TASK 1

Graded task: Gaussian Process Regression for Air Pollution Prediction.

You are in the group Strictlyfifteen consisting of rachen (rachen@student.ethz.ch (mailto://rachen@student.ethz.ch)), wanhan (wanhan@student.ethz.ch (mailto://wanhan@student.ethz.ch)) and yutang (yutang@student.ethz.ch (mailto://yutang@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 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 (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: Wikipedia (http://en.wikipedia.org/wiki/Low-rank_matrix_approximations). (Hint: for this kernel approximation method, you won't need 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**: 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 (/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. `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. You can use the provided script (see below for the workflow) to generate a cost for your solution. For reference, our baseline achieves the following cost:

BASELINE COST
118.556

You pass the task if your solution's cost is at most the cost of our baseline. The cost is immediately available to you by running the local script (details provided later) and helps you evaluate your model. It also determines position on the leaderboard, where your score will appear after submission. **This particular task has no "private" score, only a public score.**

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 $\ell(f(x), \hat{f}(x)) = (f(x) - \hat{f}(x))^2$ 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 areas with high pollution concentrations more closely (e.g. by installing permanent monitoring stations), making measurements in those areas particularly important. Hence, if a true pollution concentration is above your estimation($\hat{f}(x) \leq f(x)$) the error at location x is weighted by a factor of 25.

If your estimation is just slightly greater then real value $(f(x) \leq \hat{f}(x) < 1.01 \cdot f(x))$, error is not additionally penalized and the error at location x is weighted by a factor of 1.

However, the city fears that systematic overprediction of pollution concentrations (false alarms, (\ 1.01 · f(x) \leq \hat{f}(x) \)) drives away businesses and tourists. Our cost thus weights overprediction errors by a factor of 10.

In summary, the loss at an individual location x is

$$\ell_W(f(x), \hat{f}(x)) = (f(x) - \hat{f}(x))^2 \cdot \begin{cases} 25, & \text{if } \hat{f}(x) \leq f(x) & \text{(underprediction)} \\ 1, & \text{if } f(x) \leq \hat{f}(x) < 1.01 \cdot f(x) & \text{(slight overprediction)} \\ 10, & \text{if } 1.01 \cdot f(x) \leq \hat{f}(x) & \text{(significant overprediction)} \end{cases}$$

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 `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 3.8.5 and sklearn 0.23. 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. Note that some required Python packages do not support ARM-based MacBooks (M1 or M2) yet. On those machines, you can install a VM application such as UTM (https://mac.getutm.app/) and emulate an *x86-based* OS such as Linux or Windows. 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 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

Some tasks have a public and private score. This task has **only a public score**. Your algorithm will make predictions on a held out test set, or (for tasks 3 and 4) obtain a single score for its interactions with the environment. 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. This project task is graded with either **pass (6.0) or fail (2.0)**. To pass the project, you need to achieve a better score than the baseline. In addition, for the pass/fail decision, we consider the code and the description of your solution that you submitted. 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. The following **non-binding** guidance provides you with an idea on what is expected to pass the project: If you hand in a properly-written description, your source code is runnable and reproduces your predictions, and your submission performs better than the baseline, you can expect to have passed the assignment.

⚠ 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, 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 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 at exam the latest.