# Gaussian Process Regression on a Grid with Tensor Train Matrices

## 1 Prerequisites and First Computer Session

- This assignment assumes basic preliminary knowledge of Kernel Methods and Gaussian Processes. However, you can finish the assignment if you just read and understand the second chapter of the GP book (see below).
- For the first computer session, please, at least prepare the following:
  - If necessary read the theory about GPs and Implement GPregression function. Do the first set of exercises 3.1.
  - Start working on 3.2 exercise about TT representation of a Kernel Matrix.

## 2 Background

Gaussian processes (GPs) are powerful Bayesian function approximators commonly used for supervised learning tasks. However GPs have a downside: with a growing number of data points $N$, the computational complexity blows up. In this project, you will do GP regression on a data set, where the input lies on a grid. In this way, the kernel matrix with a special case of tensor trains and predictions can be made with a fraction of the original complexity.

To get a better idea of what GPs do, you can play around with a 1D example on this website: `http://www.it.uu.se/edu/course/homepage/apml/GP/`. To deepen your theoretical understanding of GPs, please read the second chapter of [1]: `http://gaussianprocess.org/gpml/chapters/RW.pdf`.

In the following, you are asked to work through 4 exercises. With your results, write a report that contains a clear story line and that answer all the questions. The subpoints in the exercises are there to help you go through all the necessary steps. You do not have to write answers one by one, but incorporate your results and answers into your report.

Before you start, download the provided code. It contains a function called `getdata` that creates input output data $(\mathbf{X}, \mathbf{y})$ for a GP, as well as test data $\mathbf{X}_*$ for predictions. The input data $\mathbf{X}$ is a matrix of size $N \times 3$ and the points lay on a 3 dimensional grid in a cube that in each dimension goes from -1 to 1. The output data is of size $N \times 1$ and the test data of size which are points on a horizontal slice at height 0 in the cube. The first input of the function is the number of data points per dimension $N_d$, such that the total number of data points will be $N = N_d^3$. The second input are the hyperparameters necessary for the GP saved in the variable `hyp`. Do not change the code in this function.

For questions contact `a.saiapin@tudelft.nl`.

## 3 Exercises

### 3.1 GP Complexity

The exercises in this section serve as an introduction to how GPs work. You can use your results and answers in the background section of your report.

1. Implement a function called `GPregression` that computes the mean and covariance of predictions in unseen inputs based on the noise-free equations (2.19) of [1, p.16]. To compute the kernel matrices, use the squared exponential kernel function given by $\kappa(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp(-\frac{1}{2\ell^2}\|\mathbf{x} - \mathbf{x}'\|^2)$. The function's

input should be $\mathbf{X}, \mathbf{y}, \mathbf{X}_*, \ell^2, \sigma_f^2$ and the function should return the mean and covariance, as well as the kernel matrix $\mathbf{K}$ for the training inputs. It is `hyp` $= [\ell^2, \sigma_f^2] = [.05, 1]$.

2. Next, call the function `getdata` varying its first input, the number of data points per dimension, $10, 15, 20$ to obtain $N = 10^3, 15^3, 20^3$ inputs, outputs and test inputs.

3. Call the function `GPregression` with the three different data sets and time with `tic` and `toc` how long it takes.

4. Comment on the GP's complexity in terms of the time it takes to run. What is the bottleneck of the computation?

5. Plot the 3D measurements for $N = 1000$ with `scatter3(X(:,1),X(:,2),X(:,3),[],y,'filled')`. Also plot the 3D predictions for $N = 1000$ with `scatter(Xstar(:,1),Xstar(:,3),[],mstar,'filled'`, where `mstar` are the predictions. Describe both plots. What could the second plot represent?

## 3.2 Kernel Matrix as Tensor Train

In the last exercise, you computed the kernel matrix of a GP. Having the input on a multidimensional grid, you will now investigate, how a low-rank TT can be used to represent the kernel matrix. Use your answers to the questions as well as your insights of the implementations in the main part of your report.

1. Transform the kernel matrix $\mathbf{K}$ that you computed with $10^3$ inputs into a tensor $\boldsymbol{\mathcal{K}}$ of size $I_1 \times I_2 \times I_3 \times J_1 \times J_2 \times J_3$, where all sizes are equal to 10. Then, compute a TT with the TT-SVD algorithm with $\epsilon = 10^{-10}$. You can use your or the hidden TT-SVD function from a previous exercise.

2. Now, permute the dimensions of $\boldsymbol{\mathcal{K}}$ such that it is $I_1 \times J_1 \times I_2 \times J_2 \times I_3 \times J_3$.

3. Reshape $\boldsymbol{\mathcal{K}}$ such that the sizes are $I_1 J_1 \times I_2 J_2 \times I_3 J_3$ and do another TT-SVD with $\epsilon = 10^{-10}$.

4. What do you observe when comparing the two TTs? Comment on the ranks and on what the sizes $I_1, I_2, I_3, J_1, J_2, J_3$ represent. Describe the link to a tensor train matrix.

## 3.3 Kernel Matrix as Kronecker Product

For inputs that lie on a multidimensional grid, the dimensions become independent of each other the kernel matrix can be written as a Kronecker product over dimensions, as described in [2].

1. Compute a kernel matrix per dimension with 10 data points per dimension with Equation (5.4) of [2], using one term of the product per dimension. Note that the inputs per dimension are the coordinates per dimension, e.g. `coordX1 = linspace(-1,1,10)`.

2. Compute the Kronecker product $\mathbf{K}_3 \otimes \mathbf{K}_2 \otimes \mathbf{K}_1$. You can test your result by comparing it to $\mathbf{K}$ that is returned by the function `GPregression`.

3. Compute the inverse of $\mathbf{K}$, given by $\mathbf{K}^{-1} = \mathbf{K}_3^{-1} \otimes \mathbf{K}_2^{-1} \otimes \mathbf{K}_1^{-1}$

4. Comment on how to transform the Kronecker product into a tensor train. Comment on the size of each TT-core and why this format decreases the computational complexity.

5. Draw a diagram of the Kronecker product of matrices in terms of a 3 core tensor train matrix. Hint: check the lecture notes how the indices behave in the Kronecker product.

## 3.4 GPs with Tensor Train

1. Implement a new function `KronGPregression` that receives as an input the kernel matrices per dimension, as well as the data $\mathbf{X}, \mathbf{y}$ and $\mathbf{X}_*$. Exploit the Kronecker algebra to improve the scalability of the GP regression.

2. Call the new function for $N = 10^3, 15^3, 20^3$ and time with `tic` and `toc`.

3. Do you see a speed up compared to the original function?

## References

[1] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning.* The MIT Press, 2006.

[2] Yunus Saatçi. *Scalable inference for structured Gaussian process models.* PhD thesis, Citeseer.