☰ TASK 1

Graded task: Gaussian Process Regression for Air Pollution Prediction.

> You are in the group 👥 Babedibubedi consisting of 👤 inauenma (inauenma@student.ethz.ch (mailto://inauenma@student.ethz.ch)), 👤 nisutte (nisutte@student.ethz.ch (mailto://nisutte@student.ethz.ch)) and 👤 stheeb (stuart.heeb@inf.ethz.ch (mailto://stuart.heeb@inf.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 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 suitable residental areas with low air pollution. The city already determined a couple of candidate locations for new residental areas, based on other relevant parameters such as infrastructure, distance to city center, etc.

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 (https://en.wikipedia.org/wiki/Marginal_likelihood).
- **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 a number of methods, among which you can opt for one of the following:
  - Undersampling: Sampling a subset from our initial dataset which is used for learning, either randomly or with clustering-based approaches.
  - Kernel low-rank approximations: Effectively avoid inverting the full kernel matrix. The most popular instances are the Nyström method and random Fourier features. The following excellent review on Wikipedia can serve as an introduction: <u>Wikipedia (http://en.wikipedia.org/wiki/Low-rank_matrix_approximations)</u>. (Hint: for this kernel approximation method, you won't need a sklearn.gaussian_process GP model)
  - Approximation of GP with multiple local GPs: For certain kernels, mutual dependence of distant samples diminish, so training a local GP for every region of our domain using the corresponding subset of local data samples from the original dataset can approximate the use of one global GP within that locallity.
  Of course, implementation of other methods or your unique solutions is highly encouraged.
- **3. Asymmetric cost**: In candidate areas for new residential areas, the city wants to avoid underestimating the pollution concentration. Therefore, 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 in those areas.

PROBLEM SETUP

> Download handout (/static/task1_handout_d3d63876.zip)

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 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, along with the binary variable denoting whether that 2D point lies in a candidate resdental area or not. `train_y.csv` contains the corresponding pollution measurements for each location in $^{PM2.5}/_{m^3}$. We note that the values in `train_y.csv` contain additional measurement noise. The following is an outline of the training features

lon	lat	area_id
8.575000000000000400e-01	6.862500000000000266e-01	0
4.112500000000000044e-01	6.750000000000000444e-01	1
...	...	...

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. You can use the provided script (see below for the workflow) to generate a cost for your solution. For reference, our public baselines achieve the following cost:

PUBLIC BASELINE COST	EASY	MEDIUM	HARD
	45.480	26.722	21.719

METRICS

As mentioned in the third challenge, the cost function in candidate residential areas (binary variable **area_id** value **1**) is asymmetric. We therefore provide an implementation in the solution template. The following paragraphs explain the cost function in more detail.

We use the squared error $\ell(f(x), \hat{f}(x)) = (f(x) - \hat{f}(x))^2$ to measure the difference between your predictions $\hat{f}(x)$ and the true pollution concentration $f(x)$ at location x. However, we weight different types of errors differently.

The city wants to monitor candidate residential areas closely, making measurements in those areas particularly important. Hence, if a true pollution concentration for a point in this area is above your estimation($\hat{f}(x) \leq f(x)$) the error at location x is weighted by a factor of 50. Overpredictions and errors out of the residential areas (binary variable **area_id** value **0**) are weighted by a factor of 1.

In summary, the loss at an individual location x is

$$\ell_W(f(x), \hat{f}(x)) = (f(x) - \hat{f}(x))^2 \cdot \begin{cases} 50, & \text{if } \hat{f}(x) \leq f(x) \quad and \quad area\_id(x) == 1 \quad \text{(underprediction in potential residential zor} \\ 1, & \text{if } f(x) < \hat{f}(x) \quad \text{(else)} \end{cases}$$

We aggregate individual per-sample losses by taking the mean over all n samples, yielding the following overall cost function:

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

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_d3d63876.zip)

3. The handout contains the solution template `solution.py`. You should implement the predefined `fitting_model` and `make_predictions` methods of the `Model` class in the solution template, as well as `extract_city_area_information` method, which is just a wrapper for a data preprocessing. 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 3.8.5. 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:
   - On Linux, run `bash runner.sh`. In some cases, you might need to enable Docker for your user (https://docs.docker.com/engine/install/linux-postinstall/#manage-docker-as-a-non-root-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 (https://docs.docker.com/desktop/mac/). Running over the memory limit will result in docker writing "Killed" to the terminal. You could alternatively run your solutions on the ETH euler cluster. Please follow the guide specified by *euler-guide.md* 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 test data is at most the cost of our baseline on the test data. It determines your position on the leaderboard as well as whether you pass/fail this task.

8. We limit submissions to the server to 40 per team, with at most 20 in a 24 hour period.

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 a plot visualizing your model and save them as `extended_evaluation.pdf`. The plot features a 2D map of your actual predictions (after applying your asymmetric cost rule, if you use one).

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.

GRADING

We provide you with **one test set** for which you have to compute predictions (and any other quantities we ask for, like uncertainty quantification). We have partitioned this test set into two parts (of the same size) and use it to compute a *public* and a *private* score for each submission. You only receive feedback about your performance on the public part in the form of the public score, while the private leaderboard remains secret. The purpose of this division is to prevent overfitting to the public score. Your model should generalize well to the private part of the test set. 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. We will then compare your selected submission to our baseline. Your final grade for this task is the mean of the grades of your public and private scores. We will publish 2 baselines: easy and hard that each has a public and private score. Beating the easy baseline's public/private score guarantees you a 4.0 public/private grade respectively, while beating the hard baseline's public/private score guarantees you a 6.0 public/private grade respectively. Failing to beat the easy baseline's public/private score grants you a 2.0 public/private grade. For scores better than the easy baseline but worse than the hard baseline, we will grant a grade between 4.0 to 6.0 depending on how close you are to the hard vs easy baseline. We emphasise that this will likely not be a simple linear interpolation between the hard and easy baselines. In addition, for the pass/fail decision, we consider the code and the description of your solution that you submitted. That is, your code should be runnable and reproduce your predictions (see faq) and you should include a short writeup of your submitted solution. We emphasize that the public score leaderboard is just for fun: the scores of other teams will not effect the baseline or your own grade.

⚠ Make sure that you properly hand in the task, otherwise you may obtain zero points for this task.

FREQUENTLY ASKED QUESTIONS

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

You are free to choose any programming language and use any software library. However, **we strongly encourage you to use Python**. You can use publicly available code that was not produced directly for the purposes of this course, but you should specify the source as a comment in your code.

◉ AM I ALLOWED TO USE MODELS THAT WERE NOT TAUGHT IN THE CLASS?

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

◉ IN WHAT FORMAT SHOULD I SUBMIT THE CODE?

You can submit it as a single file (main.py, etc.; you can compress multiple files into a .zip) having max. size of 1 MB. If you submit a zip, please make sure to name your main file as *main.py* (possibly with other extension corresponding to your chosen programming language).

◉ WILL YOU CHECK / RUN MY CODE?

We will check your code and compare it with other submissions. We also reserve the right to run your code. Please make sure that your code is runnable and your predictions are reproducible (fix the random seeds, etc.). Provide a readme if necessary (e.g., for installing additional libraries).

◉ SHOULD I INCLUDE THE DATA IN THE SUBMISSION?

No. You can assume the data will be available under the path that you specify in your code.

◉ 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 PRIVATE SCORES? AND THE PROJECT GRADES?

We will publish the private scores, and corresponding grades before the exam at the latest.