Examination

Linköping University, Department of Computer and Information Science, Statistics

Course code and name	732A99 Machine Learning
Date and time	2019-01-16, 14.00-19.00
Assisting teacher	Oleg Sysoev
Allowed aids	"Pattern recognition and Machine Learning" by Bishop and "The Elements of Statistical learning" by Hastie
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.

Use seed 12345 when randomness is present unless specified otherwise.

Assignment 1 (10p)

The data file **Influenza.csv** contains contains the number of registered cases of influenza and mortality. The variables in the data are:

- Year, Week
- Mortality: number of mortality cases
- Influenza: number of influenza cases
- Temperature_deficit: a temperature measurement
- Influenza_lag, Temp_lag: measurements at a previous time point (t-1)
- Influenza_lag2, Temp_lag2: measurements at a time point which is two units back (t-2)

Import data to R.

1. Assume that mortality y is Poisson distributed, i.e. $p(y = Y|\lambda) = \frac{e^{-\lambda}\lambda^{Y}}{Y!}$, where $Y! = 1 \cdot 2 \cdot \ldots \cdot Y$. Write an R code computing the minus-loglikelihood of Mortality values for a given λ (use only basic R functions, do not use implemented Poisson distribution in R). Compute the minus log-likelihood values for $\lambda = 10{,}110{,}210, \ldots, 2910$ and produce a plot showing the dependence of the minus log-likelihood on the value of λ. Define the optimal value of λ by means of visual inspection (i.e. approximately**). (2p)**
2. Scale all variables except of Mortality. Divide the data randomly (50/50) into training and test sets and fit a LASSO regression with Mortality as a Poisson distributed target and all other variables as features. Select the optimal parameters in the LASSO regression by the cross-validation and report the optimal LASSO penalization parameter and also the test MSE. Is the MSE actually the best way of measuring the error rate in this case? Report also the optimal LASSO coefficients and report which variables seem to have the biggest impact on the target. Check the value of intercept α, compute $\exp(\alpha)$ and compare it with the optimal λ in step 1. Are these quantities similar and should they be? **(3p)**
3. Implement Benjamini-Hochberg (BH) method (use only basic R functions) that applies to the original data from years 1995 and 1996, doing paired t-test (check options in t.test), and in which target is Year and all other variables are features, use $\alpha = 0.05$. Provide a plot showing the hypotheses, their p-values and the rejection line. Which hypotheses are rejected and what does it mean? Why is it in fact not so important to use BH method in this case? **(2p)**
4. Perform principal component analysis using all the variables in the training data except of Mortality and report how many principal components are needed to capture more than 90% of the variation in the data. Use the coordinates of the data in the principal component space as features and fit a LASSO regression with Mortality as a Poisson distributed target by cross-validation, check penalty factors $\lambda = 0, 0.1, 0.2, \ldots, 50$. Provide a plot that shows the dependence of the cross-validation error on $\log(\lambda)$. Does complexity of the model increase when λ increases? How many features are selected by the LASSO regression? Report a probabilistic model corresponding to the optimal LASSO model. **(3p)**

Assignment 2 (10p)

NEURAL NETWORKS - 4 POINTS

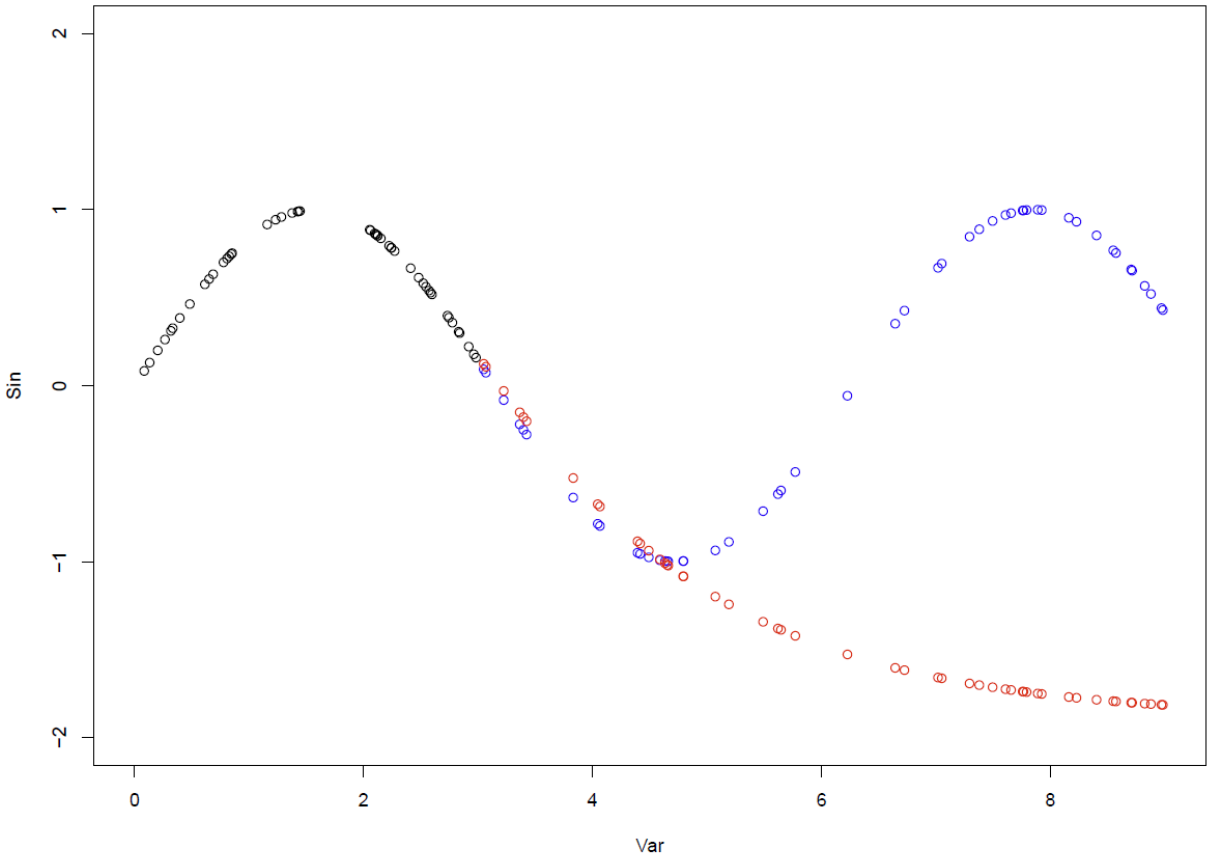
You are asked to use the function neuralnet of the R package of the same name to train a neural network (NN) to mimic the trigonometric sine function. You should run the following code to obtain the training and test data.

```
install.packages("neuralnet")
library(neuralnet)
set.seed(1234567890)
Var <- runif(50, 0, 3)
tr <- data.frame(Var, Sin=sin(Var))
Var <- runif(50, 3, 9)
te <- data.frame(Var, Sin=sin(Var))
```

(**2 p)** Produce the code to train the NN on the training data tr and test it on the data te. Use a single hidden layer with three units. Initialize the weights at random in the interval [-1,1]. Use the default values for the rest of parameters in the function neuralnet. You may need

to use the function compute. Confirm that you get results similar to the following figure. The black dots are the training data. The blue dots are the test data. The red dots are the NN predictions for the test data.

(2 p) In the previous figure, it is not surprising the poor performance on the range [3,9] because no training point falls in that interval. However, it seems that the predictions converge to -2 as the value of Var increases. Why do they converge to that particular value ? To answer this question, you may want to look into the weights of the NN learned.



MIXTURE MODELS - 6 POINTS - FOR 732A99 ONLY

You are asked to implement the EM algorithm for mixtures of multivariate Gaussian distributions. You should use the following equations in the E-step

$$p(z_{nk}|\boldsymbol{x}_n, \boldsymbol{\mu}, \boldsymbol{\pi}) = \frac{\pi_k f(\boldsymbol{x}_n|\boldsymbol{\mu}_k)}{\sum_k \pi_k f(\boldsymbol{x}_n|\boldsymbol{\mu}_k)}$$

and in the M-step

$$\pi_k^{ML} = \frac{\sum_n p(z_{nk}|\boldsymbol{x}_n, \boldsymbol{\mu}, \boldsymbol{\pi})}{N}$$

$$\boldsymbol{\mu}_k^{ML} = \frac{\sum_n \boldsymbol{x}_n p(z_{nk}|\boldsymbol{x}_n, \boldsymbol{\mu}, \boldsymbol{\pi})}{\sum_n p(z_{nk}|\boldsymbol{x}_n, \boldsymbol{\mu}, \boldsymbol{\pi})}$$

$$\boldsymbol{\Sigma}_k^{ML} = \frac{\sum_n (\boldsymbol{x}_n - \boldsymbol{\mu}_k^{ML})(\boldsymbol{x}_n - \boldsymbol{\mu}_k^{ML})^T p(z_{nk}|\boldsymbol{x}_n, \boldsymbol{\mu}, \boldsymbol{\pi})}{\sum_n p(z_{nk}|\boldsymbol{x}_n, \boldsymbol{\mu}, \boldsymbol{\pi})}$$

where f is the density function of a Gaussian distribution, which is implemented by the function dmvnorm in the R package mvtnorm. You can reuse your solution for the corresponding lab assignment. Use the following code for sampling the learning data and initializing the parameters. Note that the learning data consists of 300 points sampled from a mixture model with three equally likely components, and each component is a bivariate Gaussian distribution.

```
install.packages("mvtnorm")
library(mvtnorm)
set.seed(1234567890)
max_it <- 100 # max number of EM iterations
min_change <- 0.1 # min change in log likelihood between two consecutive EM iterations
N=300 # number of training points
D=2 # number of dimensions
x <- matrix(nrow=N, ncol=D) # training data
# Sampling the training data
mu1<-c(0,0) # component 1
Sigma1 <- matrix(c(5,3,3,5),D,D)
dat1<-rmvnorm(n = 100, mu1, Sigma1)
mu2<-c(5,7) # component 2
Sigma2 <- matrix(c(5,-3,-3,5),D,D)
dat2<-rmvnorm(n = 100, mu2, Sigma2)
mu3<-c(8,3) # component 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 guessed components
z <- matrix(nrow=N, ncol=K) # fractional component assignments
pi <- vector(length = K) # mixing coefficients
mu <- matrix(nrow=K, ncol=D) # conditional means
Sigma <- array(dim=c(D,D,K)) # conditional covariances
llik <- vector(length = max_it) # log likelihood of the EM iterations
# Random initialization of the paramters
pi <- runif(K,0,1)
pi <- pi / sum(pi)
for(k in 1:K) {
mu[k,] <- runif(D,0,5)
Sigma[,,k]<-c(1,0,0,1)
}
```

(4 p) Implement the EM algorithm described above.
(2 p) Run your code on the data provided with the true number of components, i.e. three. Show that the log likelihood increases with the number of iterations. Show also that the final parameters are close to the true ones.