TASK 1 - GAUSSIAN PROCESS REGRESSION

Graded task: Gaussian Process Regression for ground-water pollution prediction.

You are in the group Potato consisting of jacobsr (jacobsr@student.ethz.ch (mailto://[u'jacobsr@student.ethz.ch'])), pgrontas (pgrontas@student.ethz.ch (mailto://[u'pgrontas@student.ethz.ch'])) and zsheebael (zsheebael@student.ethz.ch (mailto://[u'zsheebael@student.ethz.ch'])).

1. READ THE TASK DESCRIPTION

2. SUBMIT SOLUTIONS

3. HAND IN FINAL SOLUTION

1. TASK DESCRIPTION

BACKGROUND: GAUSSIAN PROCESS REGRESSION

According to the United Nations, one in three people worldwide do not have access to safe drinking water. Unsafe water is a leading risk factor for death, especially at low incomes, and is one of the world's largest health and environmental problems. Groundwater pollution occurs when pollutants are released into the ground and make their way down into groundwater. While water contamination can occur from naturally occurring contaminants, such as arsenic or fluoride, common causes of water pollution are on-site sanitation systems, effluent from wastewater treatment plants, petrol filling stations or agricultural fertilizers.

In order to prevent outbreaks and incidents of water poisonings, detecting ground-water contamination is crucial. Geostatistics has often utilized the Gaussian Process (GP) to model the spatial pattern of pollutant concentrations in the ground. Usually, a data point in 2D represents a geological well where a sample was taken from a bore hole to measure concentration of pollutants.

In the following task, you will use Gaussian Process regression (or a similar method) in order to model groundwater pollution, and try to predict the concentration of pollutants at previously unmeasured wells (points).

CHALLENGES

We envisage that in order to solve this task you need to overcome three challenges - each requiring a specific strategy.

- **1. Model selection** - You will need to find the right kernel and its hyper-parameters that model the GP faithfully. With Bayesian models, a commonly used principle in choosing the right kernel or hyper-parameters is to use the "data likelihood", otherwise known as the marginal likelihood to find the best model. See more details at: Wiki (https://en.wikipedia.org /wiki/Marginal_likelihood)
- **2. Large scale learning** - Natively, GP inference is computationally intensive for large datasets and common-place computers. The inference requires $\mathcal{O}(N^3)$ basic operations in order find the posterior distributions. For large datasets this becomes infeasible. In order to solve this problem, practitioners use forms of low-rank approximations. The most popular are the Nyström method, using random features and/or other scalable clustering-based approaches. This excellent review on Wikipedia can serve as an introduction: Wiki (http://en.wikipedia.org /wiki/Low-rank_matrix_approximations).
- **3. Asymmetric costs**: We utilize a specifically designed cost function, where deviation from the true concentration levels is penalized, and you are rewarded for correctly predicting safe regions. Under this specific cost function, the mean prediction might not be optimal. Note that the mean prediction refers to the optimal decision with respect to a general squared loss and some posterior distribution over the true value to be predicted.

PROBLEM SETUP

Download handout (/static/task1_handout_e8050ae9.zip)

The handout you download contains the following files: *train_x.csv, train_y.csv, test_x.csv and test_y.byte.* The outline of train_x.csv file is:

9.597315436241611319e-01,	5.838926174496643640e-01
2.013422818791954505e-02,	-9.463087248322147316e-01
...	...

The two columns are the 2D coordinates of the wells in a map, with the pollutant concentrations represented by the Y values in *train_y.csv*. Notice that each measurement is repeated 3 times and part of the evaluated domain has been sampled more precisely. The test dataset is organized in the same way but is split into two parts. The PUBLIC score represents a score on the first part while the PRIVATE score is the score on the first and second parts combined. This split is not revealed to you. You can use the provided script (see below for the workflow) to generate a PUBLIC score. The PRIVATE score is revealed only once you submit to the server.

Your task is to implement a class *Model* in *solution.py*, which contains at least the two methods *fit_model* and *predict* with the arguments as given in the *solution.py* template. The following table gives PUBLIC and PRIVATE score for our baseline.

PUBLIC	**PRIVATE**
0.064	0.069

METRICS

As mentioned in the third challenge, the cost function used here is asymmetric and rather involved. We assume the existence of a threshold value $\theta = 0.5$, which tells us whether the pollution is safe (values below θ) or unsafe (values above θ) for humans. The cost function is designed to heavily penalize predictions that predict lower than true levels of contaminant. Additionally, predicting safe values for unsafe points is especially penalized. The exact form of cost is given below where $\hat{f}(x)$ is the prediction at x and $f(x)$ is the true value at x.

$$\mathcal{L}_C(f(x), \hat{f}(x)) = \begin{cases} 1 \times (f(x) - \hat{f}(x))^2 & \text{if } f(x) > \theta, \hat{f}(x) \geq f(x) \text{ or } f(x) \leq \theta, \hat{f}(x) > f(x), \\ 20 \times (f(x) - \hat{f}(x))^2 & \text{if } f(x) > \theta, f(x) > \hat{f}(x) \geq \theta \text{ or } f(x) \leq \theta, \hat{f}(x) \leq f(x), \\ 100 \times (f(x) - \hat{f}(x))^2 & \text{if } f(x) > \theta, \hat{f}(x) < \theta. \end{cases}$$

On top of that, the final score is amended such that for each correctly predicted safe datapoint the cost is reduced by a constant amount. This gives the loss

$$\mathcal{L}oss = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}_C(f(x_i), \hat{f}(x_i)) - 0.04 \times \frac{1}{n}\sum_{i=1}^{n} 1(f(x_i) < \theta \text{ and } \hat{f}(x_i) < \theta)$$

where n is the number of test data points and 1 represents the indicator function. The python implementation of this metric can be found in *solution.py*. The visual interpretation is provided below:

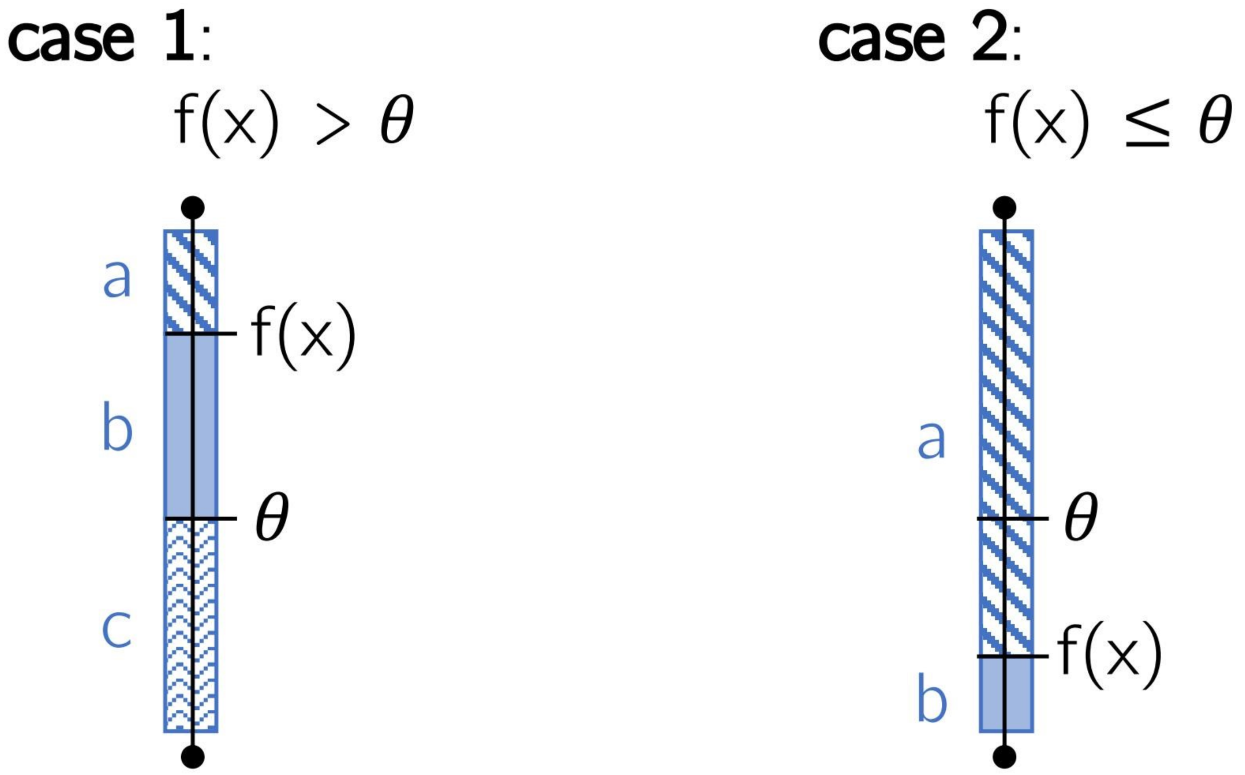

Figure 1: Cost Function: θ represents the threshold and $f(x)$ the true concentration value.

SUBMISSION WORKFLOW

1. Install and start Docker (https://www.docker.com/get-started). Understanding how Docker works and how to use it is beyond the scope of the project. Nevertheless, if you are interested, you could read about its use cases (https://www.docker.com/use-cases).
2. Download handout (/static/task1_handout_e8050ae9.zip)
3. The handout contains the solution template `solution.py`. You should write your code `# TODO:`

`enter your code here` marker in the solution template. You are supossed to implement a Model class that implements the pre-defined methods. You can add new methods but the evaluation script relies on existence of the predefined methods.
4. You should use Python 3.8.5. You are free to use any other libraries that are not already imported in the solution template. Important: please make sure that you list these additional libraries together with their versions in the `requirements.txt` file provided in the handout.
5. Once you have implemented your solution, run the checker in Docker:
   - if you are using Linux or MacOS, run `bash runner.sh`. On some operating systems, you might need to run `sudo bash runner.sh` if you see a Docker permission denied error.
   - if you are using Windows, open a PowerShell, change the directory to the handout folder and run `docker build --tag task0 .; docker run --rm -v "$(pwd):/results" task0`
6. If the checker fails, it will display an appropriate error message. If the check was successful, then a file called `results_check.byte` will be generated. The output will give you information about public score. You should upload this file to the project server to get the overall score which matters for passing this project along with the code and text description.
7. The split between public/private is so that you cannot overfit to the test set.
8. The submissions to the server are limited to 18 per team, and to 6 within 24h timeframe per team.

GRADING

When handing in the task, you need to select which of your submissions will get graded and provide a short description of your approach. This has to be done **individually by each member** of the team. Your submisssion is graded as either **pass** or **fail**. A complete submission typically consists of the following **three components**:

- **Submission file**: The `results_check.byte` file generated by the `runner.sh` script which tries to execute your code and checks whether it fulfills the requirements of the task.
- Your **code** in form of a .py or .zip file. The source code must be runnable and able to reproduce your uploaded `results_check.byte` file.
- A **description** of your approach which is consistent with your code. If you do not hand in a description of your approach, you may obtain zero points regardless of how well your submission performs.

To pass the task, your submission needs to be complete and **outperform the baseline** in terms of public and private (if the task has one) scores. Some tasks only have a single score on which you have to improve upon the baseline. Other tasks have a public and private score. In order to pass such tasks, you need to achieve a higher compound score (average of public and private score) than the baseline.

⚠ Make sure that you properly hand in the task, otherwise you may obtain zero points for this task. If you successfully completed the hand-in, you should see the respective task on the overview page shaded in green.

FREQUENTLY ASKED QUESTIONS

⦿ WHICH PROGRAMMING LANGUAGE AM I SUPPOSED TO USE? WHAT TOOLS AM I ALLOWED TO USE?

You should implement your solutions in Python 3. You can use any publicly available code, but you should specify the source as a comment in your code.

⦿ AM I ALLOWED TO USE METHODS THAT WERE NOT TAUGHT IN THE CLASS?

Yes. Nevertheless, the baselines were designed to be solvable based on the material taught in the class up to the second week of each task.

IN WHAT FORMAT SHOULD I SUBMIT THE CODE?

If you changed only the solution file, you can submit it alone. If you changed other files too, then you should submit all changed files in a zip of size max. 1 MB. You can assume that all files from the handout that you have not changed will be available to your code.

WILL YOU CHECK / RUN MY CODE?

We will check your code and compare it with other submissions. If necessary, we will also run your code. Please make sure that your code is runnable and your results are reproducible (fix the random seeds, etc.). Provide a readme if necessary.

SHOULD I INCLUDE THE HANDOUT DATA IN THE SUBMISSION?

No. You can assume the data will be available under the same path as in the handout folder.

CAN YOU HELP ME SOLVE THE TASK? CAN YOU GIVE ME A HINT?

As the tasks are a graded part of the class, **we cannot help you solve them**. However, feel free to ask general questions about the course material during or after the exercise sessions.

CAN YOU GIVE ME A DEADLINE EXTENSION?

We do not grant any deadline extensions!

CAN I POST ON PIAZZA AS SOON AS HAVE A QUESTION?

This is highly discouraged. Remember that collaboration with other teams is prohibited. Instead,

- Read the details of the task thoroughly.
- Review the frequently asked questions.
- If there is another team that solved the task, spend more time thinking.
- Discuss it with your team-mates.

WHEN WILL I RECEIVE THE PROJECT GRADES?

We will publish the project grades before the exam the latest.