732A99/TDDE01 MACHINE LEARNING

LAB 3 BLOCK 1: KERNEL METHODS AND NEURAL NETWORKS

JOSE M. PEÑA
IDA, LINKÖPING UNIVERSITY, SWEDEN

INSTRUCTIONS

Each student must submit a report with his/her solutions to the lab. The report must be concise but complete. It should include (i) the code implemented or the calls made to existing functions, (ii) the results of the code or calls, and (iii) explanations for the code and results.

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: **To be solved by 732A99/TDDE01/732A68/PhD course**

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 distance from a station to the point of interest.
- 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. Answer to the following questions:

- Show that your choice for the kernels' width is sensible, i.e. that it gives more weight to closer points. Discuss why your of definition of closeness is reasonable.
- Instead of combining the three kernels into one by summing them up, multiply them. 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: **To be solved by 732A99/732A68/PhD course**

Use the function `ksvm` from the R package `kernlab` to learn a SVM for classifying the `spam` dataset that is included with the package. Consider the radial basis function kernel (also known as Gaussian) with a width of 0.05. For the C parameter, consider values 0.5, 1 and 5. This implies that you have to consider three models.

- Perform model selection, i.e. select the most promising of the three models (use any method of your choice except cross-validation or nested cross-validation).
- Estimate the generalization error of the SVM selected above (use any method of your choice except cross-validation or nested cross-validation).
- Produce the SVM that will be returned to the user, i.e. show the code.
- What is the purpose of the parameter C ?

3. NEURAL NETWORKS: **To be solved by TDDE01**

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. 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. Consider threshold values $i/1000$ with $i = 1, \ldots, 10$. 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 initialization of the weights in the interval [-1, 1]
winit <- # Your code here
for(i in 1:10) {
  nn <- neuralnet(# Your code here)
  # Your code here
}

plot(nn <- neuralnet(# Your code here))

# Plot of the predictions (black dots) and the data (red dots)
plot(prediction(nn)$rep1)
points(trva, col = "red")
```

4. Neural Networks: **Special task, optional**

Implement the backpropagation algorithm for fitting the parameters of a NN for regression. The NN has one input unit, 10 hidden units, and one output unit. Use the `tanh` activation function. Recall that you have an example in Bishop's book as well as in the course slides. Feel free to use stochastic or batch gradient descent. Please use only basic R functions in your solution. Please comment your code appropriately. Feel free to use the following template.

```
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

# plot(trva)
# plot(tr)
# plot(va)

w_j <- runif(10, -1, 1)
b_j <- runif(10, -1, 1)
w_k <- runif(10, -1, 1)
b_k <- runif(1, -1, 1)

l_rate <- 1/nrow(tr)^2
n_ite = 5000
error <- rep(0, n_ite)
error_va <- rep(0, n_ite)

for(i in 1:n_ite) {

# error computation: Your code here

  cat("i: ", i, ", error: ", error[i]/2, ", error_va: ", error_va[i]/2, "\n")
  flush.console()

  for(n in 1:nrow(tr)) {

# forward propagation: Your code here

# backward propagation: Your code here

  }

}

w_j
```

```
b_j
w_k
b_k

plot(error/2, ylim=c(0, 5))
points(error_va/2, col = "red")
```