Introduction to Bayesian Optimization

Drew Gjerstad

Contents

1 Introduction

Bayesian optimization refers to an optimization approach that uses Bayesian inference to guide the optimizer to make "better" decisions based on *uncertainty*. In addition, this approach provides a framework that enables us to strategically tackle the uncertainty inherent in all optimization decisions. One particularly attractive property of this framework is its *unparalleled* sample efficiency, a property we will discuss more in-depth later.

In these notes, the goal is to introduce Bayesian optimization from a high-level perspective and introduce the components involved in the "Bayesian optimization workflow". Additional notes discussing the components and related topics in detail are available in the `bayesian-optimization` repository. We will begin with the motivation behind Bayesian optimization, primarily focusing on the theoretical motivation and examples of real-world Bayesian optimization applications.

2 Motivation

2.1 Theoretical Motivation

First, we consider the theoretical motivation for the Bayesian optimization approach. Typically, the theoretical motivation steps from the form or type of *objective* (the "function" we are aiming to optimize). Note that the list below is by no means exhaustive; there exists additional theoretical motivation but these are seemingly the most common, particularly with regard to the form and characteristics of the objective.

- **Black-box objective functions** are functions that we can only interact with via its inputs and outputs meaning classical, analytical methods do not work. However, we can usually use Bayesian optimization to approximate such an objective and manage the inherent uncertainty.
- **Expensive-to-evaluate objective functions** are functions that require significant computation effort to obtain their output. However, just as with black-box objectives, we can use Bayesian optimization to approximate and model these efficiently.
- More generally, this approach is very useful when the objectives lack analytical evaluation (or, if analytical evaluation is expensive).
- In some spaces such as the discrete or combinatorial ones, the objective may not have efficient gradients (if they exist). Thus, classical gradient-based optimization methods are incompatible.

2.2 Applications

The application potential of Bayesian optimization can be seen across several critical domains, especially those attempting to accelerate the identification of solutions to real-world scientific and engineering problems. Some of these applications include:

- Drug discovery
- Molecule/protein discovery
- Materials design
- AutoML (i.e., hyperparameter tuning)
- Engineering decisions
- Many more...

The diagrams in the next few sections are there to illustrate how the Bayesian optimization approach is used in real-world applications. There will also be commentary explaining why the approach is so useful in these applications. Just as with the list above, this is not (in any way) an exhaustive review of applications. Rather, the point of these sections is to showcase the real-world motivation for this optimization approach.

Application: Drug Discovery

Figure 1 below shows an example of a chemical process optimization problem, common in drug discovery. Don't stress about the details regarding the formulation of the problem but rather consider the fact that running chemical synthesis experiments require both *time* and *money*. Therefore, scientists want to accelerate finding optimal solutions, or in this case, optimal chemical processes/reactions.

In the left half of the figure, four common (classical) approaches are shown but these approaches are expensive. Alternatively, the Bayesian optimization approach is shown in the right half. That approach uses data from previous experiments to locate points that may optimize the unknown objective, while strategically handling uncertainty in the model. The located points will usually be used to plan future experiments as they represent samples of *high utility* (i.e., the point should be useful in optimizing the reaction parameter).

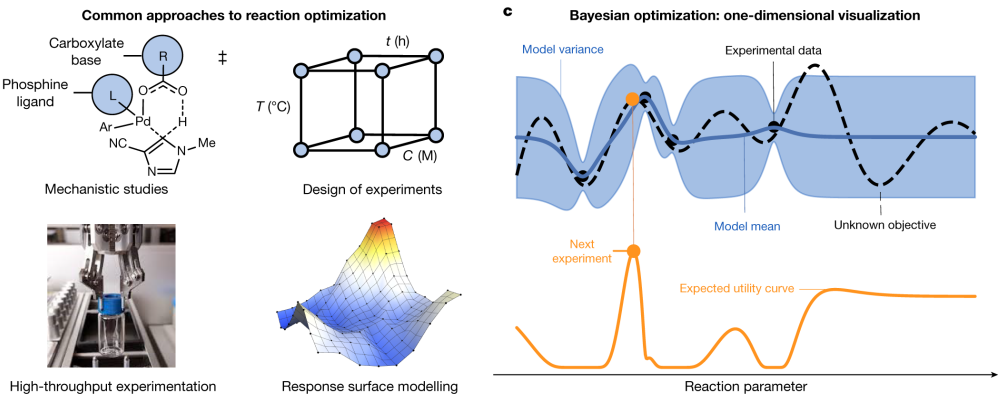


Figure 1: From *Bayesian optimization as a tool for chemical synthesis* by Shields et al. (2021)

Application: Molecule/Protein Discovery

In the field of molecule and protein design, there are similar considerations to the ones in the previous application: experiments are costly. Figure 2 shows the integration of the Bayesian optimization approach with experimentation. In most environments, scientists synthesize and test several different formulations to obtain a dataset. This dataset is used to help model the underlying objective and can be used to suggest new, promising formulations. Then, as shown in the figure, the new formulations are synthesized, tested, and added to the dataset so as to inform the model and optimizer of formulations to suggest next.

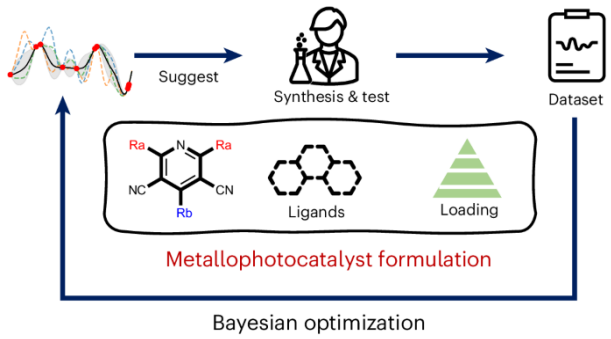


Figure 2: From *Sequential closed-loop Bayesian optimization as a guide for organic molecular metallophotocatalyst formulation discovery* by Li et al. (2024)

Application: Materials Discovery

Figure 3 is similar to the one for molecule/protein discovery but now it is showcasing some additional details specific to materials design and discovery. Again, the initial dataset is used to model the underlying objective and inform design exploration, with results from design exploration being used to augment the dataset. Additionally, in this particular example, the researchers also use the experiments to assist in calibrating a simulation of designs, another application of Bayesian optimization.

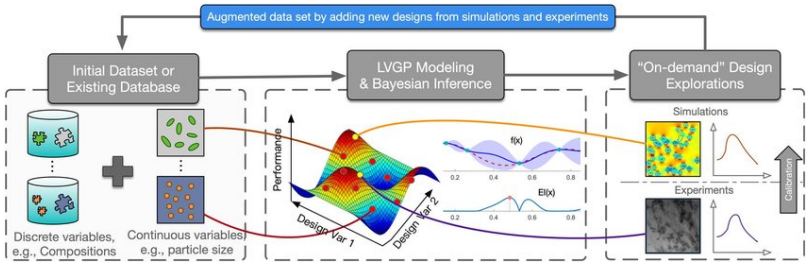


Figure 3: From *Bayesian optimization for Materials Design with Mixed Quantitative and Qualitative Variables* by Zhang et al. (2020)

Application: AutoML

Figure 4 shows a workflow to tune neural network hyperparameters using Bayesian optimization, specifically using a Gaussian process as a surrogate model. AutoML is the process of automating the machine learning workflow and typically includes tuning models' hyperparameters. We can use Bayesian optimization to perform this tuning in a more efficient manner by identifying the next set of promising hyperparameters based on the current model and its uncertainty.

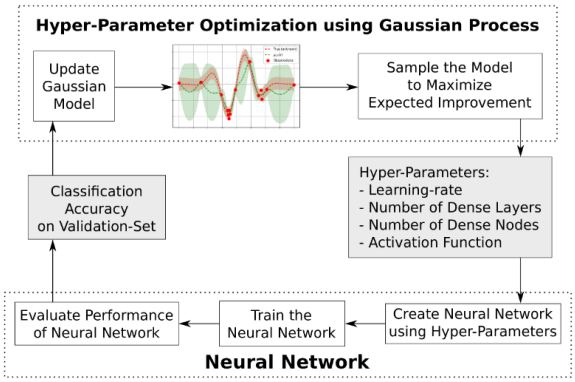


Figure 4: From *Achieve Bayesian optimization for tuning hyperparameters* by Edward Ortiz on *Medium* (2020)

Application: Engineering Decisions

Figure 5 shows how Bayesian optimization can be used to calibrate a particle accelerator in a similar manner to the previous applications. The "operator" inputs the target beam parameters while a camera inputs the observed beam parameters. Then, Bayesian optimization determines the changes (i.e., the next set of beam parameters) to ideally improve the calibration of the particle accelerator.

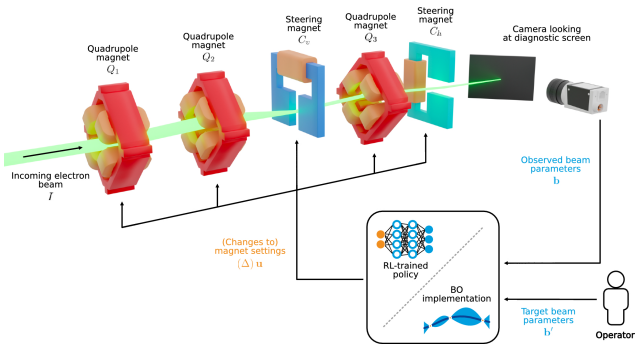


Figure 5: From *Reinforcement learning-trained optimizers and Bayesian optimization for online particle accelerator tuning* by Kaiser et al. (2024)

3 Optimization Foundations

In this section, we will examine the foundations of optimization to understand the ideas and methods that Bayesian optimization builds on. This will include formalizing optimization, objective functions, observation models, some common optimization policies, termination policies, and a diagram of the optimization process. For a more thorough examination of optimization, see Nocedal and Wright's *Numerical Optimization* (Springer Series in Operations Research).

Optimization is a process and field of study that aims to efficiently locate the optimal objective value and/or its location from the search domain and its corresponding objective values.

3.1 Formalization of Optimization

Let's first formalize a typical optimization problem. This formulation is a simple and flexible one for global optimization and is not inherently Bayesian. Additionally, it is also formulating a *continuous* optimization problem but there exists other "types" of optimization other than just continuous.

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

where $f : \mathcal{X} \to \mathbb{R}$ is a real-valued *objective function* on some domain $\mathcal{X}$, x^* is the point that obtains the global maximum value f^*. Note that the max versus min is arbitrary and depends on the specific problem.

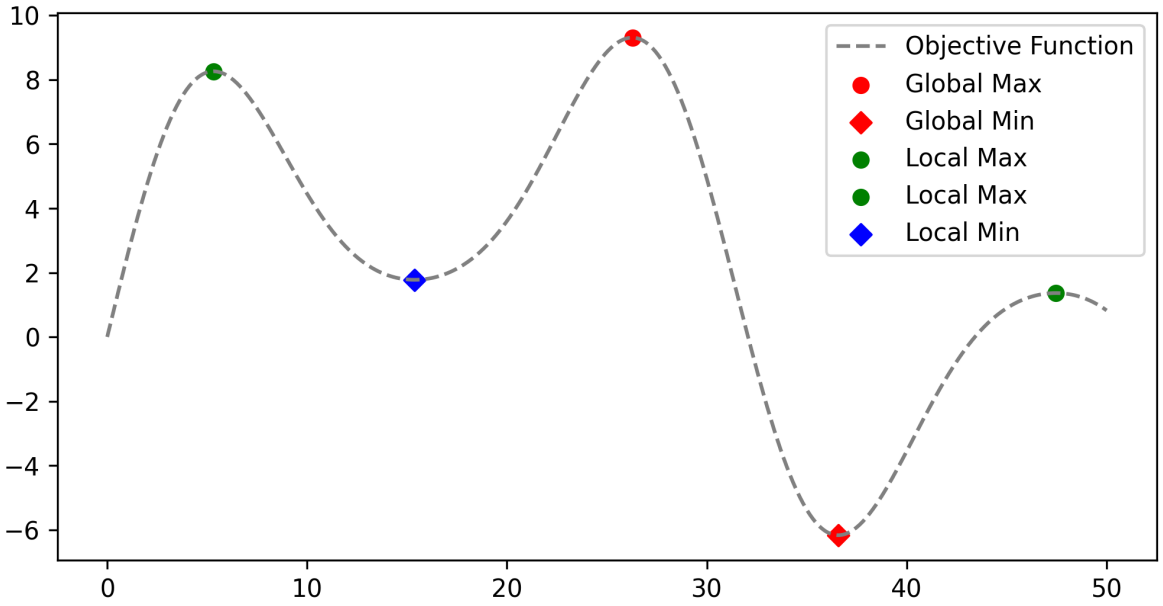
Black-box optimization arises from the fact that we do not need to have an explicit objective function f but rather only some information about the objective at identified points.

The code below creates a plot showing a objective function and points showing the *optima*. Optima are the points that either maximize or minimize (optimize) the objective function.

$$f(x) = 10 \sin(0.2x) \cos(0.1x) + \sin(0.5x) + 0.05x$$

```
# Plot Objective Function with Optima Labeled

def objective(x:np.array)->np.array:
    return (np.sin(0.2 * x) * np.cos(0.1 * x) * 10 +
            np.sin(0.5 * x) + 0.05 * x)

x = np.linspace(0, 50, 1000)
y = objective(x)
```

```
plt.figure(figsize=(8, 4))
plt.plot(x, y, label="Objective Function", c="gray", linestyle="--")
plt.scatter(x[np.argmax(y)], np.max(y), label="Global Max", c="r")
plt.scatter(x[np.argmin(y)], np.min(y), label="Global Min", c="r", marker="D")
plt.scatter(x[np.argmax(y[:200])], np.max(y[:200]), label="Local Max",
            c="g", marker="o")
plt.scatter(x[np.argmax(y[800:]) + 800], np.max(y[800:]), label="Local Max",
            c="g", marker="o")
plt.scatter(x[np.argmin(y[200:400]) + 200], np.min(y[200:400]),
            label="Local Min", c="b", marker="D")
plt.legend()
plt.show()
```



It must be emphasized that the use of NumPy's functions for finding the argument (location) and value for each minimum/maximum is purely for the purposes of visualization. In optimization, we would typically want to avoid computing the objective function at each location in the space but rather use methods to find the optima in a more efficient manner.

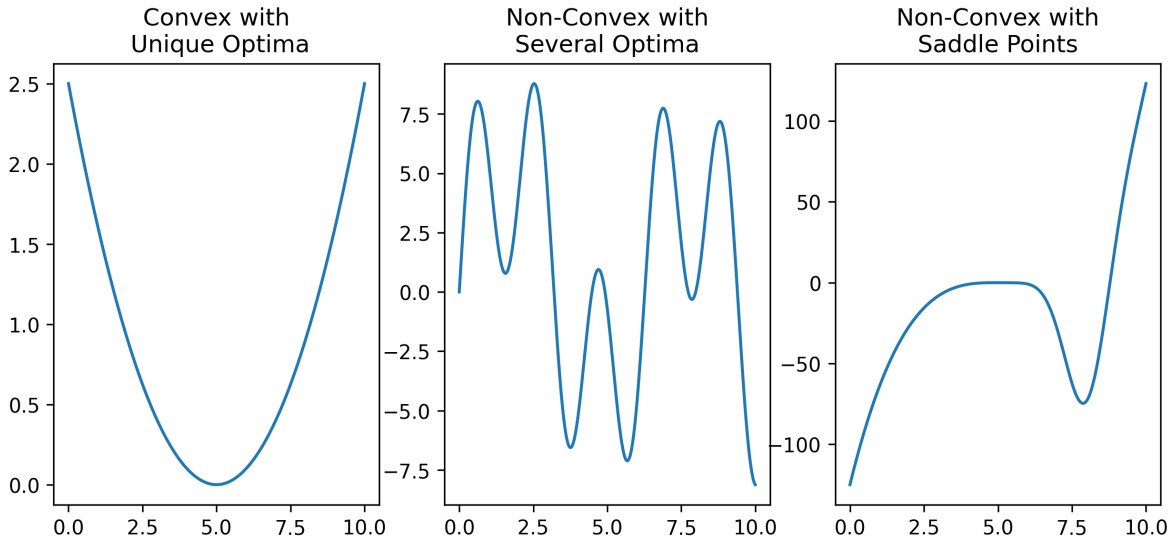
3.2 Objective Functions

The **objective function** is the function that we want to optimize. For some problems, there is a mathematical model that can be developed to describe the objective. However, in many

real-world applications, there is no analytical or mathematical model to describe the objective (one of the motivations for Bayesian optimization).

Some objective functions are convex with only a single, unique global minimum or maximum. Other functions are non-convex and might have several local optima or a flat region with many saddle points. The plot below shows these three general types of objectives.

```
# Plot General Types of Objective Functions

def objective_convex(x:np.array)->np.array:
    return 0.1*(x - 5)**2

def objective_non_convex(x:np.array)->np.array:
    return np.sin(2*x) * np.cos(x) * 10 + np.sin(0.5 * x) + 0.05 * x

def objective_non_convex_saddle(x:np.array)->np.array:
    return (x - 5)**3 - 100 * np.exp(-((x - 8)**2))

x = np.linspace(0, 10, 1000)

fig, ax = plt.subplots(nrows=1, ncols=3, figsize=(10, 4))
ax[0].plot(x, objective_convex(x))
ax[0].set_title("Convex with\n Unique Optima")
ax[1].plot(x, objective_non_convex(x))
ax[1].set_title("Non-Convex with\nSeveral Optima")
ax[2].plot(x, objective_non_convex_saddle(x))
ax[2].set_title("Non-Convex with\nSaddle Points")
plt.show()
```



While the plots showing non-convex objective functions pose difficulty to efficiently optimize such functions, there are other characteristics (listed below) of objective functions common in Bayesian optimization that also lead to issues.

- The objective is considered to be a `black-box` function meaning we can only interact with the objective via its inputs and outputs.
- The objective's returned value is corrupted by some sort of noise and does not represent the exact true objective value at that location.
- The cost of evaluating the objective function is high and requires a sample efficient method to avoid expensive probing.
- There do not exist gradients (if we did, efficient gradient-based methods could be employed to locate and evaluate optima).

3.3 Observation Models

Observation models are similar to the idea of surrogate models used in the Bayesian optimization workflow but there are some key differences. The observation is used to describe how the true objective is *observed*, usually accounting for some form of noise. On the other hand, the surrogate model is a probabilistic model (i.e., Gaussian process) used to approximate the unknown objective function.

Specifically, the **observation model** is an approach to formalize the relationship between the true objective function, the actual observation, and the noise. This is rather important since the model used to relate the true objective and actual observations must account for

uncertainty due to noise. Mathematically, this is the probability distribution of y given the sample location x and true objective function f:

$$p(y|f, x)$$

To account for uncertainty, we assume that the observations are *stochastically* dependent on the objective. Mathematically, we assume an additive noise term ε:

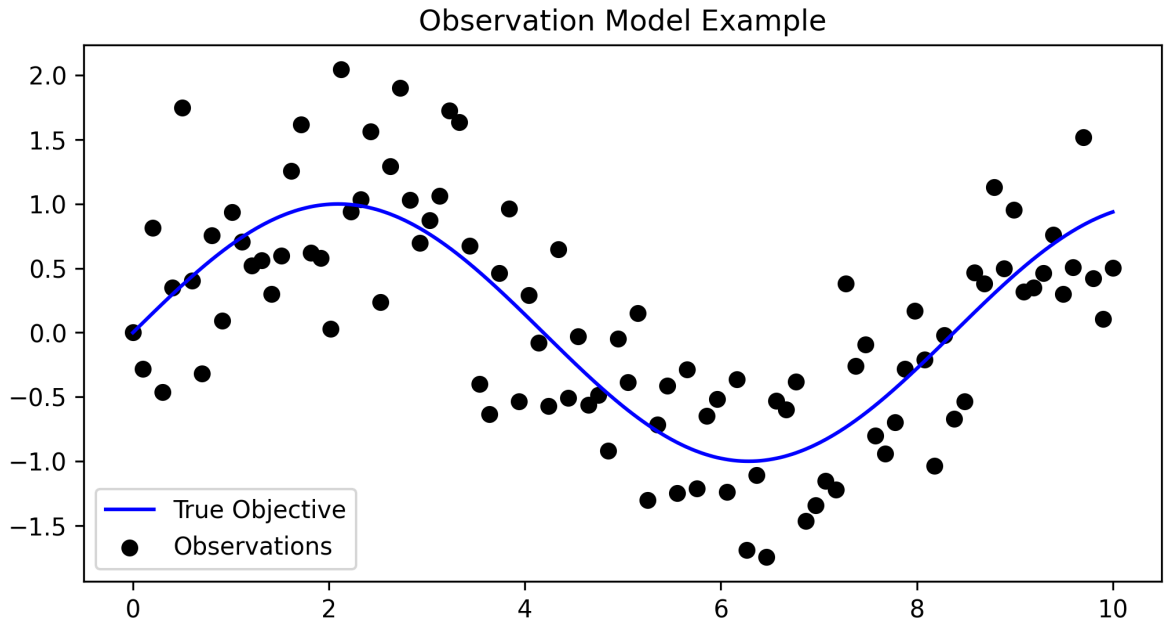
$$y = f(x) + \varepsilon$$

Let $\varepsilon \sim \mathcal{N}(0, \sigma^2)$. Then, the model becomes:

$$p(y|x, f, \sigma) = \mathcal{N}(y; f, \sigma^2)$$

Thus, the observation y at sample location x is treated as a *random variable* which follows a Gaussian, or normal, distribution with mean f and variance σ^2. This leads to the distribution of y being centered around the true objective value at sample location x, $f(x)$.

```
# Plot Example Observation Model

def true_objective(x:np.array)->np.array:
    return np.sin(0.75 * x)

def observation(x:np.array, sd:float)->np.array:
    return true_objective(x) + np.random.normal(0, sd, x.shape)

x = np.linspace(0, 10, 1000)
samples = np.linspace(0, 10, 100)

plt.figure(figsize=(8, 4))
plt.plot(x, true_objective(x), label="True Objective", c="b")
plt.scatter(samples, observation(samples, sd=0.5), label="Observations", c="k")
plt.title("Observation Model Example")
plt.legend()
plt.show()
```



In the plot above, the true objective is shown in addition to 100 observations that have an additive term of random (Gaussian, or normal) noise. The observation model will need to take this into account since noise, often called *uncertainty* in Bayesian optimization, is inherent in real-world problems. It can come from measurement errors, environmental factors, or simply the randomness in the system being optimized.

3.4 Optimization Policies

In simple terms, the optimization policy handles the repeated interactions between the "inner-workings" of the policy and the environment, with the environment typically being *noise-corrupted* (hence the need for an observation model).

More definitive, a **policy** is a mapping function that takes in a new input observation plus any existing observations and uses a *principled* sampling approach to output the next observation location. It will also decide if it should perform another iteration (select a new observation) or terminate the optimization process (see *Termination Policies* below).

For most applications, we want the policy to ideally be learning and improving such that it will guide the search toward the global optimum more effectively. Furthermore, the iterative policy improvement should lead to a good policy that retains the (typically limited) sampling budget for more promising candidate points. A policy that does not consider observed data are known as a *non-adaptive* policy and is not ideal for costly observations.

Note that in some literature, there is little distinction between the optimization policy and the termination policy (discussed next). To be clear, the termination policy is one of the several

components that make up the optimization policy. This lack of distinction is not unreasonable but rather something to be aware of when reviewing the vast literature on optimization.

3.5 Termination Policies

A **termination policy** is the final decision in the optimization loop. The policy decides whether to terminate immediately or continue with another observation (continue to optimize the objective). Such policies can be *deterministic* or *stochastic*.

- A **deterministic termination policy** is one that will stop the optimization process after reaching a goal or exhausting a pre-defined search budget.
- A **stochastic termination policy** is one that will depend on the observed data and some level of randomness or probability.

Note that this piece of the optimization process can be handled by an external agent or be dynamic (i.e., a deterministic or stochastic termination policy). For the purposes of this notebook, we will not focus on specific termination policies here. This is primarily due to the fact that the termination policy depends heavily on the approach or method and the problem/application.

3.6 Diagram of the Optimization Process

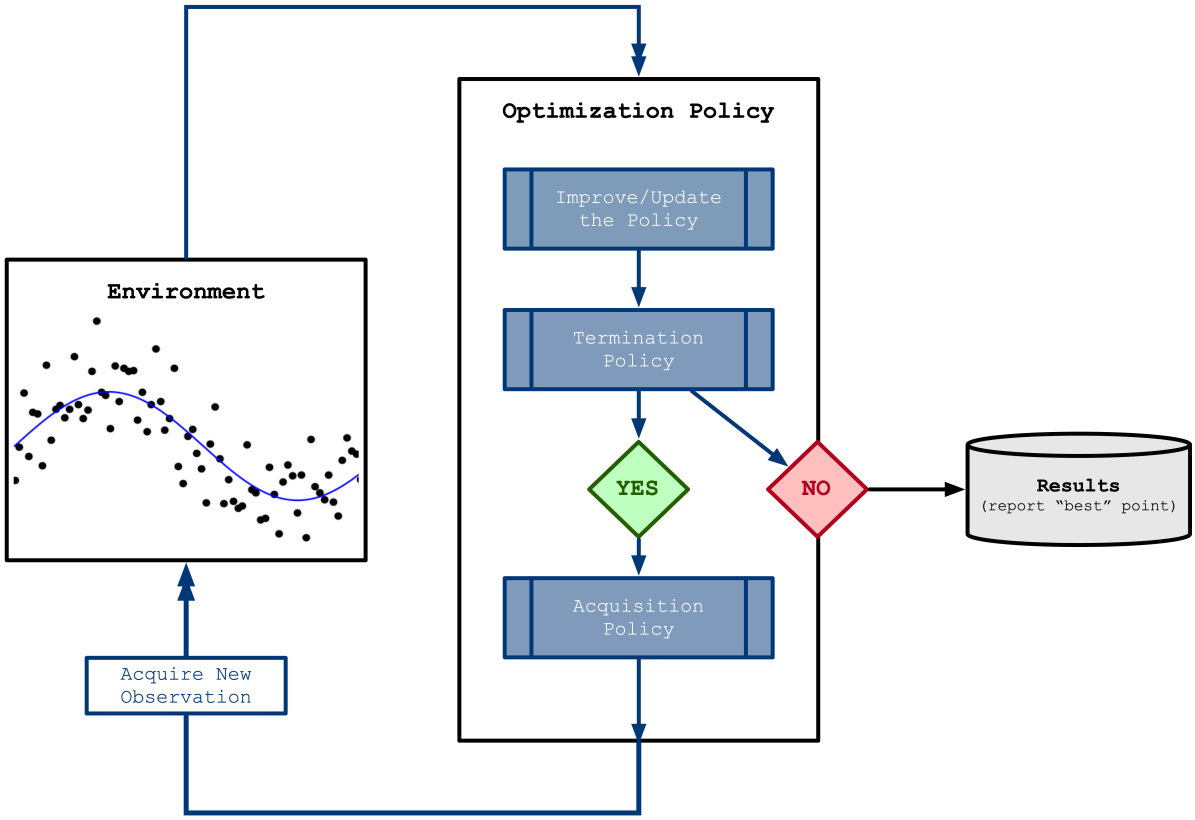


Figure 6: General Optimization Process Diagram

4 Bayesian Foundations

Before we venture into the world of Bayesian optimization, we must first review some foundations of Bayesian statistics. For a more comprehensive examination of Bayesian statistics, the reader is referred to *Mathematical Statistics and Data Analysis* by John A. Rice or *Bayesian Optimization* by Roman Garnett (which provides an optimization-focused review).

In Bayesian optimization, we use Bayesian statistics to reason in a systematic and quantitative manner about the uncertainty in the observation (surrogate) model using probabilities.

One of the core ideas in Bayesian statistics is **Bayesian inference**. Bayesian inference uses Bayes' Theorem to reason about how the prior distribution $p(\theta)$, the likelihood $p(\text{data}|\theta)$, and the posterior distribution $p(\theta|\text{data})$ interact with each other (where θ represents the parameter of interest). Bayes' Theorem is given below:

$$p(\theta|\text{data}) = \frac{p(\text{data}|\theta)p(\theta)}{p(\text{data})}$$

Let's back up a bit and break this down. First, in simpler terms, Bayesian inference is a framework used to infer uncertain features (variables) of a particular system of interest from observations using the laws of probability.

Within Bayesian inference framework, all unknown quantities are considered to be *random variables*. This enables us to represent our beliefs with probability distributions that reflect plausible values. We begin first with a **prior distribution**, $p(\theta)$ that represents our beliefs before seeing any data. For instance, if we believe that the data is normally distributed then we would define the prior distribution to be the Gaussian normal distribution with mean μ and standard deviation σ.

Next, we are able to refine our initial beliefs after we have observed some data using the **likelihood** $p(\text{data}|\theta)$. The likelihood, or likelihood function, provides the distribution of observed values (y) given the location (x) and values of interest (θ).

Using the observed value y, we can derive the **posterior distribution** of θ using Bayes' theorem, as defined previously.

$$p(\theta|x, y) = \frac{p(\theta)p(y|x, \theta)}{p(y|x)}$$

This so-called posterior distribution represents a "compromise" between our beliefs or experience from the prior and our observations from the likelihood. The posterior distribution is at the heart of Bayesian optimization since we use it to update the surrogate model as we acquire new observations.

4.1 Bayesian Inference of the Objective Function

The primary use of Bayesian inference in Bayesian optimization is with respect to the objective function, the source of uncertainty. The probabilistic belief we use over the objective function is called a *stochastic process*.

A **stochastic process** is a probability distribution over an infinite collection of random variables, for example, the objective function value at every point in the domain.

We will express any assumptions that we may have about the objective function in a **prior process** $p(\{)$. Then, a stochastic process can be defined via the distribution of the function values $\boldsymbol{\phi}$ given a finite set of points $\boldsymbol{x}$:

$$p(\boldsymbol{\phi}|\boldsymbol{x})$$

In Bayesian optimization, the gold standard stochastic process is the **Gaussian process** since many of these finite-dimensional distributions are multivariate Gaussian (or approximately so).

Returning to the discussion of Bayesian inference over the objective, suppose we make a set of observations at locations $\boldsymbol{x}$ with corresponding vaulues $\boldsymbol{y}$. Let $\mathcal{D} = (\boldsymbol{x}, \boldsymbol{y})$ be the dataset of aggregated observations.

Bayesian inference will account for these observations via the formation of the **posterior process** (akin to the posterior predictive distribution):

$$p(\{|\mathcal{D}) = \int p(\{|\boldsymbol{x}, \boldsymbol{\phi})p(\boldsymbol{\phi}|\mathcal{D}) \, \mathrm{d}\boldsymbol{\phi}$$

5 The Bayesian Approach

The notion of the "Bayesian approach" (particularly in the context of optimization) refers to a philosophical approach derived from Bayesian inference. The Bayesian approach allows us to tackle uncertainty inherent in optimization decisions, a crucial property since decisions will determine our success. To do this, it systematically relies on probability laws and Bayesian inference to reason about uncertainty during optimization.

For instance, as briefly discussed in the previous section, the objective function is regarded as a random variable that is to be informed by our prior expectations and prior observations. This approach will play an active role in the optimization as it will guide the optimization policy, specifically by evaluating the merit of a candidate observation. Furthermore, it leads to the development of **uncertainty-aware optimization policies**.

5.1 Uncertainty-Aware Optimization Policies

Recall that the optimization policy determines the decisions, or actions, taken during the optimization process. In order to handle the uncertainty of the (potentially unknown or black-box) objective function, the policy should examine the available data (i.e., pre-existing observations and corresponding values) to determine the each successive observation location optimally. The only requirement from ourselves is to establish our preferences for what data or what kind of data we want to acquire and then maximize such preferences.

Clearly, this will require some sort of framework to make decisions in this way, especially in the face of uncertainty. A natural choice is **Bayesian decision theory** which will be discussed in more detail in another notebook.

As we continue to explore Bayesian optimization and uncertainty-aware policies, a common theme will begin to emerge: all Bayesian optimization policies handle the uncertainty in the

objective function in a uniform manner, a property that is defined implicitly within an optimal *acquisition function*.

6 Bayesian Optimization Workflow

The Bayesian optimization workflow is composed of two main *primitives*: the surrogate model for expressing our beliefs and knowledge about the objective function, and the acquisition function that will guide the optimization policy to make optimal decisions about where to observe next. In this section, we will briefly discuss these two primitives to understand how they fit into this workflow. Note that these topics will be discussed more thoroughly in other notebooks.

6.1 Surrogate Models

Surrogate models are the models that we use to express our beliefs and knowledge about the objective function. While in some literature the term *observation model* is used interchangeably with *surrogate model*, they are distinguishable. In most applications, if we know some sort of mathematical formulation of the true objective then we refer to the model that relates the observations and the true objective as an *observation model*. However, if we do not know the underlying structure (as is the case with black-box functions), we refer to the model as a *surrogate model* since it "takes the place of" and acts "as a surrogate of" the true underlying objective function.

The specific use of the surrogate model is usually when we use the predictive posterior distribution (posterior process) to quantify the probabilities of observations usually in conjunction with utility functions. Therefore, the surrogate model is a powerful tool when we are reasoning about both the uncertainty and utility of candidate observations.

Recall that the probability distributions of an infinite collection of random variables are jointly used as a *stochastic process* that is used to characterize the true objective function. In Bayesian optimization, a common choice for the surrogate model in the family of stochastic processes is the **Gaussian process**. It is very common due to its flexibility, expressivity, and sufficient uncertainty quantification properties. For Gaussian processes specifically, the uncertainty is quantified via the credible interval but we will discuss such processes in more detail later.

6.2 Acquisition Functions

Acquisition functions assign a score to candidate observations with the score representing the potential benefit the observation will add to the optimization process. Ideally, acquisition functions are cheap to evaluate and should address the *exploitation-exploration tradeoff*. Within this tradeoff, exploitation refers to sampling where the objective function value is expected to

be large while exploration refers to sampling where we are more uncertain about the value of the objective function. Another important property of an acquisition function is that it will vanish at pre-existing observations where the objective function value is already known (no sense in sampling twice).

The optimization policy should be designed to maximize the acquisition function such that we select the candidate observation with the most potential to benefit the optimization process. There are two general cases of acquisition function: myopic and look-ahead. Myopic acquisition functions only consider the immediate utility while look-ahead acquisition functions will consider longer-term utility (see the notebook on decision theory for more information).

Finally, the policy that uses a particular acquisition function can be evaluated using **instant (simple) regret**. Instant regret measures the distance between the current and optimal locations whereas cumulative regret measures the cumulative distance between historical locations and the optimum location.

7 References

For full-reference details, the BibTeX entries can be found in the `bibliography.bib` file.

- *Bayesian Optimization* by Roman Garnett (2023)
- *Bayesian Optimization: Theory and Practice Using Python* by Peng Liu (2023)
- *Gaussian Processes for Machine Learning* by Rasmussen & Williams (2019)