732A96/TDDE15 ADVANCED MACHINE LEARNING

LAB 4: GAUSSIAN PROCESSES

JOSE M. PEÑA
IDA, LINKÖPING UNIVERSITY, SWEDEN

1. Instructions

- **Deadline for individual and group reports**
  See LISAM.
- **What and how to hand in**
  Each student must send a report to LISAM with his/her solutions to the lab. The file should be named FirstName_LastName.pdf. 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 such code or calls, and (iii) explanations for (i) and (ii).
  In addition, students must discuss their lab solutions in a group. Each group must compile a collaborative report that will be used for presentation at the seminar. The report should clearly state the names of the students that participated in its compilation and a short description of how each student contributed to the report. This report should be submitted via LISAM. The group reports are corrected and graded. The individual reports are also checked, but feedback on them will not be given. A student passes the lab if the group report passes the lab and the individual report has reasonable quality, otherwise the student must complete his/her individual report by correcting the mistakes in it.
  Attendance to the seminar is obligatory. In the seminar, some groups will be responsible for presenting their group reports. Each student in these groups must be prepared to individually present an arbitrary part of the report. The selection of the presenters is done randomly during the seminar. In the seminar, some groups will act as opponents to the reports provided by the presenters. The opponent group should examine the group report of the presenter group before the seminar, and prepare a minimum of three questions, comments and/or improvements. The opponent group will ask these questions during the seminar. Check LISAM for the list of presenter and opponent groups.
- **Resources**
  The lab is designed to be partially solved with the R package `kernlab`. You may want to reuse code from the R demo files available on the course website. You may also want to use the RStudio development environment. To install the packages in the university's lab environment, do the following:
  (1) Open a terminal window.
  (2) Write ”module add courses/732A96” or ”module add courses/TDDE15” in the terminal window.
  (3) Write ”rstudio” in the terminal window.
  - Literature:
    * Package documentation.

2. Questions

The purpose of the lab is to put in practice some of the concepts covered in the lectures.

2.1. Implementing GP Regression. This first exercise will have you writing your own code for the Gaussian process regression model:

$$y = f(x) + \epsilon \text{ with } \epsilon \sim \mathcal{N}(0, \sigma_n^2) \text{ and } f \sim \mathcal{GP}(0, k(x, x'))$$

You must implement Algorithm 2.1 on page 19 of Rasmussen and Willams' book. The algorithm uses the Cholesky decomposition (`chol` in `R`) to attain numerical stability. Note that L in the algorithm is a lower triangular matrix, whereas the R function returns an upper triangular matrix. So, you need to transpose the output of the R function. In the algorithm, the notation $\boldsymbol{A}\backslash\boldsymbol{b}$ means the vector $\boldsymbol{x}$ that solves the equation $\boldsymbol{Ax} = \boldsymbol{b}$ (see p. xvii in the book). This is implemented in R with the help of the function `solve`.

Here is what you need to do:

(1) Write your own code for simulating from the posterior distribution of f using the squared exponential kernel. The function (name it `posteriorGP`) should return a vector with the posterior mean and variance of f, both evaluated at a set of x-values (X_*). You can assume that the prior mean of f is zero for all x. The function should have the following inputs:
- `X`: Vector of training inputs.
- `y`: Vector of training targets/outputs.
- `XStar`: Vector of inputs where the posterior distribution is evaluated, i.e. X_*.
- `sigmaNoise`: Noise standard deviation σ_n.
- `k`: Covariance function or kernel. That is, the kernel should be a separate function (see the file `GaussianProcesses.R` on the course web page).

(2) Now, let the prior hyperparameters be $\sigma_f = 1$ and $\ell = 0.3$. Update this prior with a single observation: $(x, y) = (0.4, 0.719)$. Assume that $\sigma_n = 0.1$. Plot the posterior mean of f over the interval $x \in [-1, 1]$. Plot also 95 % probability (pointwise) bands for f.

(3) Update your posterior from (2) with another observation: $(x, y) = (-0.6, -0.044)$. Plot the posterior mean of f over the interval $x \in [-1, 1]$. Plot also 95 % probability (pointwise) bands for f.

Hint: Updating the posterior after one observation with a new observation gives the same result as updating the prior directly with the two observations.

(4) Compute the posterior distribution of f using all the five data points in the table below (note that the two previous observations are included in the table). Plot the posterior mean of f over the interval $x \in [-1, 1]$. Plot also 95 % probability (pointwise) bands for f.

x	-1.0	-0.6	-0.2	0.4	0.8
y	0.768	-0.044	-0.940	0.719	-0.664

(5) Repeat (4), this time with hyperparameters $\sigma_f = 1$ and $\ell = 1$. Compare the results.

2.2. GP Regression with `kernlab`. In this exercise, you will work with the daily mean temperature in Stockholm (Tullinge) during the period January 1, 2010 - December 31, 2015. We have removed the leap year day February 29, 2012 to make things simpler. You can read the dataset with the command:

```
read.csv("https://github.com/STIMALiU/AdvMLCourse/raw/master/GaussianProcess/
Code/TempTullinge.csv", header=TRUE, sep=";")
```

Create the variable `time` which records the day number since the start of the dataset (i.e., `time`= 1, 2, . . ., 365 × 6 = 2190). Also, create the variable `day` that records the day number since the start of each year (i.e., `day`= 1, 2, . . ., 365, 1, 2, . . ., 365). Estimating a GP on 2190 observations can take some time on slower computers, so let us subsample the data and use only every fifth observation. This means that your time and day variables are now `time`= 1, 6, 11, . . ., 2186 and `day`= 1, 6, 11, . . ., 361, 1, 6, 11, . . ., 361.

(1) Familiarize yourself with the functions `gausspr` and `kernelMatrix` in `kernlab`. Do `?gausspr` and read the input arguments and the output. Also, go through the file

`KernLabDemo.R` available on the course website. You will need to understand it. Now, define your own square exponential kernel function (with parameters ℓ (`ell`) and σ_f (`sigmaf`)), evaluate it in the point $x = 1, x' = 2$, and use the `kernelMatrix` function to compute the covariance matrix $K(X, X_*)$ for the input vectors $X = (1,3,4)^T$ and $X_* = (2,3,4)^T$.

(2) Consider first the following model:

$$temp = f(time) + \epsilon \text{ with } \epsilon \sim \mathcal{N}(0, \sigma_n^2) \text{ and } f \sim \mathcal{GP}(0, k(time, time'))$$

Let σ_n^2 be the residual variance from a simple quadratic regression fit (using the `lm` function in R). Estimate the above Gaussian process regression model using the `gausspr` function with the squared exponential function from (1) with $\sigma_f = 20$ and $\ell = 100$ (use the option `scaled=FALSE` in the `gausspr` function, otherwise these σ_f and ℓ values are not suitable). Use the `predict` function in R to compute the posterior mean at every data point in the training dataset. Make a scatterplot of the data and superimpose the posterior mean of f as a curve (use `type="l"` in the plot function). Plot also the 95 % probability (pointwise) bands for f. Play around with different values on σ_f and ℓ (no need to write this in the report though).

(3) Repeat the previous exercise, but now use Algorithm 2.1 on page 19 of Rasmussen and Willams' book to compute the posterior mean and variance of f.

(4) Consider now the following model:

$$temp = f(day) + \epsilon \text{ with } \epsilon \sim \mathcal{N}(0, \sigma_n^2) \text{ and } f \sim \mathcal{GP}(0, k(day, day'))$$

Estimate the model using the `gausspr` function with the squared exponential function from (1) with $\sigma_f = 20$ and $\ell = 100$ (use the option `scaled=FALSE` in the `gausspr` function, otherwise these σ_f and ℓ values are not suitable). Superimpose the posterior mean from this model on the posterior mean from the model in (2). Note that this plot should also have the time variable on the horizontal axis. Compare the results of both models. What are the pros and cons of each model?

(5) Finally, implement the following extension of the squared exponential kernel with a periodic kernel (a.k.a. locally periodic kernel):

$$k(x, x') = \sigma_f^2 \exp\left\{-\frac{2\sin^2(\pi|x - x'|/d)}{\ell_1^2}\right\} \exp\left\{-\frac{1}{2}\frac{|x - x'|^2}{\ell_2^2}\right\}$$

Note that we have two different length scales in the kernel. Intuitively, ℓ_1 controls the correlation between two days in the same year, and ℓ_2 controls the correlation between the same day in different years. Estimate the GP model using the time variable with this kernel and hyperparameters $\sigma_f = 20$, $\ell_1 = 1$, $\ell_2 = 100$ and $d = 365$. Use the `gausspr` function with the option `scaled=FALSE`, otherwise these σ_f, ℓ_1 and ℓ_2 values are not suitable. Compare the fit to the previous two models (with $\sigma_f = 20$ and $\ell = 100$). Discuss the results.

2.3. **GP Classification with `kernlab`.** Download the banknote fraud data:

```
data <- read.csv("https://github.com/STIMALiU/AdvMLCourse/raw/master/
GaussianProcess/Code/banknoteFraud.csv", header=FALSE, sep=",")
names(data) <- c("varWave","skewWave","kurtWave","entropyWave","fraud")
data[,5] <- as.factor(data[,5])
```

You can read about this dataset here. Choose 1000 observations as training data using the following command (i.e., use the vector `SelectTraining` to subset the training observations):

```
set.seed(111); SelectTraining <- sample(1:dim(data)[1], size = 1000,
replace = FALSE)
```

(1) Use the R package `kernlab` to fit a Gaussian process classification model for fraud on the training data. Use the default kernel and hyperparameters. Start using only the covariates `varWave` and `skewWave` in the model. Plot contours of the prediction probabilities over a suitable grid of values for `varWave` and `skewWave`. Overlay the training data for fraud = 1 (as blue points) and fraud = 0 (as red points). You can reuse

code from the file `KernLabDemo.R` available on the course website. Compute the confusion matrix for the classifier and its accuracy.

(2) Using the estimated model from (1), make predictions for the test set. Compute the accuracy.

(3) Train a model using all four covariates. Make predictions on the test set and compare the accuracy to the model with only two covariates.