☰ TASK 3

Hyperparameter tuning with Bayesian optimization

You are in the group 👥 Randomguess consisting of 👤 pingli (pingli@student.ethz.ch (mailto://pingli@student.ethz.ch)), 👤 umbacht (umbacht@student.ethz.ch (mailto://umbacht@student.ethz.ch)) and 👤 yucsun (yucsun@student.ethz.ch (mailto://yucsun@student.ethz.ch)).

📖 1. READ THE TASK DESCRIPTION

🖥 2. SUBMIT SOLUTIONS

✉ 3. HAND IN FINAL SOLUTION

📖 1. TASK DESCRIPTION

TASK

In this task, you should use Bayesian optimization to tune one hyperparameter of a machine learning model subject to a constraint on a property of the model. Let $\theta \in \Theta$ denote the hyperparameter of interest, e.g., the number of layers in a deep neural network. We aim at finding a network that makes accurate and fast predictions. To this end, we can train our model for a specific value of θ on a training data set. From this, we obtain a corresponding accuracy on the validation data set and an average prediction speed. In this context, our goal is to find θ^*, i.e., the value of the hyperparameter that induces the highest possible validation accuracy, such that a requirement on the average prediction speed is satisfied.

More formally, let us denote with $f : \Theta \rightarrow [0,1]$ the mapping from the space of hyperparameter to the corresponding validation accuracy. When we train with a given θ, we observe $y_f = f(\theta) + \varepsilon_f$, where $\varepsilon_f \sim \mathcal{N}(0, \sigma_f^2)$ is zero mean Gaussian i.i.d. noise. Moreover, we denote with $v : \Theta \rightarrow \mathbb{R}^+$ the mapping from the space of hyperparameters to the corresponding prediction speed. Similar to the accuracy, we observe a noisy value of the speed, which we denote with $y_v = v(\theta) + \varepsilon_v$, with $\varepsilon_v \sim \mathcal{N}(0, \sigma_v^2)$ zero mean Gaussian i.i.d. noise. Then, the problem is formalized as,

$$\theta^* \in \operatorname{argmax}_{\theta \in \Theta, v(\theta) > \kappa} f(\theta),$$

where κ is the minimum tolerated prediction speed. The objective of this problem does not admit an analytical expression, is computationally expensive to evaluate and is only accessible through noisy evaluations. Therefore, it is well suited to Bayesian optimization (see [1] for further reading on Bayesian optimization for hyperparameter tuning).

In this project, you need to solve the hyperparameter tuning problem presented above with Bayesian optimization. Let θ_i be the hyperparameter evaluated at the ith iteration of the Bayesian optimization algorithm. **While running the Bayesian Optimization algorithm, you may try hyperparameters for which the speed constraint is violated, i.e. $v(\theta_i) < \kappa$. However, the final solution must satisfy $v(\theta) \geq \kappa$.**

Remarks: In the motivating example above, θ takes discrete values (number of layers is a Natural number) and the objective and the constraint can be evaluated independently. However, to keep the problem simple, we let θ be continuous and we evaluate f and v simultaneously. Moreover, to avoid unfair advantages due to differences in computational power, the training of the neural network is simulated and, therefore, the time required for this step is platform independent. This task does not have a private score.

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

- The domain is $\Theta = [0, 5]$.
- The noise perturbing the observations is Gaussian with standard deviation $\sigma_f = 0.15$ and $\sigma_v = 0.0001$ for the accuracy and the speed, respectively.
- The mapping f can be effectively modeled with a Matérn (https://en.wikipedia.org/wiki/Mat%C3%A9rn_covariance_function) kernel with variance 0.5, lengthscale 0.5 and smootheness parameter $\nu = 2.5$.
- The mapping v can be effectively modeled with a constant mean of 1.5 and a Matérn (https://en.wikipedia.org/wiki/Mat%C3%A9rn_covariance_function) kernel with variance $\sqrt{2}$, lengthscale 0.5, and smootheness parameter $\nu = 2.5$.
- The minimum tolerated speed is $\kappa = v^{\min} = 1.2$. The unit of measurement is not relevant as the training is only simulated.

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 Docker's use cases (https://www.docker.com/use-cases).
2. Download handout (/static/task3_handout.zip). Alternatively you can use the handout for Macbook M1 devices (/static/task3_handout_M1.zip). See below for more details.
3. The handout contains the solution template `solution.py`. You need to implement all sections marked with `# TODO: ...`. You are free to implement the `plot` method for visualization. However, the method in the solution template is *for illustrative purposes only* and is *completely ignored* by the checker! Please make sure that your implementation preserves the names and signatures of the `run_solution` method, as well as all methods in the classes of your selected approach. Please set the `approach` to the approach that you want to submit with, since the evaluation script relies on them. However, you are free to introduce new methods and attributes. Note: The `main()` method in the solution template is *for illustrative purposes only* and is *completely ignored* by the checker! It is representative of how your `BO_algo` will be used by the workflow of the checker, just on a toy example. You *may play around* with the toy example provided in the `solution` file.

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 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 (without Apple silicon), 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. Above you can see we have a handout specifically for Macbook M1 users. We expect the handout to also work on M2 machines but we have not tested it on an M2. M2 users might want to stick with using a virtual machine or the euler cluster with the regular handout. The M1 handout also requires Docker. The workflow with an M1 and the M1 handout is identical to the workflow for an Intel Mac (paragraph above) using the regular 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`.
   - If you are having trouble running your solution using docker locally, consider using the ETH Euler cluster to run your solution. Please follow the guide specified by *euler-guide.md* in the handout. The setup time of using the cluster means that this option is only worth doing if you really can't run your solution locally.

6. If the checker fails, it will display an appropriate error message. If the checker runs successfully, it will show you your PUBLIC score, 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. You pass this task if your PUBLIC score is lower than the baseline. More details are given in the Evaluation section below.

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

EVALUATION

In this kind of problems, we are interested in minimizing the normalized regret for not knowing the best hyperparameter value. Let $\tilde{\theta}$ be the optimal solution suggested by the algorithm, define the regret as follows:

$$r = \begin{cases} \frac{f(\theta^*) - f(\tilde{\theta})}{f(\theta^*)} & v(\tilde{\theta}) \geq \kappa \\ 1 & v(\tilde{\theta}) < \kappa \end{cases}$$

Which means, if the solution violates the minimum speed criteria, then $r = 1$. To evaluate your algorithm, 75 random hyperparameter tuning tasks as the one described above are generated and your final score is computed as

$$R = \frac{1}{75} \sum_{j=1}^{75} r_j.$$

You successfully pass the project if $R < 0.25$. Note that r_j are random variables, however the baseline has been set taking into account the variance of R. In other words, you don't need to worry about this randomness.

Hint: This task is designed such that a correct implementation of a standard Bayesian optimization algorithm (i.e. unaware of the speed constraint) might not be sufficient to pass. We strongly suggest taking into account the constraint on prediction speed. For example, by making the choice of θ_i dependent on how likely it is to lead to a fast network, you should obtain a better score. For further reading on the topic (in a more challenging setting than the one presented here), see [2].

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.

REFERENCES

[1] Snoek, Jasper, Hugo Larochelle, and Ryan P. Adams. "Practical bayesian optimization of machine learning algorithms." Advances in neural information processing systems. 2012. (https://papers.nips.cc/paper/4522-practical-bayesian-optimization-of-machine-learning-algorithms.pdf)
[2] Michael A. Gelbart, Jasper Snoek, and Ryan P. Adams. "Bayesian optimization with unknown constraints." In Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence. 2014. (https://www.cs.princeton.edu/~rpa/pubs/gelbart2014constraints.pdf)