Examination

Linköping University, Department of Computer and Information Science, Statistics

Course code and name	732A99/732A68 Machine Learning
Date and time	2020-01-10, 14.00-19.00
Assisting teacher	Oleg Sysoev
Allowed aids	PDF of the course book + your help file (if submitted to LISAM in due time)

Grades:	A=19-20 points
	B=16-18 points
	C=11-15 points
	D=9-10 points
	E=7-8 points
	F=0-6 points

Provide a detailed report that includes plots, conclusions and interpretations. Give motivated answers to the questions. If an answer is not motivated, the points are reduced. Provide all necessary codes in the appendix.

Note: seed 12345 should be used in all codes that assumes randomness unless stated otherwise!

To start work in RStudio, type this in the Terminal application:

```
module add courses/732A99
rstudio
```

To submit your report:

1. Create one file (DOC, DOCX, ODT, PDF)
2. Use Exam Client to submit, and choose Assignment 1 in the drop box
3. Attach your report
4. Submit.
5. “Request Received” status implies that your report is successfully submitted.

Assignment 1 (10p)

The data file **olive.csv** contains information the origin and the content of various Italian olive oils, including:

- **region** Three super-classes of Italy: North, South and the island of Sardinia encoded by 3, 1 and 2 respectively.
- **area** Nine collection areas: three from North, four from South and 2 from Sardinia
- **palmitic, palmitoleic, stearic, oleic, linoleic, linolenic, arachidic, eicosenoic** fatty acids levels

1. Use the fatty acid variables to perform Principle Component Analysis and report how many principle components are enough to explain 90% variation in the data. Why is scaling necessary for these data? **(2p)**
2. Consider new data in which principal component scores for the first 3 principle components are features and the Region variable is target. Divide these data into training and test data (50/50) and compute a corresponding logistic regression model with multinom() from nnet package. Report training and test misclassification rates and comment on the quality of prediction. Finally, print the resulting model (by using print()) and report equations of decision boundaries between every pair of classes in terms of Principle Component variables. **(4p)**
    a. Note that **multinom** is modelling contrasts between all classes and first one, i.e. probabilistic equation for any class "i" is $log\frac{p(y=i|x)}{p(y=1|x)} = \theta_i^T x$
3. Compute a new target variable Y="South" if Region=1 and Y="Other" otherwise and split the original data into training and test (50/50). Use glmnet package to compute a logistic regression model with Ridge (L2) penalty by cross-validation from the training data in which Y is target and all acid variables are inputs. Report the optimal penalty factor value. Report a dependence of the cross-validation score on the penalty factor and interpret it in terms of bias-variance tradeoff. Use basic R functionality to implement a code to compute ROC curve for the test data predictions and plot it. According to the plot, how good is this classifier compared to the best possible classifier and to the random guess classifier? **(4p)**

Assignment 2 (10p)

GAUSSIAN MIXTURE MODELS – 6 POINTS

You are asked to implement the algorithm for semi-supervised learning of Gaussian mixture models. See Method 10.2 in page 212 of the course textbook for details or, alternatively, see the figure below. **Comment your code and the output obtained**.

Learn the GMM

```
Data: Partially labeled training data $\mathcal{T} = \{\{\mathbf{x}_i, y_i\}_{i=1}^{n_l}, \{\mathbf{x}_i\}_{i=n_l+1}^{n}\}$ (with output classes
      $m = 1, \dots, M$)
Result: Gaussian mixture model
1 Compute $\boldsymbol{\theta} = \{\widehat{\pi}_m, \widehat{\boldsymbol{\mu}}_m, \widehat{\boldsymbol{\Sigma}}_m\}_{m=1}^{M}$ according to (10.3), using only the labeled data $\{\mathbf{x}_i, y_i\}_{i=1}^{n_l}$
2 repeat
3     For each $\mathbf{x}_i$ in $\{\mathbf{x}_i\}_{i=n_l+1}^{n}$, compute the prediction $p(y \mid \mathbf{x}_i, \widehat{\boldsymbol{\theta}})$ according to (10.5) using the
        current parameter estimates $\widehat{\boldsymbol{\theta}}$
4     Update the parameter estimates $\widehat{\boldsymbol{\theta}} \leftarrow \{\widehat{\pi}_m, \widehat{\boldsymbol{\mu}}_m, \widehat{\boldsymbol{\Sigma}}_m\}_{m=1}^{M}$ according to (10.10)
5 until convergence
```

Predict as QDA, Method 10.1

$$p(y = m \mid \mathbf{x}_\star) = \frac{\widehat{\pi}_m \mathcal{N}\left(\mathbf{x}_\star \mid \widehat{\boldsymbol{\mu}}_m, \widehat{\boldsymbol{\Sigma}}_m\right)}{\sum_{j=1}^{M} \widehat{\pi}_j \mathcal{N}\left(\mathbf{x}_\star \mid \widehat{\boldsymbol{\mu}}_j, \widehat{\boldsymbol{\Sigma}}_j\right)}. \tag{10.5}$$

$$w_i(m) = \begin{cases} p(y = m \mid \mathbf{x}_i, \widehat{\boldsymbol{\theta}}) & \text{if } y_i \text{ is missing} \\ 1 & \text{if } y_i = m \\ 0 & \text{otherwise} \end{cases}$$

$$\widehat{\pi}_m = \frac{1}{n} \sum_{i=1}^{n} w_i(m), \tag{10.10b}$$

$$\widehat{\boldsymbol{\mu}}_m = \frac{1}{\sum_{i=1}^{n} w_i(m)} \sum_{i=1}^{n} w_i(m) \mathbf{x}_i, \tag{10.10c}$$

$$\widehat{\boldsymbol{\Sigma}}_m = \frac{1}{\sum_{i=1}^{n} w_i(m)} \sum_{i=1}^{n} w_i(m) (\mathbf{x}_i - \widehat{\boldsymbol{\mu}}_m)(\mathbf{x}_i - \widehat{\boldsymbol{\mu}}_m)^{\mathsf{T}}. \tag{10.10d}$$

Method 10.2: Semi-supervised learning of the GMM

Run the code below to produce the learning data, which consist of 300 points sampled from a Gaussian mixture model with three equally likely components, and each component is a bivariate Gaussian distribution. Assume that you know the true class of the first 10 points sampled from each of the mixture components. The true class labels of the rest of the points are unknown, hence we are dealing with semi-supervised learning. Finally, note also that we use the R package mvtnorm to manipulate Gaussian distributions. In Equation 10.5, you may want to use the function dmvnorm, which implements the density function of a Gaussian distribution.

```
install.packages("mvtnorm")

library(mvtnorm)

set.seed(1234567890)

min_change <- 0.001 # min parameter change between two consecutive iterations

N=300 # number of training points

D=2 # number of dimensions

x <- matrix(nrow=N, ncol=D) # training data
```

```
# Producing the training data

mu1<-c(0,0)

Sigma1 <- matrix(c(5,3,3,5),D,D)

dat1<-rmvnorm(n = 100, mu1, Sigma1)

mu2<-c(5,7)

Sigma2 <- matrix(c(5,-3,-3,5),D,D)

dat2<-rmvnorm(n = 100, mu2, Sigma2)

mu3<-c(8,3)

Sigma3 <- matrix(c(3,2,2,3),D,D)

dat3<-rmvnorm(n = 100, mu3, Sigma3)

plot(dat1,xlim=c(-10,15),ylim=c(-10,15))

points(dat2,col="red")

points(dat3,col="blue")

x[1:100,]<-dat1

x[101:200,]<-dat2

x[201:300,]<-dat3

plot(x,xlim=c(-10,15),ylim=c(-10,15))


K=3 # number of classes

w <- matrix(nrow=N, ncol=K) # fractional class assignments

pi <- vector(length=K) # mixing coefficients

mu <- matrix(nrow=K, ncol=D) # class conditional means

Sigma <- array(dim=c(D,D,K)) # class conditional covariances
```

KERNEL MODELS – 4 POINTS

In the course, you have learned about kernel models for classification and regression. Kernel models can also be used for density estimation, i.e. to model a probability distribution or density function p(x_*). In particular,

$$p(x_*) = \frac{1}{n}\sum_{i=1}^{n} k(\frac{x_* - x_i}{h})$$

where the kernel function k() must integrate to 1. To ensure this, you will hereinafter consider k() to be the density function of a Gaussian distribution with mean equal to 0 and standard deviation equal to 1. You can get it by using the command dnorm in R.

Run the code below to produce some learning data, which consist of 1500 samples from class 1 and 1000 samples from class 2. These points are stored in the variables data_class1 and data_class2. Implement the kernel model presented above to estimate the density function of the data sampled from class 1. Choose a kernel width h that you deem appropriate. Choose it manually and disregard overfitting issues. **Explain your choice**. Do the same for class 2. Finally, answer the following question: Once you have kernel models for p(x_*|class=1) and p(x_*|class=2), how would you use them to produce posterior class probabilities, i.e. p(class|x_*)? You don't need to implement the answer.

```
set.seed(1234567890)

N_class1 <- 1500

N_class2 <- 1000


data_class1 <- NULL

for(i in 1:N_class1){

  a <- rbinom(n = 1, size = 1, prob = 0.3)

  b <- rnorm(n = 1, mean = 15, sd = 3) * a + (1-a) * rnorm(n = 1, mean = 4, sd
= 2)

  data_class1 <- c(data_class1,b)

}


data_class2 <- NULL

for(i in 1:N_class2){

  a <- rbinom(n = 1, size = 1, prob = 0.4)

  b <- rnorm(n = 1, mean = 10, sd = 5) * a + (1-a) * rnorm(n = 1, mean = 15,
sd = 2)

  data_class2 <- c(data_class2,b)

}
```