Task 3 - Bayesian Optimization

1. Task description

Task

As seen in the lectures, Bayesian Optimization (BO) is a powerful framework for finding optimums while using very few function evaluations. However, in many real-world applications, the search space is often subject to constraints limiting the feasible domain. When evaluating the feasibility of a candidate solution becomes expensive, modeling the feasible domain jointly with the objective function becomes crucial. To this effect, in this task, you are asked to extend what you've seen in the lectures on Bayesian Optimization: to find the minimum of an objective function f subject to a constraint c. Let $\mathcal{X} \subset \mathbb{R}^2$ denote the input space of f and c. For a given threshold λ, your goal is to find

$$f(x^*) = \min_{c(\mathbf{x}) \leq \lambda} f(\mathbf{x}) \, .$$

However, contrary to unconstrained BO, c is also unknown. Thus, when selecting a new candidate $\widehat{x}$, you not only get an observation of the objective $f(\widehat{x})$ but also of the constraint $c(\widehat{x})$.

One important thing to note is that only the final solution returned by your BO algorithm has to respect the constraint. Let x_i be the candidate evaluated at the i^{th} iteration of the Bayesian Optimization algorithm. While running the algorithm, you may try hyperparameters for which the constraint is violated, i.e., $c(x_i) > \lambda$.

Below, you can find the quantitative details of this problem.

- The domain is $\mathcal{X} = [0, 6]^2$.
- The observation noise is Gaussian with standard deviation $\sigma_f = 0.01$ and $\sigma_v = 0.005$ for the objective and the constraint, respectively.
- The mapping f can be effectively modeled with as a product of a constant kernel with scale 1.5 and an RBF kernel with lengthscale 1.5.
- The mapping c can be effectively modeled with as a product of a constant kernel with scale 3.5 and an RBF kernel with lengthscale 2.
- For ease of implementation, we always subtract λ from the constraint so that you do not need to care about different λ values in your implementation. More precisely, for all λ, your solution observes $c'(x) = c(x) - \lambda$ and must satisfy $c'(\widehat{x}) \leq 0$. λ will take 3 different values per setting to gradually make the problem harder.

In order to get a better idea of problem as well as debug your code, we provide a toy example in $solution.py$. This toy example follows the same characteristics described above but **it will not be used to evaluate your method**. When evaluating your solution, we perform a sanity check by running your method on this toy example. You can obtain additional feedback on this toy data by setting `EXTENDED_EVALUATION = TRUE` in `solution.py` as further explained in the dedicated section **Extended Evaluation**.

Evaluation

In this kind of problems, we are interested in minimizing the normalized regret for not

knowing the best function value. However, if the solution violates the constraint, then the error should be maximal. Let $\widetilde{x}$ be the optimal solution suggested by your algorithm and $f_m = \max_{x \in \mathcal{X}} |f(x)|$. We define the normalized regret as follows:

$$r = \begin{cases} \frac{f(\widetilde{x}) - f(x^*)}{f_m} & c(\widetilde{x}) \leq \lambda \\ 1 & c(\widetilde{x}) > \lambda \end{cases}.$$

To evaluate your algorithm, we generate 9 PUBLIC random tasks ($f(\cdot)$ and $c(\cdot)$) with 3 different difficulty levels (i.e., different λ), yielding a total of 27 tasks. Your final score is the mean normalized regret over all PUBLIC tasks:

$$R = \frac{1}{27} \sum_{j=1}^{27} r_j \,.$$

You successfully pass this task if $R \leq 0.20$. Note that r_j are random variables. However we set the baseline to take the variance of R into account; you don't need to worry about this randomness. We further evaluate your solution on some additional PRIVATE tasks. The mean normalized regret over all PUBLIC and PRIVATE tasks then constitutes your PRIVATE score. The PRIVATE score determines your leaderboard position and is only provided to you upon submission. Because the PRIVATE tasks are harder, the PRIVATE score might be higher than 0.20, but this won't change whether you pass the project.

Hint: This task is designed such that a correct implementation of a standard Bayesian Optimization algorithm (i.e., unaware of the constraint) as presented in Snoek et al. (2012), might not suffice to pass. We strongly suggest that you take the constraint into account. For example, by making the choice of x_i dependent on the likelihood of fulfilling a constraint, you should obtain a better score. For further reading on the topic, see Gelbart et al. (2014).

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 write your code right below the `# TODO: enter your code here` markers in the `BO_algo` class in the solution template. You shouldn't modify any of the predefined classes or methods, but only add your code to the indicated spaces. However, you are free to introduce new methods and attributes. 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 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 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 task3 .; docker run --rm -v "$(pwd):/results" task3`.

6. If the checker fails, it will display an appropriate error message. If the checker runs successfully, it will show you your PUBLIC average normalized regret R, tell you whether your solution passes this 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 your code and text description to pass this task.
7. 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 of the toy example, set the global variable `EXTENDED_EVALUATION` in the solution script to `True`. Running the toy example again will then create a PDF output with plots of some features of your current solution **for the toy problem only**.

1. The first sub-figure is a heatmap of your modelled objective function. Overlaid in red is the true infeasible domain. The red round points correspond to the optimal solution whereas the blue cross is your solution estimate.
2. The second sub-figure contains a 3D plot of your objective function estimate.
3. The third sub-figure contains a 3D plot of your constraint function estimate.
4. The last sub-figure shows a heatmap of your modelled constraint function. Overlaid in red is the true infeasible domain. The blue cross points correspond to the successive candidates your algorithm proposed to sample from during training.

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.