Bayesian Optimization with Gaussian Processes

# 1. Introduction to Bayesian Optimization

Bayesian Optimization (BO) is a powerful strategy for optimizing black-box functions that are expensive to evaluate. Instead of randomly trying different points (like in grid search or random search), BO builds a probabilistic model of the function and uses it to choose the most promising points to evaluate next.

The key components of BO are:  
1. A Gaussian Process (GP) that models the unknown function and gives both predictions and uncertainties.  
2. An Acquisition Function that uses the GP's predictions to decide where to sample next.

# 2. Gaussian Process (GP): The Core Model

A Gaussian Process (GP) is a non-parametric model that defines a probability distribution over functions. It is particularly useful because it provides both:  
 - A mean prediction (the GP's best guess of the function).  
 - A measure of uncertainty (how confident the GP is about its guess).

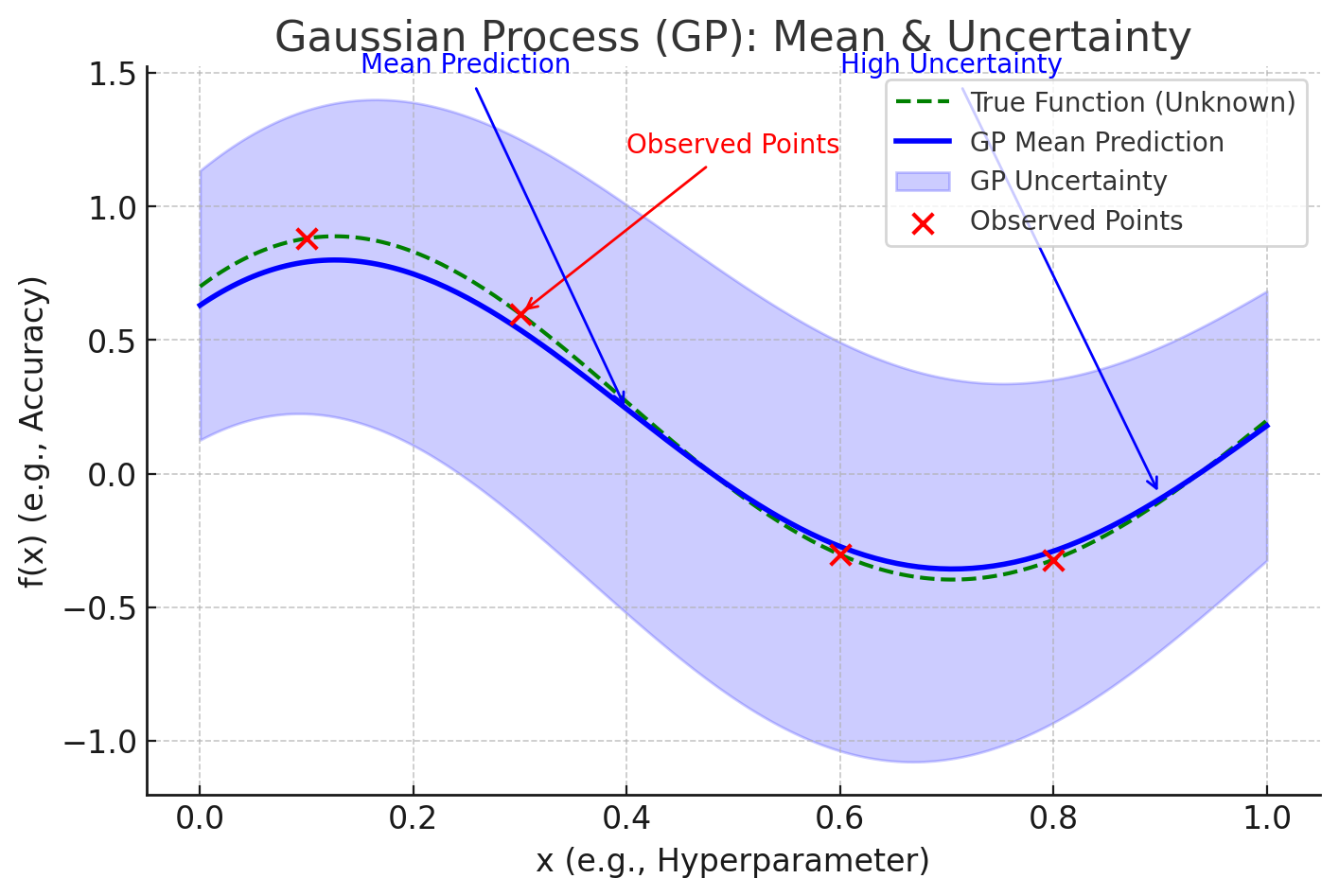
 Unlike a normal regression, it doesn’t force one fixed curve; instead, it keeps a **probability distribution over possible curves**.

 The **mean (blue line)** is the "average curve" GP believes in.

Mathematically, a GP is represented as:  
 f(x) ~ GP(m(x), k(x, x'))  
where m(x) is the mean function (often zero) and k(x, x') is the kernel function that measures similarity between two points x and x'.

The GP updates its predictions every time we observe a new data point. Near observed points, the GP is confident (narrow uncertainty), but far away, the uncertainty is high (wide confidence intervals).

The diagram below illustrates how a GP models the function:



**Bayesian Optimization doesn’t just pick high or low uncertainty — it balances two things:**

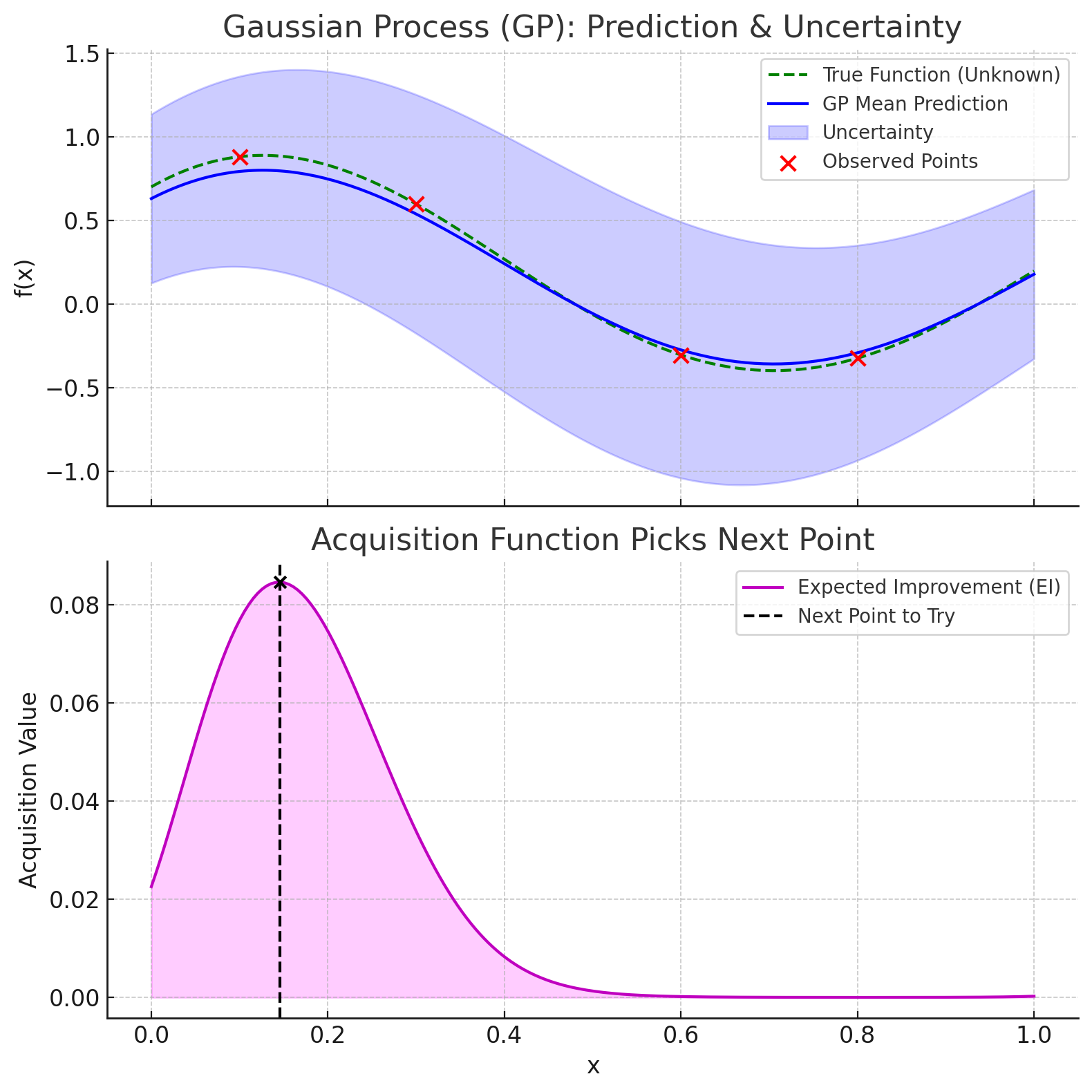
**1. Exploration vs. Exploitation**

* **Exploration:**  
  If the GP is **very uncertain** in some region (wide shaded area), the algorithm may try that region because there could be a hidden "peak" we haven't discovered yet.  
  *(Example: Trying a new hyperparameter value we haven’t tested much.)*
* **Exploitation:**  
  If the GP predicts **high performance** in some region (blue curve is high), the algorithm may try that region again to refine the best result.  
  *(Example: Trying hyperparameters around the currently best accuracy.)*

# 3. Acquisition Function: Balancing Exploration and Exploitation

The Acquisition Function is responsible for selecting the next point to evaluate. It uses both the mean and uncertainty from the GP to balance:  
 - Exploitation: Focusing on areas where the predicted value is high.  
 - Exploration: Trying areas where the model is uncertain, which could potentially have better values.

Common acquisition functions include:  
 - Expected Improvement (EI): Chooses the point expected to improve the best-known result.  
 - Probability of Improvement (PI): Chooses the point with the highest probability of beating the current best.  
 - Upper Confidence Bound (UCB): Chooses the point that maximizes (mean + uncertainty \* kappa).



**Step 1: What We Have After GP**

When we build a Gaussian Process (GP), for each possible point xxx (e.g., a hyperparameter value), we have:

* **Mean (μ(x))**– GP’s best guess for f(x)
* **Uncertainty (σ(x))**– GP’s confidence (wide if we have no data nearby).

Think of it like:

* **Mean (μ(x))** = "How good I think this point is."
* **Uncertainty (σ(x))**= "How unsure I am about this point."

**Step 2: Our Goal**

We want to choose **the next x** to evaluate that is most likely to give us a better result (improvement).

But we have two scenarios:

* We might try **near the current best point** because GP predicts it’s high (**exploitation**).
* Or we might try **in uncertain areas** that could be surprisingly better (**exploration**).

The acquisition function combines these two.

**Step 3: Expected Improvement (EI) – The Most Common Acquisition Function**

EI asks:  
**"How much improvement can I *expect* from this point over the current best f(x+)f(x^+)f(x+)?"**

Formula:

where:

* Φ\PhiΦ = Cumulative distribution function (probability that f(x)f(x)f(x) beats current best).
* ϕ\phiϕ = Probability density (how spread the possible values are).

**In simple terms:**

* If μ(x)\mu(x)μ(x) is far above the current best, EI is large.
* If σ(x)\sigma(x)σ(x) is large (uncertain), EI is also large — because we might "get lucky."
* If both are small, EI is near zero (no point checking).

**Step 4: Searching for the Next Point**

1. For every candidate x (say, 0.01 to 0.1 for learning rate), calculate EI(x).
2. Find the x where EI is maximum.
3. That x is chosen as the next point to evaluate the **real function**.

**Step 5: After Choosing the Point**

* We evaluate the true function at this x (train the model with that hyperparameter).
* Add this (x, f(x)) to the observed data.
* Update the GP with this new point.
* Repeat the process.

**Step 6: Visual Example**

Imagine a graph:

* The **blue curve** is the GP mean.
* The **shaded blue** area is uncertainty.
* The **purple curve (below)** is EI.
  + **Where EI peaks = the next point to try.**
  + Sometimes it’s just after the best point (exploit).
  + Sometimes it’s where uncertainty is high (explore).

**Step 7: Why This Works**

* EI gives **a single score per point** by combining "how promising" and "how uncertain" into one number.
* Instead of guessing randomly, it always picks the point with the **highest probability of improving our current best.**

# 4. The Bayesian Optimization Workflow

1. Start with a few random evaluations of the function.  
2. Fit a GP to these observed points.  
3. Use the acquisition function to determine the next point to evaluate.  
4. Evaluate the function at this point (expensive step).  
5. Update the GP with the new data point.  
6. Repeat steps 3-5 until the stopping condition is met (budget or convergence).

# 5. Example: Hyperparameter Tuning

In machine learning, training a model can be very expensive. Instead of trying all combinations of hyperparameters, Bayesian Optimization helps by predicting which combination is most promising. For example, if we are tuning learning rate and dropout, the GP models the validation accuracy as a function of these parameters, and the acquisition function suggests the next combination to try.

# 6. Python Example with scikit-optimize

from skopt import gp\_minimize  
from skopt.space import Real  
from skopt.utils import use\_named\_args  
  
# Define the search space  
space = [  
 Real(0.001, 0.1, name='learning\_rate'),  
 Real(0.01, 1.0, name='dropout')  
]  
  
@use\_named\_args(space)  
def objective(\*\*params):  
 lr = params['learning\_rate']  
 dropout = params['dropout']  
 return (lr - 0.02)\*\*2 + (dropout - 0.5)\*\*2 + 0.01  
  
# Run Bayesian Optimization  
result = gp\_minimize(objective, space, n\_calls=20, random\_state=42)  
  
print('Best parameters:', result.x)  
print('Best objective value:', result.fun)