# Bayesian Optimization with Gaussian Process: A Beginner's Guide

**Hoh Siew Yan (何守仁)**

**22 May 2026**


XIAMEN UNIVERSITY MALAYSIA
厦門大學馬來西亞分校

# Introduction

**Given**

Solid target parameters

Laser's parameters

Injection parameters

What is the laser energy absorbed by the target?

# Introduction

**Given**

Solid target parameters

Laser's parameters

Injection parameters

What is the laser energy absorbed by the target?

How high is good? Depends on the physics story

# Introduction

**Given**

Solid target parameters

Laser's parameters

Injection parameters

What is the laser energy absorbed by the target?

How high is good? Depends on the physics story

**For now, we wanted as maximal as possible : optimization problem**

# Motivation

## Parameter of interests

```
begin:constant
  # laser parameters
  lambda0   = 0.8 * micron
  Intensity = {{INTENSITY}}
  t_laser   = {{TLASER}} * femto
  # derived laser parameters
  omega     = (2.0*pi*c)/lambda0
  period = (2.0*pi)/omega
  sigma_t   = t_laser/(2*sqrt(loge(2)))
  n_crit    = critical(omega)
  mass_density = 8.96 / cc
  mass_number = 64
  max_dens = mass_density * 6.02e23 /
  x0 = 10 * micron
  scale_x = {{SCALEX}} * lambda0
  slab_thickness = {{SLABTHICKNESS}} *
```

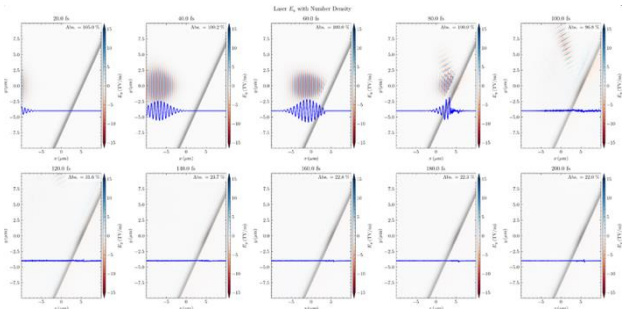
intensity

T_laser

Scale_x

Slab_thickness

## Objective function



## “The Objective”

```
# 5. evaluate objective
sdf_file = os.path.join(
    self.current_output_directory,
    "0010.sdf"
)

result = self.compute_absorption(sdf_file)
```

Absorption/Fraction of Laser Energy Absorbed (%)

# Motivation

## Parameter of interests

```
begin:constant
  # laser parameters
  lambda0    = 0.8 * micron
  Intensity = {{INTENSITY}}
  t_laser    = {{TLASER}} * femto

  # derived laser parameters
  omega      = (2.0*pi*c)/lambda0
  period = (2.0*pi)/omega
  sigma_t    = t_laser/(2*sqrt(loge(2)))
  n_crit     = critical(omega)

  mass_density = 8.96 / cc
  mass_number = 64
  max_dens = mass_density * 6.02e23 /

  x0 = 10 * micron
  scale_x = {{SCALEX}} * lambda0
  slab_thickness = {{SLABTHICKNESS}} *
```

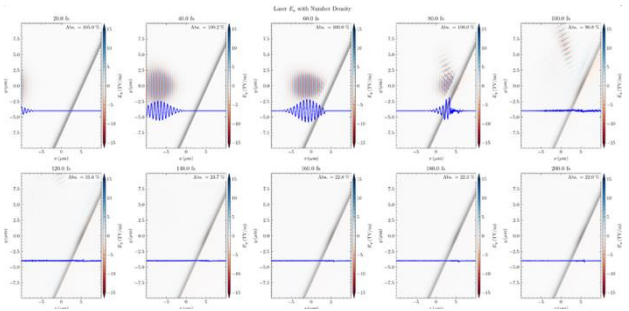
intensity

T_laser

Scale_x

Slab_thickness

## Objective function



## “The Objective”

```
# 5. evaluate objective
sdf_file = os.path.join(
    self.current_output_directory,
    "0010.sdf"
)

result = self.compute_absorption(sdf_file)
```

Absorption/Fractio
n of Laser Energy
Absorbed (%)

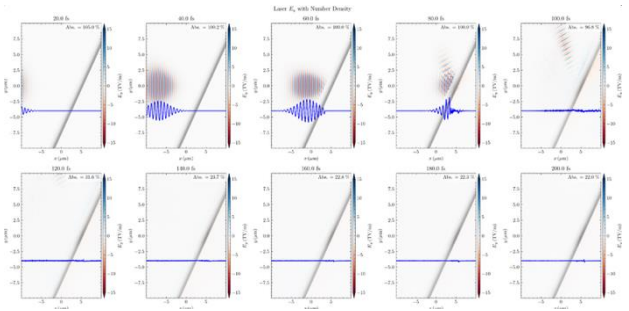
**The goal is to maximize the objective, through simulation, in which, the corresponding parameter of interests will be the set of optimal parameters, which can be used to advise experiment’s design**

# Motivation

Parameter of interests

$$\vec{X} = (\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n)$$

Objective function



"The Objective"

```
# 5. evaluate objective
sdf_file = os.path.join(
    self.current_output_directory,
    "0010.sdf"
)

result = self.compute_absorption(sdf_file)
```

Absorption/Fractio
n of Laser Energy
Absorbed (%)

**The goal is to maximize the objective, through simulation, in which, the corresponding parameter of interests will be the set of optimal parameters, which can be used to advise experiment's design**

# Motivation

Parameter of interests

$$\vec{X} = (\vec{x}_1, \vec{x}_2, \dots, \vec{x}_n)$$

Objective function

$$f(\vec{X} = (\vec{x}_1, \vec{x}_2, \dots, \vec{x}_n))$$

Aka Black Box

"The Objective"

```
# 5. evaluate objective
sdf_file = os.path.join(
    self.current_output_directory,
    "0010.sdf"
)

result = self.compute_absorption(sdf_file)
```

Absorption/Fraction of Laser Energy Absorbed (%)

**The goal is to maximize the objective, through simulation, in which, the corresponding parameter of interests will be the set of optimal parameters, which can be used to advise experiment's design**

# Motivation

Parameter of interests

$$\vec{X} = (\vec{x}_1, \vec{x}_2, \dots, \vec{x}_n)$$

Objective function

$$f(\vec{X} = (\vec{x}_1, \vec{x}_2, \dots, \vec{x}_n))$$

Aka Black Box

"The Objective"

$$Y = f(\vec{X} = (\vec{x}_1, \vec{x}_2, \dots, \vec{x}_n))$$

**The goal is to maximize the objective, through simulation, in which, the corresponding parameter of interests will be the set of optimal parameters, which can be used to advise experiment's design**

# Motivation

Parameter of interests

$$\vec{X} = (\vec{x}_1, \vec{x}_2, \dots, \vec{x}_n)$$

Objective function

$$f(\vec{X} = (\vec{x}_1, \vec{x}_2, \dots, \vec{x}_n))$$

Aka Black Box

"The Objective"

$$Y = f(\vec{X} = (\vec{x}_1, \vec{x}_2, \dots, \vec{x}_n))$$

**Goal :** $Y^* = \arg\max_{X^*} f(\vec{X} = (\vec{x}_1, \vec{x}_2, \dots, \vec{x}_n))$

**The maximal objective obtained:** $(X^*, Y^*)$

# Methods

- Uses Bayes' theorem to iteratively improve assumption:

$$posterior \propto likelihood \times prior$$

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$$

# Methods

- Uses Bayes' theorem to iteratively improve assumption:

$$posterior \propto likelihood \times prior$$

$$p(f|D_N) = \frac{p(D_N|f)p(f)}{p(D_N)}$$

Evident

# Methods: Prior

$$p(f|D_N) = \frac{p(D_N|f)p(f)}{p(D_N)}$$

- Before seeing any data, what do we believe about the objective function, $f$?
- By default, Gaussian Process (aka **Surrogate Model**) is used for prior function:

$$p(f(\boldsymbol{x})) = f(\boldsymbol{x}) \sim \mathcal{GP}(\mu, \sigma^2)$$

# Methods: Prior

$$p(f|D_N) = \frac{p(D_N|f)p(f)}{p(D_N)}$$

- Before seeing any data, what do we believe about the objective function, $f$?
- By default, Gaussian Process (aka **Surrogate Model**) is used for prior function:

$$p(f(\boldsymbol{x})) = f(\boldsymbol{x}) \sim \mathcal{GP}(\mu(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}'))$$

Injecting our assumption…

# Methods: Prior

$$f(\boldsymbol{x}) \sim \mathcal{GP}(\mu(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}'))$$

- The mean function and kernel together express our prior assumptions:
  - Typically, that $f$ is smooth, that nearby inputs give similar outputs,
  - the function varies on certain characteristic length scales

# Methods: Prior : Mean function

$$f(\boldsymbol{x}) \sim \mathcal{GP}(\mu(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}'))$$

**The Mean Function $\mu(\mathbf{x})$**

The mean function expresses what we expect $f$ to look like *before seeing any data*. In practice, we rarely have strong prior knowledge about absolute output values, so we typically set $\mu(\mathbf{x}) = 0$ (after centering the data). This is a deliberately weak assumption: "in the absence of evidence, I have no reason to think the function is large or small at any particular point." If we did have domain knowledge — say, a known baseline trend — we could bake it into $\mu(\mathbf{x})$ and let the GP model only the deviations from it.

# Methods: Prior : Kernel Function

**The Kernel $k(\mathbf{x}, \mathbf{x}')$**

The kernel is where the substantive prior assumptions live. It controls *how function values at different inputs are related*, and this single object encodes three distinct beliefs:

**Smoothness.** The choice of kernel family fixes how regular we expect $f$ to be. The RBF kernel, for instance, implies $f$ is infinitely differentiable — a strong smoothness assumption suitable for physical processes that vary gradually. Other kernels (Matérn, for example) encode rougher behavior. Picking a kernel is picking a hypothesis class for the function's regularity.

**Locality.** The kernel decays with distance: $k(\mathbf{x}, \mathbf{x}')$ is large when $\mathbf{x}$ and $\mathbf{x}'$ are close and small when they are far apart. This formalizes the intuition that *nearby inputs should produce similar outputs*. Without this assumption, an observation at one point would tell us nothing about neighboring points, and learning would be impossible.

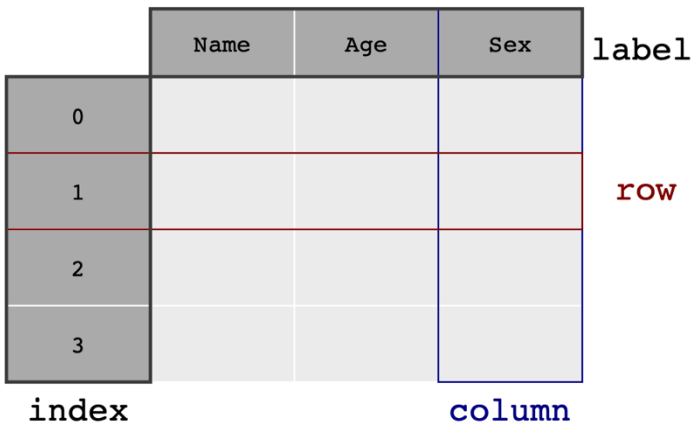
**Characteristic length scales $\ell_i$.** The length scales quantify *how quickly* correlations decay along each input dimension. A small $\ell_i$ means $f$ can change rapidly along dimension $i$ (the function is "wiggly" in that direction); a large $\ell_i$ means $f$ changes slowly (gentle, gradual variation). Crucially, each dimension gets its own $\ell_i$, so the prior can express that, say, the function depends strongly on laser energy but only weakly on incidence angle. These length scales are typically learned from data by maximizing the marginal likelihood, so the prior adapts to the geometry of the actual problem.

$$f(\boldsymbol{x}) \sim \mathcal{GP}(\mu(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}'))$$

# Methods: Likelihood

$$p(f|D_N) = \frac{p(D_N|f)p(f)}{p(D_N)}$$

$$D_N = \{(x_i, y_i)\}_{i=1}^{N}$$

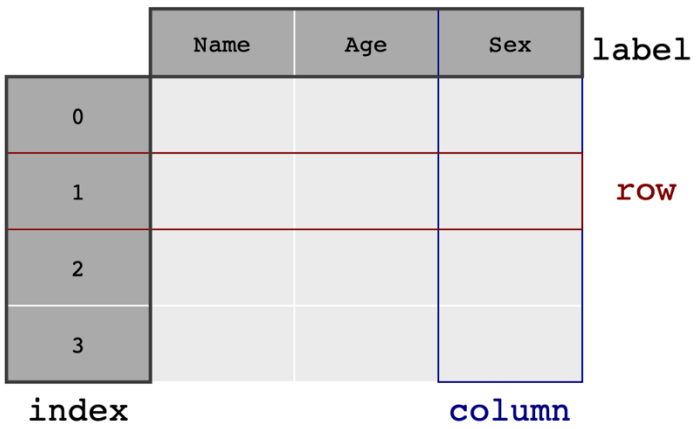



- Likelihood said : “Given a candidate function $f$, how plausible is the data we actually observed?”

# Methods: Likelihood

$$p(f|D_N) = \frac{p(D_N|f)p(f)}{p(D_N)}$$

$$D_N = \{(x_i, y_i)\}_{i=1}^{N}$$




- Likelihood said : "Given a candidate function $f$, how plausible is the data we actually observed?"
- Noise term is introduced:

$$y_i = f(x_i) + \varepsilon_i \qquad \varepsilon_i \sim \mathcal{N}(0, \sigma_{noise}^2)$$

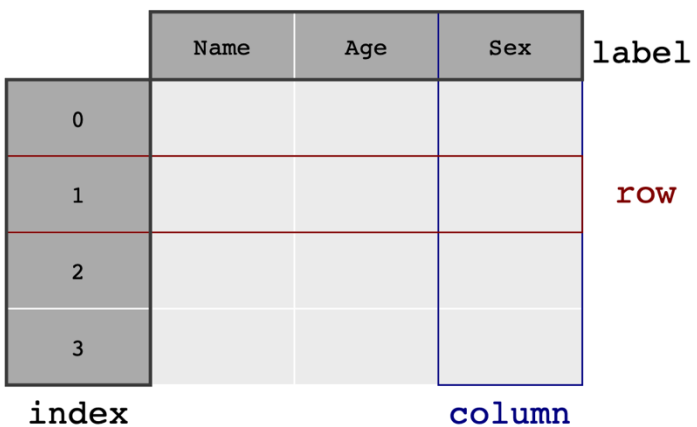
Equivalently,

$$p(y_i|f(x_i)) = \mathcal{N}(y_i|f(x_i), \sigma_{noise}^2)$$

# Methods: Posterior

$$p(f|D_N) = p(D_N|f)p(f)$$

$$D_N = \{(x_i, y_i)\}_{i=1}^{N}$$




- After applying Bayes' theorem, the posterior is *also* a Gaussian Process!
- The evident function cancels out each.
- The computed posterior, has an updated mean and variance (from the prior).
- **The prior belief change, in light of new observation (aka iteration)**

# Methods: Iterative Baye's Theorem

- Before any data present, the prior:

$$f(\boldsymbol{x}) \sim \mathcal{GP}(\mu(\boldsymbol{x}) = 0, k(\boldsymbol{x}, \boldsymbol{x}') = \sigma^2)$$

- After observing the data, the prior belief is updated:

$$f(\boldsymbol{x}) \sim \mathcal{GP}(\mu(\boldsymbol{x}) \neq 0, k(\boldsymbol{x}, \boldsymbol{x}') \neq \sigma^2)$$

# Methods: Iterative Baye's Theorem

- Before any data present, the prior:

$$f(\boldsymbol{x}) \sim \mathcal{GP}(\mu(\boldsymbol{x}) = 0, k(\boldsymbol{x}, \boldsymbol{x}') = \sigma^2)$$

- After observing the data, the prior belief is updated:

$$f(\boldsymbol{x}) \sim \mathcal{GP}(\mu(\boldsymbol{x}) \neq 0, k(\boldsymbol{x}, \boldsymbol{x}') \neq \sigma^2)$$

Precisely:

$$\mu_N(x) = k_*^T (K + \sigma_{noise}^2 I)^{-1} y \qquad \sigma_N^2(x) = k(x, x) - k_*^T (K + \sigma_{noise}^2 I)^{-1} k_*$$

# Methods: Iterative Baye's Theorem

1. The mean is no longer flat, it bends to follow the observations.
2. The variance is no longer uniform, it shrinks near training points and stays large far from them.
3. AKA, you can see that the optimization is in progress!!

$$f(\boldsymbol{x}) \sim \mathcal{GP}(\mu(\boldsymbol{x}) \neq 0, k(\boldsymbol{x}, \boldsymbol{x}') \neq \sigma^2)$$

Precisely:

$$\mu_N(x) = k_*^T (K + \sigma_{noise}^2 I)^{-1} y \qquad \sigma_N^2(x) = k(x,x) - k_*^T (K + \sigma_{noise}^2 I)^{-1} k_*$$

# Methods: Iterative Baye's Theorem

1. The mean is no longer flat, it bends to follow the observations.
2. The variance is no longer uniform, it shrinks near training points and stays large far from them.
3. AKA, you can see that the optimization is in progress!!

We need a **decision theory**, using the updated muon and variance, to predict the next data points.

$k_*$

# Methods: Acquisition Function

- The acquisition function $\alpha(x)$ takes the posterior mean $\mu_N(x)$ and variance (encoding the uncertainty) $\sigma_N^2(x)$, combines them into a single scalar score.
- The score said how desirable is it to evaluate $f$ at $\mathbf{x}$ next?
- The simple acquisition function is UCB (Upper Confident Bound):

$$\alpha_{UCB}(x) = \mu_N(x) + \kappa\sigma_N(x)$$

- The $\kappa$ is the explore rate:
  - $\kappa = 0$ is exploit
  - $\kappa \neq 0$ trade-off between exploitation and exploration

# Methods: Cheap maximization

- The next query is chosen by maximizing $\alpha$.
- This is a separate, *cheap* optimization problem solved on the surrogate — no expensive simulations involved. The maximization itself is typically handled by gradient ascent, L-BFGS, or a multi-start search.

$$\alpha_{UCB}(x) = \mu_N(x) + \kappa\sigma_N(x)$$

$$x_n = \arg\max_x \alpha(x|D_{N-1})$$

# Bayesian Optimization

**Parameter of interest**

$$\vec{X} = (\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n)$$

**Objective function**


**objective**

$$X^* = \{(x^*, y^*)\}$$

# Bayesian Optimization

**Parameter of interest**

$\vec{X} = (\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n)$

**Surrogate model**

$f(\boldsymbol{x}) \sim \mathcal{GP}(\mu, \sigma^2)$

**Objective function**

EPOCH

**objective**

$X^* = \{(x^*, y^*)\}$

# Bayesian Optimization

**Parameter of interest**

$\vec{X} = (\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n)$

**Surrogate model**

$f(\boldsymbol{x}) \sim \mathcal{GP}(\mu, \sigma^2)$

**Acquisition function**

$\alpha_{UCB}(x) = \mu_N(x) + \kappa\sigma_N(x)$

**Objective function**

EPOCH

**objective**

$X^* = \{(x^*, y^*)\}$

# Bayesian Optimization

**Parameter of interest**

$\vec{X} = (\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n)$

**Surrogate model**

$f(\boldsymbol{x}) \sim \mathcal{GP}(\mu, \sigma^2)$

**Acquisition function**

$\alpha_{UCB}(x) = \mu_N(x) + \kappa\sigma_N(x)$

**Prediction**

$x_{n+1} = \arg\max_x \alpha(x|D_{N-1})$

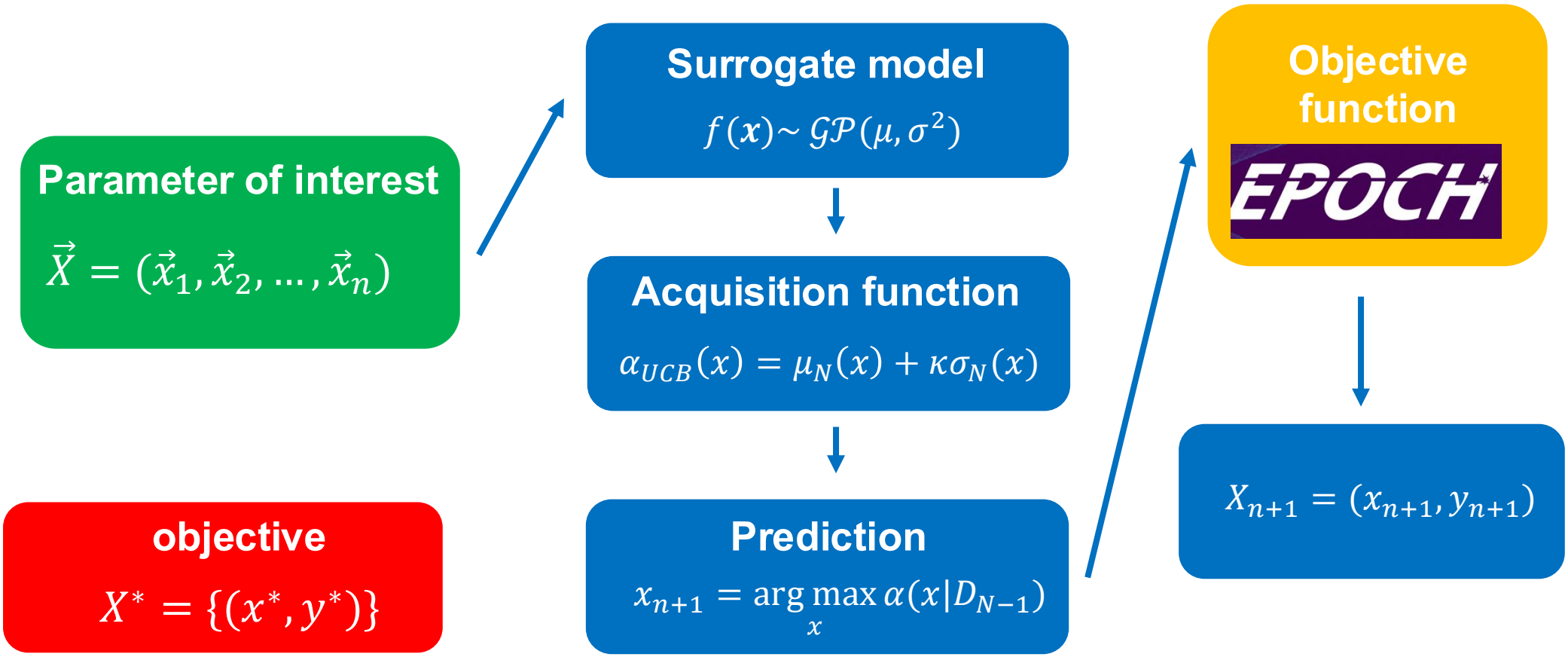
**Objective function**

EPOCH

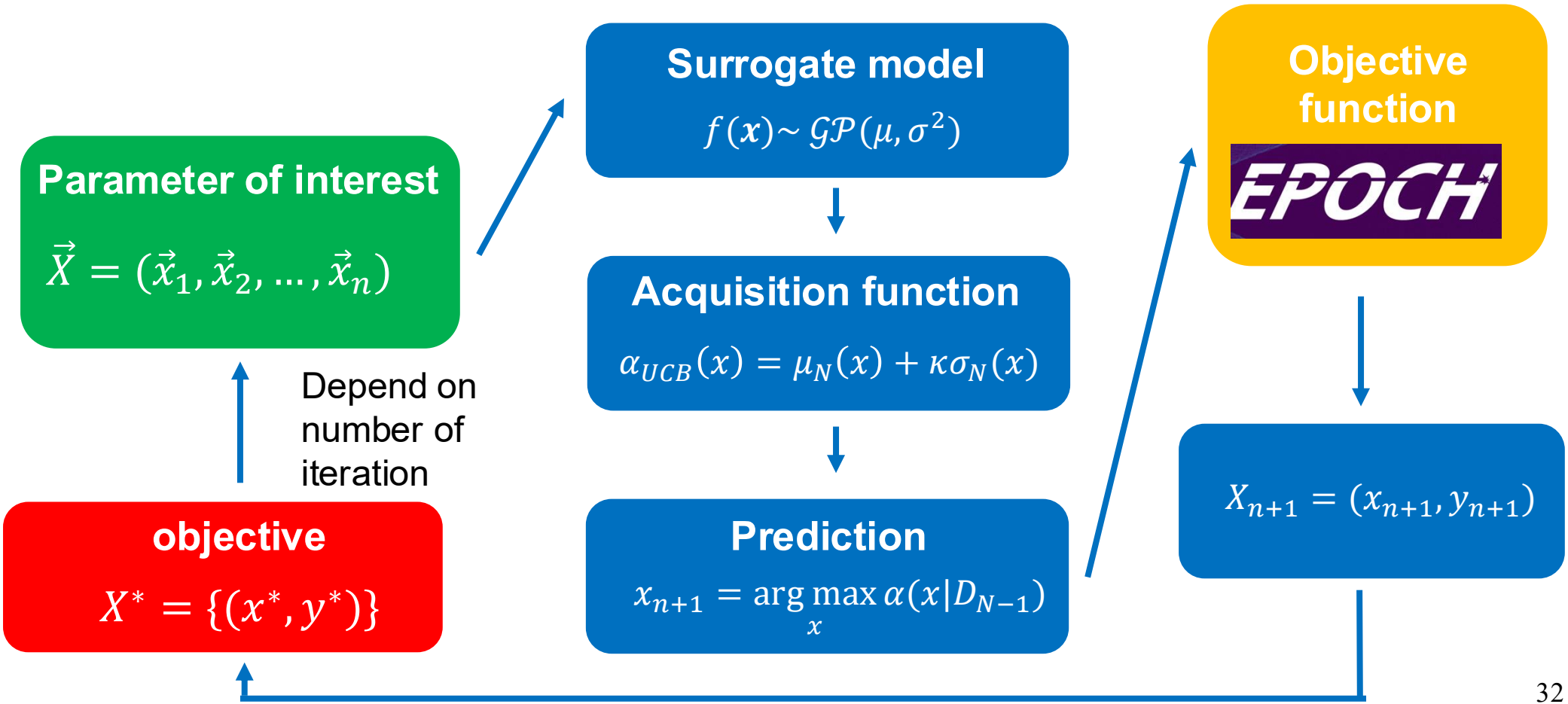
**objective**

$X^* = \{(x^*, y^*)\}$

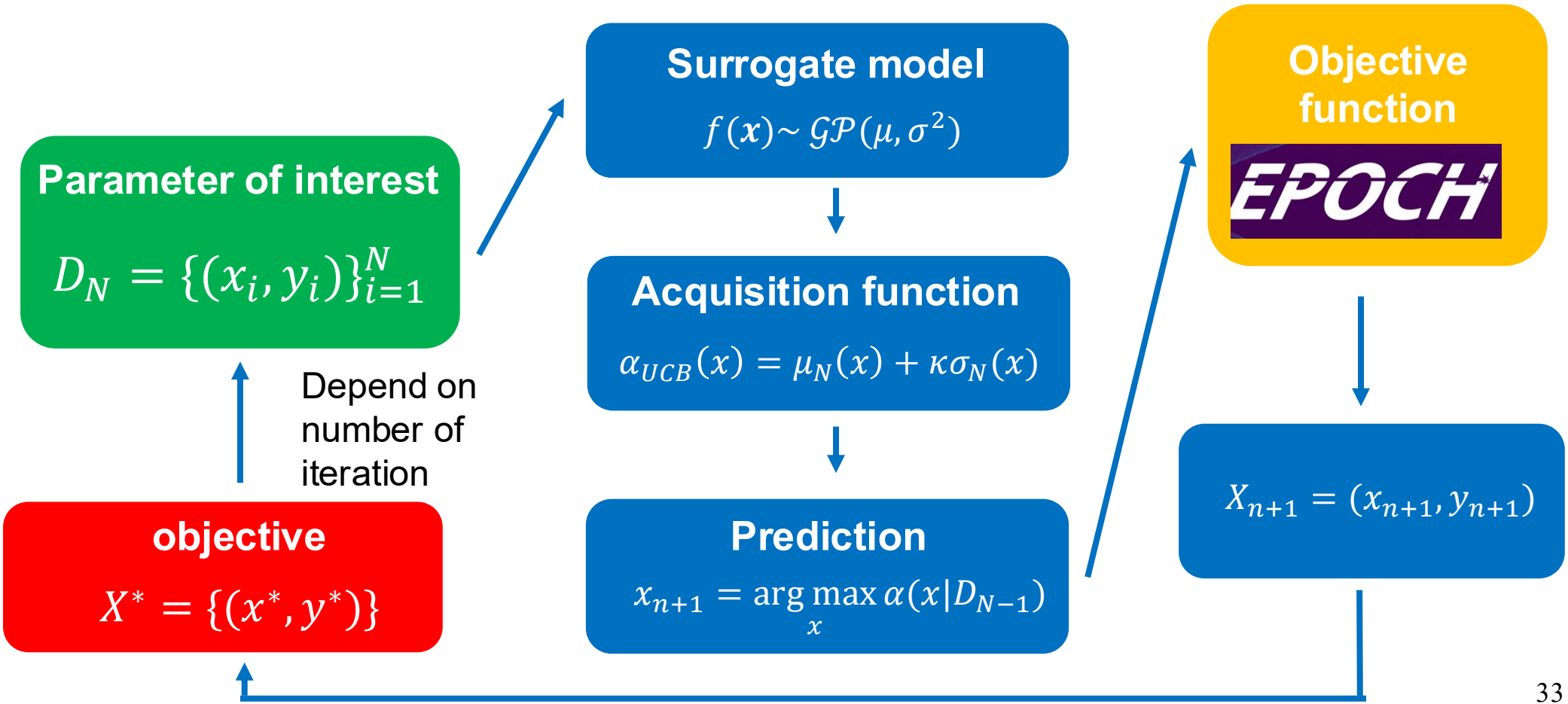
# Bayesian Optimization



# Bayesian Optimization



# Bayesian Optimization



# Bayesian Optimization

## BO components

**Parameter of interest**

$$D_N = \{(x_i, y_i)\}_{i=1}^{N}$$

Depend on number of iteration

**objective**

$$X^* = \{(x^*, y^*)\}$$

**Surrogate model**

$$f(\boldsymbol{x}) \sim \mathcal{GP}(\mu, \sigma^2)$$

**Acquisition function**

$$\alpha_{UCB}(x) = \mu_N(x) + \kappa\sigma_N(x)$$

**Prediction**

$$x_{n+1} = \arg\max_x \alpha(x|D_{N-1})$$

**Objective function**

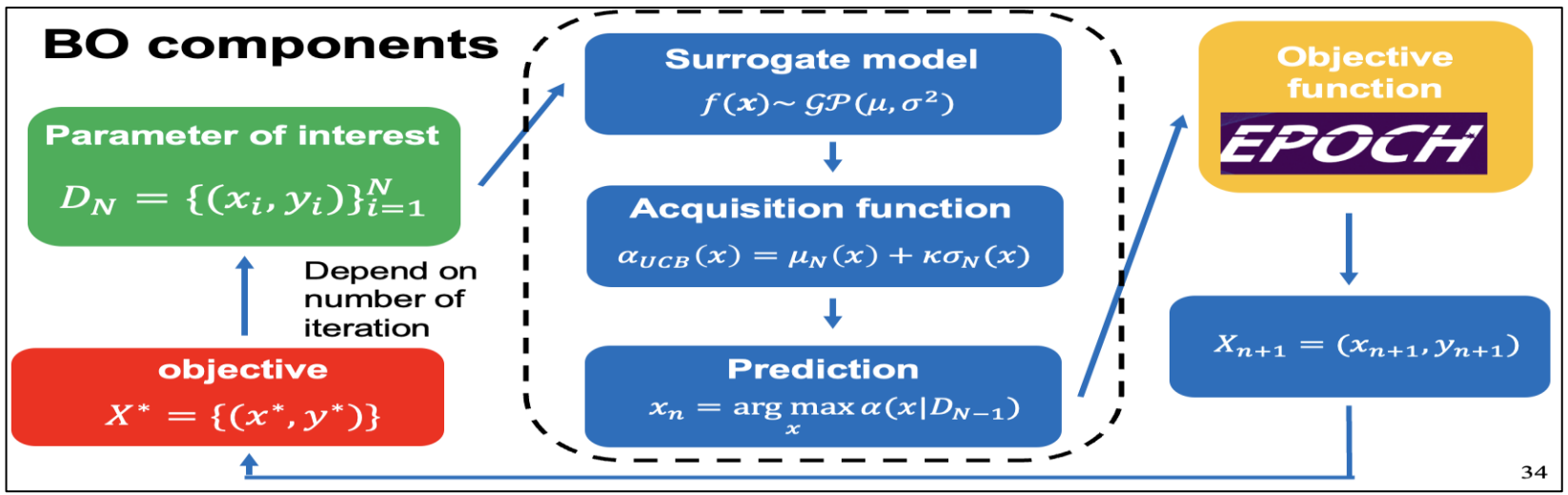
EPOCH

$$X_{n+1} = (x_{n+1}, y_{n+1})$$

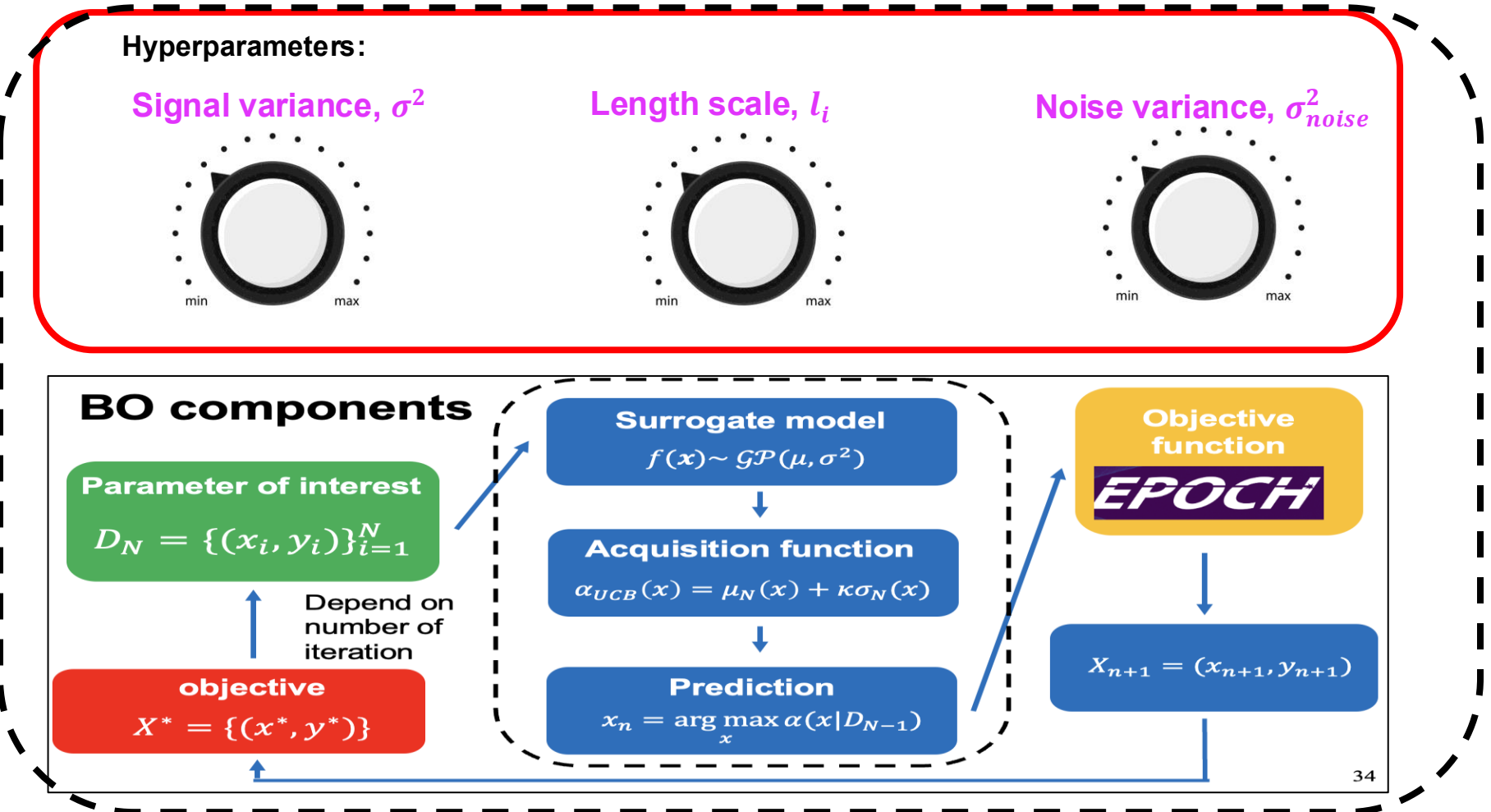
# Bayesian Optimization

**Hyperparameters:** They are not part of $f$ itself; they are the **knobs that define the shape of the prior** (and the noise model)



# Bayesian Optimization




Whole workflow is called Bayesian optimizati on

## What the Hyperparameters Are

For a standard GP with an RBF kernel and Gaussian noise, the hyperparameters are:

$$\theta = \{\sigma^2,\ \sigma^2_{\text{noise}},\ \ell_1, \ell_2, \ldots, \ell_d\}.$$

Each one controls a specific aspect of the model:

- **Signal variance** $\sigma^2$ — the overall amplitude of function variations. How far from the mean does $f$ typically swing?
- **Length scales** $\ell_i$ — how quickly correlations decay along each input dimension. How "wiggly" is $f$ in each direction?
- **Noise variance** $\sigma^2_{\text{noise}}$ — how much of the scatter in observations is measurement/simulation noise rather than real structure in $f$?

These are not learned *along with* $f$ in the same Bayesian update. They are model-level parameters that determine what the prior even looks like *before* you start doing inference about $f$.

# Conclusion

- **Bayesian Optimization with a GP surrogate** efficiently optimizes expensive, noisy, black-box objectives.
- The **GP prior** (RBF kernel + length scales) and **Gaussian likelihood** combine via Bayes' theorem to yield a closed-form posterior.
- The **UCB acquisition function** balances exploration and exploitation; **hyperparameters** are auto-tuned via marginal-likelihood maximization.
- Far fewer evaluations are needed than grid or random search, making BO well-suited to simulation-driven tasks.

# Future Work

- **Alternative acquisitions** — benchmark UCB against EI, PI, and Thompson Sampling.
- **Richer kernels** — explore Matérn kernels to capture less smooth features.
- **Multi-objective BO** — jointly optimize efficiency, beam quality, and operational cost.
- **Batch / parallel BO** — propose multiple candidates per iteration to leverage parallel compute.
- **Trust-region BO (TuRBO)** — improve scaling to higher-dimensional parameter spaces.
- **Physics-informed priors** — encode domain knowledge into the GP mean function.