Advanced Model Predictive Control
Prof. M. Zeilinger
Dr. A. Carron
Dr. L. Hewing
Dr. J. Köhler

Fall 2022

PE Exercise 7

Bayesian Optimization

Alexandre Didier and Jérôme Sieber

Exercise

Solve the following exercises using the Matlab code provided on Moodle.

1. **Graded.** Implement the lower confidence bound (LCB) as the acquisition function in the `BO.m` class, where the LCB is defined as

$$q_i = \arg\min_q \ \mu_{i-1}^h(q) - \beta\sqrt{\text{diag}\left(\Sigma_{i-1}^h(q)\right)},$$

   with β the exploitation/exploration trade-off parameter. Feel free to play around with different values for β, but make sure to set $\beta = 1$ for the remaining exercises.
   Then, proceed to run all cells of the first part and observe how BO samples and learns the simple 1D function.

2. **Graded.** Now we proceed to the second part, where we use BO to tune a PID controller, hence the parameter vector q contains all the PID gains[1], i.e., $q = [K_P, K_I, K_D]^\top$. Our goal is to perform a swing-up with the segway system, while avoiding jittering control actions. Design a cost function $J(q)$ that achieves this goal, which uses the state and input trajectories resulting from simulating the segway system with PID gains q.
   Hint: See the recitation for some ideas on how to design such a cost function.
   Remark: Implement your cost function in both the initial data sampling **and** the BO iterations.

3. **Graded.** Implement the squared exponential kernel, i.e.,

$$K(x, x') = \lambda^2 \exp\left(-\frac{1}{2\sigma^2}\|x - x'\|^2\right),$$

   where σ, λ are hyperparameters. Please make sure you correctly address the dimensionality of your data, since the parameter vector is not scalar anymore, but $q \in \mathbb{R}^3$.
   Then, proceed to run all remaining cells of the second part and observe how BO samples and learns the PID gains that are optimal according to your cost function.
   Remark: The number of samples and iterations provided in the code should be enough to achieve sufficient control performance. However, depending on your cost function you might need to run BO for more iterations, thus feel free to change these parameters until you are satisfied with the performance.

4. **Not Graded.** The provided code template can also handle the optimization of the full parameter vector, i.e., optimizing over all three PID gains. As an optional exercise you can change the range of K_D from a singleton to an interval and apply BO for the full problem. Apart from changing the range of K_D, you only need to tune the hyperparameters of the kernel and BO (number of iterations, etc.) to make this work. Additionally, you might change your cost function to speed up the optimization. However, all the plotting functionality will not work, since the resulting plots would need to be four dimensional.

[1] However, we fix the derivative gain, resulting only in an optimization over the proportional and integral gains in order to allow visualization of the fitted GP.