19th April, 2023

STK2100 - Machine learning and statistical methods for prediction and classification

Mandatory assignment 2 of 2

Submission deadline

Thursday May 4th 2023, 14:30 in Canvas (canvas.uio.no).

Instructions

You can choose between scanning handwritten notes or typing the solution directly on a computer (for instance with LaTeX). The assignment must be submitted as a single PDF file. Scanned pages must be clearly legible. The submission must contain your name, course and assignment number.

It is expected that you give a clear presentation with all necessary explanations. Remember to include all relevant plots and figures. Students who fail the assignment, but have made a genuine effort at solving the exercises, are given a second attempt at revising their answers. All aids, including collaboration, are allowed, but the submission must be written by you and reflect your understanding of the subject. If we doubt that you have understood the content you have handed in, we may request that you give an oral account.

In exercises where you are asked to write a computer program, you need to hand in the code along with the rest of the assignment. It is important that the submitted program contains a trial run, so that it is easy to see the result of the code.

Application for postponed delivery

If you need to apply for a postponement of the submission deadline due to illness or other reasons, you have to contact the Student Administration at the Department of Mathematics (e-mail: studieinfo@math.uio.no) well before the deadline.

All mandatory assignments in this course must be approved in the same semester, before you are allowed to take the final examination.

Complete guidelines about delivery of mandatory assignments:

uio.no/english/studies/admin/compulsory-activities/mn-math-mandatory.html

GOOD LUCK!

Spesific requirements to this assignment:

In order to get the assignment **accepted** you need to fullfil the following requirements:

- You have done a real attempt on **all** (sub-)questions. This includes commenting on the results you obtain. **This applies to the first submission**!
- There is a satisfactory answer in at least 2/3 of the (sub-)questions.

Remember that it is allowed to ask for help!
Within the exercises several commands that can be used in **R** are included. If some libraries are not available at your computer, you need to install them, for example by

```
install.packages("MASS")
```

All the command listed in the assignment are aslo available in a separate .R file on the course webpage.

It is allowed to use other programs, but there will then be extra requirements to good documentation on what you have done and you can not expect to obtain help with respect to implementional details.

At https://www.uio.no/studier/emner/matnat/math/STK2100/v23/r-scripts/stk2100_mandatory_assignment_2.rmd there is an R-markdown file which you can use for the assignment.

Problem 1. We will in this problem look at a data set about spam which arrives in email boxes. The data set is described in http://www.uio.no/studier/emner/matnat/math/STK2100/data/spam_info.txt and is available at http://www.uio.no/studier/emner/matnat/math/STK2100/data/spam_data.txt. The datafile is somewhat different from the description in that there is one extra column TRAIN which indicate the division in training (`TRUE`) and test (`FALSE`) sets. Further, the response column `y` is given as `TRUE` if it is a spam and `FALSE` otherwise.

In **R** the data can be read by the commands

```
fil = "http://www.uio.no/studier/emner/matnat/math/STK2100/data/spam_data.txt"
spam = read.table(fil,header=T)
```

The aim with the exercise is to look at different methods for classifying mail to spam or not spam based on 57 different explanatory variables. The explanatory variables essentially measure the frequencies of words or letters within the mail.

(a) First run standard logistic regression with fitting based on the training data and evaluation based on the test data. You may use the following **R** commands:

```
fit = glm(y~.-train,data=spam,subset=spam$train,family=binomial)
pred = predict(fit,spam[!spam$train,],type="response")>0.5
```

Specify a suitable measure for prediction performance. (You should use the same measure throughout.)

What value of the measure do you obtain in this case?

Note: You will probably get some warnings when calling the `glm` routine. You may search on the web for explanations on this. If the aim would be to *understand* the relationship between the explanatory variables and the response, one should explore this further. Here we will mainly be concerned with prediction in which case we can ignore these warnings.

(b) An alternative to use all the explanatory variables is to reduce the dimension by principal components. You can make a new data object with the principal components in **R** by the commands

```
x.prcomp = prcomp(spam[,1:57], retx=TRUE, scale=TRUE)
d = data.frame(x.prcomp$x, y=spam$y, train=spam$train)
```

where the option `scale=TRUE` result in that the variables are scaled to have variances equal to 1 before transformation.

Why is this scale transformation a reasonable option?

Fitting a logistic model based on the first 2 principal components can be performed by

```
fit.pc = glm(y~.-train, data=d[,c(1:2,58,59)], family=binomial, subset=d$train)
pred.pc = predict(fit.pc, d[!d$train,], type="response")>0.5
```

Try out and report your results.

(c) Try out logistic regression by using the first k principal components for different values of k.

Which value of k gives the best result for the test data?

By looking at your performance measure for all different values of k, would you choose the model that gives the best performance or would you make another choice?

(d) In order to understand the structures of the principal components, it may be useful to look at what linear combinations the different principal components relate to. The command

```
plot.ts(x.prcomp$rotation[,1])
```

shows the weights for the 57 different explanatory variables for the first principal component (and similarly for the other components). Use this to give some interpretation of the components that you have chosen to include.

(e) We will now look at some non-linear terms in the model. A generalized additive model (GAM) based on the first three explanatory variables can be fitted and predictions can be made by the commands

```
library(gam)
fit.gam = gam(y~s(x1)+s(x2)+s(x3),data=spam,
   subset=spam$train,family=binomial)
pred.gam = predict(fit.gam,spam[!spam$train,],type="response")>0.5
```

Compare this prediction with a similar model with only linear terms. Do you obtain any improvements by including non-linear structures?

Look a bit closer into the non-linear structures by plotting the estimated functions:

```
plot(fit.gam,se=T)
```

Discuss the results and try in particular to explain the non-linear structures based on what the explanatory variables actually measure (it might be helpful here to make some histograms of the explanatory variables).

(f) Repeat the procedure from (e) but now based on the first three principal components. The GAM fit can be made by the commands

```
fit.pc.gam = gam(y~s(PC1)+s(PC2)+s(PC3),data=d,
  subset=d$train,family=binomial)
pred.pc.gam = predict(fit.pc.gam,d[!d$train,],type="response")>0.5
```

Do you get any improvements by including the non-linear structures in this case?

Make plots of the non-linear structures also in this case and relate this to the results you obtained.

(g) We will now look at more principal components and non-linear structures. In order to avoid writing down long formulas, it will be useful to construct a character string which include the model formula, here illustrated for $k = 20$.

```
nam = names(d)[1:57]
k=20
formula = as.formula(paste("y",paste(paste("s(",nam[1:k],")",sep=""),
      collapse="+"), sep="~"))
print(formula)
fit.pc.gam = gam(formula,data=d,subset=d$train,family=binomial)
pred.pc.gam = predict(fit.pc.gam,d[!d$train,],type="response")>0.5
```

Try out these commands. Do you get any improvements compared to your earlier results?

(h) Finally, try out non-linear structures on the principal components for different values of k. Find the k which gives the best result. Also look at the errors for different values of k and make a choise.

For your choice of k, plot the non-linear structures. Relate this again to your results.

Note: This part will be computer intensive!

(i) Summarize your results. Also include a discussion about possible weaknesses with the experiments that are made.

Problem 2. We will in this exercise continue on the spam data set, but now consider the use of neural nets.

(a) A neural net with one layer, 10 hidden nodes and a decay parameter equal to 0.3 can be performed with the following commands.

```
library(MASS)
library(nnet)
library(class)
spam$y = as.factor(spam$y)
spam.nnet = nnet(y~.-train,data=spam[spam$train==TRUE,],
      size=10,decay=0.3,MaxNWts=10000,maxit=500)
pred = predict(spam.nnet,spam[spam$train==FALSE,],type="class")
show(table(spam$y[spam$train==FALSE],pred))
```

Run these commands, and also calculate performance messure you defined in Problem 1.

(b) Try out different values of the decay parameter.

Which value gave the best prediction?

Explain what the decay parameter do.

(c) Also try out different numbers of hidden layers. Did you get any improvements?

(d) Turn now to deep learning, using the `mlp` routine. The following commands are available in this case:

```
library(RSNNS)
spamTargets <- decodeClassLabels(spam$y)
spamTargets.train = spamTargets[spam$train==TRUE,]
spamTargets.test = spamTargets[spam$train==FALSE,]
spam.dnet = mlp(as.matrix(spam[spam$train==TRUE,1:57]),
                spamTargets.train, size = c(6,10,6),
                learnFuncParams=c(0.3),maxit=300)
pred.dnet = predict(spam.dnet,spam[spam$train==FALSE,1:57])
pred.dnet = apply(pred.dnet,1,which.is.max)-1
pred.dnet = pred.dnet==1
err.dnet = mean(spam$y[spam$train==FALSE]!=pred.dnet)
```

Try out this command with different numbers of maxit.

Which value of maxit performed best?

Discuss why maxit can be used as a sort of penalization.

(e) Also try out different sized of the neural net.

Do you get any improvements?