7th April, 2022

STK2100 - Machine learning and statistical methods for prediction and classification

Mandatory assignment 2 of 2

Submission deadline

Thursday 05th May 2022, 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. 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 fulfill one of the following requirements:

- You have done a real attempt on **all** (sub-)questions. This includes commenting on the results you obtain, **or**
- there is a satisfactory answer in at least 2/3 of the (sub-)questions. By satisfactory we mean both doing the actual computations *and* discussing the results.

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 programming languages or software, 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.

Problem 1. We will in this problem look at a data set on breast cancer. The data set is described in https://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+(diagnostic) and is already divided for you into a train set at https://www.uio.no/studier/emner/matnat/math/STK2100/v22/data/train.csv and a test set at https://www.uio.no/studier/emner/matnat/math/STK2100/v22/data/test.csv. Further, the response column `X` is given as `TRUE` for a cancerous tissue and `FALSE` otherwise.

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

```
dir = "https://www.uio.no/studier/emner/matnat/math/STK2100/v22/data/"
Xtr = read.table(paste(dir,"train.csv",sep=""),header=T,sep=",")[,2:32]
Xte = read.table(paste(dir,"test.csv",sep=""),header=T,sep =  ",")[,2:32]
```

Throughout the exercises you will be given help-listings, which are also available in a single file here https://www.uio.no/studier/emner/matnat/math/STK2100/v22/r-scripts/help_oblig2.r.

The aim with the exercise is to look at different methods for classifying tissues to cancerous or not based on 30 different explanatory variables. The explanatory variables essentially measure means, standard deviations and other summary statistics on physical measurements of the tissues.

(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(X~.,data=Xtr,family=binomial)
pred = predict(fit,Xte,type="response")>0.5
```

Why are you getting a warning here?[1] Explain the limitation of logistic regression corresponding to the obtained warning.

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?

(b) Choose only first two covariates ("radius_mean" "texture_mean") to fit a logistic regression and evaluate the performance of this model.

How did model perform compared to the one in (a)

Did you get the warning here? Why is it so?

(c) Another 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(x = as.matrix(rbind(Xtr,Xte)[,-31]),retx=TRUE,
scale=TRUE)
Xptr = data.frame(x.prcomp$x[1:dim(Xtr)[1],],X=Xtr$X)
Xpte = data.frame(x.prcomp$x[(dim(Xtr)[1]+1):(dim(Xtr)[1]+dim(Xte)[1]),],
X=Xte$X)
```

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(X~.,data=Xptr[,c(1:2,31)],family=binomial)
pred.pc = predict(fit.pc,Xpte[,c(1:2,31)],type="response")>0.5
```

Try out and report your results. Compare them to those in (a) and (b).

Plot the decision boundary based on a set of 50000 points generated uniformly for PC1 and PC2 from -10 to 10. This can be done through

```
X0 = runif(50000,-10,10)
X1 =  runif(50000,-10,10)
Xnew = as.data.frame(cbind(X0,X1))
colnames(Xnew) = c("PC1","PC2")
preds =predict(fit.pc,Xnew,type="response")>0.5
x1 = Xnew[preds,]
x0 = Xnew[which(preds==0),]
plot(x1,col = 4,xlim = c(-10,10),ylim = c(-10,10),pch=16)
points(x0,col = 7,pch=16)
```

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

Add training data points from both of the class to the graph. The following command may be used.

```
points(Xptr[Xptr$X==0,c(1,2)],col = 3,pch=1)
points(Xptr[Xptr$X==1,c(1,2)],col = 2,pch=1)
```

Add testing data points from both of the class to the graph. The following command may be used.

```
points(Xpte[Xpte$X==0,c(1,2)],col = 3,pch=16)
points(Xpte[Xpte$X==1,c(1,2)],col = 2,pch=16)
```

(d) Implement classifier based on LDA and QDA from scratch[2]. You may partially utilize the following function in your implementation. You may use colMeans and cov function in R to compute the required means and covariances corresponding to the settings of LDA and QDA.

```
da_mi = function(x,p1,p2,m1,S1,m2,S2)
{
  pc1 = p1*mvtnorm::dmvnorm(x,m1,S1)
  pc2 = p2*mvtnorm::dmvnorm(x,m2,S2)
  pc2/(pc1+pc2)
}
```

Compare the performance of QDA, LDA to (a)-(c) using the first two principal components (PC1 and PC2) as covariates.

Plot the decision boundaries w.r.t. PC1 and PC2 for LDA and QDA as in (c).

Discuss the decision boundaries obtained by all of the methods.

(e) Implement classifier based on 1-nearest neighbors and 5-nearest neighbors from scratch using again the first two principal components[3]. You may fully or partially utilize the following function in your implementation.

```
knn_mi = function(x1,x2, K = 3,X)
{
  p1 = array(0,dim(x1)[1])
  for(k in 1:dim(x1)[1])
    p1[k] = sqrt((X[1]-x1[k,1])^2+(X[2]-x1[k,2])^2)
  p2 = array(0,dim(x2)[1])
  for(k in 1:dim(x2)[1])
    p2[k] = sqrt((X[1]-x2[k,1])^2+(X[2]-x2[k,2])^2)
  P1 = 0
  for(el in sort(c(p1,p2))[1:K])
  {
    if(el %in% p1)
```

[2]If you struggle to implement the methods, you can use standard **LDA**, and **QDA** functions from library(MASS), but then only half point can be given for d) at most.

[3]If you struggle to implement the methods, you can use standard **knn** from library(class), but then only half point can be given for e) at most.

```
    {
      P1 = P1 + 1
    }
  }
  return((P1/K))
}
```

Compare the performance of KNNs to (a)-(d) using the first two principal components (PC1 and PC2) as covariates.

Plot the decision boundaries for the addressed KNNs as in (c) w.r.t. PC1 and PC2.

(f) Create a fully connected neural network with two hidden layers of 4 and 3 neurons each and a sigmoid activation function in library(RSNNS) as in the following listing.

```
fit.dnet = mlp(as.matrix(Xptr[,1:2]),Xptr$X, size = c(4,3),
               learnFuncParams=c(0.3),maxit=10000)
```

Then plot a decision boundary for your neural network from library(RSNNS) and compare it to those in (c)-(e), also compare in terms of the evaluation measure that you have chosen.

(g) Now create a model in torch

```
#define a model lass
net = nn_module(
  "CancerNet",
  initialize = function() {
    self$fc1 = nn_linear(length(features), 4)
    self$fc2 = nn_linear(4, 3)
    self$fc3 = nn_linear(3, 1)
  },
  forward = function(x) {
    x %>%
      self$fc1() %>%
      nnf_sigmoid() %>%
      self$fc2() %>%
      nnf_sigmoid() %>%
      self$fc3()
  }
)
```

You may use the following data set class for Breast cancer data.

```
df_dataset = dataset(
  name = "Breast cancer",
  initialize = function(df, feature_variables, response_variable) {
    self$df = df[, feature_variables]
    self$response_variable = df[[response_variable]]
  },
  .getitem = function(index) {
    response = torch_tensor(self$response_variable[index], dtype = torch_float())
```

```
    x = torch_tensor(as.numeric(self$df[index,]))
    list(x = x, y = response)
  },
  .length = function() {
    length(self$response_variable)
  }
)
```

You may further use the example in (g) in https://www.uio.no/studier/emner/matnat/math/STK2100/v22/r-scripts/help_oblig2.r.

Choose an appropriate loss and regularization. Implement an SGD optimization algorithm with a batch size of 10. Tune step size of SGD, number of epochs, regularization strength, and classification threshold to obtain a reasonable performance of your neural network. Tune until you obtain the performance comparable or better than QDA and LDA. This can be done by tuning:

```
# number of SGD epochs
M = 400
# weight decay
lambda = 0.0001
# SGD learning rate
alpha =  0.001
# classification treshold
u = 0.5
```

Report your final choice. Then plot a decision boundary for your best performing torch neural network from and compare it to those in (c)-(f).

(h) Now 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?

(i) 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 30 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.

(j) We will now look at other non-linear terms in the model. A generalized additive model (GAM) based on the first two principal components can be fitted and predictions can be made by the commands

```
library(gam)
fit.pc.gam = gam(X~s(PC1)+s(PC2),data=Xptr,family=binomial)
pred.pc.gam = predict(fit.pc.gam,Xpte,type="response")>0.5
```

Compare this prediction with a similar model with only linear terms and other used methods. 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.pc.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).

Plot a decision boundary for GAM and compare it to those of previously used models.

(k) Repeat the procedure from (j) but now based on the first three principal components.

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.

(l) 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(Xptr)[1:30]
k=20
formula = as.formula(paste("X",paste(paste("s(",nam[1:k],")",sep=""),
      collapse="+"), sep="~"))
print(formula)
```

Try out such formula for GAM. Do you get any improvements compared to your earlier results?

(m) 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!

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