Tentative solutions to exam 24 May 2018

TMA4268 Statistical learning

Mette Langaas

11 June, 2018

Problem 1: K-nearest neighbour regression

[10 points]

Q1: Write down the formula for the K-nearest neighbour regression curve estimate at a covariate value x_0, and explain your notation.

$$\hat{f}(x_0) = \frac{1}{K} \sum_{i \in \mathcal{N}_0} y_i$$

Given an integer K and a test observation x_0, the KNN regression first identifies the K points in the training data that are closest (Euclidean distance) to x_0, represented by $\mathcal{N}_0$. It then estimates the regression curve at x_0 as the average of the response values for the training observations in $\mathcal{N}_0$.

Common mistakes in the exam papers: many answered how to do KNN-classification. In addition some wrote either x_i or $f(x_i)$ in place of y_i in the formula above.

Q2: Match panels A–D to values of K (3,15,50,100). Justify your choice.

- A: $K = 100$ since we see that $\hat{f}(z_0)$ is the average over all the 100 training observations.
- B: $K = 3$ since this is the most variable curve, meaning the average (over the 3 neighbour observations) change the most.
- C: $K = 50$ since the first and last part of the curve is the average over 50 observations, and in the middle the average changes smoothly with how the 50 neighbourhood change.
- D: $K = 15$ since this is less rapid changing than B, and more rapid than C.

No common mistakes here.

Q3: Explain how 5-fold is performed, and specify which error measure you would use. Your answer should include a formula to specify how the validation error is calculated. A drawing would also be appreciated.

We look at a set of possible values for K, for example $K = (1, 2, 3, 5, 7, 9, 15, 25, 50, 100)$.

First we devide the data into a training set and a test set - and lock away the test set for model evaluation.

We work now with the training set.

5-fold CV: we divide the training data randomly into 5 folds of size $n/5$ each and call the folds $j = 1$, to $j = 5$.

For each value of K we do the following.

For $j = 1, \ldots, 5$:

- use the $4n/5$ observations from the folds except fold j to define the K-neighbourhood $\mathcal{N}_0$ for each of the observations in the jth fold
- the observations in the jth fold is left out and is the validation set, there are $n/5$ observations - and we denote them (x_{0jl}, y_{0jl}),
- we then estimate $\hat{f}(x_{0jl}) = \frac{1}{K} \sum_{i \in \mathcal{N}_0} y_i$.

- We calculate the error in the jth fold of the validation set as $\sum_l (y_{0jl} - \hat{f}(x_{0jl}))^2$ where the j is for the validation fold

The total error on the validation set is thus the validation MSE=$\frac{1}{n}\sum_{j=1}^{5}\sum_l (y_{0jl} - \hat{f}(x_{0jl}))^2$.

So, for each value of K we get an estimate of the validation MSE. Finally, we choose the value of K that gives the lowest validation MSE.

Common mistakes: many forget to shuffle data. Some forget to explain how this is related to choosing the number of neighbours in KNN (only focus on the 5-fold CV process) - that is, relate this to the setting of the problem. A few mention the error measure for classification.

Q4: In your opinion, would you prefer the leave-one-out cross-validation or the 5-fold cross-validation method? Justify your answer.

The LOOCV would give a less biased estimate of the MSE on a test set, since the sample size used $(n-1)$ is close to the sample size to be used in the real world situation (n), but the LOOCV will be variable (the variance of the validation MSE is high).

On the other hand the 5-fold CV would give a biased estimate of the MSE on a test set since the sample size used is 4/5 of the sample size that would be used in the real world situation. However, the variance will be lower than for the LOOCV.

When it comes to computational complexity we have in general that with LOOCV we need to fit n models but with 5-fold only 5. However, for KNN-regression there is really no model fitting, we only choose the K closest neighbours in the training part of the data. This means that LOOCV should not be more time consuming than 5-fold for KNN.

There is no "true" answer here – and arguments for both solutions can be given.

Common mistakes: many did not talk about bias, variance and computational complexity, a few made mistakes on the comparison of LOOCV and 5-fold CV with respect to bias and variance.

Problem 2: An important decomposition in regression

[10 points]

Q5: Write down the definition of the expected test mean squared error (MSE) at x_0.

The *expected test mean squared error (MSE) at* x_0 is defined as:

$$\mathrm{E}[Y - \hat{f}(x_0)]^2$$

Q6: Derive the decomposition of the expected test MSE into three terms.

Useful rule: $\mathrm{Var}[Y] = \mathrm{E}[Y^2] - \mathrm{E}[Y]^2$.

$$\begin{aligned}\mathrm{E}[Y - \hat{f}(x_0)]^2 &= \mathrm{E}[Y^2 + \hat{f}(x_0)^2 - 2Y\hat{f}(x_0)] \\ &= \mathrm{E}[Y^2] + \mathrm{E}[\hat{f}(x_0)^2] - \mathrm{E}[2Y\hat{f}(x_0)]\end{aligned}$$

The new Y is independent of the estimated $\hat{f}(x_0)$.

$$= \mathrm{Var}[Y] + \mathrm{E}[Y]^2 + \mathrm{Var}[\hat{f}(x_0)] + \mathrm{E}[\hat{f}(x_0)]^2 - 2\mathrm{E}[Y]\mathrm{E}[\hat{f}(x_0)]$$

Now, $\mathrm{E}[Y] = f(x_0)$.

$$\begin{aligned}&= \mathrm{Var}[Y] + f(x_0)^2 + \mathrm{Var}[\hat{f}(x_0)] + \mathrm{E}[\hat{f}(x_0)]^2 - 2f(x_0)\mathrm{E}[\hat{f}(x_0)] \\ &= \mathrm{Var}[Y] + \mathrm{Var}[\hat{f}(x_0)] + (f(x_0) - \mathrm{E}[\hat{f}(x_0)])^2\end{aligned}$$

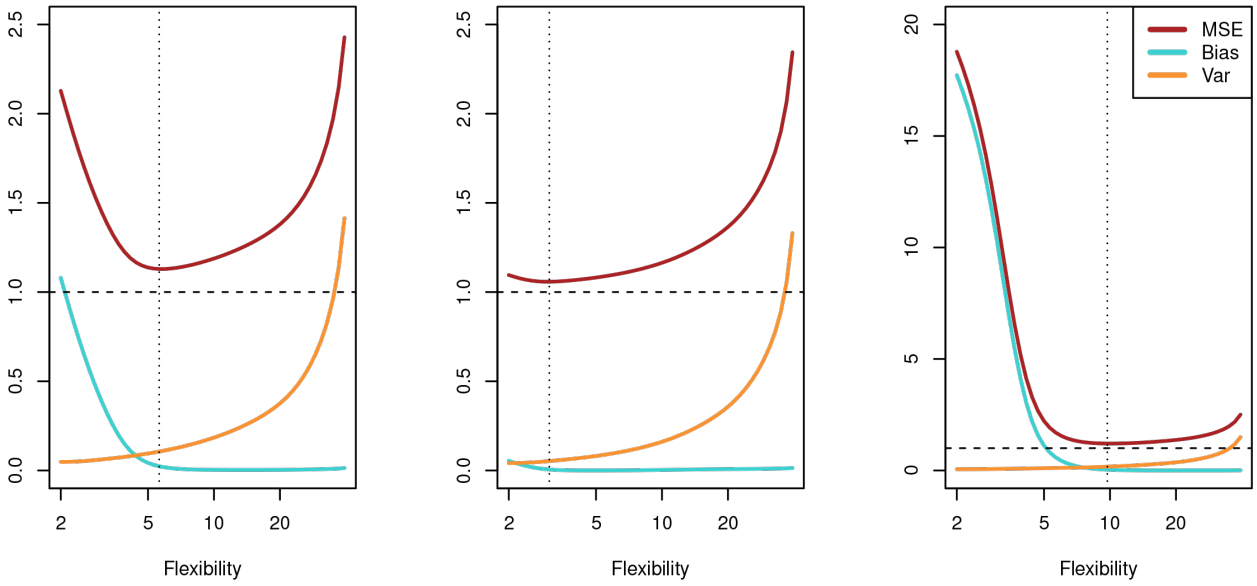


Figure 1: ISLR Figure 2.12

Finally $\text{Var}[Y] = \text{Var}(\varepsilon) = \sigma^2$.

$$= \text{Var}(\varepsilon) + \text{Var}[\hat{f}(x_0)] + [\text{Bias}(\hat{f}(x_0))]^2.$$

Therefore:

$$\text{E}[(Y - \hat{f}(x_0))^2] = \text{Var}(\varepsilon) + \text{Var}[\hat{f}(x_0)] + [\text{Bias}(\hat{f}(x_0))]^2$$

Common mistakes: Confusion on what is $\text{E}[Y]$, and different definisions of what is the bias. Since the problme said to "derive" points were deducted if the student skipped important steps in the derivation. This derivation had been given on many problem sets during the course.

Q7: Explain with words how we can interpret the three terms.

$$\text{E}[(Y - \hat{f}(x_0))^2] = \text{Var}(\varepsilon) + \text{Var}[\hat{f}(x_0)] + [\text{Bias}(\hat{f}(x_0))]^2$$

- First term: irreducible error, σ^2 and is always present unless we have measurements without error. This term cannot be reduced regardless how well our statistical model fits the data.
- Second term: variance of the prediction at x_0 or the expected deviation around the mean at x_0. If the variance is high, there is large uncertainty associated with the prediction.
- Third term: squared bias. The bias gives an estimate of how much the prediction differs from the true mean $f(x_0)$. If the bias is low the model gives a prediction which is close to the true value.

Assume that we have a method to estimate the regression function, where this method has a tuning parameter that controls the complexity of the model and that a large value of the tuning parameter gives high model complexity.

Q8: Make a sketch of how the expected test MSE (at x_0) and the decomposition into three terms could look as a function of the tuning parameter.

The figures below are from our textbook and for the expected test MSE averaged over all values of x_0, but for a given x_0 this might look similar.

This decomposition has played a central role in our course.

Common mistakes: not so many, but some swapped the bias and variance curves.

Q9: In your opinion, what is the most important implication of this decomposition? Answer with *only one* sentence.

A too complex model will give a small bias but high variance– so we cannot win by overfittng!

Problem 3: LDA

[10 points]

Given: The multivariate normal distribution function:

$$f(\mathbf{x}) = \frac{1}{(2\pi)^{p/2}|\mathbf{\Sigma}|^{1/2}} \exp(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\mathbf{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}))$$

Q10: Model assumptions for LDA:

We have two classes labelled 0 and 1, and denote the class conditional densities $f_0(\mathbf{x})$ and $f_1(\mathbf{x})$.

- For each class we assume that the class conditional distribution $f_k(\mathbf{x})$ $(k = 0, 1)$ is multivariate normal with means $\boldsymbol{\mu}_0$ and $\boldsymbol{\mu}_1$.
- We assume that both classes have the same $p \times p$ variance-covariance matrix $\mathbf{\Sigma}$.
- The prior class probability for each class is π_0 for class 0 and π_1 for class 1, and $\pi_0 + \pi_1 = 1$.

Common mistakes: missing to say that the class conditional distributions are multivariate normal. Very few mentioned the existence of prior class probabilities. The model does not involve any parameter estimation.

Q11: LDA is one of the methods that is built upon the sampling paradigm. To use LDA to classify an new observation we want to classify to the class with the highest posterior probability (0 or 1), using the Bayes rule

$$\Pr(Y = k|\mathbf{X} = \mathbf{x}) = \frac{\Pr(Y = k \cap \mathbf{X} = \mathbf{x})}{f(\mathbf{x})} = \frac{f_k(\mathbf{x})\pi_k}{f(\mathbf{x})} = \frac{\pi_k f_k(\mathbf{x})}{\pi_0 f_0(\mathbf{x}) + \pi_1 f_1(\mathbf{x})}.$$

Common mistakes: not so many, but some miss the denominator in the Bayes rule.

Q12: The class boundary is found by setting

$$\Pr(Y = 0|X = \mathbf{x}) = \Pr(Y = 1|X = \mathbf{x}).$$

Then we insert the formula for the posterior class probabilities.

$$\frac{\pi_0 f_0(\mathbf{x})}{\pi_0 f_0(\mathbf{x}) + \pi_1 f_1(\mathbf{x})} = \frac{\pi_1 f_1(\mathbf{x})}{\pi_0 f_0(\mathbf{x}) + \pi_1 f_1(\mathbf{x})}$$

The denominators cancel.

$$\pi_0 f_0(\mathbf{x}) = \pi_1 f_1(\mathbf{x})$$

We insert the formula for the multivariate normal density for $f_k(\mathbf{x})$.

$$\pi_0 \frac{1}{(2\pi)^{p/2}|\mathbf{\Sigma}|^{1/2}} e^{\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_0)^T\mathbf{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_0)} = \pi_1 \frac{1}{(2\pi)^{p/2}|\mathbf{\Sigma}|^{1/2}} e^{\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_1)^T\mathbf{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_1)}$$

More terms cancel and we take natural logs (yes ln).

$$\log(\pi_0) - \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_0)^T\mathbf{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_0) = \log(\pi_1) - \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_1)^T\mathbf{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_1)$$

We work on the quadratic terms.

$$\log(\pi_0) - \frac{1}{2}x^T\mathbf{\Sigma}^{-1}\mathbf{x} + \mathbf{x}^T\mathbf{\Sigma}^{-1}\boldsymbol{\mu}_0 - \frac{1}{2}\boldsymbol{\mu}_0^T\mathbf{\Sigma}^{-1}\boldsymbol{\mu}_0 = \log(\pi_1) - \frac{1}{2}\mathbf{x}^T\mathbf{\Sigma}^{-1}\mathbf{x} + \mathbf{x}^T\mathbf{\Sigma}^{-1}\boldsymbol{\mu}_1 - \frac{1}{2}\boldsymbol{\mu}_1^T\mathbf{\Sigma}^{-1}\boldsymbol{\mu}_1$$

Quadratic terms in $\mathbf{x}$ cancel.

$$\log(\pi_0) + \mathbf{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_0 - \frac{1}{2}\boldsymbol{\mu}_0^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_0 = \log(\pi_1) + \mathbf{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 - \frac{1}{2}\mu_1^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1$$

The result is linear in $\mathbf{x}$

$$(\log\pi_0 - \log\pi_1) - \frac{1}{2}(\boldsymbol{\mu}_0^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_0 + \boldsymbol{\mu}_1^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1) + \mathbf{x}^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1) = 0$$

Common mistakes: not performing the derivation, stopping before we get rid of the quadratic term in $\mathbf{x}$, assuming $p = 2$, assuming the covariance matrix is diagonal. Since the problem said to "derive the mathematical formula for the class boundary" students got points deducted for skipping important steps in the derivation.

Q13: The class boundary is linear in the covariate space.

Problem 4: Classification of diabetes cases

```
library(MASS)
#str(Pima.tr)
Pima.tr$diabetes=as.numeric(Pima.tr$type)-1
Pima.te$diabetes=as.numeric(Pima.te$type)-1
train=Pima.tr[,c(1:7,9)]
test=Pima.te[,c(1:7,9)]
colnames(test)=colnames(Pima.te)[c(1:7,9)]
#table(Pima.tr$diabetes)
#table(Pima.te$diabetes)
```

```
fitlogist=glm(diabetes~npreg+glu+bp+skin+bmi+ped+age,data=train,family=binomial(link="logit"))
summary(fitlogist)
```

```
##
## Call:
## glm(formula = diabetes ~ npreg + glu + bp + skin + bmi + ped +
##     age, family = binomial(link = "logit"), data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9830  -0.6773  -0.3681   0.6439   2.3154
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -9.773062   1.770386  -5.520 3.38e-08 ***
## npreg        0.103183   0.064694   1.595  0.11073
## glu          0.032117   0.006787   4.732 2.22e-06 ***
## bp          -0.004768   0.018541  -0.257  0.79707
## skin        -0.001917   0.022500  -0.085  0.93211
## bmi          0.083624   0.042827   1.953  0.05087 .
## ped          1.820410   0.665514   2.735  0.00623 **
## age          0.041184   0.022091   1.864  0.06228 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
```

```
##
##     Null deviance: 256.41  on 199  degrees of freedom
## Residual deviance: 178.39  on 192  degrees of freedom
## AIC: 194.39
##
## Number of Fisher Scoring iterations: 5
```

```
exp(fitlogist$coefficients)
```

```
##  (Intercept)        npreg          glu           bp         skin
## 5.696568e-05 1.108695e+00 1.032638e+00 9.952438e-01 9.980852e-01
##          bmi          ped          age
## 1.087220e+00 6.174392e+00 1.042043e+00
```

a) Logistic regression

Q14:

$$Y_i = \begin{cases} 1 \text{ with probability } p_i, \\ 0 \text{ with probability } 1 - p_i. \end{cases}$$

where $Y_i = 1$ denotes diabetes and $Y_i = 0$ not diabetes.

In logistic regression we *link* together our covariates $\mathbf{x}_i$ with this probability p_i using a *logistic function*.

We have one intercept and 7 covariates in our model, then the probability of diabetes for person i with covariate vector $\mathbf{x}_i$ is given as

$$p_i = \frac{e^{\beta_0+\beta_1 x_{1i}+\cdots+\beta_7 x_{7i}}}{1 + e^{\beta_0+\beta_1 x_i+\cdots+\beta_7 x_{7i}}}.$$

Alternatively we may write this as

$$\log \frac{p_i}{1-p_i} = \beta_0 + \beta_1 x_{1i} + \cdots + \beta_7 x_{7i}$$

In our model we have estimated the 8 regression parameters, and with the parameter estimates we have

$$\log \frac{\hat{p}_i}{1-\hat{p}_i} = -9.8 + 0.1 x_{\texttt{npreg}i} + 0.03 x_{\texttt{glu}i} - 0.004 x_{\texttt{bp}i} - 0.002 x_{\texttt{skin}i} + 0.08 x_{\texttt{bmi}i} + 1.82 x_{\texttt{ped}i} + 0.04 x_{\texttt{age}i}$$

Significant covariates are `glu` and `ped`.

Common mistakes: Not understanding the difference between the model and parameter estimation, not connecting the p_i and the Y_i, not connecting correctly the p_i and the x_is, believing that we do least squares and assuming that there is a $+\varepsilon_i$ in the equation for $\text{logit}(p_i)$.

Q15: How to explain the effect of `ped` on estimated probability of getting diabetes.

This is done indirectly, by looking at the odds of getting diabetes, which is the ratio between the probability of getting diabetes and the probability of not getting diabetes. The odds of getting diabetes is multiplied with $\exp \beta_{\texttt{ped}} = 6.17$ if `ped` increases with one unit.

Common mistake: interpreting β-effect on Y_i or p_i.

Q16: Would you predict that a person with the following characteristics has diabetes? Make the assumptions you need to make.

Person: $\texttt{npreg} = 2, \texttt{glu} = 145, \texttt{bp} = 85, \texttt{skin} = 35, \texttt{bmi} = 37, \texttt{ped} = 0.7, \texttt{age} = 40$.

Assumptions: When classify as diabetes? When the probability of diabetes is larger than 0.5? Rather natural choice, but the student may choose otherwise.

We know that we estimate the probability as

$$\hat{p}_i = \frac{e^{\hat{\beta}_0+\hat{\beta}_1 x_{1i}+\cdots+\hat{\beta}_7 x_{7i}}}{1+e^{\hat{\beta}_0+\hat{\beta}_1 x_i+\cdots+\hat{\beta}_7 x_{7i}}}.$$

We first calculate the estimated linear predictor $\hat{\eta} = \hat{\beta}_0 + \hat{\beta}_1 x_{1i} + \cdots + \hat{\beta}_7 x_{7i}$ inserted the estimated coefficents and the value for the covariates.

$$\hat{\eta} = -9.77 + 0.103 \cdot 2 + 0.03 \cdot 145 - 0.0048 \cdot 85 - 0.002 \cdot 35 + 0.08 \cdot 37 + 1.82 \cdot 0.7 + 0.04 \cdot 40$$

- Calulated: $\hat{\eta}$= 0.628
- Probability of diabetes: 0.6520358

Remark: big effect on linear predictor on rounding! (if less decimals were included)

Not rounding (using objects from R directly):

- Linear predictor: gives 0.6336346
- Probability: 0.6533131

```
news=data.frame("npreg"=2,"glu"=145,"bp"=85,"skin"=35,"bmi"=37,"ped"=0.7,"age"=40)
predict(fitlogist,newdata=news,type="response")
```

```
##         1
## 0.6533131
```

Also ok: say that not include not significant covariates – to make the calculations simpler – but then the logistic regression should have been refitted.... and that gives a rather different probability:

```
small=glm(diabetes~glu+ped,data=train,family=binomial(link="logit"))
summary(small)
```

```
##
## Call:
## glm(formula = diabetes ~ glu + ped, family = binomial(link = "logit"),
##     data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1273  -0.7548  -0.4829   0.7548   2.1946
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -6.400924   0.948689  -6.747 1.51e-11 ***
## glu          0.038770   0.006475   5.988 2.12e-09 ***
## ped          1.635204   0.601098   2.720  0.00652 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 256.41  on 199  degrees of freedom
## Residual deviance: 199.26  on 197  degrees of freedom
## AIC: 205.26
##
## Number of Fisher Scoring iterations: 4
```

```
eta=sum(small$coeff*c(1,145,0.7))
eta
```

```
## [1] 0.3654004
```

```
exp(eta)/(1+exp(eta))
```

```
## [1] 0.5903471
```

Common mistakes: none.

b) Neural network for logistic regression and ROC

Q17: Feed-forward network

We need 7 input notes and one bias connection (for the intercept), then we don't have a hidden layer and connect the input nodes directly to the output layer. The output layer has only one node, for the diabetes probability. We use the same logistic function as for the logistic regression, and for neural networks the logistic function is called the *sigmoid activation function*.

On the arrows below we would have written β_0 from B1 to O1, and β_1 from I1 to O1, β_2 from I2 to O1, etc. up to β_7 from I7 (age) to O1 (diabetes).

```
library(nnet)
library(NeuralNetTools)
fitnnet=nnet(diabetes~npreg+glu+bp+skin+bmi+ped+age,
             data=train,linout=FALSE,size=0,skip=TRUE,maxit=1000,
             entropy=TRUE,Wts=fitlogist$coefficients)
```

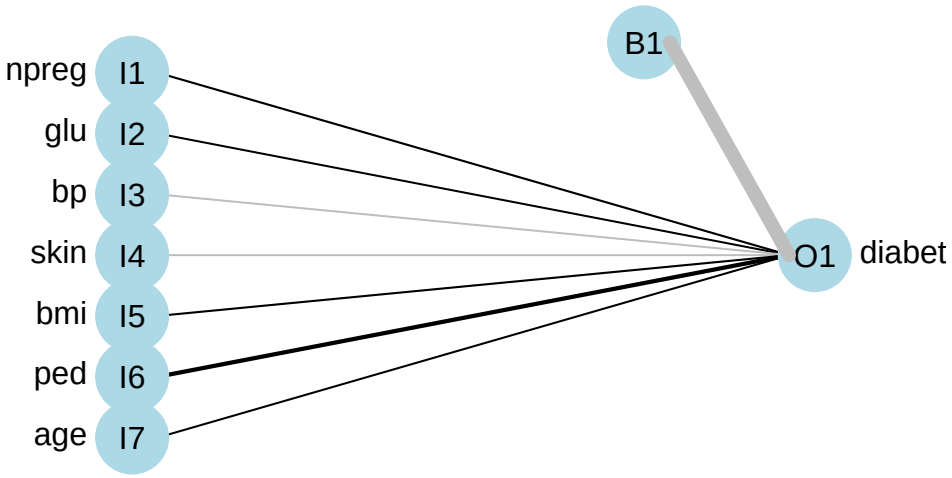
```
## # weights:  8
## initial  value 89.195333
## final  value 89.195333
## converged
```

```
# NB default is one hidden layer but avoided with size=0, but then skip=TRUE is needed,
# and if not entropy=TRUE then default is least squares
cbind(fitnnet$wts,fitlogist$coefficients)
```

```
##                      [,1]         [,2]
## (Intercept) -9.773061533 -9.773061533
## npreg        0.103183427  0.103183427
## glu          0.032116823  0.032116823
## bp          -0.004767542 -0.004767542
## skin        -0.001916632 -0.001916632
## bmi          0.083623912  0.083623912
## ped          1.820410367  1.820410367
## age          0.041183529  0.041183529
```

```
plotnet(fitnnet)
```

```
## Warning in plotnet.default(mod_in, x_names = x_names, y_names = y_names, :
## No hidden layer, plotting skip layer only with bias connections
```



Common mistakes: including hidden layers, including a summation layer with only one connection to the output layer, making weights into nodes. Using softmax instead of sigmoid. Not knowing that sigmoid and logistic function is the same.

```
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

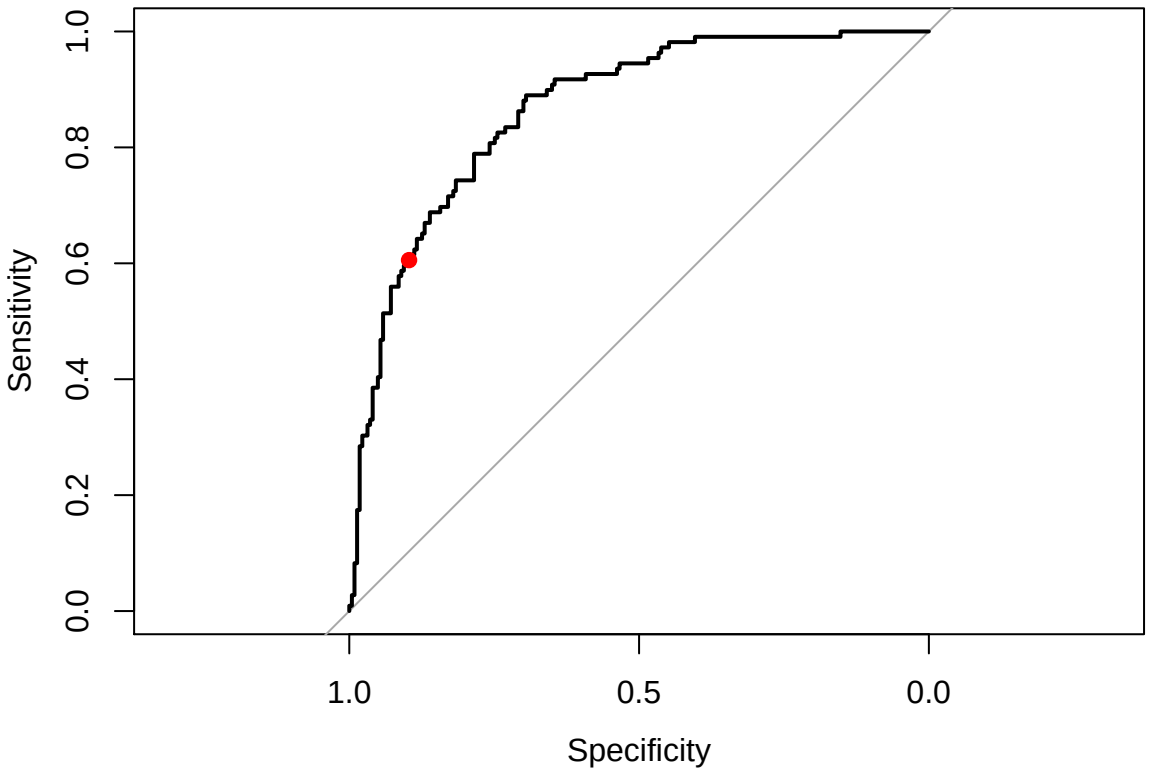
```
predlogist=predict(fitlogist,newdata=test,type="response")
testclasslogist=ifelse(predlogist > 0.5, 1, 0)
table(test$diabetes, testclasslogist)# 223 non-diabetes and 109 diabetes cases
```

```
##    testclasslogist
##       0   1
##   0 200  23
##   1  43  66
```

```
roclogist=roc(test$diabetes,predlogist)# for the logistic regression
auc(roclogist)
```

```
## Area under the curve: 0.8659
```

```
plot(roclogist, lty="solid")
points(200/223,66/109,pch=19,col=2)
```



Q18: How is a ROC curve constructed?

- We have a prediction method, and use this to predict the probability of disease for all observations in the test set.
- We choose a cut-off on the probability of disease, and construct the confusion matrix
- from the confusion matrix calculate the sensitivity and the specificity.
- The sensitivity is defined as the proportion of correctly classified positive observations (among the true disease how many correctly classified)
- The specificity is defined as the proportion of correctly classified negative observations (among the true non-disease how many correctly classified)
- These two are plotted on a graph, with 1-specificity on the horisontal axis and sensitivity on the vertical axis.
- A good ROC plot is hugging the top left corner – it performs well on all cut-off values for the probability of disease.

Common mistakes: not really mistakes, but lack of information. Some did not include information on how to construct the confusion matrix, how to calculate sensitifity and specificity from the confusion matrix, not clear what we make cut-offs on (probabilites), not clear how different cut-offs lead to one curve.

Q19:

We have observed

- 223 true non-diseases and of these 200 is correctly classified: specificity=0.896861.
- 109 true disease cases and of these 66 is correctly classified: sensitivity=0.6055046.

And the misclassification rate is $(43 + 23)/332 = 0.199$.

Common mistakes: not common, but some sum over columns instead of rows.

c) Bagging and OOB

Q20: Explain how we build a bagged set of trees, and why we would want to fit more than one tree.

Bagging is short for *bootstrap aggregation*, and to do that we resample our training data (with replacement) B times and for each bootstrapped data set we fit our classification (or regression) method and produce B predictors. We may use the average of all the B predictors as our new bagged predictor. This can be done by averaging the estimated probability of th diabetes class.

$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{*b}(x).$$

For a classification tree we may also record the predicted class (for a given observation x) for each of the B trees and use the most occurring classification (majority vote) as the final prediction.

Why many: Classification trees are in general found to have inferior prediction abilities compared to other classification methods like logistic regression. They are very sensitive to small changes in the data. A full tree might have low bias but high variance (bias-variance trade-off due when high complexity). The strength of classification trees is the interpretability and ease in taking interactions into account. With bagging we grow many trees and then take the average prections (from bootstrapped data) - to get rid of the non-robustness and high variance. The average is usually taken over the estimated probability of diabetes for each tree, but majority voting (each tree has one vote for the true class) can also be used. The reason why we get smaller variance for the average of many trees is that the variance of an average is less than the variance of one tree (if not all trees are perfectly correlated).

Common mistakes: not really mistakes, but lack of information and explanation, specifically in explaining what is bootstrapping and why we 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.

- The probability of drawing observation 1 out of n in one drawing is $\frac{1}{n}$.
- The probability of not drawing observation 1 is then $1 - \frac{1}{n}$.
- We make n independent drawings, and in each drawing the probability of not getting observation 1 in this boostrap sample is $(1 - \frac{1}{n})^n$.
- The probability of getting observation 1 in this bootstrap sample (at least one time) is then $1 - (1 - \frac{1}{n})^n$
- When $n \to \infty$ this probability goes to $1 - e^{-1} = 0.632$.

Common mistakes: mainly that there is no explanation on how to arrive at the final formula. Answers are expected to be justified and contain relevant calculations.

Q22: What is an OOB sample? An OOB sample is the set of observations that were not chosen to be in a specific boostrap sample. On average $1 - 0.632 = 0.368$ of the observations are in an OOB sample.

Common mistakes: not many, but some say that the bootstrap sample is the OOB sample.

Q23: Use the results in the figure to compare the error rates on the OOB sample to the error rates for the `test` data set and comment on your findings.

- OOB error estimation gave a misclassification rate of $(24 + 31)/200 = 0.275$.
- The test set had a misclassification rate of $(39 + 42)/332 = 0.244$.

These are not that different. This means that also we observe that the OOB error is a nice estimator for the test set error.

Common mistakes: not really mistakes, but some calculate error rates for each class, and some spend much time on trying to explain why the numerical values are different.

Q24: Which of the three methods would you recommend to use for predicting diabetes status?

The logistic regression performs better than the two other methods when we look at the full ROC curve, and the AUC is 0.87 compared to 0.81 for bagged trees and 0.80 for QDA. We want the AUC to be high because that would reflect that high values of the sensitifity and high values of specificity will give good performance of the classifier for any cut-off.

For a cut-off on probability of 0.5 the mislassification rate for the logistic regression is $(23 + 43)/332 = 0.20$ compared to 0.244 for the bagged trees and $(47 + 29)/332 = 0.229$ for QDA, so the differences between the methods are not big. In addition the logistic regression is easy to interpret (however, we could work a bit on model selection, but that is beyond the scope here).

Conclusion: logistic regression is preferred.

Common mistakes: not really, but many fail to explain why they want to have a high value for the AUC.

```
# bagged trees with call to randomForest
library(randomForest)
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
library(pROC)
set.seed(4268)
rf=randomForest(factor(diabetes)~npreg+glu+bp+skin+bmi+ped+age,
                data=train,
                mtry=7,ntree=1000,importance=TRUE)
rf$confusion #error rates based on OOB data
```

```
##     0  1 class.error
## 0 108 24   0.1818182
## 1  31 37   0.4558824
```

```
yrf=predict(rf,newdata=test)
table(test$diabetes, yrf)# 223 non-diabetes and 109 diabetes cases
```

```
##    yrf
##       0   1
##   0 184  39
##   1  42  67
```

```
predrf = predict(rf,test, type = "prob")
rocrf=roc(test$diabetes, predrf[,2])
auc(rocrf)
```

```
## Area under the curve: 0.8094
```

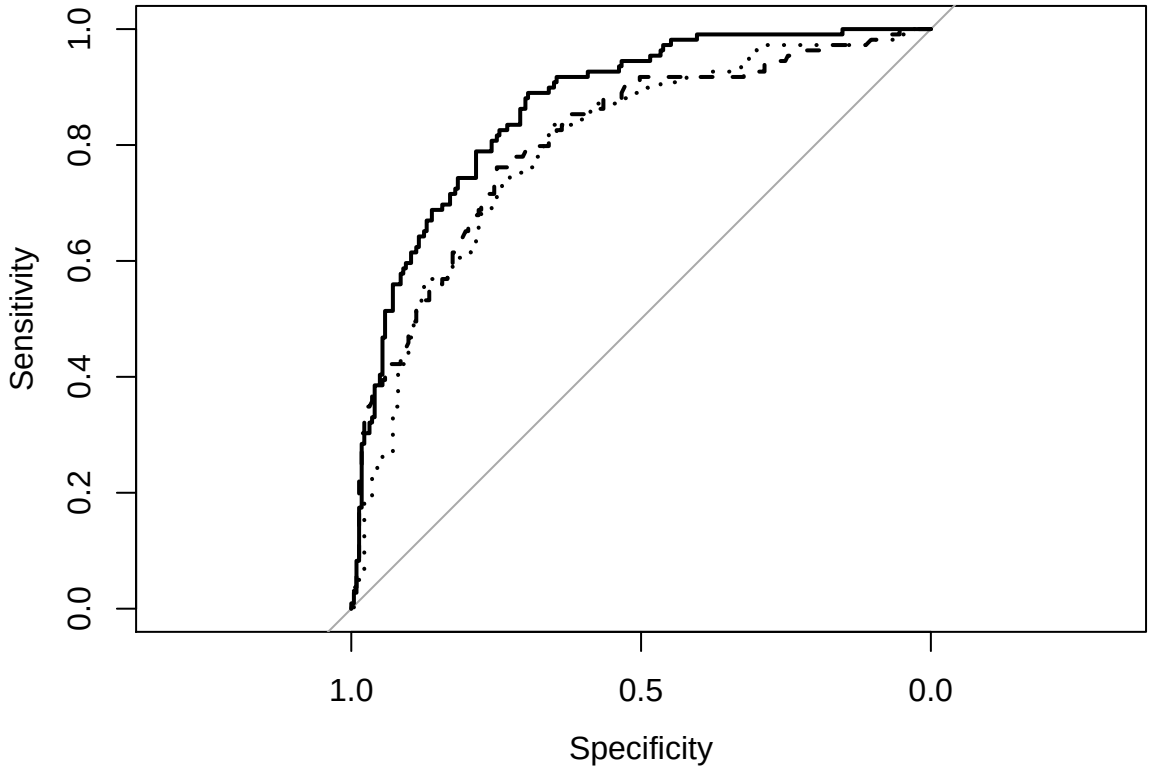
```
# fitting and ROC for QDA
fitqda=qda(diabetes~npreg+glu+bp+skin+bmi+ped+age,data=train)
predqda = predict(fitqda,newdata=test)
testclassqda=ifelse(predqda$posterior[,2] > 0.5, 1, 0)
table(test$diabetes, testclassqda)
```

```
##    testclassqda
##       0   1
##   0 194  29
##   1  47  62
```

```
rocqda=roc(test$diabetes,predqda$posterior[,2])
auc(rocqda)
```

```
## Area under the curve: 0.7962
```

```
# plotting all tree methods with ROC
plot(roclogist) #logistic regression solid line
plot(rocrf,add=TRUE,lty="dashed") #bagged trees dashed
plot(rocqda,add=TRUE,lty="dotted") #QDA dotted line
```



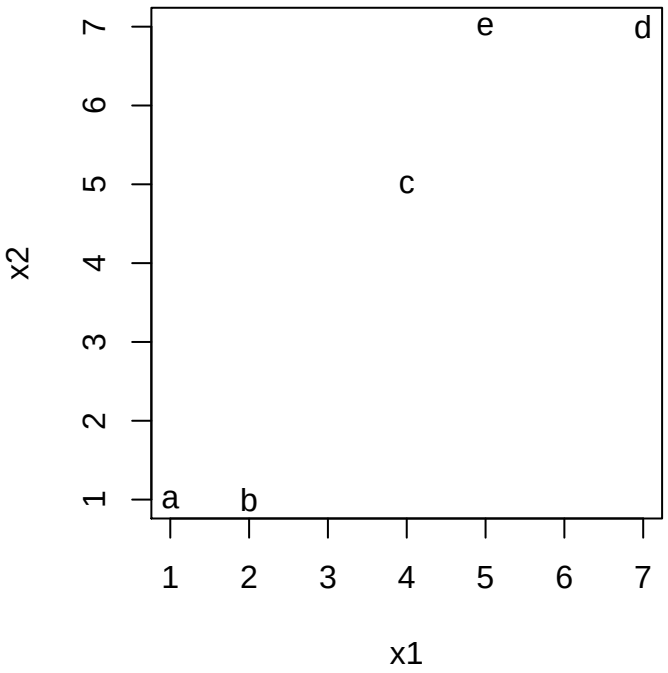
Problem 5: Clustering

[10 points]

```
library(kableExtra)
par(pty="s")
ds=rbind(c(1,1),c(2,1),c(4,5),c(7,7),c(5,7))
colnames(ds)=c("x1","x2")
rownames(ds)=letters[1:5]
print(ds)
```

```
##   x1 x2
## a  1  1
## b  2  1
## c  4  5
## d  7  7
## e  5  7
```

```
# ds%>%
#   knitr::kable("html") %>%
#   kable_styling()
plot(ds[,1],ds[,2],type="n",xlab="x1",ylab="x2")
text(ds[,1],ds[,2],labels=c("a","b","c","d","e"))
```



Q25: There are three missing entries, labelled with a question mark, in the Euclidean distance matrix. Calculate the missing entries.

Distance between

- a and b: $\sqrt{(1-2)^2+(1-1)^2}=1$.
- b and d: $\sqrt{(2-7)^2+(1-7)^2}=\sqrt{25+36}=7.8$.
- d and e: $\sqrt{(7-5)^2+(7-7)^2}=2$.

Which give the full distance matrix

```
Edist=dist(ds, method = "euclidean", diag = FALSE, upper = FALSE)
print(Edist)
```

```
##          a        b        c        d
## b 1.000000
## c 5.000000 4.472136
## d 8.485281 7.810250 3.605551
## e 7.211103 6.708204 2.236068 2.000000
```

Common mistakes: none, this is easy for the students!

Q26: Perform hierarchical clustering with complete linkage and draw the resulting dendrogram.

Using complete linkage we define the distance between clusters as the largest pairwise distance between any element of the two clusters.

1. First we "marry" a and b, since they have distance 1 (the smallest – will be the height in our dendrogram). Then we need to calculate a new distance matrix for the 4 units – that is we need the distance from the new ab cluster to c, d and e. This gives:

```
mat1=matrix(c(0,5,8.5,7.2,0,0,3.6,2.2,0,0,0,2,0,0,0,0),ncol=4)
colnames(mat1)=rownames(mat1)=c("ab","c","d","e")
print(as.dist(mat1))
```

```
##    ab   c   d
## c 5.0
```

```
## d 8.5 3.6
## e 7.2 2.2 2.0
```

2. Then we observe that the smallest distance is between d and e, and marry these – the height of the dendrogram here will be 2. Then we need to calculate a new distance matrix for the 3 units – we need the distance from de to ab and c. This gives:

```
mat2=matrix(c(0,5,8.5,0,0,3.6,0,0,0),ncol=3)
colnames(mat2)=rownames(mat2)=c("ab","c","de")
print(as.dist(mat2))
```

```
##     ab   c
## c  5.0
## de 8.5 3.6
```

3. The closest clusters are now c and de at height 3.6. Need now distance between ab and cde.

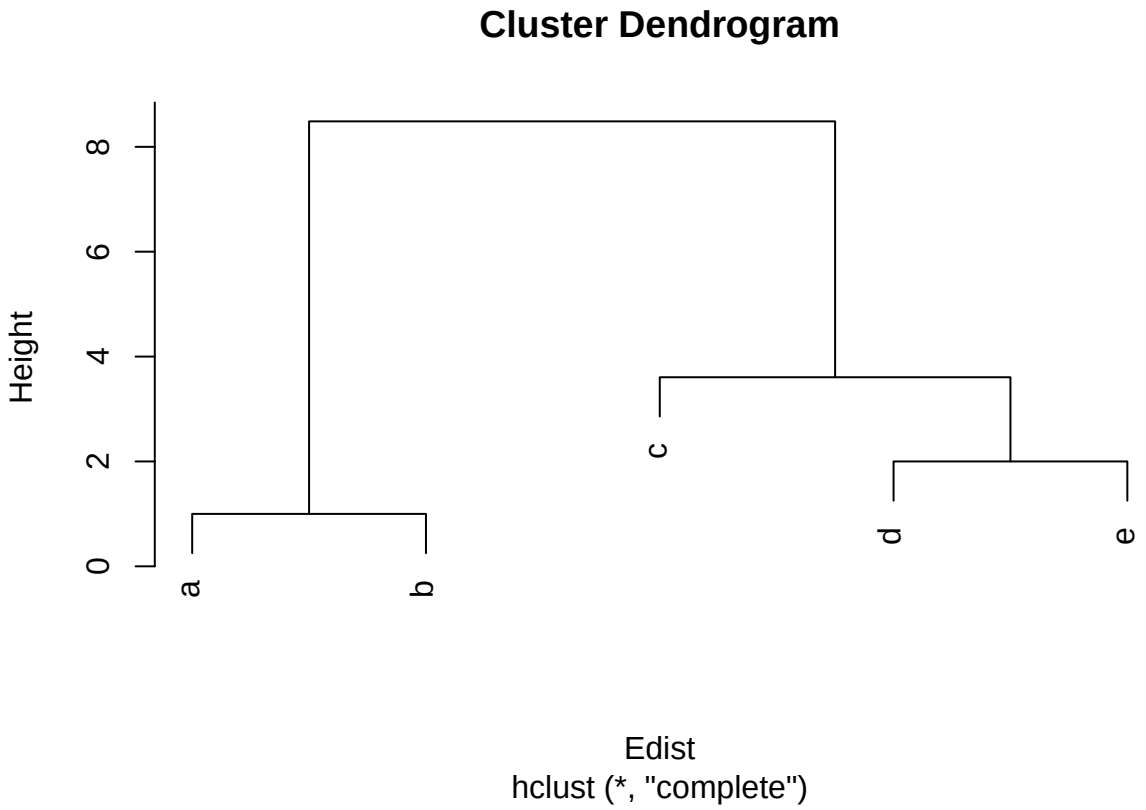
```
mat3=matrix(c(0,8.5,0,0),ncol=2)
colnames(mat3)=rownames(mat3)=c("ab","cde")
print(as.dist(mat3))
```

```
##      ab
## cde 8.5
```

4. Finally abcde all married at 8.5

This gives the dendrogram:

```
res=hclust(Edist,method="complete")
plot(res,labels=c("a","b","c","d","e"))
```



Common mistakes: many strange dendrograms, wrong fuses, upside-down, without value on the vertical axis.

And, very many have not shown how to do the distance calculations. I have not deducted points for that (even though it is clearly stated that relevant calculations should be provided). The students making wrong dendrograms and not providing distance calculations would then have to get 0 points.

Q27: Using the dendrogram assume that we want two clusters, which observations are in each cluster?

If we want two clusters we have one cluster with a and b and the other with c, d and e.

No common mistakes- all understood!

Q28: A competing clustering method is k-means. Discuss briefly two differences between hierarchical clustering and k-means clustering.

In general, with clustering we try to find homogenours subgroups among observations (or covariates).

- Need to specify k clusters for k-means – prior to starting, while hiearchical clustering just marries everybody and you choose the level you want.
- Hierarchical clustering can be presented using a dendrogram – and one single dendrogram can be used to obtain any number of clusters. Presenting the result of k-means is not so easy if the number of covariates is larger than 2.
- Hierarchical clustering is a bottom-up (agglomerative) method, and best solution for 3 groups may not be a subdivision of the solution for 2 groups – so hierarchical clustering might be bad for this situations.
- k-means: a good cluster is one where the within-cluster variation is small (compared to the between cluster variation).
- When using k-means the clusters are non-overlapping in the sense that observations have membership to the closest cluster.
- Both methods: need to specify distance measure – but more common with Euclidean or Mahloanobis for k-means.

Common mistakes: not really. But, many students say that k-means is a random method, since we need to run the algorithm many times from different random starting points. But, what we do is to look for the global optimum, and therefore need to run the algorithm several time to avoid a local minimim - but there is a global optimum that we want to achieve. So, in that sense the k-means does not produce a random result.