Task 1 - Gaussian Process Regression

1. Task description

Background: Gaussian Process Regression

According to the World Health Organization, air pollution is a major environmental health issue. Both short- and long-term exposure to polluted air increases the risk of heart and respiratory diseases. Hence, reducing the concentration of particulate matter (PM) in the air is an important task.

You are commissioned to help a city predict and audit the concentration of fine particulate matter ($PM_{2.5}$) per cubic meter of air. In an initial phase, the city has collected preliminary measurements using mobile measurement stations. The goal is now to develop a pollution model that can predict the air pollution concentration in locations without measurements. This model will then be used to determine particularly polluted areas where permanent measurement stations should be deployed

A pervasive class of models for weather and meteorology data are Gaussian Processes (GPs). In the following task, you will use Gaussian Process regression in order to model air pollution and try to predict the concentration of $PM_{2.5}$ at previously unmeasured locations.

Challenges

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

- **1. Model selection**: You will need to find the right kernel and its hyperparameters that model the data faithfully. With Bayesian models, a commonly used principle in choosing the right kernel or hyperparameters is to use the *data likelihood*, also known as the marginal likelihood. See more details here: Wikipedia.
- **2. Large scale learning**: GP inference grows computationally expensive for large datasets. In particular, posterior inference requires $\mathcal{O}(n^3)$ basic operations which becomes computationally infeasible for large datasets. Thus, you will have to mitigate computational issues. Practitioners often do so using forms of low-rank approximations. The most popular instances are the Nyström method, using random features, and clustering-based approaches. The following excellent review on Wikipedia can serve as an introduction: Wikipedia.
- **3. Asymmetric cost**: We use a specialized cost function that weights different kinds of errors differently. As a result, the mean prediction might not be optimal. Here, 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. Thus, you might need to develop a custom decision rule for your model.

Problem setup

Download handout

The handout contains the data (`train_x.csv`, `train_y.csv`, `test_x.csv` and `test_y.byte`),

a solution template (`solution.py`), and other files required to run your solution and generate a submission.

Your task is to implement a class `Model` in `solution.py` that contains at least the two methods `fit_model` and `predict` with the signature as given in the `solution.py` template. We do not expect you to implement Gaussian Process regression from scratch and encourage you to use a library.

The training features (`train_x.csv`) are the (relative) 2D coordinates of locations on a map. `train_y.csv` contains the corresponding pollution measurements for each location in $^{PM_{2.5}}/_{m^3}$. We note that the values in `train_y.csv` contain additional measurement noise. The following is an outline of the training features

lon	lat
8.575000000000000400e-01	6.862500000000000266e-01
4.112500000000000044e-01	6.750000000000000444e-01
...	...

with corresponding measurements

pm25
3.620316838370916201e+01
5.594634794425741831e+01
...

The test dataset follows the same distribution as the training dataset and is organized in the same way. However, the test data does not contain measurement noise and is split into two parts. This split is not revealed to you. We calculate the PUBLIC cost only on the first part of the data while we use all test samples for the PRIVATE cost. You can use the provided script (see below for the workflow) to generate a PUBLIC cost. However, you can observe the PRIVATE cost only once you submit to the server. For reference, our baseline achieves the following PUBLIC and PRIVATE costs:

PUBLIC	**PRIVATE**
29.202	30.020

You pass the task if your solution's PUBLIC cost is at most the PUBLIC cost of our baseline. The PRIVATE cost is *irrelevant for grading* and only counts for the leaderboard. We will invite the highest-scoring teams to present their approaches in a tutorial session after this task's hand-in deadline.

Metrics

As mentioned in the third challenge, the cost function is asymmetric and rather involved. We therefore provide an implementation in the solution template. The following paragraphs explain the cost function in more detail.

We use the squared error $\ell(f(x), \widehat{f}(x)) = (f(x) - \widehat{f}(x))^2$ to measure the difference

between your predictions $\widehat{f}(x)$ and the true pollution concentration $f(x)$ at location x. However, we weight different types of errors differently.

We assume the existence of a threshold value $\theta = 35.5$ above which pollution becomes harmful to humans. The city wants to monitor areas with pollution concentrations above the threshold more closely (e.g., by installing permanent monitoring stations), making those areas particularly important. Hence, if a true pollution concentration is above the threshold ($f(x) \geq \theta$) but your prediction is below ($\widehat{f}(x) < \theta$), the error at location x is weighted by a factor of 20.

However, the city fears that systematic overprediction of pollution concentrations drives away businesses and tourists. Our cost thus weights overprediction errors ($\widehat{f}(x) > f(x)$) by a factor of 5, independent of the threshold.

The weights for all other types of errors is simply 1. In summary, the loss at an individual location x is

$$\ell_W(f(x), \widehat{f}(x)) = (f(x) - \widehat{f}(x))^2 \times \begin{cases} 5 & \text{if } \widehat{f}(x) > f(x) \quad \text{(overprediction)} \\ 20 & \text{if } f(x) \geq \theta, \widehat{f}(x) < \theta \quad \text{(underpredicting threshold)} \\ 1 & \text{otherwise} \end{cases}$$

where $\widehat{f}(x)$ denotes your prediction and $f(x)$ the true pollution concentration. We aggregate individual per-sample losses by taking the mean over all n samples, yielding the following overall cost function:

$$\mathcal{L}(\widehat{f}) = \frac{1}{n} \sum_{i=1}^{n} \ell_W(f(x_i), \widehat{f}(x_i)) \,.$$

Submission workflow

1. Install and start Docker. 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 Docker's use cases.
2. Download handout
3. The handout contains the solution template `solution.py`. You should implement the predefined `fit_model` and `predict` methods of the `Model` class in the solution template. You may not remove or change the signature of the predefined methods since the evaluation script relies on them. However, you are free to introduce new methods if required. Note: The `main()` method in the solution template is *for illustrative purposes only* and *completely ignored* by the checker!
4. You should use Python 3.8. You are free to use any other libraries that are not already imported in the solution template. Important: please make sure that you list all additional packages 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:
    - On Linux, run `bash runner.sh`. In some cases, you might need to enable Docker for your user if you see a Docker permission denied error.
    - On MacOS, run `bash runner.sh`. Docker might by default restrict how much memory your solution may use. Running over the memory limit will result in docker writing "Killed" to the terminal. If you encounter out-of-memory issues you can increase the limits as described in the Docker Desktop for Mac user manual. We do not support ARM-based M1 MacBooks yet. If you own such a device, you can resort to the Euler cluster.

Please follow the guide specified by *euler-guide.html* in the handout.
    - On Windows, open a PowerShell, change the directory to the handout folder, and run `docker build --tag task1 .; docker run --rm -v "$(pwd):/results" task1`.
6. If the checker fails, it will display an appropriate error message. If the checker runs successfully, it will show your PUBLIC score, tell you whether your solution passes the task, and generate a `results_check.byte` file. The `results_check.byte` file constitutes your submission and needs to be uploaded to the project server along with the code and text description in order to pass the project.
7. You pass this task if your solution's cost on the PUBLIC test data is at most the cost of our baseline on the PUBLIC test data. The PRIVATE cost determines your position on the leaderboard but it does *not* determine whether you pass/fail this task.
8. We limit submissions to the server to 18 per team.

Extended evaluation

This part of the task is optional, but highly encouraged. Once your solution passes the checks, set the variable `EXTENDED_EVALUATION` in your `solution.py` to `True`. Running the checker again will then create three plots visualizing your model and save them as `extended_evaluation.pdf`:

1. A 2D map of your actual predictions (after applying your decision rule, if you use one).
2. A 2D map of the standard deviations estimated by your GP.
3. A 3D map visualizing the raw estimates of your GP. At each location, the height corresponds to the GP mean while the color corresponds to the estimated standard deviation.

The extended evaluation is disabled by default since it can be computationally expensive. You can play around with the `EVALUATION_GRID_POINTS` variable which determines the number of points your model is evaluated on, and the `EVALUATION_GRID_POINTS_3D` variable which controls the resolution of the 3D map.

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 that 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 score. Some tasks only have a single score on which you have to improve upon the baseline. Other tasks have a PUBLIC and PRIVATE score. The PRIVATE score determines your position on the server leaderboard but is irrelevant for grading.

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 3.8. 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 Moodle 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.