Stochastic models and optimization - Problem set 3

Hrvoje Stojic

May 17, 2021

This is a programming assignment in which you should work on your own. The goal is to get some hands on experience with contextual bandit algorithms. You can choose between R and Python for the assignment. You are not allowed to use any of the functions from the packages that have already coded up these algorithms. I will evaluate your submission based on correctness of the implementation (and to some extent how well the code is written). Please make sure your solution is reproducible and operating system independent, I should be able to rerun your code and obtain your results/figures ideally with a single line command.

---

1. Gaussian processes

This problem focuses on Gaussian processes that is most often used as a function learning component of Bayesian optimization algorithm. It is also a powerful supervised learning algorithm that will be an important addition in your machine learning toolbox.

Your task will be to reproduce Figure 2.5 (and Figure 2.2-a) from Rasmussen & Willams (2006) book, by implementing their Algorithm 2.1.

In these illustrations the authors are using a GP with zero mean function and squared exponential (SE) kernel. In addition they assume noisy observations.

- First, code up the SE kernel and draw some sample functions based on a GP with SE and using SE parameters from Figure 2.5-a ($(l, \sigma_f, \sigma_n) = (1, 1, 0.1)$). These are draws from the prior, before observing any training data. For test points use 100 equidistant points on $(-7, 7)$ interval. You should obtain a figure similar to Figure 2.2-a. Connect the points so they form a line. Illustrate the shaded region as well, two times the standard deviation of the prior.

- Next, instead of implementing GP inference directly through conditioning equations, you will implement the Algorithm 2.1. Note that when doing Cholesky decomposition you need to add some very small number ϵ to the diagonal of the covariance matrix, for numerical stability.

- Then you will produce some training data. Draw 20 training points randomly from $U(-7, 7)$ and compute their function values by taking a single draw from the prior for these points.

- Finally, you are ready to reproduce posterior plots from Figure 2.5. Use the data from the previous step and produce predictions for 100 equidistant points on $(-7, 7)$ interval, same as for illustration of the prior. As in 2.5-a use $(l, \sigma_f, \sigma_n) = (1, 1, 0.1)$ when computing the covariance function, while in 2.5-b and 2.5-c you should use $(l, \sigma_f, \sigma_n) = (0.3, 1.08, 0.00005)$ and $(l, \sigma_f, \sigma_n) = (3, 1.16, 0.89)$ respectively. Connect the points so they form a line. Illustrate the shaded region as well, two times the standard deviation of the posterior.

2. Bayesian optimization

This problem focuses on implementing a few basic Bayesian optimization algorithm on one of the benchmark functions widely used in the literature.

The function that will pose as “an objective function very expensive to evaluate” is Branin or Branin-Hoo function. It’s a 2-dimensional function which has somewhat challenging surface consisting of three global minima. You can think of this as if you had a big neural network model with two hyperparameters, which takes a long time to train.

More precisely, the objective function has the following form:

$$f(\mathbf{x}) = a(x_2 - bx_1^2 + cx_1 - r)^2 + s(1-t)\cos(x_1) + s$$

where parameter values are: $a = 1$, $b = 5.1/(4\pi^2)$, $c = 5/\pi$, $r = 6$, $s = 10$ and $t = 1/(8\pi)$ (though some papers do change these parameters sometimes). The function should be evaluated on the square $x_1 \in [-5, 10]$ and $x_2 \in [0, 15]$. The global min is $f(\mathbf{x}^*) = 0.397887$ at following three points: $\mathbf{x}^* = (-\pi, 12.275), (\pi, 2.275), (9.42478, 2.475)$. Note that the objective here will be to find a min of the function, not the max.

You will implement three Bayesian optimization algorithms: GP combined with UCB, EI and PI acquisition functions. For GP part you can use your own code developed in the first exercise, with squared exponential kernel, or one of the popular libraries (e.g. GPflow). You should optimize hyperparameters in every iteration, after you observe a function value add a new point to the training data. If you use your own implementation, you will need to code that part up as well. No need to do something sophisticated here, you can use the maximum likelihood approach and optimize the marginal likelihood, and you can use for example a gradient free simplex method to optimize it. For optimizing the acquisition functions you can use a simple approach of discretising the 2-dimensional space in a fine grid of points.

Start the algorithms with five randomly chosen points from the Branin functions. Optimize hyperparameters of the GP on these points, make predictions and optimize acquisition function to make a choice for the next point to sample. Do this iteratively until 100 points are observed. Repeat this for all three acquisition functions along with 3 different values of their exploration parameters. You should produce a figure (or several figures) that compares performance of these 9 acquisition function, with iterations on x-axis and squared error between achieved value in that iteration and true max on y-axis.

Don’t hesitate to send me a message if anything is unclear.

Deadline is **May 24, 23:59 BCN time**. Submit your code and figures by uploading them to Classroom.