**'Applied Maschine Learning – Exercise 1 (28.04.17)**

**Lie Hong, Amnon Bleich, Ben Wulf**

**Task 1**

It is ‘just’ loading some libraries, load the ‘trainingData’ from the paprbag package and set the seed for reproducible random numbers.

library**(**paprbag**)**

library**(**randomForest**)**

library**(**boot**)**

data**(**'trainingData'**)**

set.seed**(**42**)** # set seed to get always the same random numbers

**Task 2**

We split the ‘trainingData’ dataset in to 2 randomly chosen parts with equal label distribution (TRUE [50]/FALSE [50]). We chose them randomly to avoid effects resulting from sorted input data.

#Task 2

#Choose some random values.

postive\_samples\_positions**<-**sample.int**(**100,50**)** #get 50 randomly chosen postiv labled datapoints

negative\_samples\_positions**<-**sample.int**(**100,50**)+**100 #get 50 randomly chosen negativ labled datapoints

trainingselection**<-** rep**(**F,200**)** # Creats a vector with 200 FALSE entries

trainingselection**[**postive\_samples\_positions**]<-TRUE**

trainingselection**[**negative\_samples\_positions**]<-TRUE**

# Select the datapoints for training

data\_training**<-**trainingData**[**trainingselection,**]**

# select the datapoints for testing

data\_test**<-**trainingData**[!**trainingselection,**]**

# Train the randomForest model and test it automaticly

rf\_model **<-** randomForest**(**Labels **~** ., data**=**data\_training**)**

# Get the confusion-matrix for the training dataset and calculate sensitivity an specifity.

cat**(**'Training-Set confusion-matrix'**)**

print**(**rf\_model**$**confusion**)**

cat**(**paste**(**"sensitivity train-set:",rf\_model**$**confusion**[**2,2**]/(**rf\_model**$**confusion**[**2,2**]+**rf\_model**$**confusion**[**2,1**])** **)**,'\n'**)**

cat**(**paste**(**"specifity train-set:",rf\_model**$**confusion**[**1,1**]/(**rf\_model**$**confusion**[**1,1**]+**rf\_model**$**confusion**[**1,2**])** **)**,'\n'**)**

# Apply predict manualy

model\_test **<-** as.logical**(**predict**(**rf\_model,data\_test**[**,**-**1**]))**

TP**<-**sum**(**model\_test**==**T **&**model\_test**==**as.logical**(**data\_test**[**,1**]))**

FP**<-**sum**(**model\_test**==**T **&**model\_test**!=**as.logical**(**data\_test**[**,1**]))**

FN**<-**sum**(**model\_test**==**F **&**model\_test**!=**as.logical**(**data\_test**[**,1**]))**

TN**<-**sum**(**model\_test**==**F **&**model\_test**==**as.logical**(**data\_test**[**,1**]))**

conf**<-** matrix**(**c**(**TP,FN,FP,TN**)**,ncol**=**2**)**

colnames**(**conf**)<-**c**(**'pred=T','pred=F'**)**

rownames**(**conf**)<-**c**(**'ground=T','ground=F'**)**

print **(**conf**)**

cat**(**paste**(**"sensitivity test-set:",TP**/(**TP**+**FN**)**,'\n'**))**

cat**(**paste**(**"specifity test-set:",TN**/(**TN**+**FP**)** **)**,'\n'**)**

Output for trainingset:

Confusion matrix:

FALSE TRUE class.error

FALSE 35 15 0.30

TRUE 17 33 0.34

Sensitivity: 0.66

Specificity : 0.7

For the test set:

Confusion matrix:

pred=T pred=F

ground=T 34 17

ground=F 16 33

Sensitivity: 0.68

Specificity : 0.66

That means that the Sensitivity is a bit better than in the training set. We don’t expect that, but we think that it is caused by a too small dataset. The Specificity is not so good as in the training. That is what we expect because the training error should be smaller than the test- error.

**Task 3**

We wrote a leave one out function that test the random forests. It gets a data set and a set of indices which are used for resampling the dataset (bootstrapping). In the end, it calculates an sensitivity and specificity for the given resampled dataset. A problem by this function is that need around 5min to return the results. What bring big trouble for the bootstrap   
confidence interval calculation.

LOOCV**<-** **function(**data,indices**)**

**{**

subdata**<-** data**[**indices,**]** # allow boot to select rows

testresults**<-**matrix**(**c**(**0,0**)**,ncol**=**2,nrow**=**2**)** # to store the results

**for** **(**i **in** 1**:(**nrow**(**data**)))** # for each data point

**{**

rf\_model **<-** randomForest**(**subdata**[-**i,**-**1**]**,subdata**[-**i,1**])** # remove it from the training Dataset

model\_test **<-** as.logical**(**predict**(**rf\_model,subdata**[**i,**-**1**]))** # and predict for the single Datapoint

matrix\_position**=**2**\***as.logical**(**subdata**[**i,1**])+**model\_test**+**1 # if reference TRUE select second row, otherwise first. if test TRUE select Second column otherwise first. + R offset for legal intervall 1:4

# P0 P1

# Ground 0 TN FP

# Ground 0 FN TP

testresults**[**matrix\_position**]<-**testresults**[**matrix\_position**]+**1

**}**

sens**<-**testresults**[**4**]/(**testresults**[**4**]+**testresults**[**3**])** # calculate sensitivtiy

spec**<-**testresults**[**1**]/(**testresults**[**1**]+**testresults**[**2**])** # calculate specifity

cat**(**'.'**)**

return**(**c**(**sens,spec**))**

**}**

We try to do the bootstrapping stuff in the same way like it is done her: <http://www.statmethods.net/advstats/bootstrapping.html>

bootstrapObj**<-** boot**(**data **=** trainingData, statistic**=**LOOCV, R**=**1000**)**

# need around 100h

# try to get confidence intervalls

sen**<-**boot.ci**(**bootstrapObj, index**=**1**)**

spe**<-**boot.ci**(**bootstrapObj, index**=**2**)**

Because the bootstrap calls the leave one out function 1000 times it takes 5\*1000 minutes to calculate that or nearly 4 days. We try a lot to speed that up but we didn’t find a solution for the calculation time problem. We tried also a smaller subset and a smaller R but then the conditions for the confidence interval are not fulfilled.