Bayesian Optimization with Gaussian Processes: A Beginner's Guide

Hoh Siew Yan

May 22, 2026

1 What Problem Are We Trying to Solve?

Imagine you are trying to find the best settings for a complicated experiment – for example, the optimal laser pulse energy, focal spot size, incidence angle, and pulse duration to maximize the energy absorbed by a solid target during a laser–matter interaction. You can model the situation with a particle-in-cell or hydrodynamic simulation, but each run takes hours and the results contain noise. You cannot afford to try every possible combination of settings, and you cannot easily write down a mathematical formula (**objective function**) that says "this is how the absorbed energy depends on the inputs.

This is a classic situation where Bayesian Optimization (BO) shines. BO is a global optimization technique designed for problems where:

- Evaluating the objective function is **expensive** (each evaluation takes a lot of time or computational resources).
- The function is a **black box** (you cannot easily compute its derivatives or write down its formula).
- The results may be **noisy** (running the same input twice can give slightly different outputs).

The goal of BO is simple to state: find the input $\mathbf{x}^*$ that maximizes (or minimizes) an unknown objective function $f(\mathbf{x})$, while using as few evaluations as possible.

2 The Big Idea: Build a Cheap Model of an Expensive Function

The central trick of BO is to build a cheap, probabilistic *surrogate model* of the expensive objective function. Instead of repeatedly querying the real function, we query our surrogate, which gives us not only a prediction of the output but also a measure of how uncertain that prediction is. We then use these two pieces of information—prediction and uncertainty—to decide intelligently where to look next.

The loop looks like this:

1. Start with a small set of observations $(\mathbf{x}_i, y_i)$.
2. Fit a surrogate model to the data.
3. Use an *acquisition function* to decide the next promising point $\mathbf{x}_n$ to evaluate.
4. Run the expensive simulation at $\mathbf{x}_n$ to get y_n.
5. Add the new observation to the dataset and repeat.

Data: $t \geq 0$
Result: $D_t = [X, Y]_t$
$Y \leftarrow f_{obj}(\mathbf{x_n})$;
$\mathbf{X} \leftarrow \mathbf{x_0}$;
$N \leftarrow 1$;
while $N < t$ **do**
  $\mathbf{X} \leftarrow \text{argmax } \alpha(\mathbf{x_{t-1}}|D_{t-1})$;
  $Y \leftarrow f_{obj}(\mathbf{X})$;
  $N \leftarrow N + 1$;
end

Algorithm 1: Bayesian optimization algorithm

3 The Surrogate Model: Gaussian Processes

For the surrogate, BO commonly uses a *Gaussian Process* (GP). A GP is a non-parametric, probabilistic way to model functions. Rather than committing to a specific functional form (like a polynomial or a neural network with a fixed architecture), a GP defines a probability distribution over all possible functions consistent with the observed data.

A GP is fully specified by two ingredients:

- A **mean function** $\mu(\mathbf{x})$, which encodes prior beliefs about the typical value of f at input $\mathbf{x}$.
- A **covariance function** (or *kernel*) $k(\mathbf{x}, \mathbf{x}')$, which encodes how strongly the function values at two inputs $\mathbf{x}$ and $\mathbf{x}'$ are expected to be correlated.

We write this compactly as

$$f(\mathbf{x}) \sim \mathcal{GP}\big(\mu(\mathbf{x}),\, k(\mathbf{x}, \mathbf{x}')\big). \tag{1}$$

The key intuition is that points close together in input space are likely to have similar function values, while points far apart are not. The kernel quantifies "close" and "far."

3.1 The RBF Kernel

A very common choice is the *Radial Basis Function* (RBF) kernel, also called the squared exponential kernel:

$$k_{\mathrm{RBF}}(\mathbf{x}, \mathbf{x}') = \exp\left[-\frac{1}{2}\sum_{i=1}^{d}\left(\frac{x_i - x_i'}{\ell_i}\right)^2\right]. \tag{2}$$

Here d is the number of input dimensions and ℓ_i is the *length scale* in dimension i. The length scale controls how quickly correlations decay with distance: a small ℓ_i means the function can vary rapidly along dimension i, while a large ℓ_i means it varies slowly. The RBF kernel produces smooth, infinitely differentiable functions, which is often a reasonable prior assumption for physical systems.

3.2 Handling Noise

Real measurements and simulations are noisy. To accommodate this, we add a Gaussian noise term to the diagonal of the kernel:

$$k(\mathbf{x}, \mathbf{x}') = \sigma^2\, k_{\mathrm{RBF}}(\mathbf{x}, \mathbf{x}') + \sigma^2_{\mathrm{noise}}\, \delta_{ij}. \tag{3}$$

The Kronecker delta δ_{ij} ensures that the noise only affects the variance of a point with itself, not the correlation between distinct points. The quantities σ^2 (signal variance), $\sigma^2_{\mathrm{noise}}$ (noise variance), and the length scales ℓ_i are the *hyperparameters* of the GP. They are typically learned from data by maximizing the log-likelihood of the observations.

4 The Acquisition Function: Where to Look Next?

Once we have a GP that gives us a predicted mean $\mu(\mathbf{x})$ and a predicted uncertainty $\sigma(\mathbf{x})$ at every point, we need a strategy to choose the next evaluation point. This is the role of the *acquisition function* $\alpha(\mathbf{x})$.

A good acquisition function balances two competing desires:

- **Exploitation**: sample where the predicted mean is high, because those points are likely to give good objective values.
- **Exploration**: sample where the uncertainty is high, because those regions might hide an even better optimum that we have not yet discovered.

A simple and popular choice is the *Upper Confidence Bound* (UCB):

$$\alpha_{\mathrm{UCB}}(\mathbf{x}) = \mu(\mathbf{x}) + \kappa\, \sigma(\mathbf{x}). \tag{4}$$

The hyperparameter κ controls the trade-off. A large κ emphasizes the uncertainty term and encourages exploration; a small κ favors exploitation near regions where the model already predicts good values. A common refinement is to gradually increase κ as iterations proceed, which provides theoretical guarantees that the search will eventually converge to the global optimum.

The next point to evaluate is then

$$\mathbf{x}_n = \arg\max_{\mathbf{x}} \alpha(\mathbf{x} \mid D_{t-1}), \tag{5}$$

where D_{t-1} is the data collected up to the previous iteration. Note that maximizing the acquisition function is itself an optimization problem, but it is cheap because it operates on the surrogate, not the expensive real objective.

Other acquisition functions exist, such as *Expected Improvement* (EI), which weights potential improvements by their probability. UCB is often preferred for its simplicity and the transparency of its exploration–exploitation knob.

5 Getting Started: Initial Samples

Before the BO loop begins, we need an initial set of observations to fit the GP. A naive approach is to pick random points, but this can leave gaps in the input space. A better strategy is to use a *low-discrepancy sequence* such as the Sobol sequence, which spreads points more evenly than random sampling. Generating, say, 10 Sobol points across the ranges of all input parameters typically gives the GP enough information to make sensible predictions in the first few iterations.

6 Why Bayesian Optimization Works Well

Putting the pieces together, BO is powerful because each ingredient is doing useful work:

- The **GP surrogate** turns a few expensive observations into a full predictive distribution over the entire input space.
- The **kernel** encodes smoothness assumptions, so information from one point propagates to nearby points.
- The **noise term** prevents the model from overfitting to noisy observations.
- The **acquisition function** provides a principled way to balance learning about the function (exploration) with using what we already know (exploitation).
- The **iterative loop** continually refines the model and focuses on the most promising regions, so the optimum is typically found in far fewer evaluations than grid search, random search, or many alternative methods would require.

7 When to Use Bayesian Optimization

BO is a strong choice when:

- Each evaluation of the objective is expensive (long simulations, costly experiments).
- The objective is noisy.

- The number of input dimensions is moderate (roughly 1–20; very high dimensions become harder).
- You have no convenient analytical form or gradient information.

It is less attractive when evaluations are cheap (just run a grid), when gradients are available (use gradient-based methods), or when the parameter space is enormous (thousands of dimensions).

8 Summary

Bayesian Optimization is, at its heart, a disciplined way of being curious. It maintains a probabilistic belief about an unknown function, updates that belief as new data arrive, and uses the belief—including the parts it is uncertain about—to decide where to look next. With a Gaussian Process as the surrogate, an RBF kernel to encode smoothness, a noise term to handle real-world messiness, and the UCB acquisition function to trade off exploration against exploitation, BO offers a remarkably efficient route to the optimum of an expensive black-box function.