Bayesian Algorithm

September 28, 2021

1 What is Gaussian Process [1]

Definition 1.1. *(**Gaussian Process**) A sequence of random variables $f_1, f_2, f_3, \cdots, f_n, \cdots$ are called a Gaussan process iff any combination of these random variables satisfies (multivariable) Gaussian distribution.*

$$\forall i, j, k, \cdots \in \mathbb{N}, \quad (f_i, f_j, f_k, \cdots) \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

Example 1.1. *The Gaussian Process we are using is:*

A sequence of random variables $f(x_1), f(x_2), \cdots$ *labeled by* $x_1, x_2, \cdots$ is a **Gaussian process** if they satisfy the **multivariable** Gaussian distribution

$$N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

where $\boldsymbol{\mu}$ is mean and $\boldsymbol{\Sigma}$ is covariance matrix. and satisfies $\boldsymbol{\Sigma}^T = \boldsymbol{\Sigma}$. The matrix elements are $\Sigma_{ij} = \Sigma_{ji}$

We confine our discussion to a *predefined* mean and variance by a **mean function** and **covariance function**(They are functions of *labels* x). We define $\Sigma_{ij} = \Sigma_{ij}(x_i, x_j)$ (to only depend on x_i and x_j)

$$\boldsymbol{\mu} = \boldsymbol{\mu}(x_1, \cdots), \quad \boldsymbol{\Sigma}(x_1, \cdots; x_1, \cdots) = \begin{pmatrix} \Sigma_{11}(x_1) & \Sigma_{12}(x_1, x_2) & \cdots \\ \Sigma_{21}(x_1, x_2) & \Sigma_{22}(x_2) & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix}$$

The covariance matrix is called **Kernel** in Bayesian Optimization and is crucial for the algorithm. It mainly describes the possible correlation between function values at different points, like $\Sigma(x_1, x_2)$ describes the correlation between $f(x_1)$ and $f(x_2)$, as in fig. 1

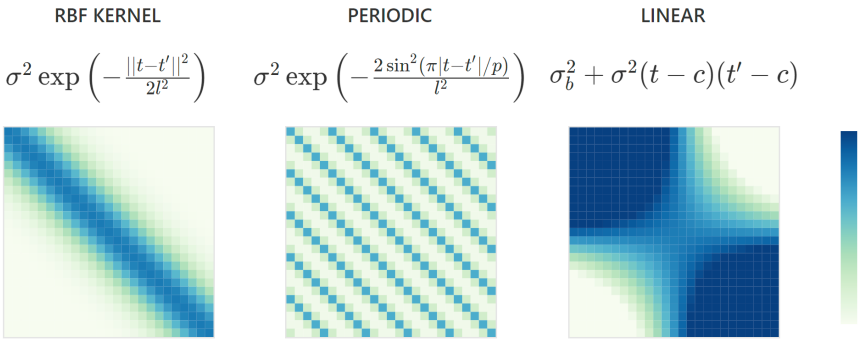

Figure 1: A visualization of a kernel. x_1 and x_2 is represented by t and t', which is also represented by the x and y axis. The color of the graph represents the value of the kernel matrix.

Theorem 1.1. *The conditional and marginal distribution of a Gaussian is also Gaussian. [2] Suppose $\boldsymbol{X} = (X_1, X_2, \cdots, X_n) \sim N(\boldsymbol{\mu}, \Sigma)$, then*

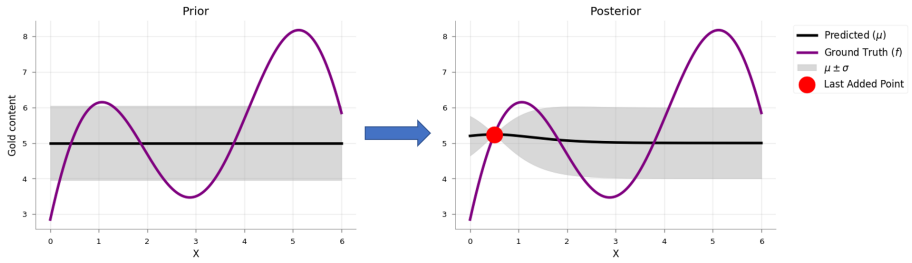

Figure 2: prior: estimate/prediction of function values without any information; posterior: estimate of function values with the information of $f(x_1)$. Shaded parts indicate the σ values at each point.

1. $X_i \sim N(\mu_i, \Sigma_{ii})$
2. conditional probability

$$x_n \,|X_1 = x_1, \cdots, X_{n-1} = x_{n-1} \sim N(\mu, \sigma) \tag{1}$$

where

$$\mu = \mu_n + \Sigma_{n,[1:n-1]}\Sigma^{-1}_{[1:n-1],[1:n-1]}(y) \begin{pmatrix} x_1 - \mu_1 \\ \vdots \\ x_{n-1} - \mu_{n-1} \end{pmatrix}$$

$$\sigma = \Sigma_{nn} - \Sigma_{n,[1:n-1]}\Sigma^{-1}_{[1:n-1],[1:n-1]}\Sigma_{[1:n-1],n}$$

Despite the complicated expression, we only need to take away that they are *calculable*.

Posterior distribution

eq. (1) implies that if we have *observed* the values $f(x_1), f(x_2), \cdots, f(x_{n-1})$ in our Gaussian process, we could *infer* the distribution of $f(x_n)$. This is called *posterior* distribution.

$$f(x_n) \sim N(\mu(x_n), \sigma(x_n)))$$

2 Bayesian Optimization [3]

The whole algorithm utilizes the Gaussian process defined in our Example 1.1. Taking an unknown function f, we construct a Gaussian process as defined earlier.[1] After probing the function value at one point, say, $f(x_1)$, utilizing eq. (1), we can *predict* the the function value of next point $f(x_2)$ and so on. The prediction will become more and more precise after iterations. This is shown in fig. 2

2.1 Probing the function

The algorithm looks as follows:

The process of determining x_{i+1} is called **acquisition function**

Definition 2.1. *The next point is the determined as the extremum of* ***acquisition function*** *F.*

$$x_{i+1} = \underset{x}{\text{argmax}} F(x) \quad or \quad \underset{x}{\text{argmin}} F(x)$$

[1] Which assumes the function values at all points are random variables.

Algorithm 1: probing the function

Result: The prediction of function values
probe a random point x_1;
while *probed points* $i <$ *desired number of points to be probed* **do**
    update prediction of function values according to eq. (1);
    find out the next point x_{i+1} to probe;
    probe the next point $f(x_{i+1})$;`i++`;

Example 2.1. *(active learning)* $F = \sigma(x)$, $x_{i+1} = \underset{x}{\operatorname{argmax}} f(x)$, *meaning that the point with the greatest uncertainty should be probed. As a result, the algorithm 'learns' and predicts the unknown function better and better after each iterations.*

2.2 Bayesian optimization

Now, we don't want to know how the entire function behaves, we want to find the extremums. Therefore, we want to just probe the points that gives a high promise of maximum. Acquisition function can be chosen as follows:

Acquisition Function: probability of improvement (PI)

Example 2.2. *probability of improvement (PI)*

$$F = P(f(x) \geq f(x^+) + \varepsilon) = \Phi\left(\frac{\mu(x) - f(x^+) - \varepsilon}{\sigma(x)}\right), \quad x_{i+1} = \underset{x}{\operatorname{argmax}} F(x)$$

where x^+ *represents the current maximum,* Φ *indicates CDF. This acquisition function chooses the next query point as the one which has the highest probability of improvement over the current max* $f(x^+) + \varepsilon$

Example 2.3. *expected improvement (EI) choose the next query point as the one which has the highest expected improvement over the current max* $f(x^+)$[2]

Example 2.4. *Thompson Sampling. At every step, we sample a function from the surrogate's posterior(according to posterior distribution) and optimize it. See fig. 3*

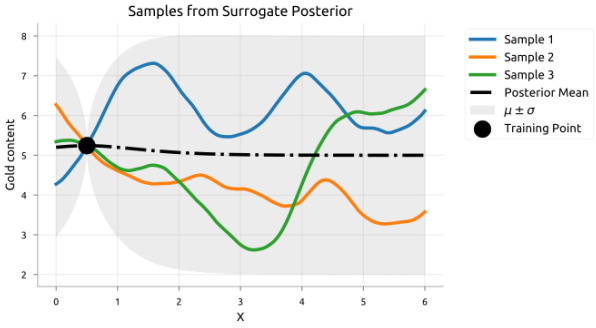

Figure 3: Thompson Sampling

2.3 exploration vs exploitation

We don't know if we are stuck at the local maxima or near the global maxima. Therefore we need to balance exploring uncertain regions, which might unexpectedly be global maxima(**exploration**), against focusing on regions we already know have have a large function

[2]details see https://people.orie.cornell.edu/pfrazier/Presentations/2018.11.INFORMS.tutorial.pdf
https://thuijskens.github.io/2016/12/29/bayesian-optimisation/

value (a kind of **exploitation**). This can be done by *adjusting the parameters* of the **acquisition function**. Example using PI as acquisition function is shown in figs. 4 and 5

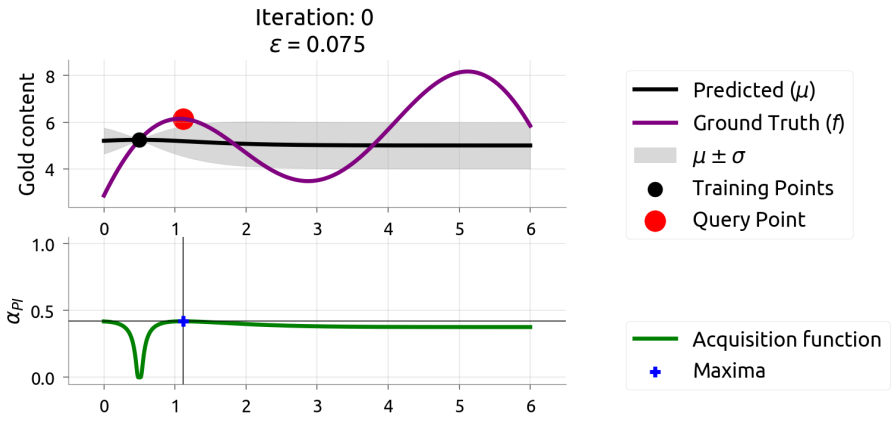

Figure 4: setting $\varepsilon = 0.075$ to be small means we only need a small improvement, resulting the next search point to be near local maxima

2.4 sequential domain reduction

As the title suggests [4].
https://github.com/fmfn/BayesianOptimization/blob/master/examples/domain_reduction.ipynb

3 Convergence

In principle, the Bayesian optimization algorithm just does a "fit" to our unknown function. The function values probed might not be always increasing, or converging to the maxima.

The sequential domain reduction looks like could converge because the domain is getting smaller and smaller thus could converge. Details are discussed in [4]

References

[1] A visual exploration of gaussian processes. https://distill.pub/2019/visual-exploration-gaussian-processes/. Accessed: 2021-09-24.

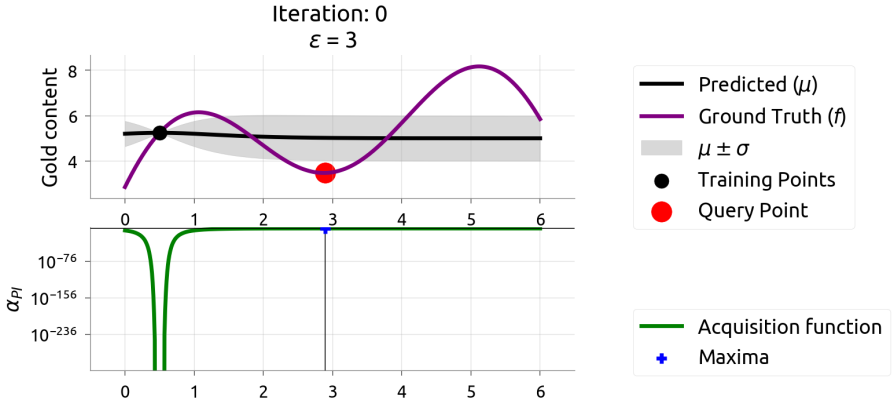

Figure 5: setting $\varepsilon = 3$ to be large means we want to find a large potential improvement. The local maxima could not satisfy that, therefore the algorithm explores other positions of parameter space.

[2] The equation can be found at any probability textbooks like http://www.math.chalmers.se/~rootzen/highdimensional/SSP4SE-appA.pdf.

[3] Exploring bayesian optimization. https://distill.pub/2020/bayesian-optimization/. Accessed: 2021-09-24.

[4] Nielen Stander and KJ Craig. On the robustness of a simple domain reduction scheme for simulation-based optimization. *Engineering Computations*, 2002.