

## Gaussian Processes - Computer Lab

---

**Deadline:** See LISAM  
**Teacher:** Mattias Villani  
**Grades:** Pass/Fail  
**Submission:** Via LISAM

You should use R to solve the lab since the computer exam will be in R.  
Output: an individual report and a group report to be presented on the seminar.  
Attach your code in LISAM as separate files.

---

1. **Implementing Gaussian process regression from scratch.** This first exercise will have you writing your own code for the Gaussian process regression model:

$$y = f(x) + \varepsilon \quad \varepsilon \sim N(0, \sigma_n^2) \\ f \sim GP[0, k(x, x')].$$

When it comes to the posterior distribution for  $f$ , I **strongly** suggest that you implement Algorithm 2.1 on page 19 of Rasmussen and Williams (RW) book. That algorithm uses the Cholesky decomposition (`chol()` in R) to attain numerical stability. Here is what you need to do:

- (a) Write your own code for simulating from the posterior distribution of  $f(x)$  using the squared exponential kernel. The function (name it `posteriorGP`) should return vectors 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:
  - i. `x` (vector of training inputs)
  - ii. `y` (vector of training targets/outputs)
  - iii. `xStar` (vector of inputs where the posterior distribution is evaluated, i.e.  $x^*$ . As in my slides).
  - iv. `hyperParam` (vector with two elements  $\sigma_f$  and  $\ell$ )
  - v. `sigmaNoise` ( $\sigma_n$ ).[Hint: I would write a separate function for the Kernel (see my `GaussianProcess.R` function on the course web page) which is then called from the `posteriorGP` function.]
- (b) Now let the prior hyperparameters be  $\sigma_f = 1, \ell = 0.3$ . Update this prior with a single observation:  $(x, y) = (0.4, 0.719)$ . Assume that the noise standard deviation is known to be  $\sigma_n = 0.1$ . Plot the posterior mean of  $f$  over the interval  $x \in [-1, 1]$ . Plot also 95% probability (pointwise) bands for  $f$ .

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

Table 1: Simple data set for GP regression.

- (c) Update your posterior from 1b) 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 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. Bayes is beautiful!]
- (d) Compute the posterior distribution of  $f$  using all 5 data points in Table 1 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 intervals for  $f$ .
- (e) Repeat 1d), this time with the hyperparameters  $\sigma_f = 1, \ell = 1$ . Compare the results.

2. **Gaussian process regression on real data using the kernlab package.** This exercise lets you explore the kernlab package on a data set of daily mean temperature in Stockholm (Tullinge) during the period January 1, 2010 - December 31, 2015. I have removed the leap year day February 29, 2012 to make your life 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 \cdot 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's thin out the data by only using every fifth observation. This means that your time variable is now `time = 1, 6, 11, ..., 2186` and `day = 1, 6, 11, ..., 361, 1, 6, ..., 361`.

- (a) Familiarize yourself with the following functions in `kernlab`, in particular the `gausspr` and `kernelMatrix` function. Do `?gausspr` and read the input arguments and the output. Also, go through my `KernLabDemo.R` carefully; you will need to understand it. Now, define your own square exponential kernel function (with parameters  $\ell$  (`ell`) and  $\sigma_f$  (`sigmaf`)), evaluate it in the point  $x = 1$ ,  $x' = 2$ , and use the `kernelMatrix` function to compute the covariance matrix  $K(\mathbf{x}, \mathbf{x}_*)$  for the input vectors  $\mathbf{x} = (1, 3, 4)^T$  and  $\mathbf{x}_* = (2, 3, 4)^T$ .
- (b) Consider first the model:

$$\begin{aligned} temp &= f(time) + \varepsilon \quad \varepsilon \sim N(0, \sigma_n^2) \\ f &\sim GP[0, k(time, time')] \end{aligned}$$

Let  $\sigma_n^2$  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 squared exponential function from 2a) with  $\sigma_f = 20$  and  $\ell = 0.2$ . Use the `predict` function to compute the posterior mean at every data point in the training datasets. Make a scatterplot of the data and superimpose the posterior mean of  $f$  as a curve (use `type="l"` in the plot function). Play around with different values on  $\sigma_f$  and  $\ell$  (no need to write this in the report though).

- (c) **Kernlab** can compute the posterior variance of  $f$ , but I suspect a bug in the code (I get weird results). Do you own computations for the posterior covariance matrix of  $f$  (hint: Algorithm 2.1 in RW), and use information in that matrix to plot 95% (pointwise) posterior probability bands for  $f$ . Use  $\sigma_f = 20$  and  $\ell = 0.2$ . Superimpose those bands on the figure with the posterior mean in 2b).
- (d) Consider now the model

$$\begin{aligned} \text{temp} &= f(\text{day}) + \varepsilon \quad \varepsilon \sim N(0, \sigma_n^2) \\ f &\sim GP[0, k(\text{day}, \text{day}')] \end{aligned}$$

Estimate the model using the squared exponential function from 2a) with  $\sigma_f = 20$  and  $\ell = 6 \cdot 0.2 = 1.2$ . (I multiplied  $\ell$  by 6 compared to when you used **time** as input variable since **kernlab** automatically standardizes the data which makes the distance between points larger for **day** compared to **time**). Superimpose the posterior mean from this model on the fit (posterior mean) from the model with **time** using  $\sigma_f = 20, \ell = 0.2$ . Note that this plot should also have the **time** variable on the horizontal axis. Compare the results from using **time** to the ones with **day**. What are the pros and cons of each model?

- (e) Now implement a generalization of the periodic kernel given in my slides from Lecture 2 of the GP topic (Slide 6)

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

Note that we have two different length scales here, and  $\ell_2$  controls the correlation between the same day in different years ( $\ell_2$ ). So this kernel has four hyperparameters  $\sigma_f, \ell_1, \ell_2$  and the period  $d$ . Estimate the GP model using the **time** variable with this kernel with hyperparameters  $\sigma_f = 20, \ell_1 = 1, \ell_2 = 10$  and  $d = 365/\text{sd}(\text{time})$ . The reason for the rather strange period here is that **kernlab** standardized inputs to have standard deviation of 1. Compare the fit to the previous two models (with  $\sigma_f = 20, \ell = 0.2$ ). Discuss the results.

### 3. Gaussian process classification using the kernlab package. 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)
```

- (a) Use **kernlab** to fit a Gaussian process classification model for **fraud** on the training data, using **kernlab**. Use **kernlab**'s the default kernel and hyperparameters. Start with using only the first two 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 take a lot of code for this from my **KernLabDemo.R**. Compute the confusion matrix for the classifier and its accuracy.
- (b) Using the estimated model from 3a), make predictions for the testset. Compute the accuracy.

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

Good luck!

- Mattias