Compulsory exercise 3, version 16.04.2018

TMA4268 Statistical Learning V2018

Contact person: Andreas Strand, andreas.strand@ntnu.no

To be handed in on Blackboard: deadline May 4 at 16.00

(12.04: added extra supervision times, 16.04: added random seed in svm-task.)

Introduction

Maximal score is 10 points. You need a score of 4/10 for the exercise to be approved. Your score will make up 10% points of your final grade.

Supervision

- Fridays in April 12-14 in Smia, in addition
- Wednesday April 25, 10-12 in EL1
- Wednesday May 2, 10-12 in 656, 6.etg. sentralbygg 2.
- If needed: Friday May 4 at 10-12 in Andreas' office at Materialteknisk 3-1281.

Practical issues

- Maximal group size is 3 - join a group (self enroll) before handing in on Blackboard.
- Remember to write your names and group number on top of your submission.
- The exercise should be handed in as one R Markdown file and a pdf-compiled version of the R Markdown file (if you are not able to produce a pdf-file directly please make an html-file, open it in your browser and save as pdf - no, not landscape - but portrait please). We will print the pdf-files (and you get written comments) and use the Rmd file in case we need to check details in your submission.
- In the R-chunks please use both `echo=TRUE` and `eval=TRUE` to make it simpler for us to read and grade.
- Please do not include all the text from this file (that you are reading now) - we want your R code, plots and written solutions - see the template below.
- Please not more than 10 pages in your pdf-file! (This is a request, not a requirement.)
- Please save us time and NOT submit word or zip - or only Rmd - that only results in extra work for us!

Template file for submission of Compulsory exercise 3 is here: https://www.math.ntnu.no/emner/TMA4268/2018v/CompEx3mal.Rmd

You need to install the following packages in R:

```
install.packages("caret")
install.packages("tree")
install.packages("pROC")
install.packages("RandomForest")
install.packages("corrplot")
install.packages("knitr")
install.packages("e1071")
install.packages("keras")
```

The `keras` package will call Python, so Python must be installed on your machine. After installing the `keras` package, the installation of keras and dependencies should be done only once:

```
library(keras)            # Load R package Keras
install_keras()           # Install Keras and dependencies
```

Problem 1 - Classification with trees [4 points]

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",
germancredit$response = as.factor(germancredit$response) #2=bad
table(germancredit$response)
str(germancredit) # to see factors and integers, numerics

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

```
##
##   1   2
## 700 300
## '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 ...
## [1] 750  21
## [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?

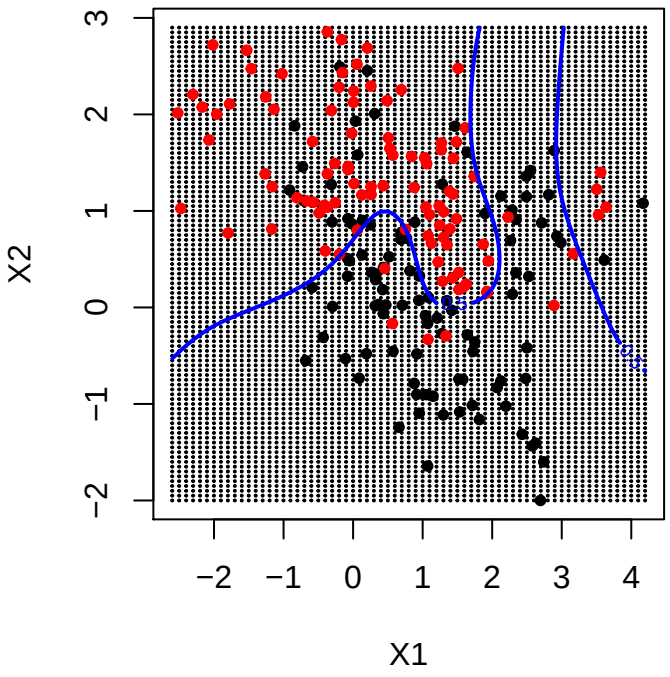
- Q10. Would you prefer to use bagging or random forest to classify the credit risk data?

Problem 2 - Nonlinear class boundaries and support vector machine [2 points]

a) Bayes decision boundary [1 point]

We will study classification applied to a simulated data set with two classes from @ESL, where the data set is supplied together with the Bayes decision boundary. The boundary is plotted below, together with a training set.

```
load(url("https://web.stanford.edu/~hastie/ElemStatLearn/datasets/ESL.mixture.rda"))
#names(ESL.mixture)
#prob gives probabilites for each class when the true density functions are known
#px1 and px2 are coordinates in x1 (length 69) and x2 (length 99) where class probabilites are calculat
rm(x,y)
attach(ESL.mixture)
dat=data.frame(y=factor(y),x)
xgrid=expand.grid(X1=px1,X2=px2)
par(pty="s")
plot(xgrid, pch=20,cex=.2)
points(x,col=y+1,pch=20)
contour(px1,px2,matrix(prob,69,99),level=0.5,add=TRUE,col="blue",lwd=2) #optimal boundary
```



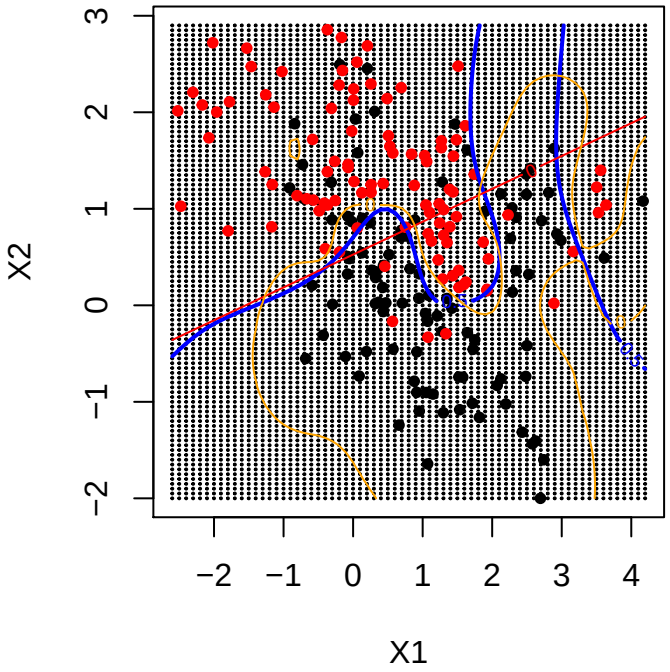
- Q11. What is a Bayes classifier, Bayes decision boundary and Bayes error rate? Hint: pages 37-39 in @ISL.
- Q12. When the Bayes decision boundary is known, do we then need a test set?

b) Support vector machine [1 point]

```
library(e1071)
# support vector classifier
svcfits=tune(svm,factor(y)~.,data=dat,scale=FALSE,kernel="linear",ranges=list(cost=c(1e-2,1e-1,1,5,10)))
summary(svcfits)
svcfit=svm(factor(y)~.,data=dat,scale=FALSE,kernel="linear",cost=0.01)
# support vector machine with radial kernel
set.seed(4268)
svmfits=tune(svm,factor(y)~.,data=dat,scale=FALSE,kernel="radial",ranges=list(cost=c(1e-2,1e-1,1,5,10),
summary(svmfits)
svmfit=svm(factor(y)~.,data=dat,scale=FALSE,kernel="radial",cost=1,gamma=5)

# the same as in a - the Bayes boundary
par(pty="s")
plot(xgrid, pch=20,cex=.2)
points(x,col=y+1,pch=20)
contour(px1,px2,matrix(prob,69,99),level=0.5,add=TRUE,col="blue",lwd=2) #optimal boundary

# decision boundaries from svc and svm added
svcfunc=predict(svcfit,xgrid,decision.values=TRUE)
svcfunc=attributes(svcfunc)$decision
contour(px1,px2,matrix(svcfunc,69,99),level=0,add=TRUE,col="red") #svc boundary
svmfunc=predict(svmfit,xgrid,decision.values=TRUE)
svmfunc=attributes(svmfunc)$decision
contour(px1,px2,matrix(svmfunc,69,99),level=0,add=TRUE,col="orange") #svm boundary
```




```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
```

```
## - best parameters:
##  cost
##   0.1
##
## - best performance: 0.285
##
## - Detailed performance results:
##    cost error dispersion
## 1  0.01 0.300 0.08819171
## 2  0.10 0.285 0.08181958
## 3  1.00 0.290 0.11254629
## 4  5.00 0.290 0.11254629
## 5 10.00 0.290 0.11254629
##
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##     1     5
##
## - best performance: 0.155
##
## - Detailed performance results:
##     cost gamma error dispersion
## 1   0.01  0.01 0.455 0.17551511
## 2   0.10  0.01 0.440 0.16964014
## 3   1.00  0.01 0.295 0.09264628
## 4   5.00  0.01 0.290 0.10219806
## 5  10.00  0.01 0.295 0.09264628
## 6   0.01  1.00 0.435 0.19155794
## 7   0.10  1.00 0.210 0.07745967
## 8   1.00  1.00 0.170 0.10327956
## 9   5.00  1.00 0.165 0.08181958
## 10 10.00  1.00 0.180 0.07527727
## 11  0.01  5.00 0.395 0.22785229
## 12  0.10  5.00 0.315 0.20145305
## 13  1.00  5.00 0.155 0.06851602
## 14  5.00  5.00 0.190 0.06992059
## 15 10.00  5.00 0.200 0.09128709
## 16  0.01 10.00 0.430 0.19032137
## 17  0.10 10.00 0.415 0.18265024
## 18  1.00 10.00 0.175 0.06770032
## 19  5.00 10.00 0.245 0.10394977
## 20 10.00 10.00 0.260 0.11005049
```

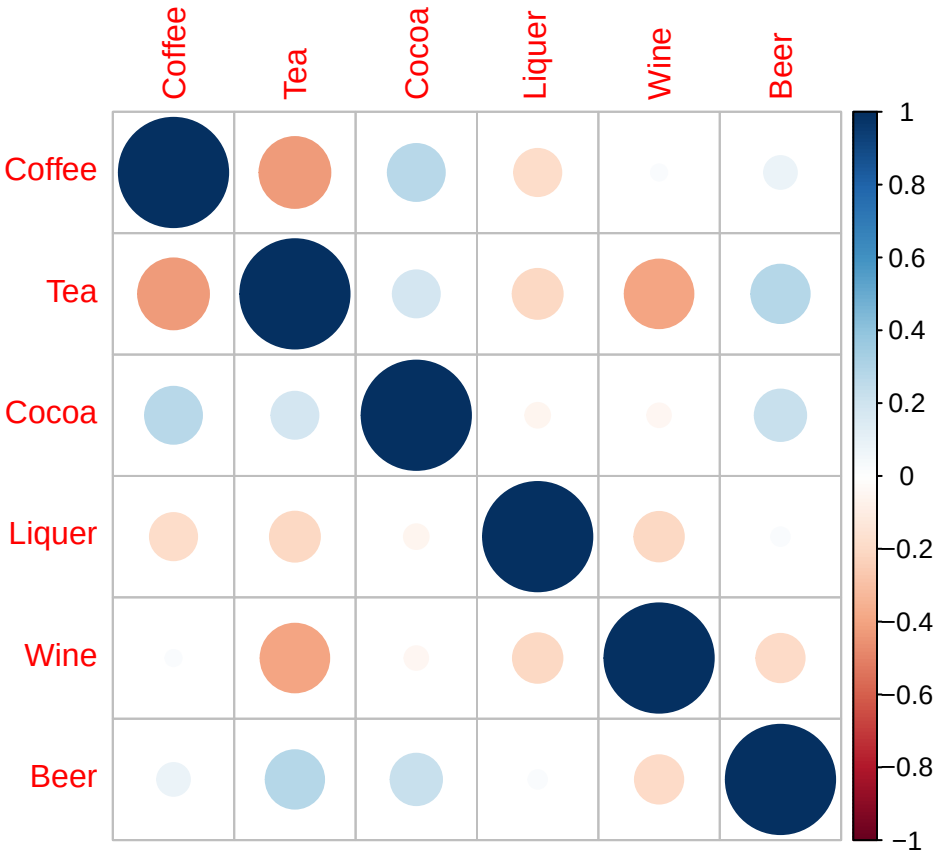
- Q13. What is the difference between a support vector classifier and a support vector machine?
- Q14. What are parameters for the support vector classifier and the support vector machine? How are these chosen above?
- Q15. How would you evaluate the support vector machine decision boundary compared to the Bayes decision boundary?

Problem 3 - Unsupervised methods [2 points]

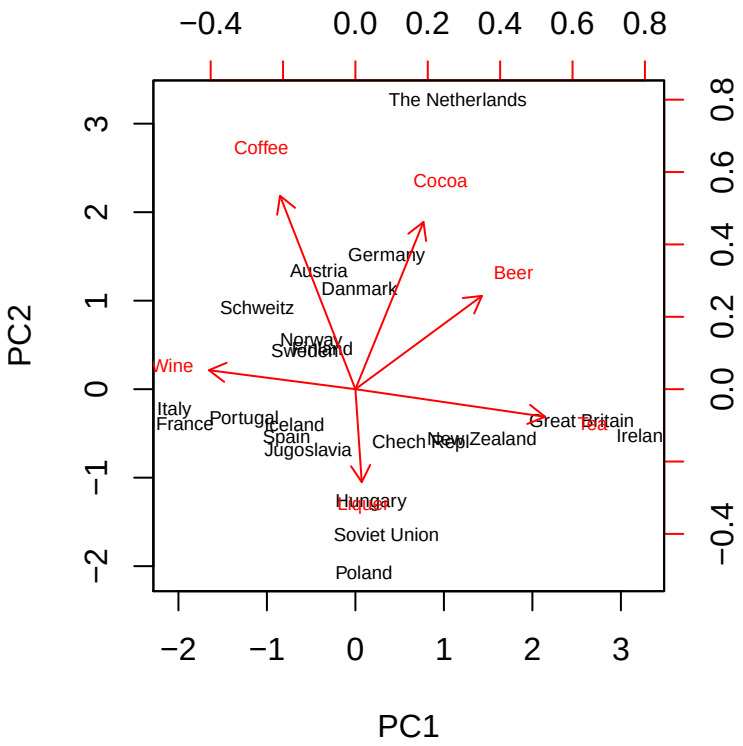
a) Principal component analysis [1 point]

We will study data on consumption of different beverages for a selection of countries, and perform principal component analysis.

```
# reading data on consumption of different beverages for countries
drink <- read.csv("https://www.math.ntnu.no/emner/TMA4267/2017v/drikke.TXT",sep=",",header=TRUE)
drink <- na.omit(drink)
# looking at correlation between consumptions
drinkcorr=cor(drink)
library(corrplot)
corrplot(drinkcorr,method="circle")
```




```
# now for PCA
pcaS <- prcomp(drink,scale=TRUE) # scale: variables are scaled
pcaS$rotation
summary(pcaS)
biplot(pcaS,scale=0,cex=0.6) # scale=0: arrows scaled to represent the loadings
```




```
##                PC1         PC2         PC3          PC4         PC5
## Coffee -0.26029733  0.66788815 -0.22475187  0.4132467433  0.07431918
## Tea     0.65540048 -0.09539757  0.36756357 -0.0002927055 -0.12503940
## Cocoa   0.23510209  0.57754726 -0.06603093 -0.4200858712 -0.61199325
## Liquer  0.02190508 -0.32118904 -0.79997824 -0.3292322714 -0.12307455
## Wine   -0.50599685  0.06551597  0.37109534 -0.6765579799  0.15862233
## Beer    0.43693234  0.32219426 -0.17985159 -0.2943302832  0.75099779
##                PC6
## Coffee  0.5092751
## Tea     0.6407898
## Cocoa  -0.2362164
## Liquer  0.3644878
## Wine    0.3450672
## Beer   -0.1493533
## Importance of components%s:
##                             PC1    PC2    PC3    PC4    PC5     PC6
## Standard deviation     1.3117 1.1957 1.0681 0.8793 0.8576 0.44782
## Proportion of Variance 0.2867 0.2383 0.1901 0.1288 0.1226 0.03342
## Cumulative Proportion  0.2867 0.5250 0.7151 0.8440 0.9666 1.00000
```

- Q16. Explain what you see in the `biplot` in relation to the loadings for the first two principal components.
- Q17. Does this analysis give you any insight into the consumption of beverages and similarities between countries?

b) Hierarchical clustering [1 point]

Within the research area of phylogeny similarities between species are studied, and below you find a distance matrix for five species. The distances are calculated from mitochondrial DNA sequences from the species.

```
library(knitr)
species=c("Human","Chimpanzee","Gorilla","Orangutan","Gibbon")
distJC <- matrix(c(0,1,3,9,12,
                   1,0,2,8,11,
                   3,2,0,6,11,
                   9,8,6,0,11,
                   12,11,11,11,0),5,5)
dimnames(distJC) <- list(species,species)
kable(distJC)
```

	Human	Chimpanzee	Gorilla	Orangutan	Gibbon
Human	0	1	3	9	12
Chimpanzee	1	0	2	8	11
Gorilla	3	2	0	6	11
Orangutan	9	8	6	0	11
Gibbon	12	11	11	11	0

Above you see three dendrogrammes labelled A, B and C, constructed from the distance matrix above. The dendrograms show agreement on how to cluster the species, but differ in branch lenghts when species are fused together in clusters.

- Q18. Describe how the distance between *clusters* are defined for single, complete and average linkage.
- Q19. Identify which of the three dendrograms (A, B, C) correspond to the three methods single, complete and average linkage. Justify your solution.

Problem 4 - Neural networks [2 points]

During our lectures, we have used a densely connected neural network to classify a movie review as either positive or negative using the IMDB data set. We will use the same data here, refer to the lecture slides for reading the data set and setting up the network.

More specifically, we have defined a linear stack of dense layers, containing two intermediate layers containing 16 hidden units each with reLu activation functions.

```
model <- keras_model_sequential() %>%
  layer_dense(units = 16, activation = "relu", input_shape = c(10000)) %>%
  layer_dense(units = 16, activation = "relu") %>%
  layer_dense(units = 1, activation = "sigmoid")
```

- Q20. What is the advantage of using a non-linear activation function such as `relu`?
- Q21. Why do we need to use a different activation function (`sigmoid`) in the output layer instead of using `relu` again?

In one of our lectures, we have fitted the model above and plotted training and validation loss as well as accuracy, as illustrated in the figure below.

Keep two intermediate layers in the network. Try creating a simpler model with 4 hidden units instead of 16 and a more complex model with 32 hidden units instead of 16.

- Q22. Plot the training and validation loss and accuracy for the simpler and more complex model mentioned above. How do they compare with the model with 16 hidden units?
- Q23. Besides reducing the network's size, what other methods can be used to avoid overfitting with neural network models? Briefly describe the intuition behind each one.

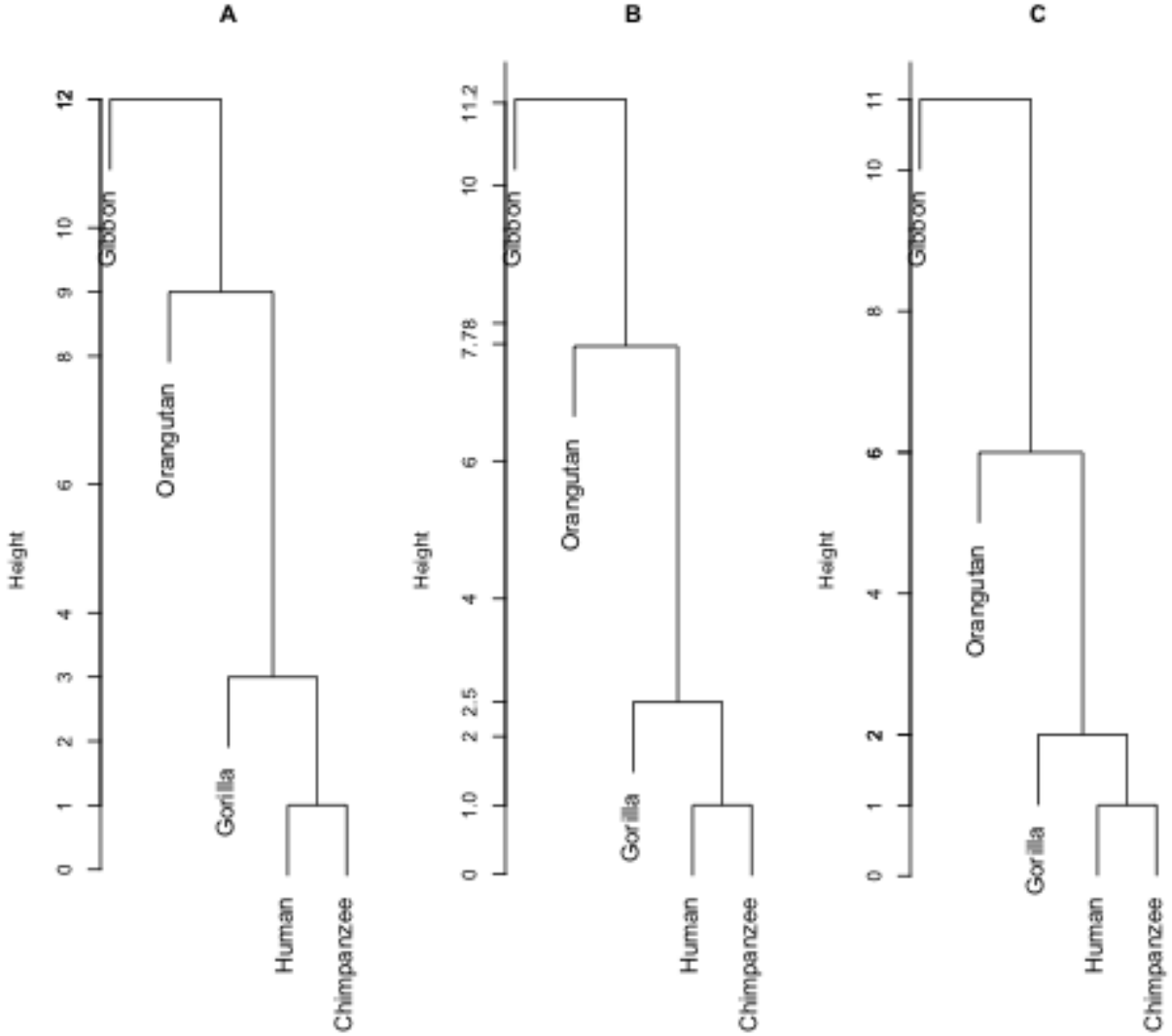

A
B
C
Height
Gibbon
Orangutan
Gorilla
Human
Chimpanzee
0
1
2
3
4
6
8
9
10
12
1.0
2
2.5
7.78
11.2
11

Figure 1:

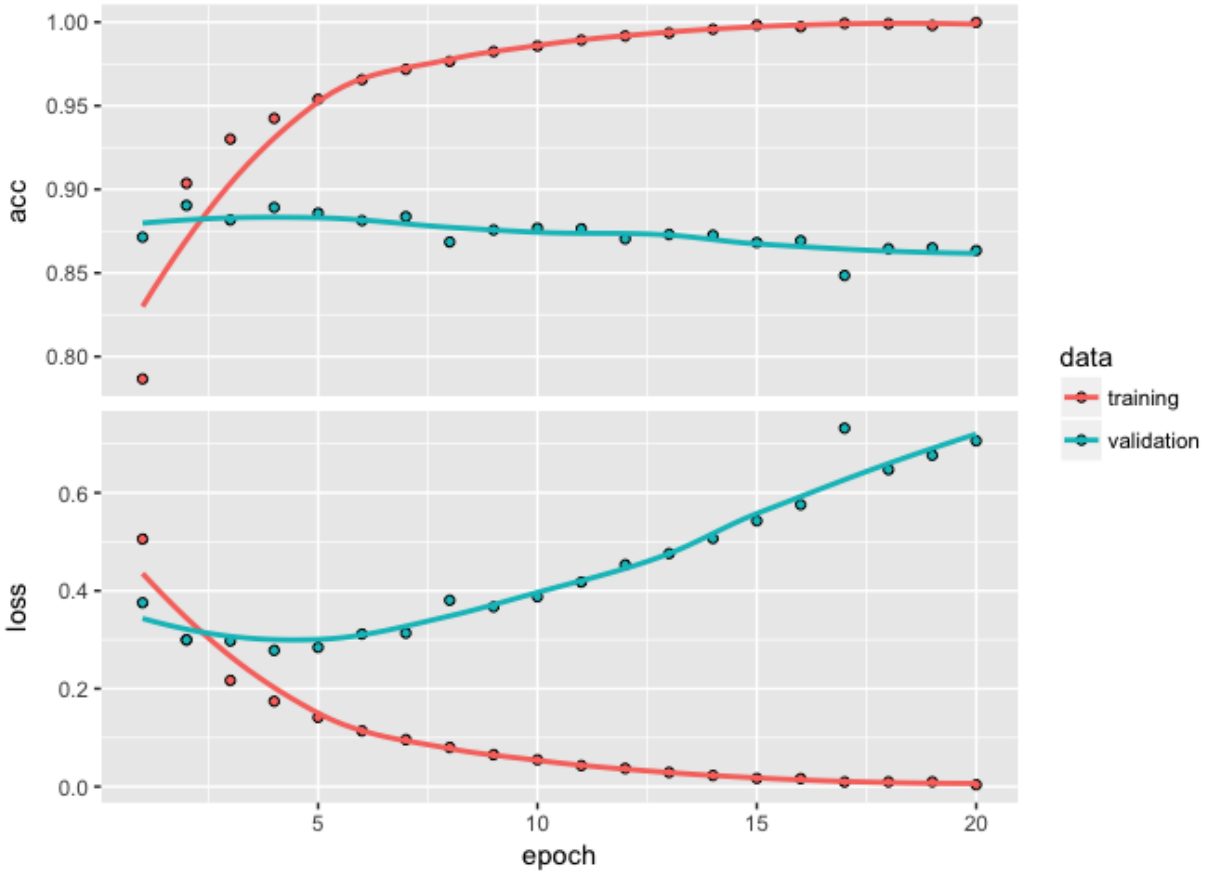

1.00
0.95
0.90
0.85
0.80
acc
0.6
0.4
0.2
0.0
loss
5
10
15
20
epoch
data
training
validation

Figure 2:

References