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 2 is the `geosphere` package (specifically, the function `distHaversine`). The assignment 3 is designed to be solved with the package `kernlab`. The assignment 4 is designed to be solved with the `neuralnet` package.

1. THEORY

In this task, you are ask to find answers to some questions in the main course book (MLFC). The answer to each question should be up to five sentences long, and should also include the page numbers in the book where this information can be found.

- What is the kernel trick?
- In the literature, it is common to see a formulation of SVMs that makes use of a hyperparameter C. What is the purpose of this hyperparameter?
- In neural networks, what do we mean by mini-batch and epoch?

2. 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. **Help**: 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.

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.

The only R package that is allowed to solve this assignment is the `geosphere` package (specifically, the function `distHaversine`). Feel free to use the template below to solve the assignment.

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

stations <- read.csv("stations.csv", fileEncoding = "latin1")
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")
```

3. Support Vector Machines

The code in the file `Lab3Block1_2021_SVMs_St.R` performs SVM model selection to classify the `spam` dataset. To do so, the code uses the function `ksvm` from the R package `kernlab`, which also includes the `spam` dataset. All the SVM 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_2021_SVMs_St.R` and answer the following questions.

(1) Which filter do you return to the user ? `filter0`, `filter1`, `filter2` or `filter3`? Why?
(2) What is the estimate of the generalization error of the filter returned to the user? `err0`, `err1`, `err2` or `err3`? Why?
(3) Once a SVM has been fitted to the training data, a new point is essentially classified according to the sign of a linear combination of the kernel function values between the support vectors and the new point. You are asked to implement this linear combination for `filter3`. You should make use of the functions `alphaindex`, `coef` and `b` that return the indexes of the support vectors, the linear coefficients for the support vectors, and the **negative** intercept of the linear combination. See the help file of the `kernlab` package for more information. You can check if your results are correct by comparing them with the output of the function `predict` where you set `type = "decision"`. Do so for the first 10 points in the `spam` dataset. Feel free to use the template provided in the `Lab3Block1_2021_SVMs_St.R` file.

4. Neural Networks

This assignment is to be solved with the `neuralnet` package.

(1) 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 value pairs are the data points available to you. Use 25 of the 500 points for training and the rest for test. Use **one hidden layer with 10 hidden units**. 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.

(2) In question (1), you used the default logistic (a.k.a. sigmoid) activation function, i.e. `act.fct = "logistic"`. Repeat question (1) with the following custom activation functions: $h_1(x) = x$, $h_2(x) = \max\{0, x\}$ and $h_3(x) = \ln(1 + \exp x)$ (a.k.a. linear, ReLU and softplus). See the help file of the `neuralnet` package to learn how to use custom activation functions. Plot and comment your results.

(3) Sample 500 points uniformly at random in the interval $[0, \mathbf{50}]$, and apply the sine function to each point. Use the NN learned in question (1) to predict the sine function value for these new 500 points. You should get mixed results. Plot and comment your results.

(4) In question (3), the predictions seem to converge to some value. Explain why this happens. To answer this question, you may need to get access to the weights of the NN learned. You can do it by running nn or $nn\$weights$ where nn is the NN learned.

(5) 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 goal was to predict $sin(x)$ from x. Use the learned NN to predict the **training data**. You should get bad results. Plot and comment your results. **Help**: Some people get a convergence error in this question. It can be solved by stopping the training before reaching convergence by setting `threshold = 0.1`.

Feel free to use the following template to solve the exercises above.

```
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)
```