**Introducing the black-box functions**

In this folder you will find initial data to guide your search, as well as some background on each function that should give you hints towards finding a good solution. We note that each function is synthetic (using real black-box functions would be too expensive for the contest to actually work), and the descriptions below are here to inform you about the properties of the functions, as well as giving a few examples of where Bayesian Optimization is applied in the real world.

*Function 1:* **Searching for Contamination Sources.** This may sound simple because you only have a two-dimensional input, however it is a very difficult problem. It corresponds to trying to find the source of radiation in some square area. However, you can only detect the radiation once you are *very close* to it, meaning most of the readings will be zero. There are two sources, one is not too dangerous, so make sure you try to find both modes of the function.

*Function 2*: **Optimizing Noisy Models.** This corresponds to trying to optimize an unknown machine learning model. However, the initialization of the model is very important, meaning your observations will be very noisy, and the problem might have a lot of local optimums! You are trying to make the model's log-likelihood as large as possible.

*Function 3*: **Drug Discovery Problem.** In this example, you are doing drug discovery! You can select three compounds to create a drug, and receive a measurement of the people's adverse reaction to the drug. You want to make this as close as possible to zero. (*hint*: one of the variables may not cause any effects on the person).

*Function 4:* **Fast, but Inaccurate Modelling.** This example is for a particular business relying heavily on online sales. It can run very accurate calculations to figure out what is the optimal placement of their product across warehouses. Unfortunately, the calculations are extremely expensive (computationally) to run, so they can only do it once every two weeks. Instead, they propose using a machine learning model which *approximates* the solution quickly (in a few hours). The model has four hyper-parameters you need to tune, and the output corresponds to the difference between the expensive calculation, and the model. Since you are modelling a dynamical system, expect a lot of local optima!

*Function 5*: **Yield in a Chemical Reaction.** This time you are trying to optimize another four-dimensional black-box. It corresponds to the yield of a chemical process after processing in some factory. This type of process tends to be unimodal. Try to find the combination of chemicals that maximizes the yield!

*Function 6:* **Cake and Stuff.** Time to get cooking! You are optimizing a cake recipe. There are five ingredients. The outputs correspond to the *sum* of different *objectives*: flavor, consistency, calories, waste and cost. Each objective receives negative points by our expert taster. You want this sum to be as close to zero as possible!

*Function 7:* **Sometimes Lazy is Best.** You are now optimizing six hyper-parameters of a machine learning model. Note that it is a popular and frequently used model, so maybe you could search to see if anyone else has optimized it before?

*Function 8:* **High-dimensional Optimization.** You've reach the final, 8-dimensional search space. High-dimensional black-box optimization can be very difficult, so sticking to local solutions is not the worst idea here.

In the folder for each function, you will find some initial data to help you begin to search for the optima. The data is in numpy format, so you can read it using `np.load()`.