Task 1: GPR for fine particulate matter prediction

Probabilistic Artificial Intelligence

Wolfgang Böttcher

October 24, 2021

Problem Analysis

The given task requires to implementation of a Gauss-Process-Regression (GPR) for the given provided, which represents the fine particulate matter as measured at defined 2D positions. As already outlined in the task description, there are three main challenges to be addressed. *Firstly*, the amount of data has a size that makes naive GPR implementations computationally inadmissible. *Secondly*, a proper kernel that is suited for the type of data needs to be selected. *Thirdly*, the parameters of the kernel need to be tuned such that the model performs well on the specific cost function specified for this task.

Problem Solution

Computational challenges Given that the input data we are working on is only 2D, using an inducing point method with a regular grid is an expedient choice. Such a method might be infeasible for higher-dimensional data but speeds up computations significantly for 2D-4D data. Precisely, we chose to implement our code using the *gpytorch* library, which provides the class `gpytorch.kernels.GridInterpolationKernel`. A benefit of using this kernel wrapper class is that we can provide it with any stationary kernel that one wants to test on the data. The grid size was determined automatically using the helper function `gpytorch.utils.grid.choose_grid_size(x)`.

Parameter selection *GPytorch* provides an easy to use framework for optimising the hyper-parameters of a GPR-model. Therefore, we used the recommended optimisation flow with a first-order optimisation on the marginal-log-likelihood values using Adam as the optimisation algorithm. The number of epochs for training and the learning rate were determined empirically by analysis of the training/validation loss. For the training-test split, the `train_test_split()` function form *scikit-learn* was used. While the GPytorch kernel-parameter optimisation didn't use any spectial loss metric, the hyperparameter optimisation was set up to use the provided custom loss function for the task.

Kernel selection The best working kernel was determined by trying out different kernel types and having the program optimise their parameters. It was found that RBF kernels were the most suitable choice for the given data. That said, we didn't explore compositions of different kernel types as our simple solution already clearly passed the public and private tests.

Results

Our GPR-model with RBF kernel, Grid Interpolation inducing points method and hyper-parameter tuning via a custom loss yielded a score of 9.22 on the public and 11.386 the private testset.