General hints for code submissions To make it easier for us and others to understand your solutions please follow these guidelines:

- If available use the template file to create your solution.
- Please add comments so we can understand your solution.
- Please make sure to load all required packages at the beginning of your code.
- Only use relative file paths for `source()`, `load()`, etc.
- Each exercise directory contains a `skeleton` folder where preliminary `R` files are located.
- Use these `R` files as a basis for creating your solution which should be contained in a main `R` file named as the content of the exercise, e.g., `evaluation.R`.
- You can (and sometimes have to) reuse code from previous exercises.
- The points indicate the difficulty of the task.
- If not stated otherwise, we will use exclusively R 4.0 or greater.

Now that you have learned about the theory behind GPs, you will have to use that theory to implement GPs yourself.

1. **Gaussian Processes** [14 points]
The exercise is mostly concerned with equations 2.11 and 2.12 in chapter 2 of "Gaussian Processes for Machine Learning[1]"

 (a) Your first task is to implement 2.11 to predict the mean and variance given some observations. [4.pt]

$$f_\star|\boldsymbol{x}_\star, \boldsymbol{X}, \boldsymbol{y} \sim \mathcal{N}(\frac{1}{\sigma_n^2}\phi(\boldsymbol{x}_\star)^\intercal A^{-1}\Phi\boldsymbol{y}, \phi(\boldsymbol{x}_\star)^\intercal A^{-1}\phi(\boldsymbol{x}_\star))$$

 where $\Phi = \phi(\boldsymbol{X})$ and $A = \sigma_n^{-2}\Phi\Phi^\intercal + \Sigma_p^{-1}$.

 We provide a function prototype that takes the observations, a function ϕ, and an array of points, $[\mathbf{x}_1, \mathbf{x}_2, \dots]$, as arguments. Compute the mean μ and the variance σ^2 at all $\mathbf{x}$. For matrix inversion you can use `solve(..., tol = 0)`. Use the data provided in the script to create a plot that shows the mean and the 2σ confidence interval around it for a feature space of size $N = 2$. Use $\sigma_n = 1$, $\Sigma_p = I$ and

$$\phi(x) = (1, x)^T$$

 which corresponds to Bayesian *linear* regression for one dimensional input.

 (b) Implement a second function based on equation 2.12: [5.pt]

$$f_\star|\boldsymbol{x}_\star, \boldsymbol{X}, \boldsymbol{y} \sim \mathcal{N}(\phi_\star^\intercal\Sigma_p\Phi(\Phi^\intercal\Sigma_p\Phi + \sigma_n^2 I)^{-1}\boldsymbol{y}, \phi_\star^\intercal\Sigma_p\phi_\star - \phi_\star^\intercal\Sigma_p\Phi(\Phi^\intercal\Sigma_p\Phi + \sigma_n^2 I)^{-1}\Phi^\intercal\Sigma_p\phi_\star)$$

 Convince yourself, that both yield the same values, by checking the output for the given input[2]. Compare the time it takes for both implementations to compute the output for the given data, and points $\mathbf{x}_i$, for a growing number of features. Again use $\sigma_n = 1$ and $\Sigma_p = I$ and

$$\phi_n(x) = (1, x, x^2, \dots, x^{n-1})^T \tag{1}$$

 to plot the computing time for $n = 2, 4, 8, \dots, 2048, 4096$.

[1] `http://www.gaussianprocess.org/gpml/chapters/RW2.pdf`

[2] By the nature of numerical calculations, the results will not be identical, but the difference will be very small. Use `all.equal` to test for equality.

(c) Apart from the reliable uncertainty estimates, a GP has the additional advantage to allow for easy optimization of its own hyperparameters by gradient based optimization of the log marginal likelihood (equation 5.8). [5.pt]

$$\log p(\mathbf{y}|X,\theta) = -\frac{1}{2}\mathbf{y}^T K_y^{-1}\mathbf{y} - \frac{1}{2}\log|K_y| - \frac{n}{2}\log 2\pi$$

where θ denotes the kernel hyperparameters (see eq. 5.1). In this exercise you will use equation 2.12 to optimize hyperparameters of the kernel. By replacing the inner products with an explicit kernel function equation $K_f = k(X,X) = \Phi^\intercal\Sigma_p\Phi$ where $K_y = K_f + \sigma_n^2 I$, (σ_n noise level of the data) equation 2.12 becomes:

$$f_\star|\boldsymbol{X}_\star, \boldsymbol{X}, \boldsymbol{y} \sim \mathcal{N}(K_*^T K_y^{-1}\mathbf{y}, K_{**} - K_*^T K_y^{-1} K_*)$$

where $K_* = k(\mathbf{X}, \mathbf{X}^\star) = \phi_\star^\intercal\Sigma_p\Phi$ and $K_{**} = k(\mathbf{X}^\star, \mathbf{X}^\star) = \phi_\star^\intercal\Sigma_p\phi_\star$. Here we will use the squared exponential kernel:

$$k(\mathbf{X_p}, \mathbf{X_q}) = \sigma_f^2 \exp\left(-\frac{1}{2}(\mathbf{X_p} - \mathbf{X_q})^T l^{-2} I (\mathbf{X_p} - \mathbf{X_q})\right)$$

For this exercise you are given the implementation of 2.12 in the kernel formulation and are asked to implement the squared exponential kernel and the log marginal likelihood (σ_f, L). The optimization is carried out via `optimize` and already implemented for you. Try to play around with the random seed to see how different training data effects to the HPO of the GP.