Examination

Linköping University, Department of Computer and Information Science, Statistics

Course code and name	TDDE01 Machine Learning
Date and time	2022-01-14, 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:	
	5=18-20 points
	4=14-17 points
	3=10-13 points
	U=0-9 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/TDDE01
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)

File **adult.csv** shown data collected in a population census in 1994. The following metrics are available:

1. age: continuous.
2. workclass: Private, Self-emp-not-inc, etc.
3. fnlwgt: a population index.
4. education: Bachelors, Some-college, etc.
5. education-num: ordered Education variable.
6. marital-status: Married-civ-spouse, Divorced, etc.
7. occupation: Tech-support, Craft-repair, etc.
8. relationship: Wife, Own-child, etc.
9. capital-gain: continuous.(when you sell an asset and get more for it than you paid originally)
10. capital-loss: continuous.
11. hours-per-week: continuous.
12. native-country: United-States, Cambodia etc.
13. Income level

1. Read data into R with option stringsAsFactors=T and divide data randomly (60/20/20) into training, validation and test sets. Use the hold-out principle to first grow a decision tree with default settings and with Income level as target and all other variables except Native Country as inputs. Then use this tree to compute decision trees with various number of leaves and select the optimal tree according to the deviance criterion. Provide the plot of the dependence of the training and validation errors on the number of leaves and interpret this plot from the perspective of bias-variance tradeoff. Report the optimal tree and make some interpretations. Why training a logistic regression model with these features and target is not possible directly? **(4p)**
2. Use the optimal tree computed in step 1 to predict the target probabilities for the test data and estimate predictions for the decision thresholds $\pi = 0.1, 0.2, \ldots, 0.9$ in the rule $\hat{Y} = '>50K' \; if \; p(Y = '> 50K' | X) > \pi, otherwise \; \hat{Y} = '\leq 50K'$. Use these predictions to report a table showing test accuracies and test F1 scores for various π. Which threshold is the best to use for these data and why? **(3p)**
3. Assume now that hours-per-week is a target variable and x_1=age, x_2=Capital gain, x_3=Capital loss are features. Use cross-validation to estimate a LASSO model with this target and these features from the training data. Report the cross-validation error plot, the optimal penalty factor value and how many features are selected by the model. Report also an equation of the estimated LASSO model. Does the cross-validation plot suggest that having 0 features is significantly better than having 3 features? **(3p)**

Assignment 2 (10p)

KERNEL MODELS – 6 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 1000 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. Do the same for class 2. Use only 800 samples from class 1 and 800 samples from class 2.

```
set.seed(123456789)

N_class1 <- 1000

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)

}
```

Once you have kernel models for the class conditional density functions p(x_*|class=1) and p(x_*|class=2), you can use them to produce posterior class probabilities p(class|x_*) via Bayes theorem. Specifically,

p(class=1|x_*) = p(x_*|class=1) p(class=1) / [p(x_*|class=1) p(class=1) + p(x_*|class=2) p(class=2)]

Use these probabilities to compute the correct classification rate on 200 samples that you did not use before, 100 from class 1 and 100 from class 2. Use this classification rate to select the kernel width h from among the values 0.1, 0.2, …, 4.9, 5. Finally, use the 200 samples that you have not used so far to estimate the generalization error of the kernel model selected.

In summary, you should use 1600 samples to build kernel models of the class conditional density functions that you should convert into a probabilistic classifier via Bayes theorem. To select the kernel width, you should use 200 samples as validation set. Finally, you should use 200 samples to estimate the generalization error of the model selected.

NEURAL NETWORKS – 4 POINTS

Train a neural network to learn the trigonometric sine function. To do so, sample 50 points uniformly at random in the interval [0, 10]. Apply the sine function to each point. The resulting pairs are the data available to you. Use 25 of the 50 points for training and the rest for validation. The validation set is used for early stop of the gradient descent. That is, you should use the validation set to detect when to stop the gradient descent and so avoid overfitting (you can read more about early stopping in page 95 of the course textbook). Stop the gradient descent when the partial derivatives of the error function are below a given threshold value. Check the argument `threshold` in the documentation of the R package `neuralnet`. Consider threshold values i/1000 with i = 1,…, 10. Initialize the weights of the neural network to random values in the interval [−1,1]. Use a neural network with a single hidden layer of 10 units. **Use the default values for the arguments not mentioned here**. Choose the most appropriate value for the threshold. **Motivate your choice.** Provide the final neural network learned with the chosen threshold. Feel free to use the following template.

```
library(neuralnet)
set.seed(1234567890)

Var <- runif(50, 0, 10)
trva <- data.frame(Var, Sin=sin(Var))
tr <- trva[1:25,] # Training
va <- trva[26:50,] # Validation

# Random initializaiton of the weights in the interval [-1, 1]
# Your code here

for(i in 1:10) {
  nn <- neuralnet(# Your code here)

  # Your code here
}

# Your code here
```