Hyperparameter tuning - Bayesian optimization

December 2, 2022

1 Introduction

1.1 Folder content

You can find in the folder:

- A dataset (TODO little explanation)
- A jupyter notebook that you have to complete following this document and the course.
- requirements.txt for convenience.

2 Toy problem optimization

2.1 Definition

You can find a function called "toy function". It corresponds to a black box function, you are not supposed to use the math of this function to optimize it.

$$f(x) = -\alpha exp(-b\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}) - exp(\frac{1}{d}\sum_{i=1}^{d} cos(cx_i)) + \alpha + exp(1)$$

w.r.t:

- $\alpha = 20$
- $b = 0.2$
- $c = 2\pi$
- The space of the function is define in the class variable "domain"
- Number of dimensions can be changed, start with 1.

2.2 Random optimization

Complete the corresponding cell.

- Take N random points in your hyper-parameter space (you can find an example in the notebook)
- Evaluate your function at this N points.
- You can use this as the starting X, Y of the bayesian optimization algorithm.

3 Bayesian Optimization

3.1 Random initialization

Already done in 2.2.

3.2 Predictive model

Install a package that handles Gaussian Processes (I'm not so sadistic, you won't program a GP). GPy is a good one.

In the python class "mymodel", program the methods :

- "__init__()", where you instantiate your model (the GP)"
- "fit(X, Y)" where you fit your model on the data passed as arguments
- "predict(X)" where you make a prediction from the input. The output have be composed by two values...

3.3 Acquisition function

At this point, you should have :

- the 5 random points, save as "X_init, Y_init".
- a predictive model that you can instantiate and fit on "X_init, Y_init".

In the cell "Acquisition function", program an acquisition function similar to the one we have seen during the class (the same one, adapted for minimization).

3.4 Acquisition function optimization

A ready-to-use optimization tool is available in scipy.optimize.

Suggestions :

- Use scipy.optimize.minimize(f, xo).
- If you want to maximize, minimize the negative.
- Use a wrapper function for scipy.

3.5 Bayesian Optimization algorithm

At this point, you should have all the fundamentals of BO.

- Program the algorithm
- Run it on the toy problem to validate its behaviour (TODO, plotterfunction1D)
- Repeat the optimization multiple times and plot the mean of the best score evolution

4 Hyper-parameter tuning

You can find at the very end of the notebook a cell already completed. This cell is a regression evaluation of a standard Random Forest.

The optimization problem is already defined :

- Domain : inputs variables
- f : objective

You can optimize this model with your BO algorithm, or do the same for a model/data set of your choice !