Module 8: Recommended Exercises

TMA4268 Statistical Learning V2020

Stefanie Muff, Department of Mathematical Sciences, NTNU

March xx, 2020

Contents

Recommended exercises

1. Theoretical questions:

a) Show that each bootstrap sample will contain on average approximately 2/3 of the observations.

2. Understanding the concepts and algorithms:

a) Do Exercise 1 in our book (page 332)

Draw an example (of your own invention) of a partition of two-dimensional feature space that could result from recursive binary splitting. Your example should contain at least six regions. Draw a decision tree corresponding to this partition. Be sure to label all aspects of your figures, including the regions R1,R2,. . . , the cutpoints t1,t2,. . . , and so forth.

If the class border of the two dimensional space is linear, how can that be done with recursive binary splitting?

b) Do Exercise 4 in the book (page 332).

Suppose that we want to build a regression tree based on the following dataset:

i	(x_{i1}, x_{i2})	y
1	(1,3)	2
2	(2,2)	5
3	(3,2)	3
4	(3,4)	7

Answer the following questions without using R:

c) Find the optimal splitting variable and split point for the first binary splitting for these data according to the recursive binary splitting algorithm. *Hint*: Draw a figure and look at possible divisions.

d) Continue the tree construction for the toy data until each terminal node in the tree corresponds to one single observation. Let the resulting tree be denoted T_0.

e) For what values of α in the cost-complexity criterion $C_\alpha(T)$ will the unpruned tree T_0 be the optimal tree? *Hint* : Prune the tree by cost complexity pruning.

f) Suppose that we want to predict the response y for a new observation at $\mathbf{x}$=(2,3). What is the predicted value when using the tree T_0 constructed above?

3. Implementation:

In this exercise you are going to implement a spam filter for e-mails by using tree-based methods. Data from 4601 e-mails are collected and can be uploaded from the kernlab library as follows:

```
library(kernlab)

data(spam)
```

Each e-mail is classified by *type* (*spam* or *nonspam*), and this will be the response in our model. In addition there are 57 predictors in the dataset. The predictors describe the frequency of different words in the e-mails and orthography (capitalization, spelling, punctuation and so on).

a) Study the dataset by writing *?spam* in R.

b) Create a training set and a test set for the dataset.

c) Fit a tree to the training data with *type* as the response and the rest of the variables as predictors. Study the results by using the *summary()* function. Also create a plot of the tree. How many terminal nodes does it have?

d) Predict the response on the test data. What is the misclassification rate?

e) Use the *cv.tree()* function to find the optimal tree size. Prune the tree according to the optimal tree size by using the *prune.misclass()* function and plot the result. Predict the response on the test data by using the pruned tree. What is the misclassification rate in this case?

f) Create a decision tree by using the bagging approach. Use the function *randomForest()* and consider all of the predictors in each split. Predict the response on the test data and report the misclassification rate.

g) Apply the *randomForest()* function again, but this time consider only a subset of the predictors in each split. This corresponds to the random forest-algorithm. Study the importance of each variable by using the function *importance()*. Are the results as expected based on earlier results? Again, predict the response for the test data and report the misclassification rate.

h) Use *gbm()* to construct a boosted classification tree. Predict the response for the test data and report the misclassification rate.

i) Compare the misclassification rates in d-h. Which method gives the lowest misclassification rate for the test data? Are the results as expected?

Compulsory exercise 3 from 2018 - Classification with trees

We will use the *German credit data set* from the UC Irvine machine learning repository. Our aim is to classify a customer as *good* or *bad* with respect to credit risk. A set of 20 covariates (attributes) are available (both numerical and categorical) for 300 customers with bad credit risk and 700 customers with good credit risk.

More information on the 20 covariates are found that the UCI archive data set description

```
library(caret)
# read data, divide into train and test
germancredit = read.table("http://archive.ics.uci.edu/ml/machine-learning-databases/statlog/german/germa
colnames(germancredit) = c("checkaccount", "duration", "credithistory",
    "purpose", "amount", "saving", "presentjob", "installmentrate", "sexstatus",
    "otherdebtor", "resident", "property", "age", "otherinstall", "housing",
    "ncredits", "job", "npeople", "telephone", "foreign", "response")
germancredit$response = as.factor(germancredit$response)  #2=bad
table(germancredit$response)
```

```
##
##   1   2
## 700 300
```

```
str(germancredit)  # to see factors and integers, numerics
```

```
## 'data.frame':    1000 obs. of  21 variables:
##  $ checkaccount   : Factor w/ 4 levels "A11","A12","A13",..: 1 2 4 1 1 4 4 2 4 2 ...
##  $ duration       : int  6 48 12 42 24 36 24 36 12 30 ...
##  $ credithistory  : Factor w/ 5 levels "A30","A31","A32",..: 5 3 5 3 4 3 3 3 3 5 ...
##  $ purpose        : Factor w/ 10 levels "A40","A41","A410",..: 5 5 8 4 1 8 4 2 5 1 ...
##  $ amount         : int  1169 5951 2096 7882 4870 9055 2835 6948 3059 5234 ...
##  $ saving         : Factor w/ 5 levels "A61","A62","A63",..: 5 1 1 1 1 5 3 1 4 1 ...
##  $ presentjob     : Factor w/ 5 levels "A71","A72","A73",..: 5 3 4 4 3 3 5 3 4 1 ...
##  $ installmentrate: int  4 2 2 2 3 2 3 2 2 4 ...
##  $ sexstatus      : Factor w/ 4 levels "A91","A92","A93",..: 3 2 3 3 3 3 3 3 1 4 ...
##  $ otherdebtor    : Factor w/ 3 levels "A101","A102",..: 1 1 1 3 1 1 1 1 1 1 ...
##  $ resident       : int  4 2 3 4 4 4 4 2 4 2 ...
##  $ property       : Factor w/ 4 levels "A121","A122",..: 1 1 1 2 4 4 2 3 1 3 ...
##  $ age            : int  67 22 49 45 53 35 53 35 61 28 ...
##  $ otherinstall   : Factor w/ 3 levels "A141","A142",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ housing        : Factor w/ 3 levels "A151","A152",..: 2 2 2 3 3 3 2 1 2 2 ...
##  $ ncredits       : int  2 1 1 1 2 1 1 1 1 2 ...
##  $ job            : Factor w/ 4 levels "A171","A172",..: 3 3 2 3 3 2 3 4 2 4 ...
##  $ npeople        : int  1 1 2 2 2 2 1 1 1 1 ...
##  $ telephone      : Factor w/ 2 levels "A191","A192": 2 1 1 1 1 2 1 2 1 1 ...
##  $ foreign        : Factor w/ 2 levels "A201","A202": 1 1 1 1 1 1 1 1 1 1 ...
##  $ response       : Factor w/ 2 levels "1","2": 1 2 1 1 2 1 1 1 1 2 ...
```

```
set.seed(4268)  #keep this -easier to grade work
in.train <- createDataPartition(germancredit$response, p = 0.75, list = FALSE)
# 75% for training, one split
germancredit.train <- germancredit[in.train, ]
dim(germancredit.train)
```

```
## [1] 750  21
```

```
germancredit.test <- germancredit[-in.train, ]
dim(germancredit.test)
```

```
## [1] 250  21
```

We will now look at classification trees, bagging, and random forests.

Remark: in description of the data set it is hinted that we may use unequal cost of misclassification for the two classes, but we have not covered unequal misclassification costs in this course, and will therefore not address that in this problem set.

a) Full classification tree [1 point]

```
# construct full tree
library(tree)
library(pROC)
fulltree = tree(response ~ ., germancredit.train, split = "deviance")
summary(fulltree)
plot(fulltree)
text(fulltree)
print(fulltree)
fullpred = predict(fulltree, germancredit.test, type = "class")
testres = confusionMatrix(data = fullpred, reference = germancredit.test$response)
print(testres)
1 - sum(diag(testres$table))/(sum(testres$table))
predfulltree = predict(fulltree, germancredit.test, type = "vector")
testfullroc = roc(germancredit.test$response == "2", predfulltree[, 2])
auc(testfullroc)
plot(testfullroc)
```

Run the code and study the output.

- Q1. Explain briefly how `fulltree` is constructed. The explanation should include the words: greedy, binary, deviance, root, leaves.

b) Pruned classification tree [1 point]

```
# prune the full tree
set.seed(4268)
fullcv = cv.tree(fulltree, FUN = prune.misclass, K = 5)
plot(fullcv$size, fullcv$dev, type = "b", xlab = "Terminal nodes", ylab = "misclassifications")
print(fullcv)
prunesize = fullcv$size[which.min(fullcv$dev)]
prunetree = prune.misclass(fulltree, best = prunesize)
plot(prunetree)
text(prunetree, pretty = 1)
predprunetree = predict(prunetree, germancredit.test, type = "class")
prunetest = confusionMatrix(data = predprunetree, reference = germancredit.test$response)
print(prunetest)
1 - sum(diag(prunetest$table))/(sum(prunetest$table))
predprunetree = predict(prunetree, germancredit.test, type = "vector")
testpruneroc = roc(germancredit.test$response == "2", predprunetree[,
    2])
auc(testpruneroc)
plot(testpruneroc)
```

Run the code and study the output.

- Q2. Why do we want to prune the full tree?
- Q3. How is amount of pruning decided in the code?
- Q4. Compare the the full and pruned tree classification method with focus on interpretability and the ROC curves (AUC).

c) Bagged trees [1 point]

```
library(randomForest)
set.seed(4268)
bag = randomForest(response ~ ., data = germancredit, subset = in.train,
    mtry = 20, ntree = 500, importance = TRUE)
bag$confusion
1 - sum(diag(bag$confusion))/sum(bag$confusion[1:2, 1:2])
yhat.bag = predict(bag, newdata = germancredit.test)
misclass.bag = confusionMatrix(yhat.bag, germancredit.test$response)
print(misclass.bag)
1 - sum(diag(misclass.bag$table))/(sum(misclass.bag$table))
predbag = predict(bag, germancredit.test, type = "prob")
testbagroc = roc(germancredit.test$response == "2", predbag[, 2])
auc(testbagroc)
plot(testbagroc)
varImpPlot(bag, pch = 20)
```

Run the code and study the output.

- Q5. What is the main motivation behind bagging?
- Q6. Explain what the importance plots show, and give your interpretation for the data set.
- Q7. Compare the performance of bagging with the best of the full and pruned tree model above with focus on interpretability and the ROC curves (AUC).

d) Random forest [1 point]

```
set.seed(4268)
rf = randomForest(response ~ ., data = germancredit, subset = in.train,
    mtry = 4, ntree = 500, importance = TRUE)
rf$confusion
1 - sum(diag(rf$confusion))/sum(rf$confusion[1:2, 1:2])
yhat.rf = predict(rf, newdata = germancredit.test)
misclass.rf = confusionMatrix(yhat.rf, germancredit.test$response)
print(misclass.rf)
1 - sum(diag(misclass.rf$table))/(sum(misclass.rf$table))
predrf = predict(rf, germancredit.test, type = "prob")
testrfroc = roc(germancredit.test$response == "2", predrf[, 2])
auc(testrfroc)
plot(testrfroc)
varImpPlot(rf, pch = 20)
```

Run the code and study the output.

- Q8. The parameter `mtry=4` is used. What does this parameter mean, and what is the motivation behind choosing exactly this value?
- Q9. The value of the parameter `mtry` is the only difference between bagging and random forest. What is the effect of choosing `mtry` to be a value less than the number of covariates?
- Q10. Would you prefer to use bagging or random forest to classify the credit risk data?

Exam problems

V2018 Problem 4 Classification of diabetes cases

c)

Q20: Explain how we build a bagged set of trees, and why we would want to fit more than one tree.
Q21: Assume we have a data set of size n, calculate the probability that a given observation is in a given bootstrap sample.
Q22: What is an OOB sample?