

# Advanced Machine Learning - Lab 04

*G. Hari Prasath (hargo729), Julius Kittler (julki029), Hector Plata (hecpl268) and  
Maximilian Pfundstein (maxpf364)*

2019-10-08

## Contents

<b>1</b>	<b>Implementing GP Regression</b>	<b>1</b>
1.1	Simulating from the Posterior . . . . .	1
1.2	Prior Observation . . . . .	2
1.3	Another Observation . . . . .	2
1.4	Five Observations . . . . .	2
1.5	Different Hyperparameters . . . . .	2
<b>2</b>	<b>GP Regression with kernlab</b>	<b>2</b>
2.1	Computing the Covariance Matrix . . . . .	3
2.2	Estimate the above GP regression model . . . . .	3
2.3	Posterior Variance . . . . .	3
2.4	Another Model . . . . .	3
2.5	Periodic Kernel Implementation . . . . .	3
<b>3</b>	<b>GP Classification with kernlab</b>	<b>4</b>
3.1	Using kernlab for Training a Model (Limited Covariates) . . . . .	4
3.2	Predicting on the Test Set . . . . .	4
3.3	Using all Covariates . . . . .	4

## 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 Williams' 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  $\mathbf{A}/\mathbf{b}$  means the vector  $\mathbf{x}$  that solves the equation  $\mathbf{Ax} = \mathbf{b}$  (see p. xvii in the book). This is implemented in R with the help of the function `solve`.

### 1.1 Simulating from the Posterior

**Task:** 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.  $X_*$ .

- **hyperParam:** Vector with two elements  $\sigma_f$  and  $\ell$ .
- **sigmaNoise:** Noise standard deviation  $\sigma_n$ .

**Hint:** Write a separate function for the kernel (see the file `GaussianProcess.R` on the course web page).

## 1.2 Prior Observation

**Task:** 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 percent probability (pointwise) bands for  $f$ .

## 1.3 Another Observation

**Task:** 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 percent 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.

## 1.4 Five Observations

**Task:** 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 percent probability (pointwise) bands for  $f$ .

```
##      [,1] [,2] [,3] [,4] [,5]
## x -1.0 -0.2 0.800 -0.044 0.719
## y -0.6 0.4 0.768 -0.940 -0.664
```

## 1.5 Different Hyperparameters

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

# 2 GP Regression with kernlab

**Task:** 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 \times 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`.

## 2.1 Computing the Covariance Matrix

**Task:** 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  $\ell$  (`ell`) and  $\sigma_f$  (`sigmaf`)), evaluate it in the point  $x = 1$ ,  $x' = 2$  and use the `kernelMatrix` function to compute the covariance matrix  $K(X, X_*)$  for the input vectors  $X = (1, 3, 4)^T$  and  $X_* = (2, 3, 4)^T$ .

## 2.2 Estimate the above $\mathcal{GP}$ regression model

**Task:** Consider first the following model:

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

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 (1) with  $\sigma_f = 20$  and  $\ell = 0.2$ . 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). Play around with different values on  $\sigma_f$  and  $\ell$  (no need to write this in the report though).

## 2.3 Posterior Variance

**Task:** `kernlab` can compute the posterior variance of  $f$ , but it seems to be a bug in the code. So, do your own computations for the posterior variance of  $f$  and plot the 95 percent probability (pointwise) bands for  $f$ . Superimpose these bands on the figure with the posterior mean that you obtained in (2).

**Hint:** Note that Algorithm 2.1 on page 19 of Rasmussen and Williams' book already does the calculations required. Note also that `kernlab` scales the data by default to have zero mean and standard deviation one. So, the output of your implementation of Algorithm 2.1 will not coincide with the output of `kernlab` unless you scale the data first. For this, you may want to use the R function `scale`.

## 2.4 Another Model

**Task:** Consider first the following model:

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

Estimate the model using the squared exponential function with  $\sigma_f = 20$  and  $\ell = 0.2$ . 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?

## 2.5 Periodic Kernel Implementation

**Task:** Finally, implement a generalization of the periodic kernel given in the lectures:

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

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

### 3 GP Classification with kernlab

Download the banknote fraud data:

```
data = read.csv("https://github.com/STIMALiU/AdvMLCourse/raw/master/\n
               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)
```

#### 3.1 Using kernlab for Training a Model (Limited Covariates)

**Task:** 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.

#### 3.2 Predicting on the Test Set

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

#### 3.3 Using all Covariates

**Task:** Train a model using all four covariates. Make predictions on the test set and compare the accuracy to the model with only two covariates.

```
knitr::opts_chunk$set(echo = TRUE)
set.seed(12345)

res = matrix(c(-1, -0.6, -0.2, 0.4, 0.8, 0.768, -0.044, -0.940, 0.719, -0.664),
             nrow = 2)
rownames(res) = c("x", "y")
res

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

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

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