the test data
mortdatatest <- read.csv("~/MATP-4400/data/mimic_synthetic_test.data", sep=' ', header=F)[ ,-1]

# Apply the same transformation to the test that we used on the training data.
mortdatatest <- as.matrix(mortdatatest[,bool.feats][ ,3:51]) # only keep age and numerical chemical dat
colnames(mortdatatest) <- cnames[bool.feats][3:51] # only keep age and numerical chemical data to avoid
```

Predict the testing data

Now we predict the testing data using the Mean method function made on the training data.

```r
# This code is provided for you
# For the test set,
# Make a prediction using the mean method
#ypred<-sign(mortdatatest%*%w-t)
# Convert all the labels < 1 to  0 for the purposes of the contest
ypred[ypred<1]<- 0
```

```r
#verify all predictions are 0 or 1
summary(as.factor(ypred))
```

```
##     0     1 
## 45721 26279
```

Prepare the testing data predictions for submission.

Now we create the entry file and zip it up for submission.

```r
# Here is the mean prediction file for submission to the website
write.table(ypred,file = "mimic_synthetic_test.csv", row.names=F, col.names=F)

#This automatically generates a compressed (zip) file
system("zip -u MMentry.csv.zip mimic_synthetic_test.csv")
```

Download the file "MMentry.csv.zip" your laptop so you can enter the challenge.

Enter the challenge using the predictions in MMentry.csv.zip

Enter the challenge and see your score.

- Go to https://competitions.codalab.org/competitions/22873, and create an account using the sign up tab. Use your RCS ID as your user name. Log in.
- Go to the participate tabe https://competitions.codalab.org/competitions/22873.
- Select Submit/View Results
- Describe your submission
- Hit Submit to upload your file.
- Hit Refresh status to make sure that your submission was successful. It can take a few minutes for the update to happen.
- Look at the 'View Scoring Output Log' tab to see how you did. You should get 60% balanced accuracy for the mean method.
- Go to the results page to see how you did with repect to everyone.
- If you see a negative number, then something went wrong and go back to the Output log to see what the problem was.
- Only your latest submission will count as your submission.

Congratulations, you get three points if you sucessfully submit the Mean Method.

Write your score with the mean method here ———.

Exercise 6

Now you will create your own entry into the challenge. Surely you can do better than the Mean Method. ** Note the Mean Method was made up for educational purposes. You should never use it real-life. LDA and many other methods will probably always do better. **

Create a zip file with the name of your choice containing "mimic_synthetic_test.csv" which contains your test data predictions. Use the same proceduce as you did for the mean method. You should predict either 0 or 1 for each test datapoint. Export that file from Rstudio using the export function under More in the Files Pane. Submit to the challenge on the website. If your score is a negative number, then you made a mistake. Check out 'View scoring output log' on the website to see what might have gone wrong. Common mistakes are to not have the correct number of predictions (for example if you predict the training set instead of the testing set or to predict labels other than 0 or 1.

NOTE: Your filename must** be: `mimic_synthetic_test.csv` and it must be ziped to submit. **

- Give your RCS ID/entry name and the score that you received on the website here:
- ID = hajjam Balanced Accuracy= 0.4

The winning entry gets 5 points extra-credit.

Conclusion

Provide a conclusion which summarizes your results and recommendations briefly, and add any observations/suggestions that you have for GetUHealthy about the data, model, or future work. To conclude, I believe the LDA method was the most effiecient method as it is simpler to use and it gave very accurate recordings. I found that using my method which I had chosing to be using the nearest neighbor similar to the face recgonition model was not as accurate than the LDA method but more accurate than the mean method which showed to be the least accurate.

$ExtraCredit(3points)$

You can learn more about this problem by reading research papers about it. This data here was inspired by the MIMIC III dataset [1] but it isn't the actual MIMIC III baby. To preserve patient privacy, only researchers who have done special training can access the MIMIC data. The data you use here is a synthetic copy of the MIMIC data generated by a machine learning method created at RPI [3]. Many papers have been used to do mortality prediction. A link is provided below to a paper that looks at those many papers [3].

[1] Johnson AE, Pollard TJ, Shen L, Li-wei HL, Feng M, Ghassemi M, Moody B, Szolovits P, Celi LA, Mark RG. MIMIC-III, a freely accessible critical care database. Scientific data. 2016 May 24;3:160035. https://mimic.physionet.org/

[2]Yale A, Dash S, Dutta R, Guyon I, Pavao A, Bennett K. Privacy preserving synthetic health data. ESANN 2019 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2019, i6doc.com publ., ISBN 978-287-587-065-0 https://www.elen.ucl.ac.be/Proceedings/esann/esannpdf/es2019-29.pdf

[3] Johnson AE, Pollard TJ, Mark RG. Reproducibility in critical care: a mortality prediction case study. In Machine Learning for Healthcare Conference 2017 Nov 6 (pp. 361-376). http://proceedings.mlr.press/v68/johnson17a.html

Find a published paper that does mortality prediction using the MIMIC II data (Google Scholar is a good place to look). Summarize the problem the papers solves, the data set that is used, and the results. Speculate how methods like these could be used to fight the COVID pandemic.