Computational Finance
FIN-472
Homework 13 and Take Home 6

December 16, 2022

- **Please upload your solutions on moodle by Friday 23.12.2020 at 08h15.**
- **You should upload one PDF with the Matlab code and results you obtained and do not forget to upload in moodle the Matlab codes.**
- **Exercise 2 is part of Take Home.**

Exercise 1: (Visualizing the priors) The specification of the covariance function k implies a distribution over functions. Given two feature vectors $\boldsymbol{x}_1$, $\boldsymbol{x}_2$ of dimension d in the input space $\mathcal{X}$, consider the following covariance functions

a) **Squared Exponential:**

$$k(\boldsymbol{x}_1, \boldsymbol{x}_2) = \sigma^2 \exp\left(-\frac{r^2}{2\ell^2}\right).$$

b) **Linear:**

$$k(\boldsymbol{x}_1, \boldsymbol{x}_2) = \sigma_0^2 + \sigma_1^2 (\boldsymbol{x}_1 - c)^T (\boldsymbol{x}_2 - c).$$

c) **Exponential:**

$$k(\boldsymbol{x}_1, \boldsymbol{x}_2) = \sigma^2 \exp\left(-\frac{r}{2\ell^2}\right).$$

d) **Periodic:**

$$k(\boldsymbol{x}_1, \boldsymbol{x}_2) = \sigma^2 \exp\left(-\frac{2}{\ell^2}\sin^2\left(\frac{\pi r}{p}\right)\right).$$

where $r^2 = ||\boldsymbol{x}_1 - \boldsymbol{x}_2||_2^2$. In this exercise, we wish to visualize the samples from prior distribution over f for the above covariance functions.

(i) Write four MATLAB functions, one for each covariance function. Each function should return the covariance matrix $K(\boldsymbol{X}_1, \boldsymbol{X}_2)$ of size $n \times m$ given two input matrices $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ of sizes $n \times d$ and $m \times d$. The function should take the hyper-parameters of the co-variance function as input. The parameters involved in the above definitions are real-valued and positive.

(ii) For this exercise, assume that the input space $\mathcal{X} = \mathbb{R}$. Write a MATLAB function that plots the samples drawn from the Gaussian prior. More specifically, the function calls the covariance function in 1(i) with inputs $\boldsymbol{X}_1 = \boldsymbol{X}_2 = \boldsymbol{X}$, where $\boldsymbol{X}$ is a matrix of size $n \times 1$ that consists of n inputs and use the resulting $n \times n$ co-variance matrix $K = (K(\boldsymbol{x}_i, \boldsymbol{x}_j))_{\{i,j=1,\dots,n\}}$ to draw N samples from the multivariate normal distribution with mean 0 and co-variance matrix K. Finally, plot these N functions. MATLAB command to draw N random samples from a multivariate normal distribution is mvnrnd(mu,sigma,N).

(iii) Implement the code written in 1(ii) for the each of the kernels defined in 1(i). Starting with the initial values of each of the parameter equal to 1, play with different values of each of the parameter and observe it's effect on the function properties, for example, smoothness etc. You can take the training inputs as $\boldsymbol{X} = linspace(-5, 5, 100)$.

Exercise 2: (part of Take Home)

(i) Modify the function written in Exercise 1 for exponential kernel and squared-exponential kernel such that the new functions return the following

- the kernel matrix
- matrices corresponding to partial derivative of kernel with respect to each hyper-parameter (two matrices if there are two hyper-parameters)

Note that the partial derivative of the kernel is required only in the optimization step. Therefore, your function should return the gradient only if the inputs are such that $\mathbf{X}_1 = \mathbf{X}_2$; otherwise it should return only the kernel matrix $K(\mathbf{X}_1, \mathbf{X}_2)$.

(ii) Write a MATLAB function fitGPR.m that implements Gaussian process regression (see next page for summary). Your function should take the training data $\mathcal{D}$, the choice of kernel (defined in part 2(i) above), initial choice of hyper-parameters (depending on the kernel choice) and the test features $\boldsymbol{X}_*$. The function should maximize the log marginal likelihood function using the gradient information (use any optimizer that accepts the derivative of the objective function, for instance, "FMINLBFGS", `https://www.mathworks.com/matlabcentral/fileexchange/23245-fminlbfgs-fast-limited-memory-optimizer`). The function should return optimal hyper-parameters, posterior mean and posterior co-variance at $\boldsymbol{X}_*$. Take σ_n to be a fixed parameter.

(iii) Repeat Exercise 2 of Exercise Sheet 12 using the fitGPR.m you built in part 2(ii).

Summary: Gaussian Process Regression

Consider the following regression problem

$$y = f(\boldsymbol{x}) + \epsilon,$$

where

- $\boldsymbol{x}$ is a d dimensional feature in an input domain $\mathcal{X}$ (for example $\mathbb{R}^n$, but in general, anything)
- y is a scalar output variable
- the i.i.d. Gaussian noise, i.e., $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$ (independent of f)
- an unknown function $f : \mathcal{X} \to \mathbb{R}$.

Given noisy observations of the function, i.e., a training set $\mathcal{D}$, $\mathcal{D} = \{(\boldsymbol{x}_i, y_i) | i = 1, \ldots, n\}$, our goal is to obtain predictions $f(\boldsymbol{x}_j^*)$ for new features $\{\boldsymbol{x}_j^*\}_{j=1,2,\ldots,m}$ (not present in the training set). Under Gaussian prior, we have the following joint distribution of the observed training outputs $\boldsymbol{y}$, and testing outputs $\boldsymbol{f}_*$

$$\begin{pmatrix} \boldsymbol{y} \\ \boldsymbol{f}_* \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \mathbf{0} \\ \mathbf{0} \end{pmatrix}, \begin{pmatrix} K(\boldsymbol{X}, \boldsymbol{X}) + \sigma_n^2 I & K(\boldsymbol{X}, \boldsymbol{X}_*) \\ K(\boldsymbol{X}_*, \boldsymbol{X}) & K(\boldsymbol{X}_*, \boldsymbol{X}_*) \end{pmatrix} \right), \tag{1}$$

where

- I is an identity matrix
- $\boldsymbol{y} = (y_1, \ldots, y_n)$ and $\boldsymbol{f}_* = (f(\boldsymbol{x}_1^*), \ldots, f(\boldsymbol{x}_m^*))$
- $\boldsymbol{X}$, of size $n \times d$, where ith row is the ith feature vector $\boldsymbol{x}_i$ in the training data.
- $\boldsymbol{X}_*$, of size $m \times d$, where jth row is the jth feature vector $\boldsymbol{x}_j^*$ in the testing data.

Training: Generally the kernel function depends on hyper-parameters. We estimate the optimal hyper-parameters (denote the vector of hyper-parameters θ that also includes the noise variance σ_n^2), by maximizing the following log marginal likelihood

$$\log p(\boldsymbol{y} \mid \boldsymbol{X}, \theta) = -\frac{\boldsymbol{y}^T K_y^{-1} \boldsymbol{y}}{2} - \frac{\log det(K_y)}{2} - \frac{n \log(2\pi)}{2}, \tag{2}$$

where $K_y = K(\boldsymbol{X}, \boldsymbol{X}) + \sigma_n^2 I$. Since the computational overhead of computing derivatives is small, so using a gradient based optimizer is advantageous. The partial derivatives of the marginal likelihood w.r.t. the hyper-parameters is given by

$$\begin{aligned} \frac{\partial}{\partial \theta_j} \log p(\boldsymbol{y} \mid \mathbf{X}, \theta) &= -\frac{\mathbf{y}^T K_y^{-1} \frac{\partial K_y}{\partial \theta_j} K_y^{-1} \mathbf{y}}{2} - \frac{1}{2} tr\left(K_y^{-1} \frac{\partial K_y}{\partial \theta_j} \right) \\ &= \frac{1}{2} tr\left((\boldsymbol{\alpha}\boldsymbol{\alpha}^T - K_y^{-1}) \frac{\partial K_y}{\partial \theta_j} \right) \end{aligned} \tag{3}$$

where $\boldsymbol{\alpha} = K_y^{-1}\mathbf{y}$. You can use the following (wherever required) for more efficient computations,

- $A^{-1}y = L^T \setminus (L \setminus y)$ where L is the Cholesky factor from A.
- $$\log(det(A)) = 2 \sum_{i=1}^{n} \log(L_{ii}).$$

Predicting: Finally, conditional on $\mathcal{D}$, the posterior distribution is given by

$$\boldsymbol{f}_* \mid \boldsymbol{X}_*, \boldsymbol{X}, \boldsymbol{y} \quad \sim \quad \mathcal{N}(\boldsymbol{K}_*^T K_y^{-1} \boldsymbol{y}, \boldsymbol{K}_{**} - \boldsymbol{K}_*^T K_y^{-1} \boldsymbol{K}_*),$$

where

$$\boldsymbol{K}_* = K(\boldsymbol{X}, \boldsymbol{X}_*), \quad \boldsymbol{K}_{**} = K(\boldsymbol{X}_*, \boldsymbol{X}_*).$$

For an algorithm, refer to Slide xxxx of Lecture 13.