732A99/TDDE01/732A68 MACHINE LEARNING

LAB 3 BLOCK 1: KERNEL METHODS, SUPPORT VECTOR MACHINES AND NEURAL NETWORKS

JOSE M. PEÑA
IDA, LINKÖPING UNIVERSITY, SWEDEN

INSTRUCTIONS

The instructions and submission procedure from the previous labs apply to this lab as well.

RESOURCES

The only R package that is allowed to solve the assignment 1 is the `geosphere` package (specifically, the function `distHaversine`). The assignment 2 is designed to be solved with the package `kernlab`. The assignment 3 is designed to be solved with the `neuralnet` package.

1. KERNEL METHODS

Implement a kernel method to predict the hourly temperatures for a date and place in Sweden. To do so, you are provided with the files `stations.csv` and `temps50k.csv`. These files contain information about weather stations and temperature measurements in the stations at different days and times. The data have been kindly provided by the Swedish Meteorological and Hydrological Institute (SMHI).

You are asked to provide a temperature forecast for a date and place in Sweden. The forecast should consist of the predicted temperatures from 4 am to 24 pm in an interval of 2 hours. Use a kernel that is the **sum** of three Gaussian kernels:

- The first to account for the **physical** distance from a station to the point of interest. For this purpose, use the function `distHaversine` from the R package `geosphere`.
- The second to account for the distance between the **day** a temperature measurement was made and the day of interest.
- The third to account for the distance between the **hour** of the day a temperature measurement was made and the hour of interest.

Choose an appropriate smoothing coefficient or width for each of the three kernels above. No cross-validation should be used. Instead, choose manually a width that gives large kernel values to closer points and small values to distant points. Show this with a **plot** of the kernel value as a function of distance.

Finally, repeat the exercise above by combining the three kernels into one by **multiplying** them, instead of summing them up. Compare the results obtained in both cases and elaborate on why they may differ.

Note that the file `temps50k.csv` may contain temperature measurements that are posterior to the day and hour of your forecast. You must **filter** such measurements out, i.e. they cannot be used to compute the forecast. Feel free to use the template below to solve the assignment.

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

stations <- read.csv("stations.csv")
temps <- read.csv("temps50k.csv")
st <- merge(stations,temps,by="station_number")
```

```
h_distance <- # These three values are up to the students
h_date <-
h_time <-
a <- 58.4274 # The point to predict (up to the students)
b <- 14.826
date <- "2013-11-04" # The date to predict (up to the students)
times <- c("04:00:00", "06:00:00", ..., "24:00:00")

temp <- vector(length=length(times))

# Students' code here

plot(temp, type="o")
```

2. Support Vector Machines

The code in the file `Lab3Block1_2020_SVMs.R` performs SVM model selection by using the function `ksvm` from the R package `kernlab`, in order to learn a SVM for classifying the `spam` dataset that is included with the package. All the models to select from use the radial basis function kernel (also known as Gaussian) with a width of 0.05. The C parameter varies between the models. Run the code in the file `Lab3Block1_2020_SVMs.R` and answer the questions there.

3. Neural Networks

Train a neural network to learn the trigonometric sine function. To do so, sample 500 points uniformly at random in the interval $[0, \mathbf{10}]$. Apply the sine function to each point. The resulting pairs are the data available to you. Use 25 of the 500 points for training and the rest for test. Use any number of layers and hidden units that you consider appropriate. You do not need to apply early stopping. Plot the training and test data, and the predictions of the learned NN on the test data. You should get good results. Comment your results.

Then, sample 500 points uniformly at random in the interval $[0, \mathbf{20}]$, and apply the sine function to each point. Use the **previously** learned NN to predict the sine function value for these new 500 points. You should get mixed results. Plot and comment your results.

Finally, sample 500 points uniformly at random in the interval $[0, \mathbf{10}]$, and apply the sine function to each point. Use all these points as training points for learning a NN that tries to predict **$\boldsymbol{x}$ from $\boldsymbol{sin(x)}$**, i.e. unlike before when the goals was to predict $sin(x)$ from x. You should get bad results. Plot and comment your results.

Feel free to use the following template.

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

Var <- runif(500, 0, 10)
mydata <- data.frame(Var, Sin=sin(Var))
tr <- mydata[1:25,] # Training
te <- mydata[26:500,] # Test

# Random initialization of the weights in the interval [-1, 1]
winit <- # Your code here
nn <- neuralnet(# Your code here)

# Plot of the training data (black), test data (blue), and predictions (red)

plot(tr, cex=2)
```

```
points(te, col = "blue", cex=1)
points(te[,1],predict(nn,te), col="red", cex=1)
```