02463 "Active Machine Learning and Agency"
Lecture 2: Bayesian Optimization

Federico Bergamin

1 Introduction

In the last lecture we have reviewed the properties of multivariate Gaussian and introduced Gaussian Processes. We focused on how to use them to perform regression and how to choose the kernel parameters given the specific dataset. In this lecture we are going to use Gaussian Process to solve optimization problems. If you have still doubts about how to fit a Gaussian Process to some data-points you should go back and revisit Lecture 1.

1.1 From decision problems to optimization problems

In our everyday life we face a lot of choices even during a single day. Some decisions we make unconsciously, that means we are not realizing that we are making those decisions. In contrast on other occasions, we think a lot about decisions we are faced with. Let's think about at the situation in which you have decided to have a beer in your favourite pub on a Saturday evening. You have quite different options to choose from, but you know that if you choose an IPA, every-time you are quite satisfied. To express this in a more explicit language, you have a prior about the level of satisfaction you get when you choose a specific beer style. Therefore, when it is time to order, you can reduce the options to main two possibilities: either you choose an IPA, knowing that you would be quite satisfied, or you can try something new and choose different beer style. In case you try a new thing, either you like the new beer style you have chosen, or you regret your choice and you are not as satisfied as if you would have chosen an IPA.

Although this is a really silly example, it allows us to understand a lot of concepts that are at the core of this lecture. If we have to come up with a decision we can start by defining an objective function over some goal we want to achieve. In our example above this would be the level of satisfaction. To decide, we can then optimize (minimize or maximize, depending on the context) this objective function. However, if we have a lot of possibilities, we should trade-off between *exploration* and *exploitation* in order to optimize our objective. In the problem described in our example, we can keep choosing the same beer (exploitation) or we can try something new (exploration). If we try only few candidates and we rank them, we are tempted to choose always the actual best candidate, but at the same time there is a possibility that we are not trying something that can give us a better return.

Therefore, as a general rule, if we are dealing with a decision problem and we are able to write it as an optimization problem using an objective function, we can obtain our decision by solving the optimization problem. So in this context we will restrict ourselves to consider optimization problems that can be fomulated as maximizing an objective function. Concisely, in the realm of optimization, we are interested in finding a global maximizer of a function $f : \mathcal{X} \to \mathbb{R}$:

$$\mathbf{x}^* = \arg\max_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}) \tag{1}$$

where $\mathcal{X}$ is some design space of interest. In the Global Optimization setting, $\mathcal{X}$ is usually a subset of $\mathbb{R}^d$, but in the case of Bayesian Optimization (BO), as we are going to see, this search space can contain both continuous and categorical inputs. Most of the time, we have to deal with a combinatorial search over some of these input sets. In case you are interested in computing the minimum of a function $f(\mathbf{x})$, you can use the methods we are going to describe to compute the

arg max of $-f(\mathbf{x})$ since the minimum of the original function would be at the $\mathbf{x}$ you have just found.

In general, we already know different methods that can be use to solve the optimization problem in Eq. 2. In the most simple case, if we have an explicit form of the objective function $f(\mathbf{x})$ we can compute the gradient, set to zero, and find the critical point that maximize the function. However, in most practical cases, the problems we are trying to solve do not have a closed form solution when we set the gradient to zero. Therefore, if the gradient is known, we can minimize the objective function using gradient descent. But in case we cannot compute the gradient we have only two possible approach: either we try to numerically estimate the gradient or we use random search over the possible inputs or sample them in a uniform way, assuming that the function evaluation is cheap in terms of time and computation. What complicates this situation even further is that many functions encountered in real problem setting have local minima and we can only hope to obtain the nearest local minimum using gradient descent. This would be an example of a non-convex objective function, where we would refer to the procedure of finding the global (over the input space considered) as a so-called global optimization problem.

It might also happen that we want to optimize some function $f(\mathbf{x})$ that has no mathematical representation and, at the same time, every function evaluation is expensive and/or time-consuming, meaning that we cannot rely on both gradient descent and random search. In the literature, people refers to this kind of unknown functions as *black-box* function, because we can observe them only through point evaluations or queries, also called *bandit feedback*. Since to obtain a bandit feedback is usually expensive in terms of money and/or time, we want to solve the problem using only a certain *budget*. In the litterature the problem we are trying to solve is related to the *multi-armed bandit* problem, but we will not consider this problem explicitly in these lecture notes.

The real world is full of problems that cannot be solved using random search because the objective function is too expensive to evaluate. Think about the problem of finding the right coordinates for drilling for oil extraction (not the best sustainable example, but back in the days this was a real problem). We would like to optimize the amount of oil in the underground $f_{\text{oil}}(\mathbf{x})$ where $\mathbf{x}$ are the geographic coordinates. We cannot perform random search because each sample, which would correspond to drilling in a specific location is really expensive. Likewise, in the pharmacological sciences researchers are interested in optimizing the effectiveness of a certain drug taken from a range of candidate drugs which would be effective against a specific disease. Therefore they want to optimize the effectiveness $f_{\text{eff}}(\mathbf{x})$, where $\mathbf{x}$ are the substances contained in the medication. To perform a single experiments is expensive and most of the time they have to wait at least a couple of months until the effects of the treatment against the disease can be observed. Therefore, in this setting random trials are not a option. Instead, since in most practical settings we can use only a fixed amount of money and/or time, we would like to have an heuristic that tell us what $\mathbf{x}$ to try next, given the fact that it could lead to a better result in maximizing $f(\mathbf{x})$.

We are going to face the same problem of time and budget constraint in a smaller scale when we want to optimize the hyper-parameters of a machine learning algorithms. Indeed, machine learning algorithms are rarely parameter-free, and we would typically have to specify these parameters before training our model. In some machine learning algorithms we have to define only a few hyper-parameters. However, in the case of deep neural networks we should take into account both the network parameters, such as the number of layers, the layer sizes, the dropout rate and the use of batch-normalization, and the training parameters such as the learning rate and the optimization algorithm to use. In this scenario, the parameter space is actually quite large. Furthermore, it might take several days for the training of the model to converge and be associated with considerable cost, in case we use an external service to train our model. In the settings it would be extremely useful to have a heuristic which would be able to choose the best hyper-parameters for your problem and reduce the number of trials.

2 Bayesian Optimization

Bayesian Optimization (BO) is a powerful sequential strategy for finding the optimum, of noisy and multi-modal *black-box* objective functions f, that, at the same time, are expensive to evaluate

in terms of time and cost. As before, we define our objective function $f : \mathcal{X} \to \mathbb{R}$, where $\mathcal{X}$ is a certain space of interest, and we want to find its global maximizer

$$\mathbf{x}^* = \arg\max_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}) \tag{2}$$

It is called Bayesian because, as we are going to see, it uses Bayes' theorem, which states that the posterior probability of a model M given the data D, or evidence, is proportional to the likelihood of the data given the model multiply by the prior of the model M:

$$P(M \mid D) \propto P(E \mid M)P(M) \tag{3}$$

To highlight the fact that this simple equation is the key to optimize the objective function, we try to introduce the key points of BO with a simple running example. Suppose we want to find the maximum of a 1-dimensional objective function $f(x)$ and we have sampled 4 different points at random (See upper plot in Fig. 1). If we ask random people to guess the position of the maximum of this function given only the four different samples and then plot an histogram of their answers, we would get something similar to the middle plot of Fig. 1. This is because each person is thinking of a possible function that goes through those four points and therefore we get a different maximum each time. If we try to consider an infinite amount of persons, we should get something similar to the bottom plot of Fig. 1, which is a function that describes the possible position of the maximum of our objective function, given only the information they get from the four samples. This example show us that if we can create a probabilistic model of our functions we can use the information that this model carries to build a function that guides our sampling strategy.

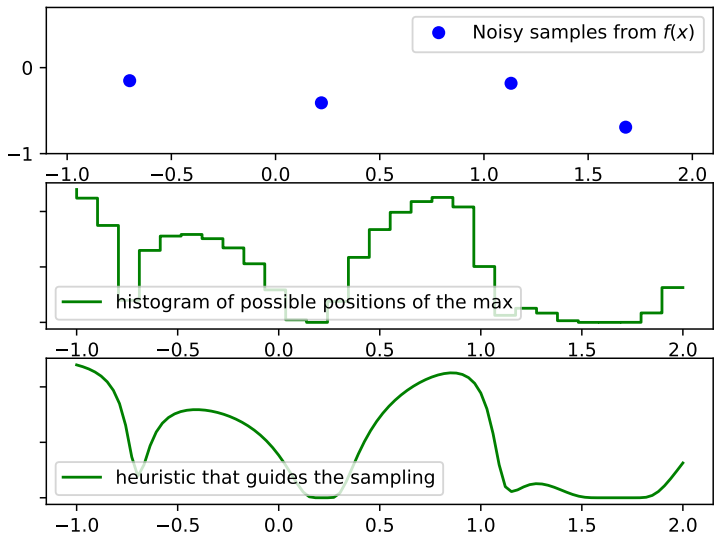

Figure 1: Initial example to introduce the key concepts of Bayesian Optimization. *Upper plot*: Four random samples from our true, unknown and expensive objective function. Our goal is to find the maximum of this function. *Middle plot*: hypothetical histogram of the possible position of the maximum of the function if we ask a certain number of people where they think the maximum might be. *Bottom plot*: if we ask infinite number of people, we should get something that looks like a function. This can inform us on what point we should sample next.

The examples above suggests that we should use all the information that the samples carry. Indeed, we have seen, that if we are able to create a probabilistic model of the possible functions that goes

through the sampled points, also known as a generative process of the functions, we can use this information to create a function that suggests the next point where we wish to evaluate the objective and guide our sampling strategy in order to find the maximum. Intuitively, the probabilistic model is doing what the group of people was doing in the example: approximate the unknown objective function by taking into account all the possible functions that goes through the points. We repeat this process, updating our model posterior when a new point is sampled, until we have enough budget to query the objective function. Therefore, to implement such technique, there are the two key ingredients that we should define: a *probabilistic model* and an *acquisition function*.

2.1 Probabilistic model

The *probabilistic model*, as the name already says, models the unknown objective function as a probability distribution, that means it learns a full statistical model of it. The initial ingredient consists of a prior distribution over the space of possible objective functions. This corresponds to make some assumptions about the properties and behaviour of the objective function that we do not know. For example, we can suppose that our function is smooth and it does not wiggle too much. This will make some possible objective functions more plausible than others. As we sample new points, we combine our prior distribution of the function with the likelihood function to obtain our posterior distribution, as in Eq. 4, that describes our updated belief about the objective function after we have seen some data. The posterior is given by

$$P(f \mid D_{1:t}) \propto P(D_{1:t} \mid f)P(f) \tag{4}$$

where $D_{1:t}$ contains the samples form our objective function $(\mathbf{x_i}, f(\mathbf{x_i}))$ with $\mathbf{x_i} \in \mathcal{X}$ and f is the unknown objective function.

This step of the Bayesian Optimization, where we are computing the posterior distribution of the unknown function, can be interpreted as approximating the true unknown objective function with a *surrogate model*, usually assumed to be given by the mean function of the distribution. There are some properties of the probabilistic model that we would like to have. It should approximate arbitrary complex functions well, therefore non-parametric methods, where we do not have to specify a fixed number of parameters in advance, seems to be preferable. It has also to be cheaper to evaluate compared to the objective function, otherwise there is no convenience in using Bayesian Optimization. In addition to that, we would like a probabilistic model that quantifies the uncertainty to make informed decision.

In this lecture, we will use a Gaussian Process as probabilistic model. We previous considered GPs as non-parametric models that define a probability distribution over functions, where the mean function is the most likely function given the training points and covariance can be used to quantify the uncertainty of the function across the input space. Another possible candidate as surrogate model would be a Random Forest. However, these will not be considered in these notes. We are not going to introduce GPs and GP regression again. If you have still doubts about those topics, you should go back and review the previous lecture.

2.2 Acquisition Functions

Thus far, we have just presented the probabilistic model used to represent our belief about the unknown objective. The second key ingredient of Bayesian Optimization that we are still missing, as we have seen from the simple example, is a mechanism that uses the information from the probabilistic model and decides which point $\mathbf{x}_{t+1}$ we should evaluate in the next iteration $t + 1$. These auxiliary functions, usually denoted by $a : \mathcal{X} \to \mathbb{R}^+$, are called *acquisition functions*, because they define the way we acquire samples from the objective function. In general these functions depends on the previous observations and the probabilistic model hyper-parameters. These dependencies are denoted by $a(\mathbf{x}, \{\mathbf{x}_n, y_n\}, \theta)$. These functions should represents the goal we want to achieve, which in our case is to find the $\mathbf{x}$ that maximizes the unknown objective function. Therefore, as we have seen from the beer example, this functions should trade-off between *exploration* and *exploitation*. Indeed, suppose we have acquired already two samples at random from our objective function $\{(\mathbf{x}_1, f(\mathbf{x}_1)), (\mathbf{x}_2, f(\mathbf{x}_2))\}$ and suppose that we have $f(\mathbf{x}_2) \geq f(\mathbf{x}_1)$.

We would like that our function selects the next points $\mathbf{x}_i$ close to $\mathbf{x}_2$, because we want to stay close to the highest sample (*exploitation*), but at the same time, if this is a local maximum, we would like to escape it and explore points far from the highest output (*exploration*) where we are also uncertain about the objective function value.

Here, we are going to consider three different possible acquisition functions: *Probability of Improvement*, *Expected Improvement*, and *Upper Confidence Bound*.

Algorithm 1: Bayesian Optimization

1) Given an objective function $f(\mathbf{x})$ and its input space $\mathcal{X}$, an acquisition function $a(\mathbf{x})$, and a set for the samples $D = \{\emptyset\}$, initially empty.
2) Acquire M random samples from our objective function $f(\mathbf{x})$, this way we have

$$D_0 = \{(\mathbf{x}_m, y_m))_{m=1}^{M}\}$$

3) **For** $t = 1, 2, \ldots$ until the budget ends **do**
   a) Fit the probabilistic model on the samples in D. When relevant, as when using GP, kernel hyper-parameters should also be optimized.
   b) Compute the acquisition function $a(\mathbf{x}|D_{t-1})$ on the input space $\mathcal{X}$ combining the attributes from the probabilistic model. In case of GPs, as we are going to see, we will use the standard deviation and the mean of the posterior distribution.
   c) Find the point $\mathbf{x}_t \in \mathcal{X}$ that maximizes the acquisition function $a(\mathbf{x}|D_{t-1})$:

$$\mathbf{x}_t = \arg\max_{\mathbf{x}\in\mathcal{X}} a(\mathbf{x}|D_{t-1})$$

   d) Sample the objective function $f(x)$ at $\mathbf{x}_t$ to obtain y_t
   e) Augment the sample set

$$D_t = \{D_{t-1}, (\mathbf{x}_t, y_t))\}$$

End for

2.2.1 Probability of improvement

The first acquisition function we introduce is one of the earliest strategies employed in the literature and it belongs to the *improvement-based* acquisition functions family because it aims to select points that are likely to improve upon the current best target. In this case, the incumbent (current best) solution is given by the best obtained function value up to the current iteration. We indicate this best current sample with $f(\mathbf{x}^+)$ where $\mathbf{x}^+ = \arg\max_{\mathbf{x}_i\in\mathbf{x}_{1:t}} f(\mathbf{x}_i)$. Therefore, for each $\mathbf{x} \in \mathcal{X}$ in our space of interest, we are interested in computing

$$\mathrm{PI}(\mathbf{x}) = \mathrm{P}(f(\mathbf{x}) \geq f(\mathbf{x}^+)) \tag{5}$$

If we consider a simple 2D example, as illustrated in Fig. 7 and we use a Gaussian Process as the probabilistic model, we have that for each $\mathbf{x} \in \mathbb{X}$ we get that $f(\mathbf{x})$ is Gaussian distributed with respect to $\mathcal{N}(\mu(\mathbf{x}), \sigma^2(\mathbf{x}))$, where $\mu(\mathbf{x})$ is the mean function value in $\mathbf{x}$ and $\sigma^2(\mathbf{x})$ is the variance we get from the covariance function of the GP. When everything is normal distributed, it is possible to compute the probability that $P(f(\mathbf{x}) \geq f(\mathbf{x}^+))$, in the following section we will recall the procedure to do this briefly.

How to compute $\mathbf{P}(X \geq x)$ Suppose you have a Normal distributed random variable $X \sim \mathcal{N}(\mu, \sigma^2)$ and we are interested in computing the probability $P(X \geq x)$. We know that we can compute the probability of an event of the random variable X using a random variable $Z \sim \mathcal{N}(0, 1)$,

Visualization of Bayesian Optimization

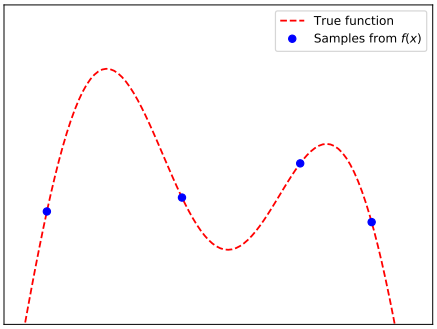

Figure 2: *Step 1*: Initial random samples from our unknown objective function.

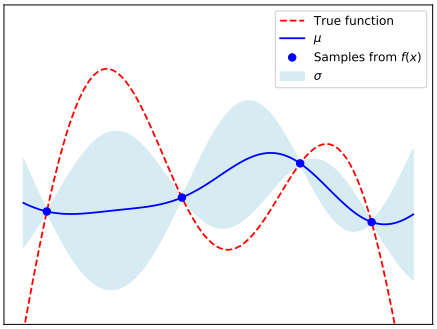

Figure 3: *Step 2*: We fit a probabilistic model, in this case a Gaussian Process on the samples.

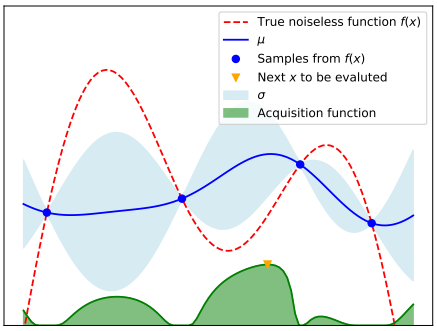

Figure 4: *Step 3*: Using the standard deviation and the mean function of the GP, we compute an acquisition function for all the points $\mathbf{x}_t \in \mathcal{X}$. We also can select the next $\mathbf{x}$ to evaluate (orange triangle in the plot) on the objective.

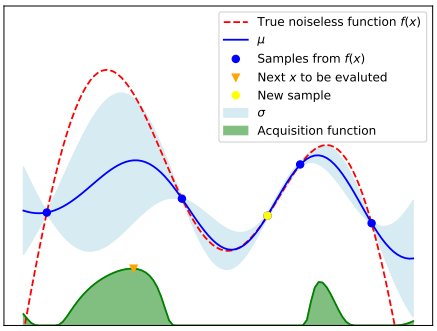

Figure 5: *Step 4*: We add the sample computed in the previous step (yellow circle in the plot) in D and re-fit the GP and compute again the acquisition function. We can then select the $\mathbf{x}$ that maximizes it and sample the unknown objective.

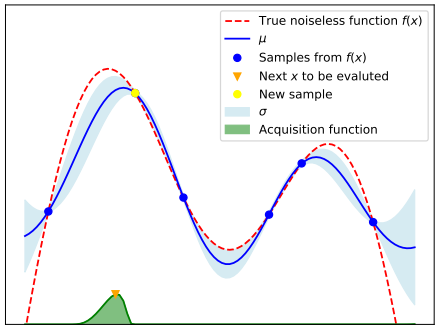

Figure 6: *Step 5*: We can repeat the `for` loop of Algorithm 1 another time, and we see that we are getting closer to the true maximum.

that follows the standard normal distribution. The trick is to define Z as $Z \coloneqq \frac{X-\mu}{\sigma}$, where μ is the mean of the distribution and $\sigma \coloneqq +\sqrt{\sigma^2}$ is the standard deviation. Therefore, if we are working with a random variable $Z \sim \mathcal{N}(0,1)$ we can express the probability of events such as $\mathrm{P}(Z \geq z)$ in terms of the cumulative distribution function (cdf) $\Phi(z)$:

$$\begin{aligned} \mathrm{P}(Z \leq z) &= \Phi(z) \\ \mathrm{P}(Z \geq z) &= 1 - \Phi(z) = \Phi(-z) \\ \mathrm{P}(|Z| \leq z) &= 2\Phi(z) - 1 \\ \mathrm{P}(|Z| \geq z) &= 2(1 - \Phi(z)) = 2\Phi(-z) \end{aligned} \tag{6}$$

The cumulative distribution function Φ of the standard Normal distribution is given by:

$$\Phi(z) \coloneqq \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} e^{-\frac{1}{2}u^2} du \tag{7}$$

Therefore, if we are interested in computing $\mathrm{P}(X \geq x)$, we should first consider the standardized version of the random variable and then use the rules above

$$\mathrm{P}(X \geq x) = \mathrm{P}\left(\frac{X-\mu}{\sigma} \geq \frac{x-\mu}{\sigma}\right) = 1 - \Phi\left(\frac{x-\mu}{\sigma}\right) = \Phi\left(\frac{\mu - x}{\sigma}\right) \tag{8}$$

If we go back to the probability of improvement acquisition function, we are interested in computing $\mathrm{P}(f(\mathbf{x}) \geq f(\mathbf{x}^+))$ for all points $\mathbf{x}$ in the space of interest. From this it follows that we can calculate this probability as indicated below, i.e. by standardizing the random variable:

$$\begin{aligned} \mathrm{PI}(\mathbf{x}) &= \mathrm{P}(f(\mathbf{x}) \geq f(\mathbf{x}^+)) = \\ &= \mathrm{P}\left(\frac{f(\mathbf{x}) - \mu(\mathbf{x})}{\sigma(\mathbf{x})} \geq \frac{f(\mathbf{x}^+) - \mu(\mathbf{x})}{\sigma(\mathbf{x})}\right) = \\ &= 1 - \Phi\left(\frac{f(\mathbf{x}^+) - \mu(\mathbf{x})}{\sigma(\mathbf{x})}\right) = \\ &= \Phi\left(\frac{\mu(\mathbf{x}) - f(\mathbf{x}^+)}{\sigma(\mathbf{x})}\right) \end{aligned} \tag{9}$$

A visual illustration of this acquisition function can be seen in Fig. 7. When we are computing the probability in Eq. 9 for each point $\mathbf{x}$ in the space of interest, we are calculating the area under the curve of the distribution from $-\infty$ to the value of the best samples of the unknown function we get. At iteration $t+1$, we will sample the unknown objective at the candidate point $\mathbf{x}_{t+1}$ where we get the biggest area under the curve.

This acquisition function, as expressed above, will result in mostly exploitation and no exploration. Indeed, we will always prefer points with high probability of being infinitesimally grater then the current best samples $f(\mathbf{x}^+)$ over points that are less certain. To overcome this drawback, we add a trade-off parameters $\xi \geq 0$ that has to trade-off exploration and exploitation. Therefore, the probability of improvement acquisition function we are going to use is defined as:

$$\begin{aligned} \mathrm{PI}(\mathbf{x}) &= \mathrm{P}(f(\mathbf{x}) \geq f(\mathbf{x}^+) + \xi) = \\ &= \Phi\left(\frac{\mu(\mathbf{x}) - f(\mathbf{x}^+) - \xi}{\sigma(\mathbf{x})}\right) \end{aligned} \tag{10}$$

that can be computed repeating the same steps as before. There is no principled way to determine an optimal value for ξ, the original paper suggest an annealing heuristic for ξ: where the value of

$$\xi$$

is initially high in order to have preference on exploration and then gradually decreased towards 0 to later prefer exploitation.

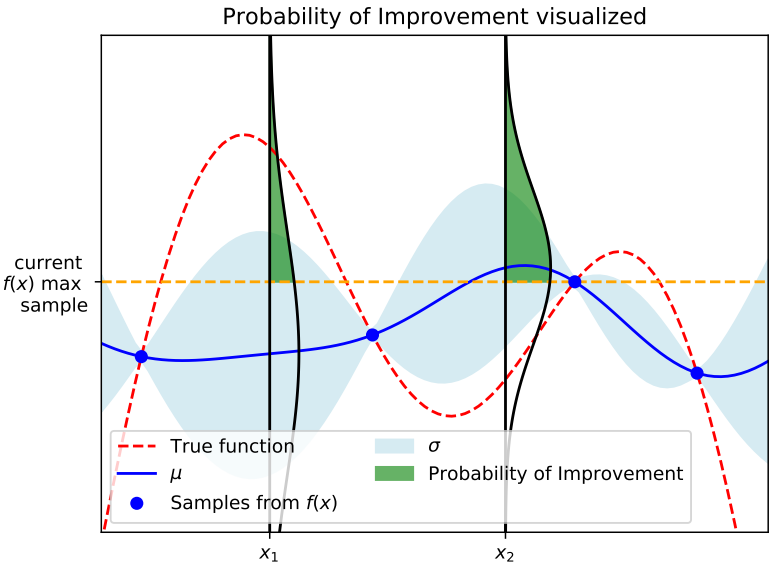

Figure 7: Visual example of the computation of the Probability of Improvement acquisition function for all the points

2.2.2 Expected Improvement

The second acquisition function that we are going to consider is called *Expected Improvement*. The idea, as then name says, is to measure the expected improvement with respect the current best sample $f(\mathbf{x}^+)$ if we sample the objective function at $\mathbf{x}$ at the next iteration. The expected improvement tries to quantify the amount of improvement and not the probability of improving as the previous acquisition function did. The proposed improvement function in the original paper is given by

$$\mathrm{I}(\mathbf{x}) = \max\{0, f_{t+1}(\mathbf{x}) - f(\mathbf{x}^+)\} \tag{11}$$

Since we are using a probabilistic model of the objective function, we have a distribution for each value of the unknown objective function. In case of the Gaussian Process, we know that each $f(\mathbf{x})$ follows a Gaussian distribution $\mathcal{N}(\mu(\mathbf{x}), \sigma^2(\mathbf{x}))$. Therefore, we can quantify the average value of the improvement we get if we sample our objective function at $\mathbf{x}$. In other words, we are interested in computing the *expected value* of the improvement function at a certain $\mathbf{x}$. That is:

$$\mathrm{EI}(\mathbf{x}) = \mathbb{E}_{f_{t+1}(\mathbf{x}) \sim \mathcal{N}(\mu(\mathbf{x}), \sigma^2(\mathbf{x}))}[\max\{0, f_{t+1}(\mathbf{x}) - f(\mathbf{x}^+)\}] \tag{12}$$

We can see that we can write $f_{t+1}(\mathbf{x}) = \mu(\mathbf{x}) + \epsilon\sigma(\mathbf{x})$ where $\epsilon \sim \mathcal{N}(0, 1)$. This is also known as the *reparametrization trick*. Therefore, we can write the expected improvement as

$$\mathrm{EI}(\mathbf{x}) = \mathbb{E}_{\epsilon \sim \mathcal{N}(0,1)}[\max\{0, \mu(\mathbf{x}) + \epsilon\sigma(\mathbf{x}) - f(\mathbf{x}^+)\}] \tag{13}$$

We can split the function instead of using the max function:

$$\mathrm{EI}(\mathbf{x}) = \begin{cases} \mathbb{E}_{\epsilon \sim \mathcal{N}(0,1)}[\mu(\mathbf{x}) + \epsilon\sigma(\mathbf{x}) - f(\mathbf{x}^+)] & \text{if } \epsilon \geq \frac{\mu(\mathbf{x}) - f(\mathbf{x}^+)}{\sigma(\mathbf{x})} \\ \mathbb{E}_{\epsilon \sim \mathcal{N}(0,1)}[0] & \text{otherwise} \end{cases} \tag{14}$$

If we now use the definition of expectation and solve the integrals, we will find the closed-form expression of the *expected improvement* acquisition function when we are using a Gaussian Process as probabilistic model.

$$
\begin{aligned}
\mathrm{EI}(\mathbf{x}) &= \mathbb{E}_{\epsilon\sim\mathcal{N}(0,1)}[\max\{0, \mu(\mathbf{x}) + \epsilon\sigma(\mathbf{x}) - f(\mathbf{x}^+)\}] \\
&= \int_{-\infty}^{\infty} \mathrm{I}(\epsilon)\phi(\epsilon)\mathrm{d}\epsilon \\
&= \int_{-\infty}^{\frac{f(\mathbf{x}^+)-\mu(\mathbf{x})}{\sigma(\mathbf{x})}} 0\phi(\epsilon)\mathrm{d}\epsilon + \int_{\frac{f(\mathbf{x}^+)-\mu(\mathbf{x})}{\sigma(\mathbf{x})}}^{+\infty} (\mu(\mathbf{x}) + \epsilon\sigma(\mathbf{x}) - f(\mathbf{x}^+))\phi(\epsilon)\mathrm{d}\epsilon \\
&= (\mu(\mathbf{x}) - f(\mathbf{x}^+)) \int_{\frac{f(\mathbf{x}^+)-\mu(\mathbf{x})}{\sigma(\mathbf{x})}}^{+\infty} \phi(\epsilon)\mathrm{d}\epsilon + \sigma(\mathbf{x}) \int_{\frac{f(\mathbf{x}^+)-\mu(\mathbf{x})}{\sigma(\mathbf{x})}}^{+\infty} \epsilon\phi(\epsilon)\mathrm{d}\epsilon \\
&= (\mu(\mathbf{x}) - f(\mathbf{x}^+)) \left[\Phi(\epsilon)\Big|_{\frac{f(\mathbf{x}^+)-\mu(\mathbf{x})}{\sigma(\mathbf{x})}}^{+\infty}\right] + \sigma(\mathbf{x}) \int_{\frac{f(\mathbf{x}^+)-\mu(\mathbf{x})}{\sigma(\mathbf{x})}}^{+\infty} \epsilon\frac{1}{\sqrt{2\pi}} e^{\frac{\epsilon^2}{2}}\mathrm{d}\epsilon \\
&= (\mu(\mathbf{x}) - f(\mathbf{x}^+))\left(1 - \Phi\left(\frac{f(\mathbf{x}^+) - \mu(\mathbf{x})}{\sigma(\mathbf{x})}\right)\right) + \frac{\sigma(\mathbf{x})}{\sqrt{2\pi}} \int_{\frac{f(\mathbf{x}^+)-\mu(\mathbf{x})}{\sigma(\mathbf{x})}}^{+\infty} \epsilon e^{\frac{\epsilon^2}{2}}\mathrm{d}\epsilon \\
&= (\mu(\mathbf{x}) - f(\mathbf{x}^+))\left(1 - \Phi\left(\frac{f(\mathbf{x}^+) - \mu(\mathbf{x})}{\sigma(\mathbf{x})}\right)\right) + \frac{\sigma(\mathbf{x})}{\sqrt{2\pi}} \left[-e^{\frac{\epsilon^2}{2}}\Big|_{\frac{f(\mathbf{x}^+)-\mu(\mathbf{x})}{\sigma(\mathbf{x})}}^{+\infty}\right] \\
&= (\mu(\mathbf{x}) - f(\mathbf{x}^+))\Phi\left(\frac{\mu(\mathbf{x}) - f(\mathbf{x}^+)}{\sigma(\mathbf{x})}\right) + \sigma(\mathbf{x})\phi\left(\frac{\mu(\mathbf{x}) - f(\mathbf{x}^+)}{\sigma(\mathbf{x})}\right)
\end{aligned}
\tag{15}
$$

where Φ is the cumulative distribution function and ϕ the probability density function of the standard Gaussian distribution and we use the fact that the cumulative distribution function of a continuous random variable can be obtained by the probability density function by integrating it. If we write $Z = \frac{\mu(\mathbf{x})-f(\mathbf{x}^+)}{\sigma(\mathbf{x})}$, we obtain:

$$
\mathrm{EI}(\mathbf{x}) = (\mu(\mathbf{x}) - f(\mathbf{x}^+))\Phi(Z) + \sigma(\mathbf{x})\phi(Z) \tag{16}
$$

Noticing that we cannot compute Z if $\sigma(\mathbf{x}) = 0$, we can write the expected improvement acquisition function when we use Gaussian Process as

$$
\mathrm{EI}(\mathbf{x}) = \begin{cases} (\mu(\mathbf{x}) - f(\mathbf{x}^+))\Phi(Z) + \sigma(\mathbf{x})\phi(Z) & \text{if } \sigma(\mathbf{x}) > 0 \\ 0 & \text{if } \sigma(\mathbf{x}) \leq 0 \end{cases} \tag{17}
$$

As we did in the probability of improvement acquisition function, also in the expected improvement people introduce an additional parameter ξ to trade-off exploration and exploitation. Therefore, it is common to define Z as $Z = \frac{\mu(\mathbf{x})-f(\mathbf{x}^+)-\xi}{\sigma(\mathbf{x})}$.

2.2.3 Gaussian Process Upper Confidence Bound

The last acquisition function that we are going to introduce is the Gaussian Process Upper Confidence Bound (GP-UCB). The basic idea of this acquisition function is to directly use the predictive mean $\mu(\mathbf{x})$ and variance $\sigma^2(\mathbf{x})$ of the Gaussian Process to guide the exploration and exploitation. The initial formulation of this acquisition function is based on an algorithm called *"Sequential Design for Optimization"*, which selects the points $\mathbf{x}$ to query the unknown objective function based on the upper confidence bound:

$$
\mathrm{UCB}(\mathbf{x}) = \mu(\mathbf{x}) + \kappa\sigma(\mathbf{x}) \tag{18}
$$

where the trade-off parameter κ, which is left to the user, balances exploration and exploitation. Indeed, for high value of κ, we are preferring points $\mathbf{x} \in \mathcal{X}$ that have high uncertainty and therefore

a high value of κ means that we will attempt to reduce the uncertainty for all points of the function. As we will see in the later lectures this is the goal of active learning. In contrast, if a small value of κ is used we will prefer to query the function in points close to the ones already sampled, and therefore have more focus on exploitation.

Another formulation of this acquisition function has been proposed later in the literature in the context of Bayesian Optimization considering a multi-armed bandit setting. In this setting, if we want to maximize the unknown function f, that has the maximum in $\mathbf{x}^* = \arg\max_{\mathbf{x}} f(\mathbf{x})$, in each step, t, in which we query $\mathbf{x}_t$, we can consider the *regret function*, defined as

$$r(\mathbf{x}_t) = f(\mathbf{x}^*) - f(\mathbf{x}_t) \tag{19}$$

If we sum all the regrets we get in the optimization sequence, we can consider the *cumulative regret*, given by:

$$R_T = \sum_{t=1}^{T} r(\mathbf{x}_t) \tag{20}$$

The goal is to find an algorithm that has the asymptotic property of leading to *no-regret*, that is $\lim_{T\to\infty} \frac{R_T}{T} = 0$. As we are going to see, we are not using r_t and R_T explicitly in the acquisition function, but they have been used to study the convergence of the upper confidence bound. It has been shown that if we consider the following formulation of the upper confidence bound:

$$\text{GP-UCB}(\mathbf{x}) = \mu(\mathbf{x}) + \sqrt{\nu\beta_t}\sigma(\mathbf{x}) \tag{21}$$

with $\nu = 1$ and $\beta_t = 2\log(\frac{t^{\frac{d}{2}+2}\pi^2}{3\delta})$[1], where t is the iteration number, d is the dimensionality of the problem, and $\delta \in (0, 1)$ is left to the user, we would have *no-regret* asymptotically. This means, that if we have the possibility to run the algorithm for $T \to \infty$ we get the true maximum of the function.

A simple approximation of the conditions above, is given by

$$\text{GP-UCB}(\mathbf{x}) = \mu(\mathbf{x}) + \epsilon\log(t)\sigma(\mathbf{x}) \tag{22}$$

where now we still have to decide on the value of ϵ.

[1]Some implementations uses $\beta_t = 2\log(\frac{dt^2\pi^2}{6\delta})$.

References and useful links

The following references were used to write the following notes. These references are also meant to be used to get a different introduction to the topic of Bayesian Optimization using Gaussian Processes regression and/or uncertainty in machine learning.

- *"Taking the human out of the loop: A review of Bayesian optimization."*, Shahriari, Bobak, et al., Proceedings of the IEEE 104.1 (2015): 148-175.
- *"A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning."*, Brochu, Eric, Vlad M. Cora, and Nando De Freitas. arXiv preprint arXiv:1012.2599 (2010).
- *"Practical bayesian optimization of machine learning algorithms."*, Snoek, Jasper, Hugo Larochelle, and Ryan P. Adams. Advances in neural information processing systems. 2012.
- The following PhD thesis:
  - *"Bayesian Optimization and Semiparametric Modelswith Applications to Assistive Technology"*, Jasper Snoek, Diss. PhD thesis, University of Toronto, 2013. You can access it here
  - *"Robust Adaptive Decision Making: Bayesian Optimization and Beyond"*, Ilija Bogunovic, Diss. PhD thesis, EPFL Lausanne , 2019. You can access it here
- The following YouTube video lectures:
  - "Machine learning - Bayesian optimization and multi-armed bandits", lecture by Nando De Freitas at University of British Columbia. Link
  - "Introduction to Bayesian Optimization", lecture by Javier Gonzalez at Gaussian Process Summer School 2017, Sheffield. Link
  - " Bayesian optimization", lecture by Daniil Polykovsky, researcher at Insilico Medicine. This video is part of the Coursera course on Bayesian Methods for Machine Learning, but can be seen at this link