Statistical learning lab 4 - Classification

Lab 3 - Regression

Abigail Gutman and Shahar Shalom

29/6/2021

# **Classification Lab**

We will try to classify handwritten digits in 28x28 greyscale values by their digit. The images are from the MNIST dataset; you can (and should) read more about the dataset here: <http://yann.lecun.com/exdb/mnist/>.

For your convenience, I have supplied a script (written by Prof. David Dalpiaz from UIUC) that downloads the data and prepares it for R. load MNIST.R

Your goal is to build a classifier for telling apart the digit 8 from the digit 3. The data includes all digits, so first separate the required digits from the training set and the test set. Please use only the 4000 images from the training set for fitting.

## prep:

## Q1:

compare two methods of classifiers.

###Model 1 : Naive bayes classifier model A generative model that uses the Bayes Law. When 25 distributions are estimated using BOOTSTRAP method for P(X|Y) while P(Y)is estimated on the train. The model chosen on threshold is 0.75 where we get the best balance as can be seen in the ROC graph we built below. Distributions are estimated by using gaussian kernel with h = 0.19 The model is chosen by its accuracy on the train forecasts (later we will examine the accuracy).

naive\_bayes\_model <- train(y ~., data = train4000 ,method = 'naive\_bayes',  
 trControl = trainControl(classProbs = TRUE,savePredictions= "final"))

###Model 2 : gardient boosting - classifiar

A model that in every step try to minimize the loss when adding the next tree. we choose the max depth of the tree to 3 to keep it simple and not get over fitting in our model. since we have 4000 train samples we did not use a sub sample to create the model, and use all the rows, but since we have large p we randomly pick 0.6 from the columns to be sampled once for each tree.

The model have been chosen on the accuracy by using cross validation 3 times. (what we could do since the package is efficient). The threshold that been chosen has we can see in the ROC below is the one that is the closest to the left upper corner.

gbm\_model\_hyperparmeters <- train(y ~ ., data = train4000, method = 'xgbTree',  
 trControl = trainControl("cv", number = 3,  
 classProbs = TRUE,savePredictions= "final"))

## Q2:

The next function calculate: (a) the confusion matrix (b) the precision - the proportion of True Positive from what the model predict as positive. (c) the recall - the proportion of True Positive from what is positive (True positive and False Negative). (d) JacardIndex

confusion\_matrix\_fun <- function(Y,Yhat){  
 Negative\_line <- Yhat[which(Y == 'X8')]  
 Postive\_line <- Yhat[which(Y == 'X3')]  
 Negative\_line <- c(sum(Negative\_line == 'X8'),  
 sum(Negative\_line == 'X3'),  
 sum(Negative\_line == 'X8')/length(Negative\_line),  
 sum(Negative\_line == 'X3')/length(Negative\_line))  
 Postive\_line <- c(sum(Postive\_line == 'X8'),  
 sum(Postive\_line == 'X3'),  
 sum(Postive\_line == 'X8')/length(Postive\_line),  
 sum(Postive\_line == 'X3')/length(Postive\_line))  
   
   
 confus <- rbind(Negative\_line,Postive\_line)  
 colnames(confus) <- c("Negtive (8)","Postive (3)","% Negtive Prediction (8)"," % Postive Prediction (3)")  
 row.names(confus) <- c("Negtive (8)","Postive (3)")  
 Pr <- confus[2,2] / sum(confus[,2])  
 Re <- confus[2,2] / sum(confus[2,])  
 Ja <- confus[2,2] / (confus[1,2] + confus[1,1] + confus[2,1])  
  
   
 return(list(confusion = confus,precision = Pr,recall = Re, JacardIndex = Ja))  
}

#### Gardient Boosting

#Gardiante boosting No overfiting  
confusion\_matrix\_fun(test$y,pred\_gbm\_model)

## $confusion  
## Negtive (8) Postive (3) % Negtive Prediction (8)  
## Negtive (8) 967 7 0.99281314  
## Postive (3) 17 993 0.01683168  
## % Postive Prediction (3)  
## Negtive (8) 0.007186858  
## Postive (3) 0.983168317  
##   
## $precision  
## [1] 0.993  
##   
## $recall  
## [1] 0.9821958  
##   
## $JacardIndex  
## [1] 1.002018

#### Naive Bayes

#Naive bayes # The important class been detcted less  
confusion\_matrix\_fun(test$y,preds\_naive\_bayes)

## $confusion  
## Negtive (8) Postive (3) % Negtive Prediction (8)  
## Negtive (8) 956 18 0.9815195  
## Postive (3) 273 737 0.2702970  
## % Postive Prediction (3)  
## Negtive (8) 0.01848049  
## Postive (3) 0.72970297  
##   
## $precision  
## [1] 0.9761589  
##   
## $recall  
## [1] 0.7289812  
##   
## $JacardIndex  
## [1] 0.5910184

We can’t see a strong over fiting in the models. But we can see that for the naive bayes is harder than the Gradient Boosting model to classified the positive.

## Q3:

Function that draws a response operating curve (ROC). Draw ROCs for both classifiers

Roc\_plot <- function(model,title){  
 t <- thresholder(model, threshold = seq(0,1,by = 0.05))  
 g <- ggplot(t, aes(y = Sensitivity,x = 1-Specificity)) + geom\_line() + labs(title = title)  
 return(g)  
}

## Q4:

For the second classifier - Gradient boosting model, display four examples (displying sample number) that were classified incorrectly.

incorrectly\_naive\_bayes <- test[-which(test$y == preds\_naive\_bayes),]  
incorrectly\_naive\_bayes <- incorrectly\_naive\_bayes[1:4,]  
  
row.names(incorrectly\_naive\_bayes)

## [1] "7" "15" "26" "29"

Naive Bayes classifier assumes the features are independent, the probabilities are incorrect if this assumption is not correct.

We work on MNIST data, cause of it we know what our data include. We can be sure that not all of the feature are independent, the feature which define the outline of the numbers 3 and 8 are dependent with each other what can be misleading.

## Q5:

Here is an image of a white digit (the digit 3) on a dark background. Do you expect both of your fitted classifiers to work well on this image? Why or why not? [Hint: Think how would this image be coded into numbers? what would happen if you try to classify using your method?]

For simplifying how the the number will be coded: Lets take the mean of each feature,with y = 3. our new picture will be coded for every feature in the area of (255 - E(feature|Y = 3)). There can be errors, but that only for the example.

The first classifier: Naive bayes might predict right since by using a gaussian kernel we allowing symmetry. The second classifier: Gradient boosting will probably predict wrong, we need to remember that that classifier learning on trees, and in this case we will almost always take the wrong turn in the cross section (black values where we are looking for white values). lets check:

predict(naive\_bayes\_model,y3)

## [1] X3  
## Levels: X3 X8

predict(gbm\_model\_hyperparmeters,y3)

## [1] X8  
## Levels: X3 X8