Data-Driven Design & Analysis of Structures & Materials (3dasm)

Instructor: Miguel A. Bessa

Brown University

Homework 5

Deliver **a short PDF report** of this assignment containing the answers to the questions listed here. **UPLOAD to CANVAS in the Assignments section (Homework 5) by the due date.**

Due date: until 11:59pm of day announced in course home page.

This Homework aims to help you reason about Gaussian Processes by writing and running code, as well as by reflecting on the content of Lectures 12, 13 and 14. This is a **computational homework**, where you need to include the Python code used to solve it – **we advise you to solve it in a Jupyter Notebook and printing it as PDF including the answers**.

Exercise 1

1. Conceptual questions.
   1.1. What is the difference between parameters and hyperparameters?

   1.2. What is a kernel function?

   1.3. In machine learning literature, the concept of 'kernel trick' is commonly mentioned. Explain this concept and why it is useful.

   1.4. Explain why Gaussian Processes are a generalization of Bayesian Linear Regression. You can use the mathematical derivations of Lecture 12, but explain in your own words what enables that generalization.

   1.5. What is the role of adding a noise term to the diagonal of the covariance matrix (kernel matrix)? In your answer also include an explanation for why it is good practice to use a very small noise value (but not zero) even if your data is noiseless.

   1.6. Explain what is aleatoric uncertainty.

   1.7. Explain what is epistemic uncertainty.

   1.8. Explain how we can use gridsearch and k-fold cross validation to find better hyperparameters (and even to find better models)[1]. In your answer, please explain why only performing a gridsearch without cross-validation is less robust, although it would still make it possible to choose hyperparameters (or models).

   1.9. (**BONUS**[2] **question**) The technique mentioned in 1.8. for hyperparameter (and model) selection is simple, but there are more interesting strategies. One alternative is to perform hyperparameter optimization (using optimization algorithms). Although we have not covered optimization yet, explain what is the difference and potential advantages between using this strategy for estimating better hyperparameters as opposed to the strategy discussed in 1.8.

Exercise 2

In this exercise you will redo what you already did in Homework 4 for Linear Least Squares regression, but now using Gaussian processes. This will allow you to establish a comparison between these models.

[1]For example, in Homework 4 you noticed that k-fold cross-validation provides a robust way to quantify the quality of approximation for a given hyperparameter or model choice (in that homework the polynomial degree was the hyperparameter). For clarity, we refer to a "model" or "model structure" as the group of choices we make including several hyperparameters and/or parameters that define it.

[2]This question is not mandatory. Only do the BONUS questions if you finished all regular questions. These questions award additional points (up to a top mark of 100% in the Homework)

Therefore, consider again the function to be learned as $f(x) = x \sin(x)$ within the domain $x \in [0, 10]$.

2. Gaussian process model for dataset without noise.

   2.1. Load the dataset without noise that you created in Homework 4:

   - Load the dataset as a pandas dataframe from the file "HW4_noiseless_dataset.csv" that you previously saved on the folder "your_data".
   - Like you did before, split the dataset in two sets (training and testing sets) using the "train_test_split" function of scikit-learn and consider 80% of the data is included in the training set. Set the "random_state" seed to 123.
   - **Important**: make sure that you use the same training and testing sets for fitting all models for this question as well as the remaining ones.

   2.2. Calculate the R^2 and MSE on the testing set after training Gaussian processes and where the columns of the table become the choice of different kernels: Constant*RBF[3], Mattern 5-2, Mattern 3-2, and Exponential-Sine-Squared. Table 1 presents an example of a possible way to present the results. Report the final parameters that result from the optimization.

Table 1: Suggested table to report R^2 (similar for MSE) in each cell.

kernel / n_{train}	Constant*RBF	Mattern 5-2	Mattern 3-2	Exponential-Sine-Squared
6				
11				
21				

Exercise 3

3. Gaussian process model for dataset **with** noise.

   3.1. Load the dataset with noise that you created in Homework 4:

   - Load the dataset "HW4_noisy_dataset.csv" from the "your_data" folder.
   - As before, split the dataset in two sets (training and testing sets) using the "train_test_split" function of scikit-learn and consider 80% of the data is included in the training set. Set the "random_state" seed to 123, as previously.

   3.2. Repeat exercise 2.2. but now for the dataset with noise. For this exercise, you can assume that you know the noise level (standard deviation) at every training point[4]. Present the previously mentioned error metrics as suggested in Table 1 and show 4 figures with the plots for each kernel choice when considering 11 training points. **What can you conclude from these results**?

   3.3. Use a kernel that results from adding a White Kernel to a Constant*RBF kernel, and try to estimate the noise level directly from the data when considering 6, 11 and 21 training points. **Include the 3 figures** with the approximations for different number of training points. **Comment the results you obtained, and do not forget to report the bounds you used for the different hyperparameters of your kernel** as well as the final hyperparameter values that were obtained for each case.

   - Note 1: you can only use the information of the training points, i.e. now (unlike Exercise 2) you cannot provide the standard deviation at each training point[5].
   - Note 2: you can try to adjust the bounds you consider for the hyperparameter definition in the kernel, but once you settled on particular bounds, make sure that you use the same bounds for all training sets, i.e. 6, 11 and 21 points.

[3]Some references simply call the Constant*RBF kernel as RBF kernel. However, scikit-learn does not.

[4]If you did not save the noise that you used in Homework 4, just generate a new dataset following the procedure you used in that homework and using the same seed number.

[5]This is not just an academic exercise. It is very common to have data for which we do not know the aleatoric uncertainty associated to each point!.