Acquisition Function

Pankaj Chouhan

August 2021

1 Introduction

Acquisition functions are mathematical techniques that guide the exploration of the parameter space during Bayesian optimization. Consider a function $f : X \to \mathbb{R}$ where we aim to find its minimum within the domain $x \in X$.

$$x_* = \underset{x \in X}{\operatorname{argmin}}\, f(x) \tag{1}$$

Typically, if the functional form of f is known, an optimization routine can be applied to find its minimum. However, when f is unknown, we approximate it using a surrogate that is easy to interpret and construct. In Bayesian optimization, f is usually modeled by a Gaussian Process Regression (GPR). GPR assumes a Gaussian prior over f:

$$p(f) = \mathcal{N}(f; \mu, K) \tag{2}$$

Given n observations of f, $\mathcal{D} = (X, Y) = (\{x_1, \cdots, x_n\}, \{y_1, \cdots, y_n\})$, a positive semi-definite $n \times n$ matrix $K_{n\times n}$ can be defined using the kernel function k, where $[K]_{ij} = k(x_i, x_j)$. Conditioning $p(f)$ on $\mathcal{D}$ results in:

$$P(f|\mathcal{D}) = \mathcal{N}(f; \mu_{f|\mathcal{D}}, K_{f|\mathcal{D}}) \tag{3}$$

Now, having a functional form of f and an understanding of how to model it, the task is to use GPR for Bayesian optimization. As a Gaussian Process is defined by a mean function μ and covariance K, acquisition functions cleverly utilize these values to intelligently select the location for the next observation.

Question:- How to fit a GP?
To train a Gaussian Process Regression (GPR), we minimize the negative log marginal likelihood, given by:

$$\mathcal{L}(\theta) = -\log p(Y|X, \theta) = \frac{1}{2} Y^T (K_{n\times n} + \sigma_\epsilon^2 I_{n\times n})^{-1} Y + \frac{1}{2} \log \left| K_{n\times n} + \sigma_\epsilon^2 I_{n\times n} \right| + \frac{n}{2} \log 2\pi.$$

Once the best hyperparameters $\hat{\theta}$ are found, inference on a new point x_* can be made. The advantage of GPR is that everything is Gaussian, allowing closed-form evaluation. The predictive distribution $p(y_*) = \mathcal{N}(\mu_*, \sigma_*^2|\mathcal{D}, x_*)$ is also Gaussian.

2 Probability of improvement

The easiest acquisition function designed for Bayesian optimization was the probability of improvement. Let $f_0 = \min f$ represent the minimal value of f observed thus far. The probability of improvement assesses f at the point most likely to improve this value. It corresponds to the utility function $u(x)$ associated with evaluating f at a given point x:

$$u(x) = \begin{cases} 0 & \text{if } f(x) > f_0 \\ 1 & \text{if } f(x) \leq f_0 \end{cases} \tag{4}$$

Here, a unit reward is received if $f(x)$ turns out to be less than or equal to f_0, and no reward is given otherwise. The probability of improvement acquisition function is then the expected utility as a function of x:

$$\text{PI}(x) = E[u(x)|x, D] = \int_{-\infty}^{f_0} N\left(f; \mu(x), K(x,x)\right) df = \Phi(f_0; \mu(x), K(x,x)) \tag{5}$$

For 1-D case this simplifies to:

$$\text{PI}(x) = E[u(x)|x, D] = \Phi\left(\frac{f_0 - \mu(x)}{\sigma(x)}\right) \tag{6}$$

Here Φ represent the CDF of standard normal. The point with the highest probability of improvement (maximal expected utility) is selected, which represents the Bayes action under this loss function.

3 Expected improvement

The loss function associated with probability of improvement is somewhat odd: we get a reward for improving upon the current minimum independent of the size of the improvement! This can sometimes lead to odd behavior, and in practice can get stuck in local optima and under-explore globally.
An alternative acquisition function that does account for the size of the improvement is expected improvement. Again, suppose that f_0 is the minimal value of f observed so far. Expected improvement evaluates f at the point that, in expectation, improves upon f_0 the most. This corresponds to the following utility function:

$$u(x) = \max(0, f_0 - f(x)) \tag{7}$$

Here, we receive a reward equal to the "improvement" $f_0 - f(x)$ if $f(x)$ turns out to be less than f_0, and no reward is given otherwise. The expected improvement acquisition function is then the expected utility as a function of x:

$$\text{EI}(x) = E[u(x)|x, D] = \int_{-\infty}^{f_0} (f_0 - f) N(f; \mu(x), K(x,x)) df \tag{8}$$

$$= (f_0 - \mu(x))\Phi(f_0; \mu(x), K(x,x)) + K(x,x) N(f_0; \mu(x), K(x,x)). \tag{9}$$

For a 1-D case the equation simplifies to

$$\text{EI}(x) = E[u(x)|x, D] = (f_0 - \mu(x))\Phi\left(\frac{f_0 - \mu(x)}{\sigma(x)}\right) + \sigma(x)\phi\left(\frac{f_0 - \mu(x)}{\sigma(x)}\right) \tag{10}$$

Here Φ and ϕ represent the CDF, and PDF of standard normal. The point with the highest expected improvement (the maximal expected utility) is selected. The expected improvement has two components. The first can be increased by reducing the mean function $\mu(x)$. The second can be increased by increasing the variance $K(x, x)$. These two terms can be interpreted as explicitly encoding a tradeoff between exploitation (evaluating at points with low mean) and exploration (evaluating at points with high uncertainty).
Here's a short snippet to calculate the EI acquisition function.

```
# Define the acquisition function (Expected Improvement)
def expected_improvement(X, gaussian_process, evaluated_loss, xi=0.01):
    mean, std = gaussian_process.predict(X, return_std=True)
    Z = (evaluated_loss - mean - xi) / std
    return (evaluated_loss - mean - xi) * norm.cdf(Z) + std * norm.pdf(Z)
```

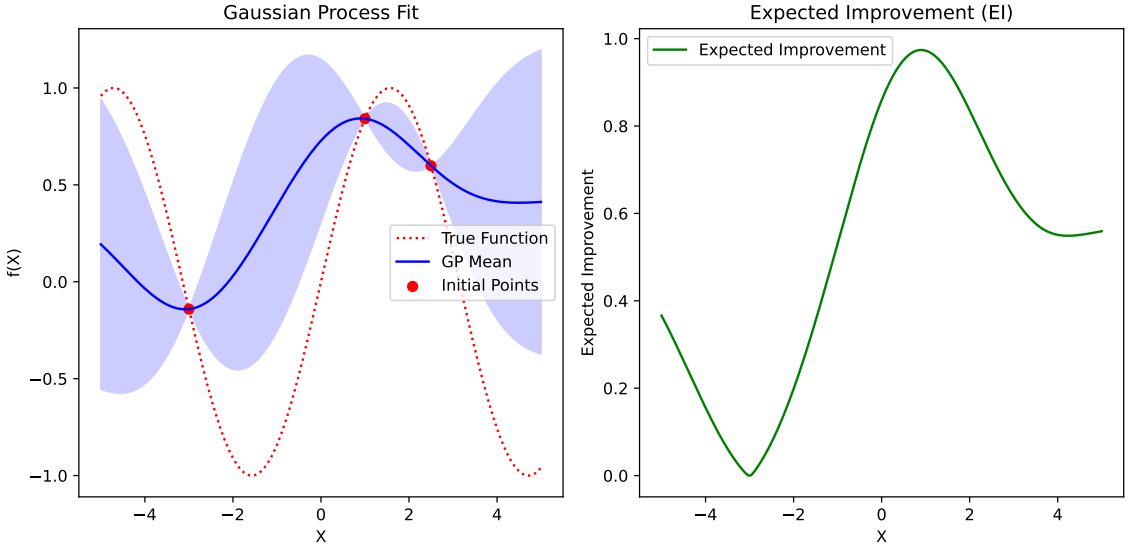


Figure 1: The figure illustrates that EI is highest around the function's minimum, concentrating the search in the area of optimal performance. To balance this bias, introducing a trade-off value ξ is crucial. In Expected Improvement, this value determines the sacrificed performance (in original units) when assessing improvement, serving as a compromise between exploration and exploitation.

4 Entropy search

Entropy can be interpreted as a measure of uncertainty or lack of predictability associated with a random variable. For example, consider a sequence of symbols $c_n \sim p$ generated from a distribution p. If p has high entropy, predicting the value of each observation c_n becomes challenging. Uniform distributions have maximum entropy, whereas Dirac delta functions have minimum entropy. Here, we seek to minimize the

uncertainty we have in the location of the optimal value $x_* = \arg\min_{x \in X} f(x)$. Since we have defined a distribution over f, we can always induces a distribution over x_*, $p(x_*|D)$. Unfortunately, there is no closed-form expression for this distribution.
Entropy search seeks to evaluate points so as to minimize the entropy of the induced distribution $p(x_*|D)$. Here the utility function is the reduction in this entropy given a new measurement at x, $f(x)$:

$$u(x) = H[p(x_*|D)] - H[p(x_*|D \cup \{x, f(x)\})] \tag{11}$$

As in probability of improvement and expected improvement, we may build an acquisition function by evaluating the expected utility provided by evaluating f at a point x. Due to the nature of the distribution $p(x_*|D)$, this is somewhat complicated, and a series of approximations must be made for both LHS and RHS term. Please look at `https://botorch.org/tutorials/max_value_entropy` and `https://gregorygundersen.com/blog/2020/10/28/predictive-entropy-search/` to see how max value is calculated.

5 Knowledge Gradient

Please look at the original article [1]. The article is very well written, I am borrowing/copying this material. This is not the complete knowledge gradient method, but serve a purpose of understanding what's happening under the hood. It only showed a (naïve) approach to calculating the KG at a given location. Suffice it to say, there is still quite a gap between this and being able to efficiently minimize KG within a sequential decision-making algorithm. For a guide on incorporating KG in a modular and fully-fledged framework for BO please look at `https://botorch.org/tutorials/one_shot_kg`.

Essentially, the steps to implement KG are:

- Find the maximum of the mean of the posterior predictive distribution from the Gaussian Process (GP) model. This is the mean $\mu_n(x) = E[y|x, D] = \mu(x; D_n)$, and its maximum is denoted as $\tau_n(D_n) = \max(\mu_n(x))$.
- Augment the dataset by adding a new point: define the new dataset $D_{n+1} = \{D_n \cup \{x_{n+1}, y_{n+1}\}\}$.
- Find the maximum of the posterior predictive mean of the GP model using the dataset D_n, $\tau_{n+1}(D_{n+1}) = \max(\mu_{n+1}(x))$.
- Define the acquisition function as $\alpha(x; D_n) = \mathbb{E}_{p(y|x,D_n)}[\tau_n - \tau_{n+1}]$.
- As the acquisition function can't be solved analytically, use Monte Carlo simulation. Approximate $\alpha(x; D_n)$ as:

$$\alpha(x; D_n) \approx \frac{1}{M}\left[\sum_{i=1}^{M} \tau_n - \tau_{n+1}^{(m)}\right] \tag{12}$$

  Here, $\tau_{n+1}^{(m)} = \tau(D_{n+1}^{(m)})$ and $D_{n+1}^{(m)} = \{D_n \cup \{x_{n+1}, y_{n+1}^{(m)}\}\}$. Multiple y samples are obtained for a single x, and the x is fixed initially.

- This approximation to the knowledge gradient is essentially the average difference between the predictive minimum values based on simulation-augmented data $\tau_{n+1}^{(m)}$ and that based on observed data τ_n across M simulations.

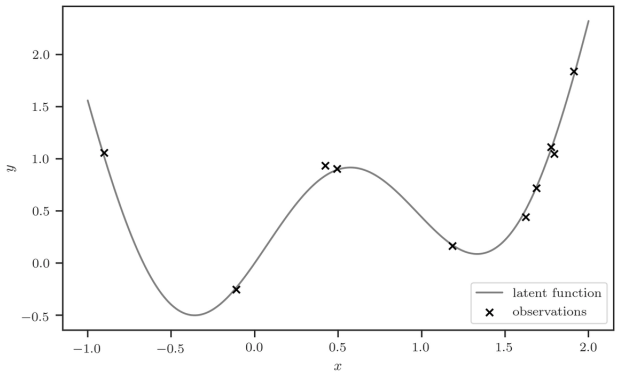
5.1 One dimensional example

Synthetic function defined by

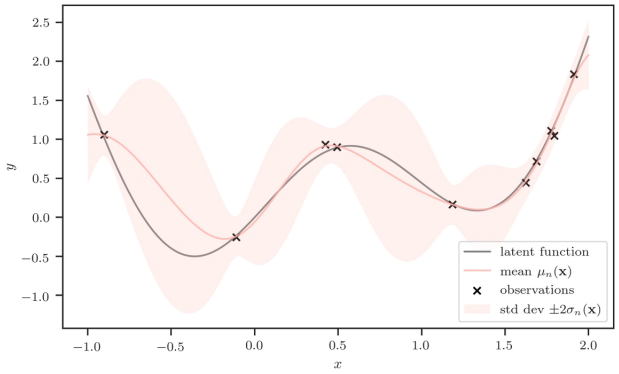
$$f(x) = sin(3x) + x^2 - 0.7x \tag{13}$$

We generate $n = 10$ observations at locations sampled uniformly at random.

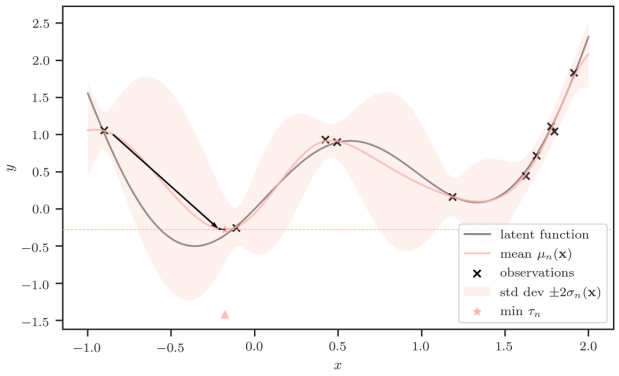
- True Function



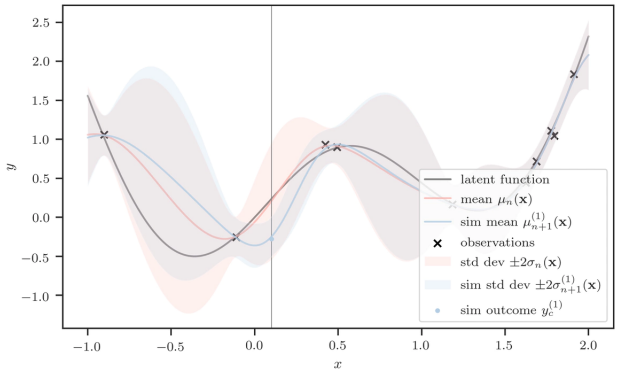
- Fitted GP



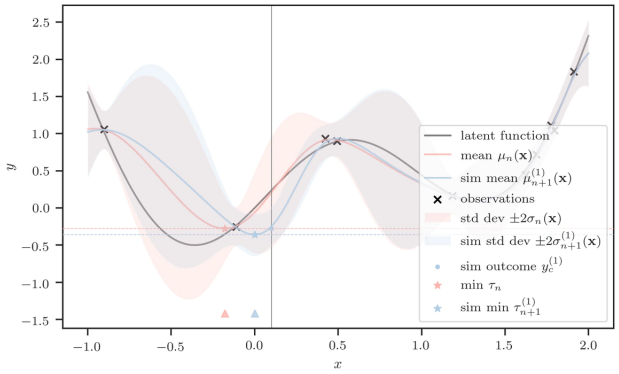
- Get minimum of posterior predictive mean.



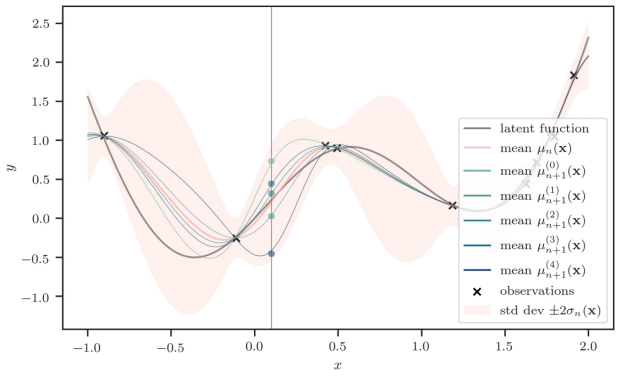
- Posterior predictive with augmented observation.



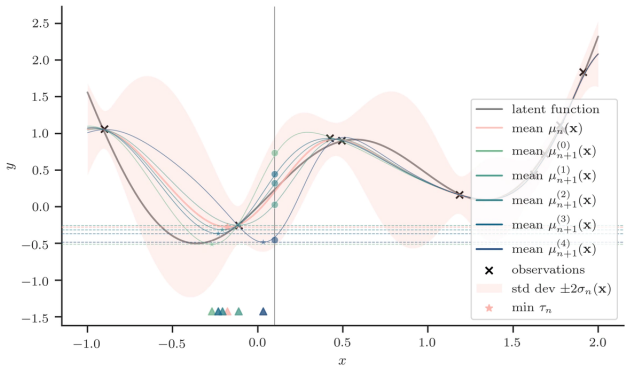
- Posterior predictive minimum with augmented observation. Note we are not sampling position of x, and assuming to be constant.



- Various Posterior predictive with different augmented observation. x is constant, and corresponding y is changing.



- Various Posterior predictive minimum with different augmented observation.



- Finally, taking the average difference between the orange dashed line and every other dashed line gives us the estimate of the knowledge gradient at input x_c.

6 Continuous fidelity knowledge gradient

Objective is to find the x_* such

$$x_* = \underset{x \in X}{\operatorname{argmin}} f(x) \tag{14}$$

They[2] assume a function, $g(x, s)$, where $f(x) = g(x, 1_m)$, and $s \in [0, 1]^m$ denotes the m fidelity.

$$x_* = \underset{x \in X}{\operatorname{argmin}} f(x) \tag{15}$$

Assume n observation. Let $\mu^{(n)}(x, 1_m)$ represent the minimum of posterior predictive mean at highest fidelity. With one additional sample $x^{(n+1)}$ at fidelity $s^{(n+1)}$, the $\underset{x \in X}{\operatorname{argmin}} \mu^{(n+1)}(x, 1_m)$ represent the minimum of posterior predictive mean of model trained with augmented observation. The minimum of augmented posterior predictive mean depend on $x^{(n+1)}$, and $s^{(n+1)}$.

The value of the information gained by sampling at $(x_{(n+1)}$ with the fidelity $s_{(n+1)}$ conditioned on any particular outcome $y_{(n+1)}$ is thus the difference of these two expected validation errors

$$\min_{x \in X} \mu^{(n)}(x, 1_m) - \min_{x \in X} \mu^{(n+1)}(x, 1_m). \tag{16}$$

They take the expectation of this difference, over the random outcome $(y_{(n+1)}$ (For a single x_{n+1}, expectation over y_{n+1}), to obtain the value of the information gained, and take the ratio of this value with the cost of obtaining it to obtain the cfKG acquisition function,

$$\text{cfKG}(x, s) = \frac{\min_{x' \in X} \mu^{(n)}(x', 1_m) - \mathbb{E}_n \left[\min_{x' \in X} \mu^{(n+1)}(x', 1_m) \,|\, x^{(n+1)} = x, s^{(n+1)} = s \right]}{\text{cost}^{(n)}(x, s)} \tag{17}$$

where $\text{cost}^{(n)}(x, s)$ is the estimated cost of evaluating at x with the fidelity s based on the observations available at iteration n. $\mathbb{E}_n$ indicates the expectation taken with respect to the posterior given $x^{(1:n)}, s^{(1:n)}, y^{(1:n)}$. cFKG is used to jointly sample to maximize

$$\underset{x \in X,\, s \in [0,1]}{\text{argmax}} \quad cFKG(x, s) \tag{18}$$

This expectation is intractable, and they use envelope theorem to make it work. Please take a look at the following article.
Instead of sampling one x, when you sample q candidates ($x's$), they jointly optimize for all q candidates and calls it q-cFKG.

References

[1] Louis C Tiao. An Illustrated Guide to the Knowledge Gradient Acquisition Function. *tiao.io*, 2021.

[2] Jian Wu and Peter I Frazier. Continuous-fidelity bayesian optimization with knowledge gradient. 2018.