

Chapter 7: Bayesian Optimization

Advanced Topics in Statistical Machine Learning

Tom Rainforth

Hilary 2022

rainforth@stats.ox.ac.uk

Problem Setting

Consider the problem of **globally** maximizing (or minimizing) some function $f(x)$ within some domain $\mathcal{X}$,

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

in a setting where evaluations of $f(x)$ are very **expensive** and potentially also **noisy**, e.g.

- Optimizing hyperparameters of a machine learning algorithm where evaluating $f(x)$ requires the algorithm to be trained
- Real world optimization, e.g. finding the perfect cookie recipe where $f(x)$ requires us to make, taste, and rate them

Here we need to perform our optimization in a way that is extremely **evaluation efficient** and typically also **derivative free** (as we generally do not have access $\nabla_x f(x)$ in such settings)

Optimization Through Bayesian Eyes

We can reason about such an optimization using **Bayesian probability**: through lack of information we are unsure about the location of the true optimum x^* and we can represent our beliefs about where it might be probabilistically

We can further construct a Bayesian **model** for this: we can specify a prior $p(x^*)$ and then condition this on function evaluations

Alternatively, we can reason about the **function itself** (e.g. using a GP), which will in turn imply a distribution over the location of the optimum

Optimization can now be framed as reducing our uncertainty in the **location** of the optimum

Optimization as Experimental Design

Given a computational budget, and thus a budget on function evaluations, there are two things we need to control:

1. Where we **evaluate** the function
2. Where we **predict** the optimum to be given our evaluations

The second of these only needs to be done **after** we have expended our function evaluation budget and is generally straightforward (e.g. if we have fit a GP we can take the evaluation with the highest posterior mean).

The former is the real challenge of optimization and constitutes a **sequential experimental design problem**

- We want to choose points in a way that maximizes the **information gained** about the location of the optimum

Exploration–Exploitation Trade–Off

By modeling the target function with a probabilistic model, often known as a **surrogate model**, we can reason about its values at points we have not yet evaluated.

In general, we then want to evaluate points x where

a. We believe that $f(x)$ is large relative to our current best evaluation (**exploitation**) and/or;
b. We have high uncertainty in the value of $f(x)$ (**exploration**).

Exploitation encodes our desire to learn about the function in promising regions, while exploration encoders our desire to reduce uncertainty in regions we currently know little about.

Managing the relative merit of these is known as the **exploration–exploitation** trade–off.

Acquisition Functions

One way to manage this trade–off is to introduce an **acquisition function** $\zeta : \mathcal{X} \mapsto \mathbb{R}$ that expresses the perceived **utility** of evaluating each input next, encapsulating both the needs for exploration and exploitation.

By optimizing this acquisition function we can choose the **best point to evaluate** at each iteration of our design process.

Provided that it is **cheap** to evaluate $\zeta(x)$, this optimization can often be done more cheaply than a single evaluation of $f(x)$, justifying the expense

Note that we often do not need to optimize $\zeta(x)$ perfectly: it is not directly affecting our prediction for the optimum of $f(x)$, only where we choose to gather information.

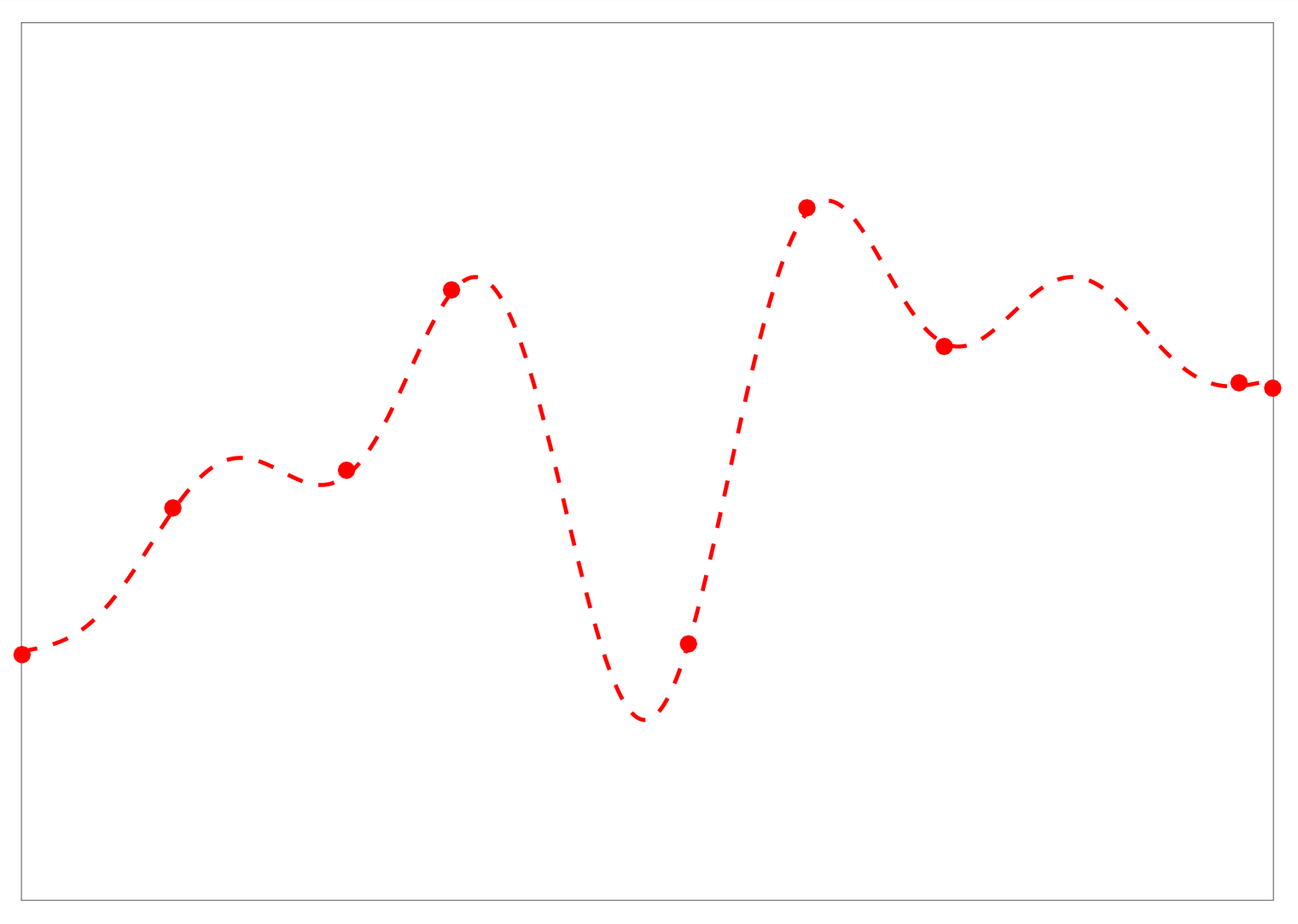


Figure 1: BO example. Red points are evaluations, dotted red line is true underlying function to maximize, solid blue line is GP posterior mean, shading is ± 2 standard deviations, and green line is acquisition function.

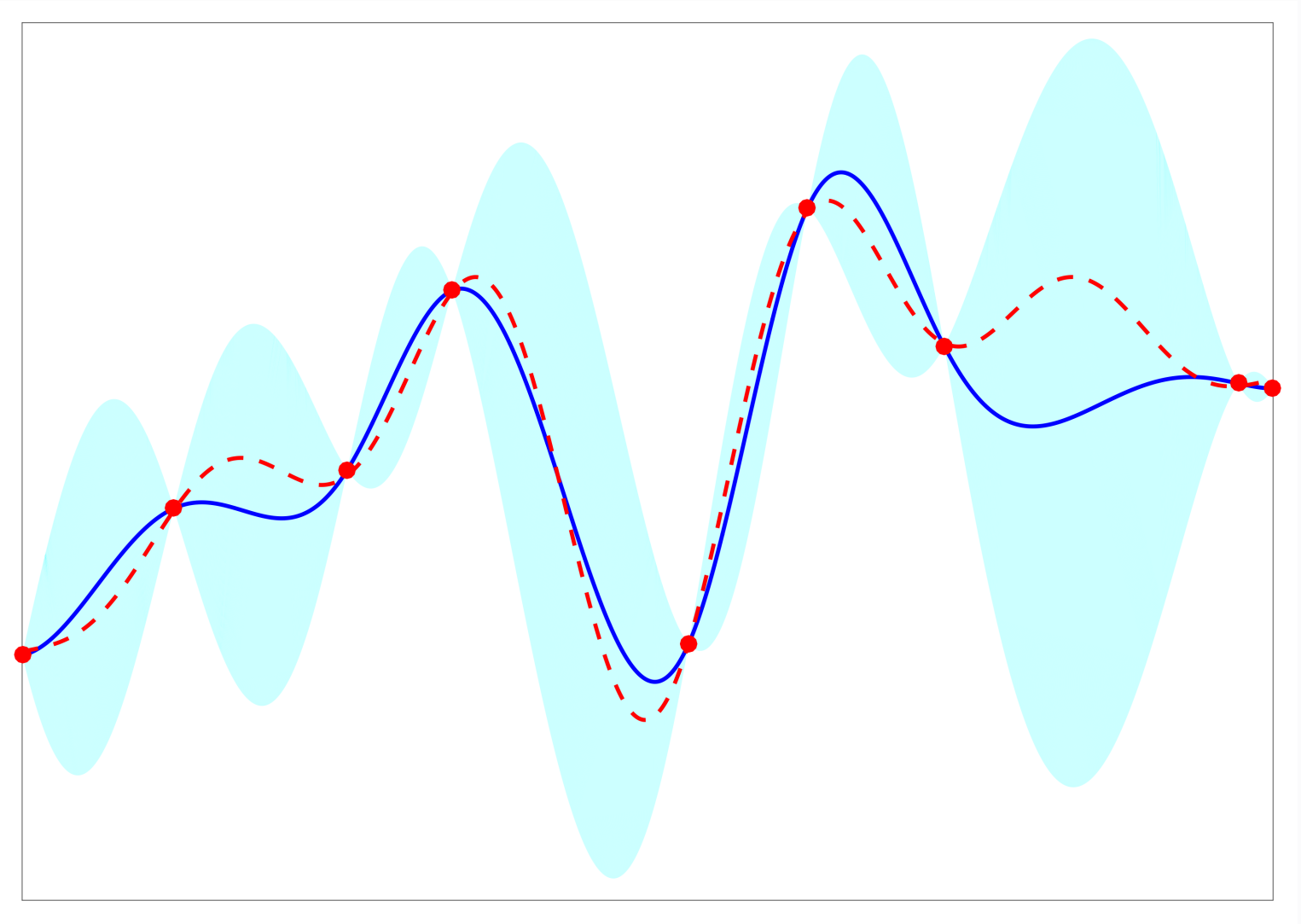


Figure 1: BO example. Red points are evaluations, dotted red line is true underlying function to maximize, solid blue line is GP posterior mean, shading is ± 2 standard deviations, and green line is acquisition function.

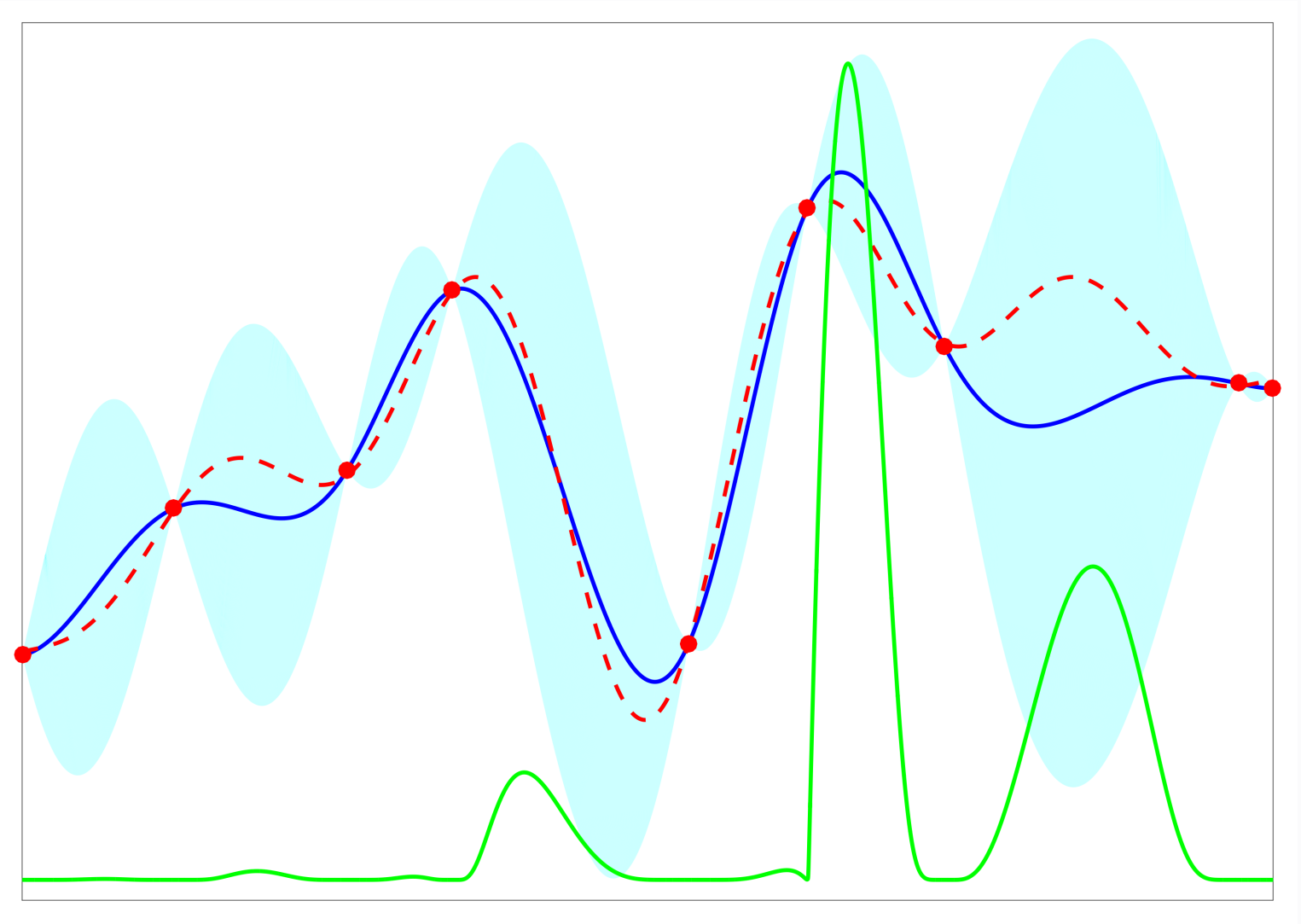


Figure 1: BO example. Red points are evaluations, dotted red line is true underlying function to maximize, solid blue line is GP posterior mean, shading is ± 2 standard deviations, and green line is acquisition function.

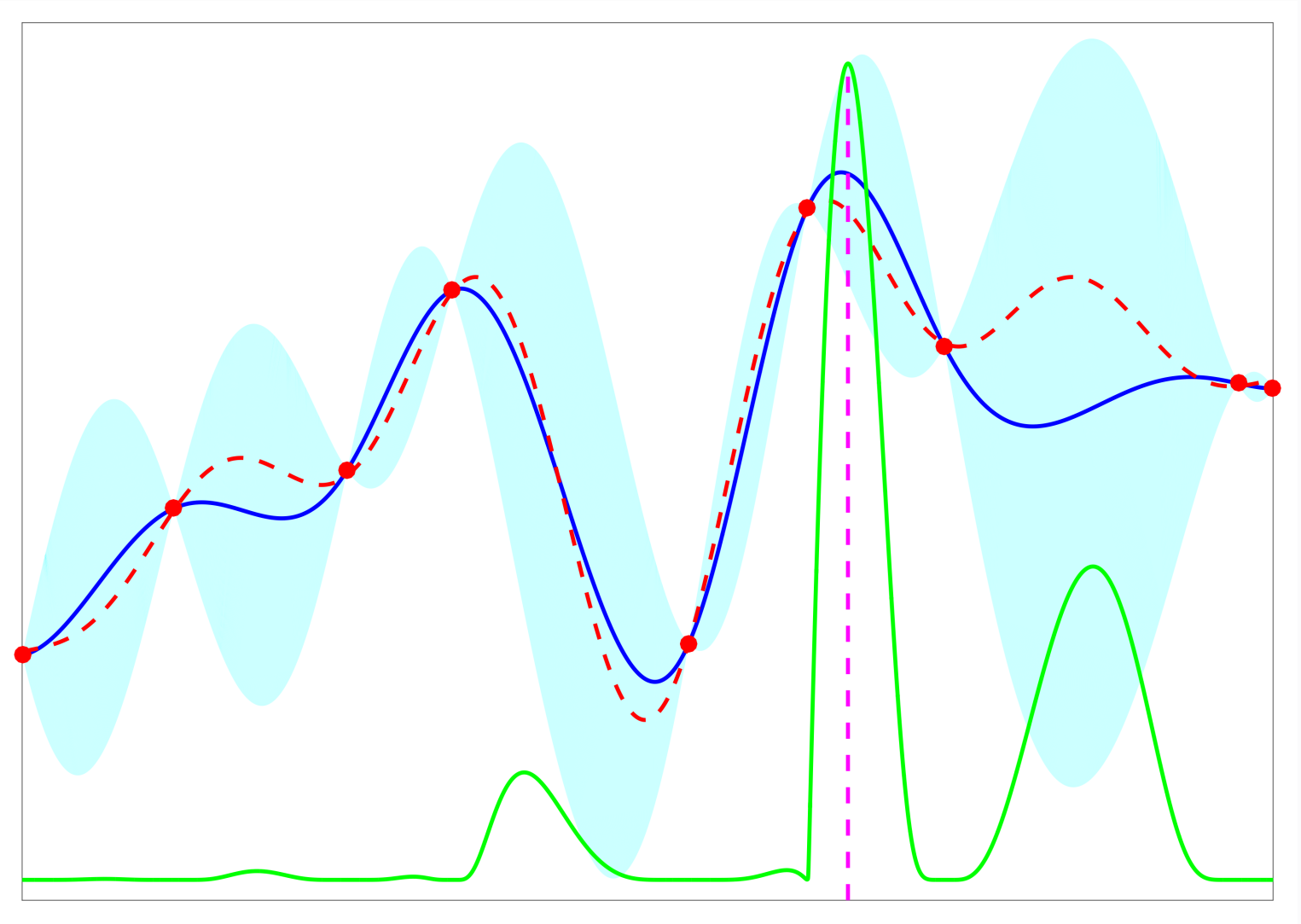


Figure 1: BO example. Red points are evaluations, dotted red line is true underlying function to maximize, solid blue line is GP posterior mean, shading is ± 2 standard deviations, and green line is acquisition function.

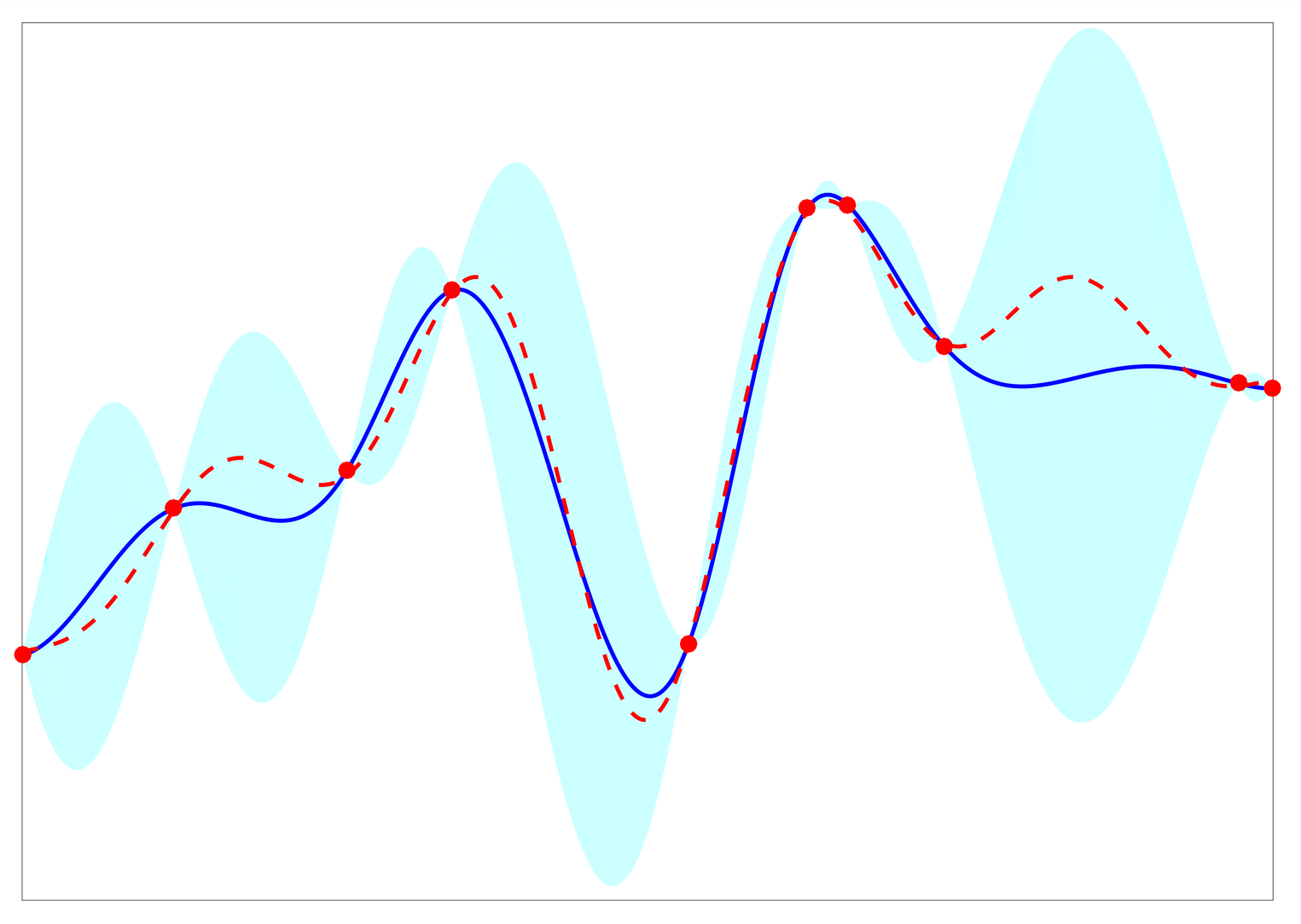


Figure 1: BO example. Red points are evaluations, dotted red line is true underlying function to maximize, solid blue line is GP posterior mean, shading is ± 2 standard deviations, and green line is acquisition function.

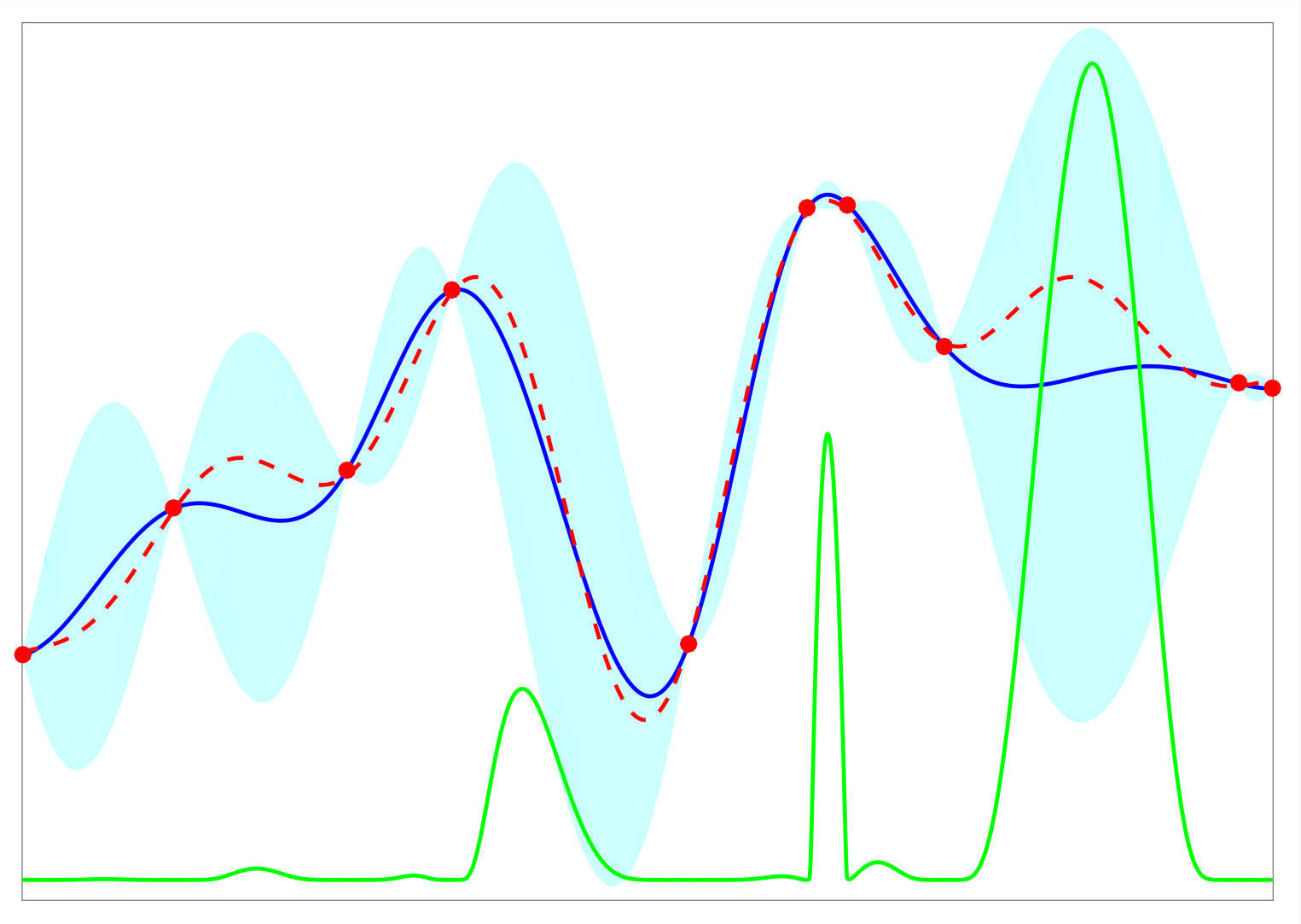


Figure 1: BO example. Red points are evaluations, dotted red line is true underlying function to maximize, solid blue line is GP posterior mean, shading is ± 2 standard deviations, and green line is acquisition function.

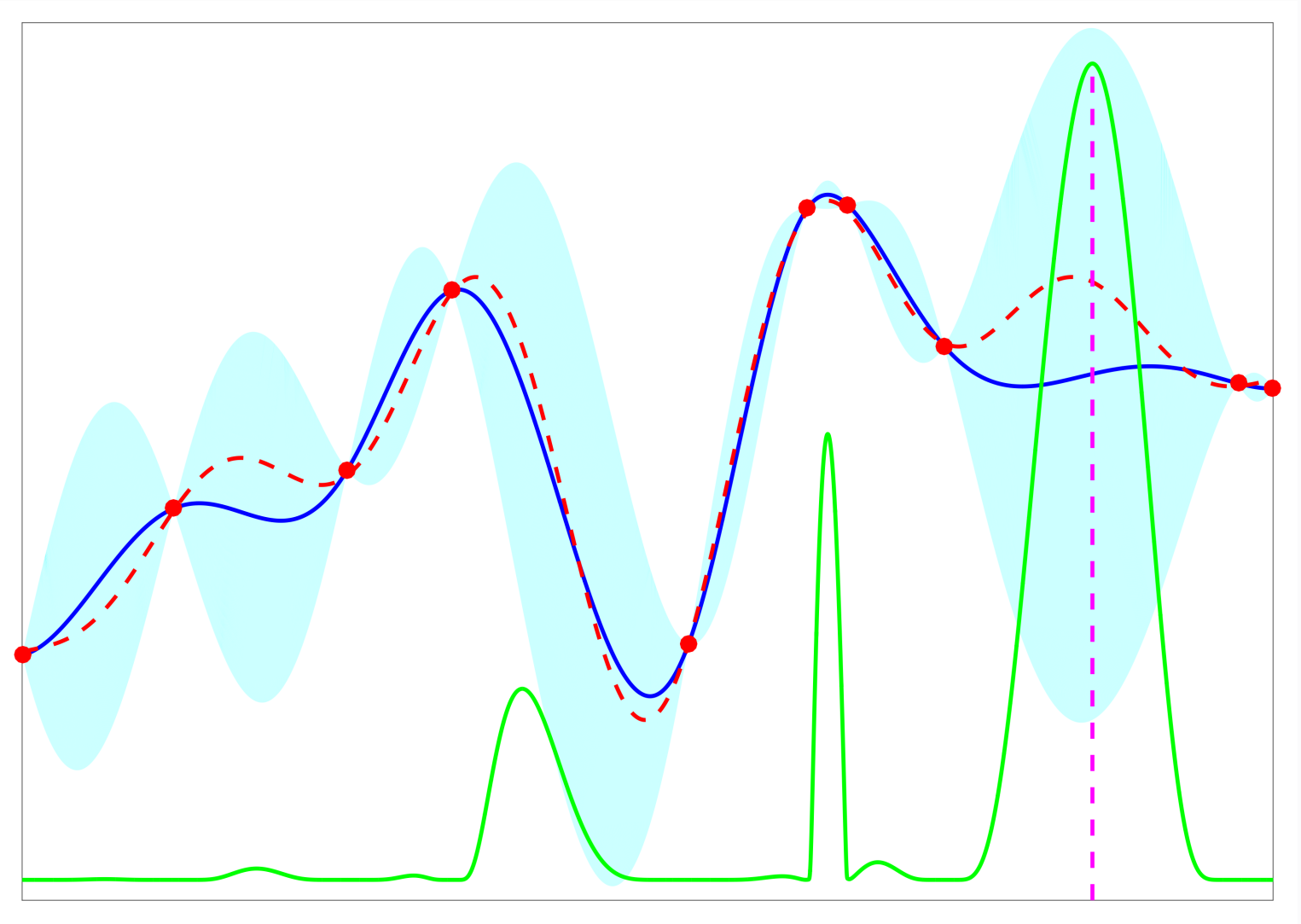


Figure 1: BO example. Red points are evaluations, dotted red line is true underlying function to maximize, solid blue line is GP posterior mean, shading is ± 2 standard deviations, and green line is acquisition function.

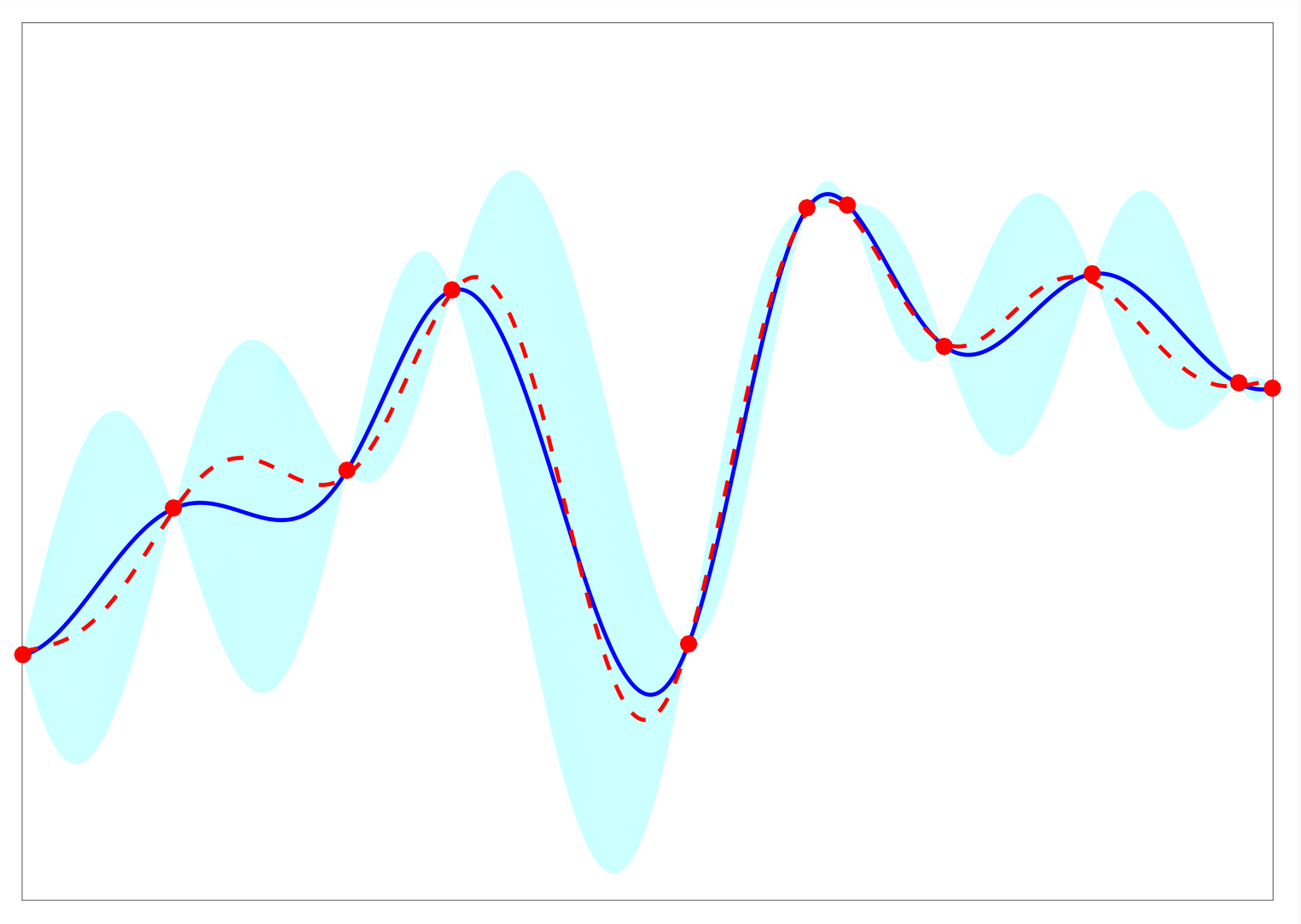


Figure 1: BO example. Red points are evaluations, dotted red line is true underlying function to maximize, solid blue line is GP posterior mean, shading is ± 2 standard deviations, and green line is acquisition function.

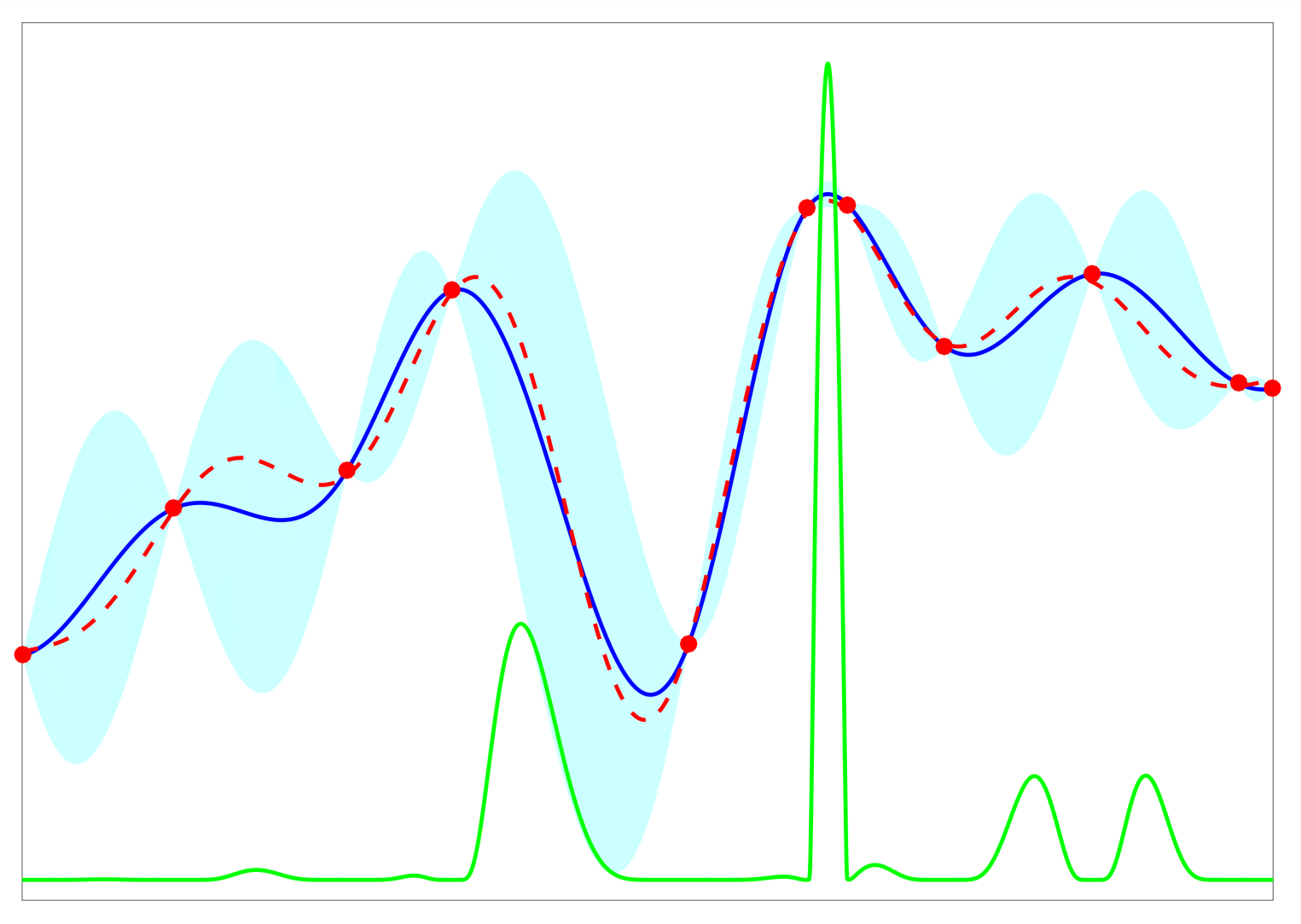


Figure 1: BO example. Red points are evaluations, dotted red line is true underlying function to maximize, solid blue line is GP posterior mean, shading is ± 2 standard deviations, and green line is acquisition function.

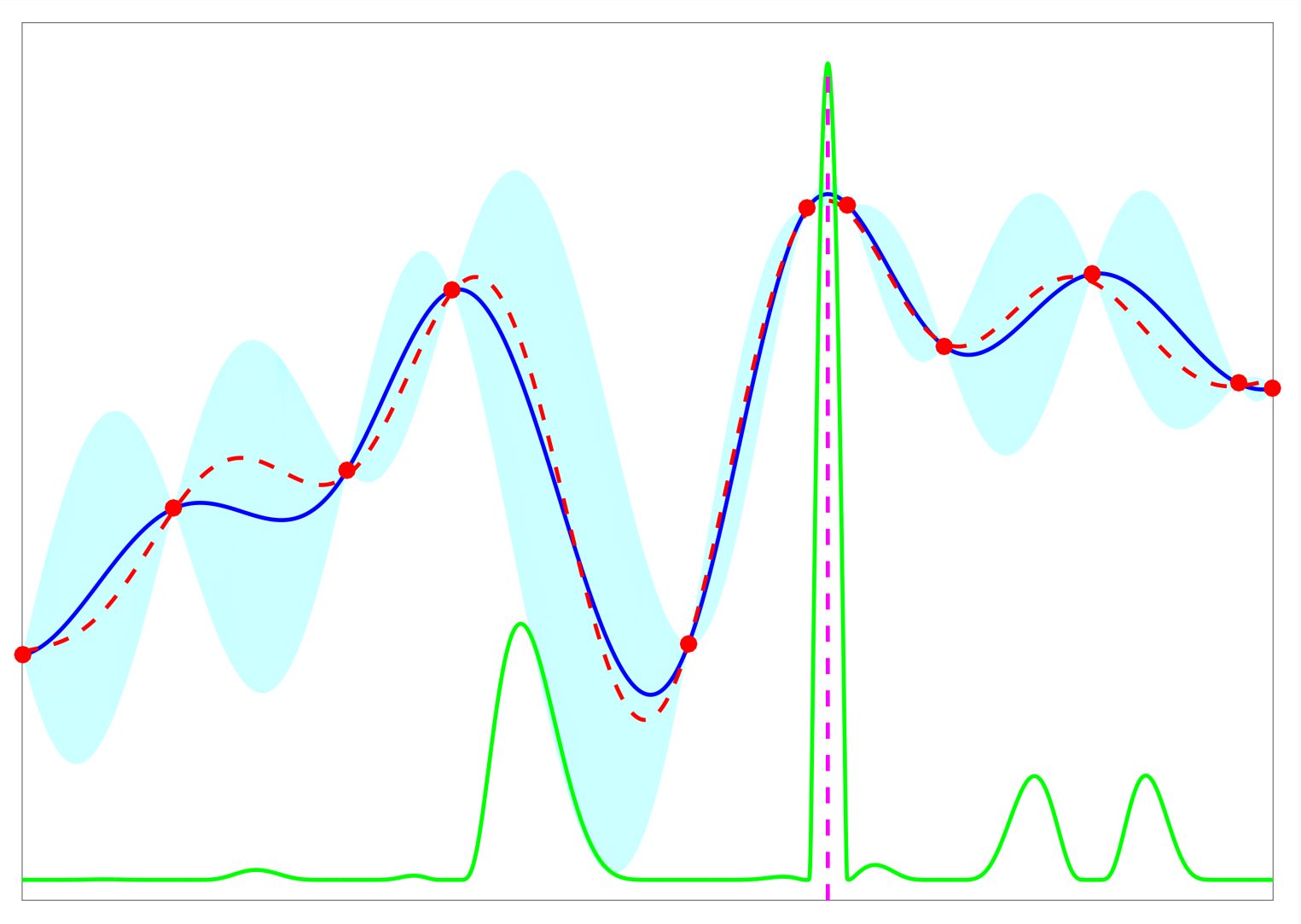


Figure 1: BO example. Red points are evaluations, dotted red line is true underlying function to maximize, solid blue line is GP posterior mean, shading is ± 2 standard deviations, and green line is acquisition function.

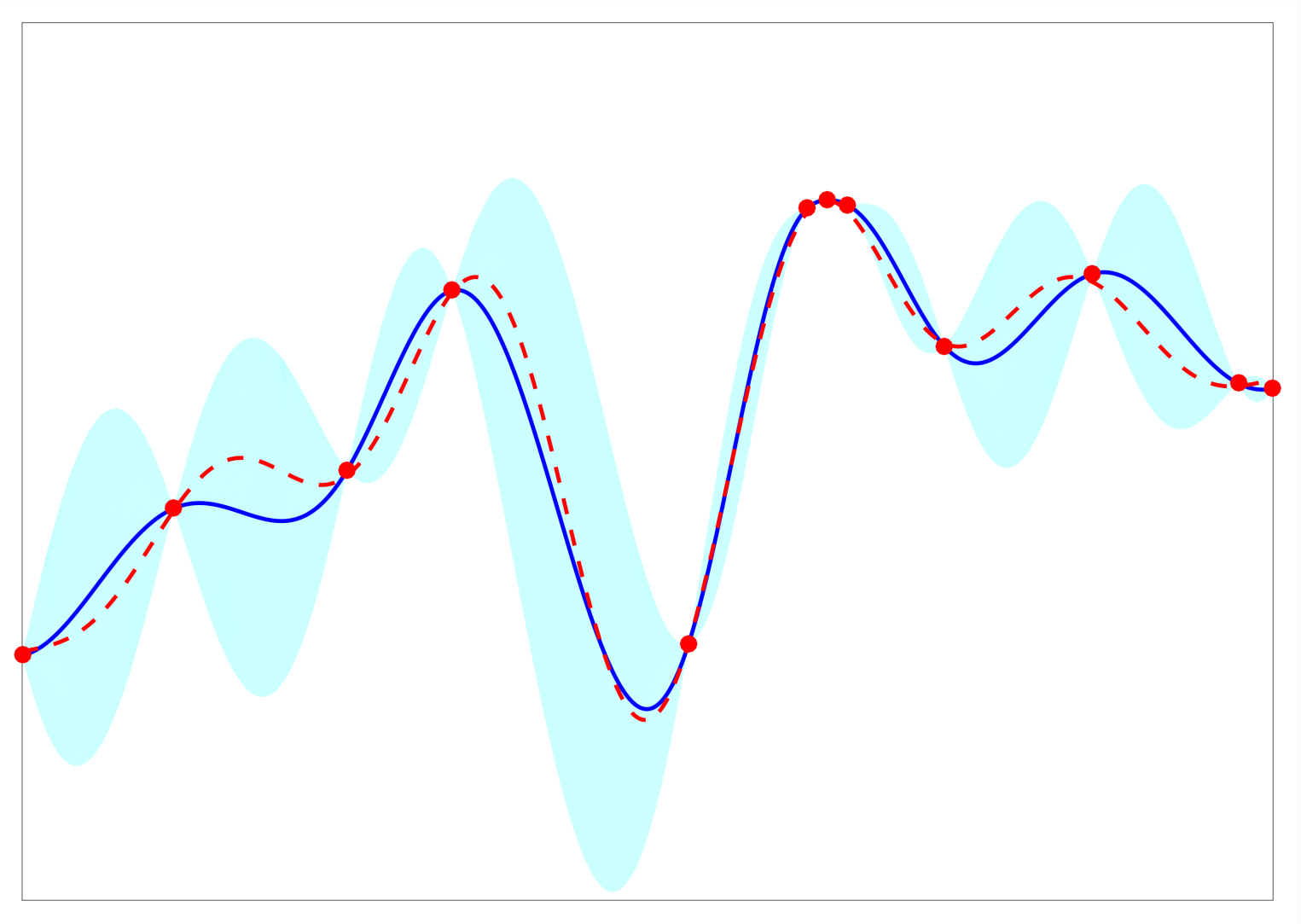


Figure 1: BO example. Red points are evaluations, dotted red line is true underlying function to maximize, solid blue line is GP posterior mean, shading is ± 2 standard deviations, and green line is acquisition function.

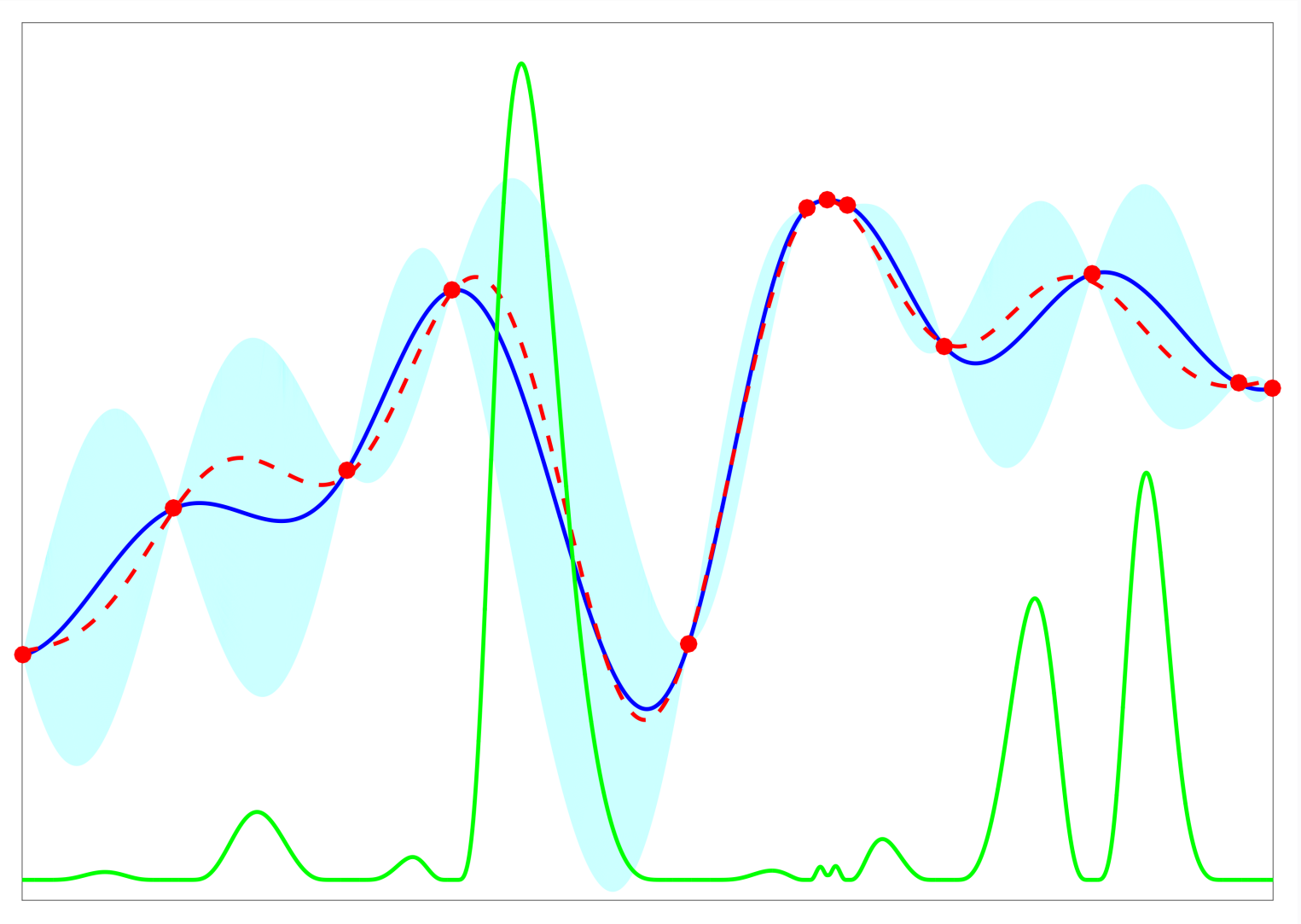


Figure 1: BO example. Red points are evaluations, dotted red line is true underlying function to maximize, solid blue line is GP posterior mean, shading is ± 2 standard deviations, and green line is acquisition function.

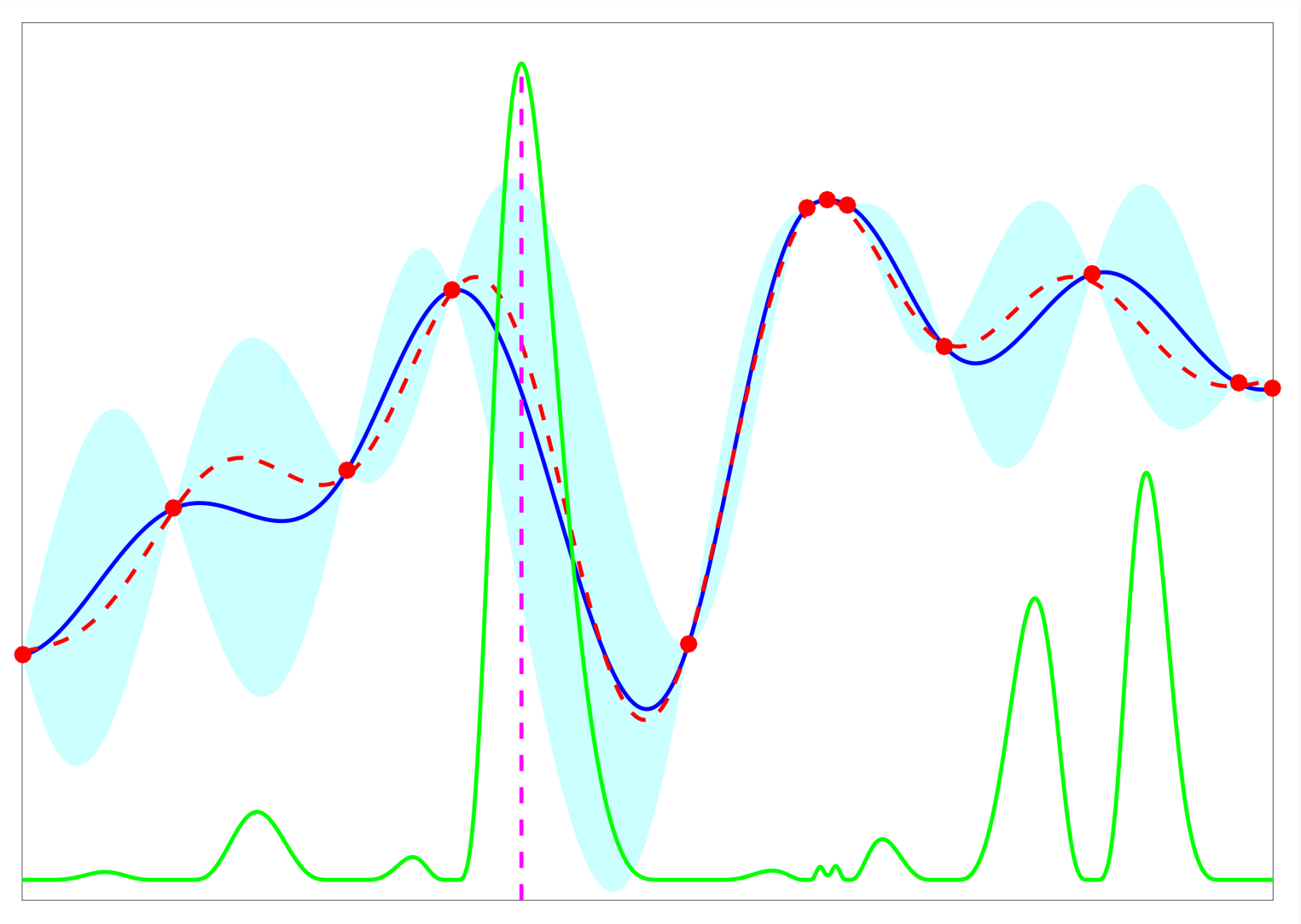


Figure 1: BO example. Red points are evaluations, dotted red line is true underlying function to maximize, solid blue line is GP posterior mean, shading is ± 2 standard deviations, and green line is acquisition function.

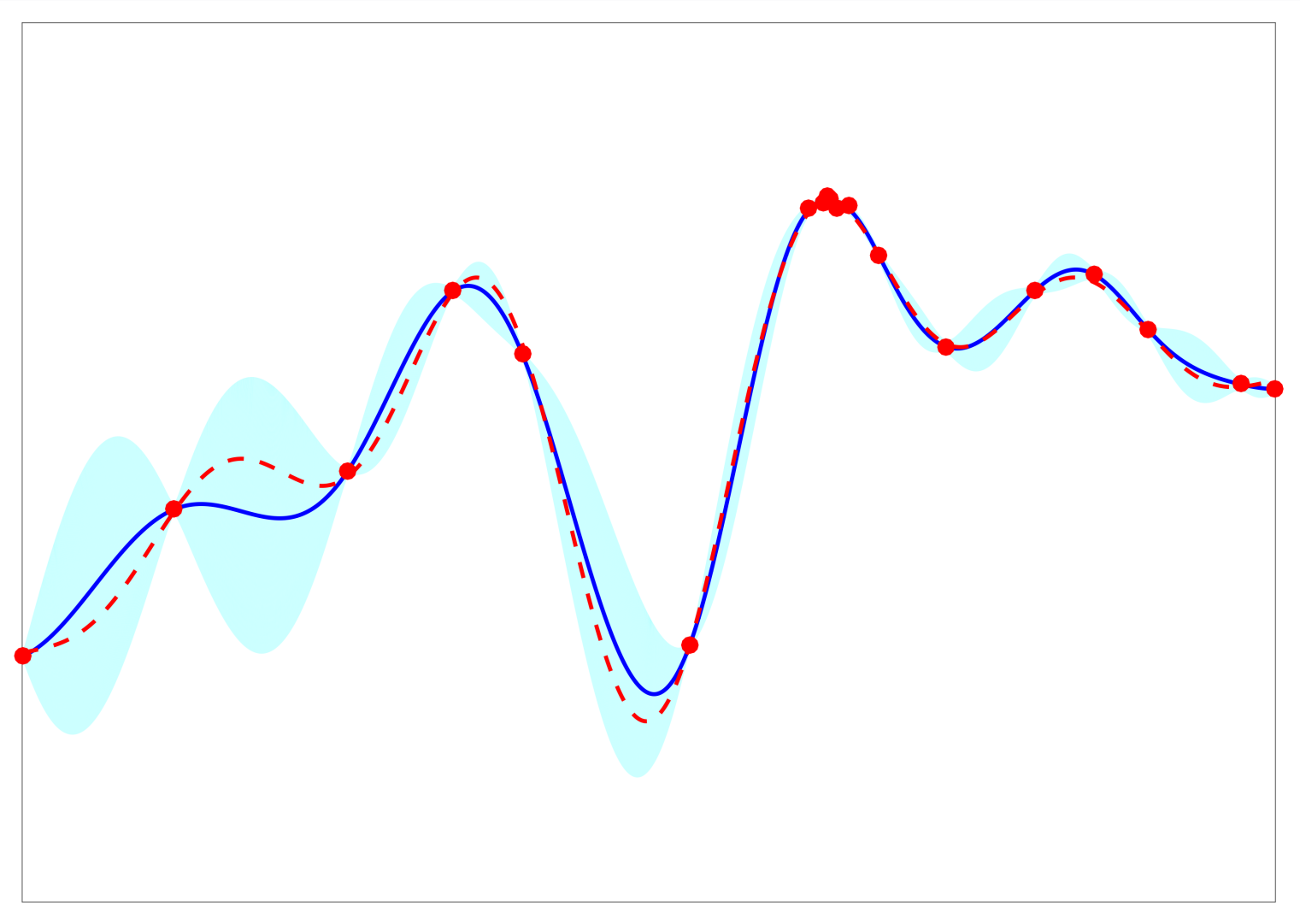


Figure 1: BO example. Red points are evaluations, dotted red line is true underlying function to maximize, solid blue line is GP posterior mean, shading is ± 2 standard deviations, and green line is acquisition function.

Algorithm for Bayesian Optimization

1. Initialize surrogate by training on small number of initial datapoints $\{x_i, y_i\}_{i=1}^{n_{\text{init}}}$ ($n_{\text{init}} = 0$ sometimes possible)
2. Derive acquisition function ζ from surrogate
3. (Approximately) optimize acquisition function to find next point to evaluate

$$x_{n+1} = \arg\max_{x \in \mathcal{X}} \zeta(x)$$

4. Evaluate the target function at x_{n+1}, yielding $y_{n+1} = f(x_{n+1}) + \epsilon_{n+1}$, where ϵ_{n+1} represents any evaluation noise, and add (x_{n+1}, y_{n+1}) to the dataset
5. Refit/update surrogate to incorporate new data
6. If computational budget remaining go back to Step 2, else return final surrogate (estimate for optimum usually taken as x_{i^*} where $i^* = \arg\max_{i \in \{1,\ldots,n\}} \mathbb{E}[f_{\text{surrogate}}(x_i)]$)

Surrogate Models

Good surrogate models should, in general,

- Be effective in the small data regime (if the function is expensive, we will typically not make many evaluations)
- Permit noisy observations
- Provide effective uncertainty estimates
- Lead to effective and cheap acquisition functions

Gaussian processes are often a great fit for these requirements and the most popular choice used in practice.

Alternatives that are sometimes used:

- Random forests; to allow scaling to large numbers of evaluations and faster acquisition of new points
- Bayesian neural networks; for better scaling in dimensionality

Acquisition Functions: Upper Confidence Bounding

Assume we are using a GP surrogate and let $\mu(x)$ and $\sigma(x) = \sqrt{k_{\text{post}}(x, x)}$ denote the current corresponding marginal posterior mean and standard deviation.

One simple, but surprisingly principled, acquisition function is an **upper confidence bound** on the value of the function:

$$\zeta_{\text{UCB}}(x) := \mu(x) + \beta\sigma(x), \quad \beta > 0.$$

Here β is a hyperparameter known as an **optimism boost** (higher β = more exploration) and $\zeta_{\text{UCB}}(x)$ represents the $\Phi(\beta)$ quantile for our distribution on $f(x)$, where Φ is the cdf of a unit Gaussian.

Though a fixed β can lead to pathologies, there are strategies for increasing β with the number of iterations (typically logarithmically or near logarithmically) that provide theoretical guarantees.

Acquisition Functions: Probability of Improvement

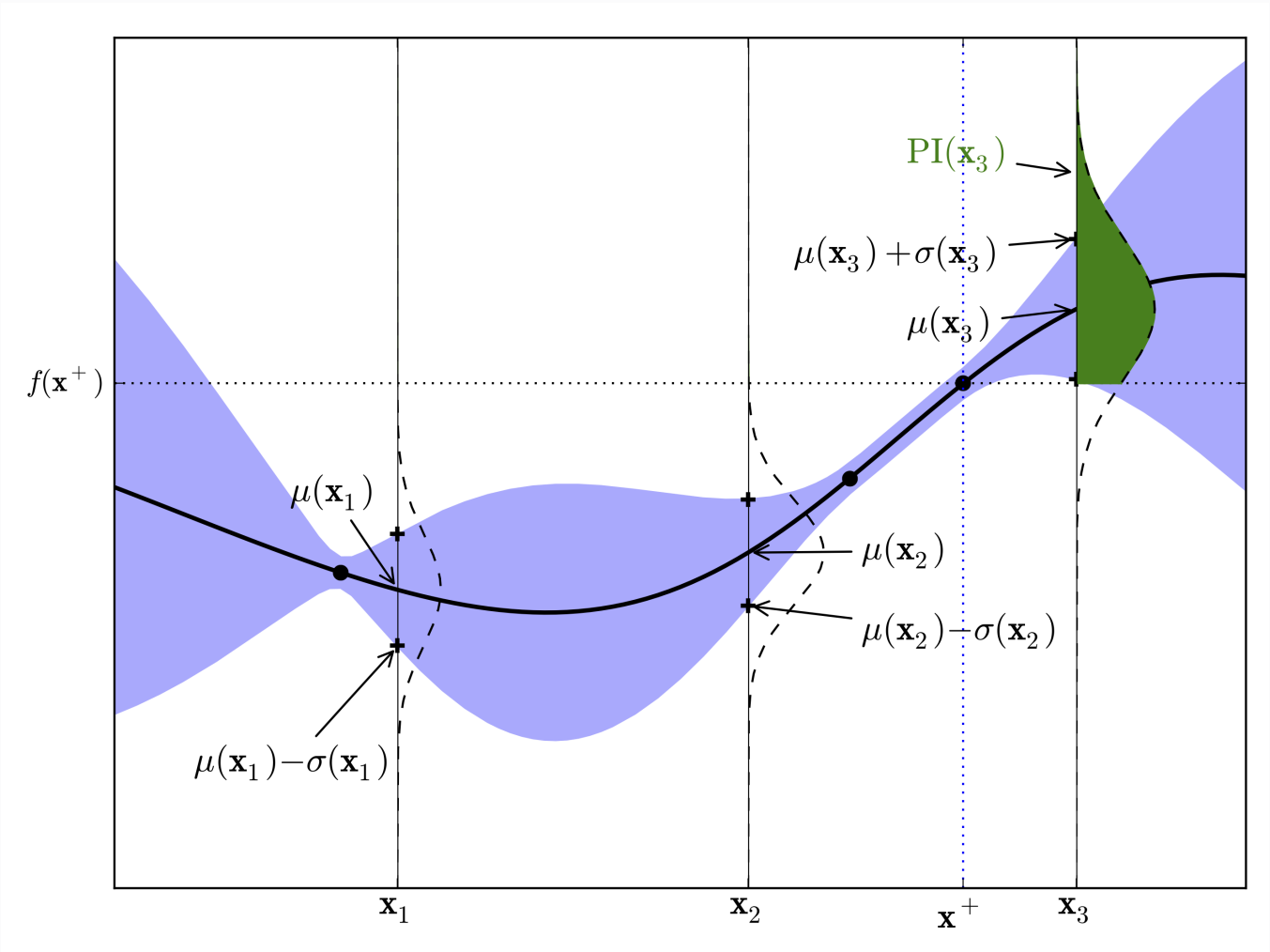


Figure 2: Choose point with the highest probability of being larger than our current best (note $\xi = 0$ in this example). Figure credit: Eric Brochu.

Acquisition Functions: Probability of Improvement (2)

Denote the current largest mean of any of our tested datapoints as

$$\mu^+ = \max_{i \in \{1,\ldots,n\}} \mu(x_i).$$

The probability of improvement of a candidate input x is defined as the probability that the **true function** $f(x)$ is larger than μ^+ by at least some (user–chosen) threshold ξ,

$$\begin{aligned}
\zeta_{\mathsf{PI}}(x) :&= P(f(x) \geq \mu^+ + \xi | \mathcal{D}) \\
&= \int_{\mu^+ + \xi}^{\infty} \mathcal{N}(f(x); \mu(x), \sigma^2(x)) df(x) \\
&= 1 - \Phi\left((\mu^+ + \xi - \mu(x))/\sigma(x)\right).
\end{aligned}$$

Can control level of exploration vs exploitation through ξ.

Rarely used in practice as fails to properly encapsulate the **magnitude** of the improvement and difficult to tune ξ.

Acquisition Functions: Expected Improvement

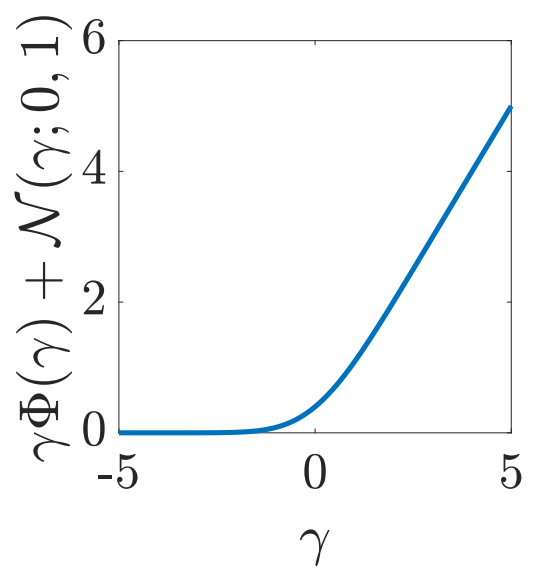
Many of the shortfalls of ζ_{PI} can be corrected by instead using the **expected improvement**

$$\zeta_{\mathsf{EI}}(x) := \int_{\mu^+ + \xi}^{\infty} (f(x) - \mu^+ - \xi)\mathcal{N}(f(x); \mu(x), \sigma^2(x))df(x).$$

Denoting, $\gamma(x) = (\mu(x) - \mu^+ - \xi)/\sigma(x)$, this (see notes)

$$= \sigma(x)\Big(\gamma(x)\Phi(\gamma(x)) + \mathcal{N}(\gamma(x); 0, 1)\Big).$$

Unlike for ζ_{PI}, $\xi = 0$ here is still a viable choice (a small but non–zero value is generally best), while performance is generally less sensitive to its exact value.



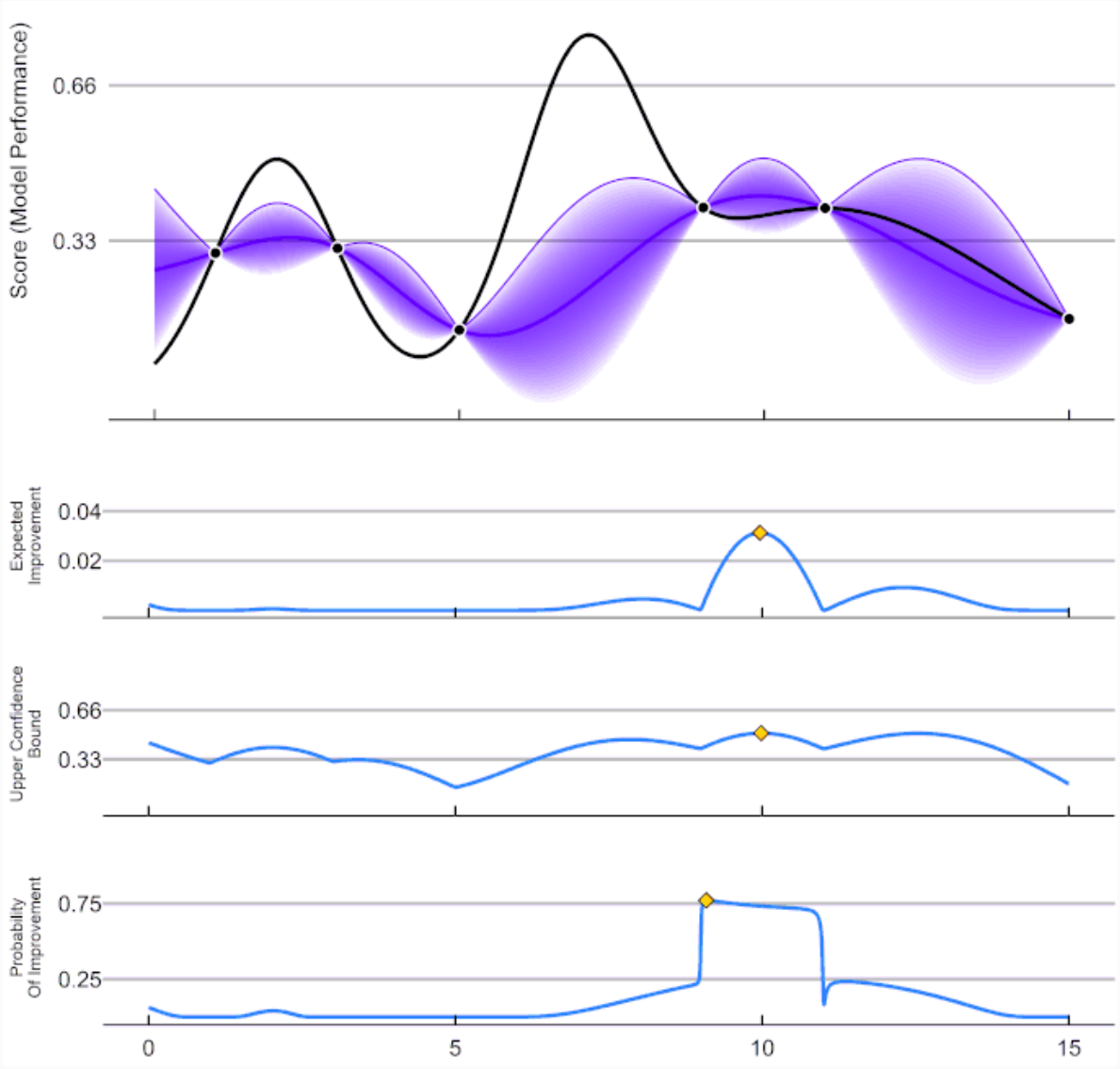


Figure 3: Comparison of behavior of different acquisition functions.

Figure from `https://github.com/AnotherSamWilson/ParBayesianOptimization`

Entropy Based Strategies

If our model is well–specified, the optimal myopic strategy[1] is to choose the input that leads to the biggest expected reduction in the entropy of our posterior of the location of the optimum,

$$\arg\max_{x \in \mathcal{X}} \; - \mathbb{E}_{y_{n+1}} \left[\mathsf{H} \left[p\left(x^* | \{x_i, y_i\}_{i=1}^{n+1} \right) \right] \middle| \, x_{n+1} = x \right].$$

Though this is horribly intractable to calculate directly, even for a GP, some of the most effective (but generally complicated and costly) acquisition strategies are based around trying to directly approximate it and then optimizing the approximation.

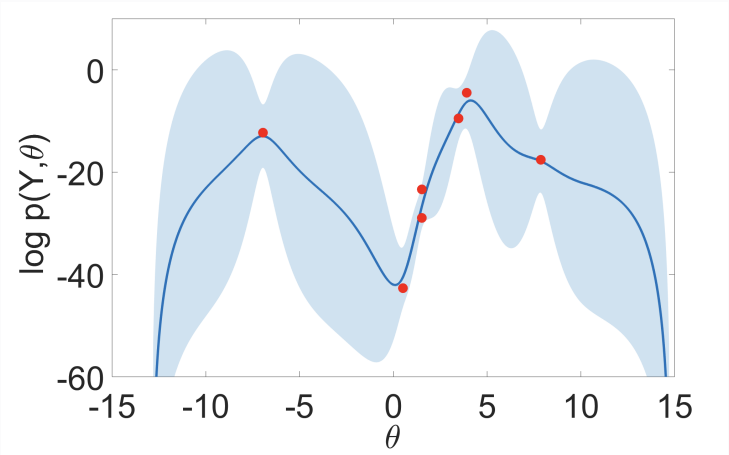
[1] That is the optimal strategy which does not take into account of own future decision making.

Input Bounds

Note that we generally require that either

1. $\mathcal{X}$ is a bounded domain that is not overly large, or
2. Our GP has a non–constant prior mean that falls away outside a region of interest (or some other mechanism that softly constrains us to acquire points in some region).

This stops our acquisition strategy continuously sampling points further and further away from our initialization, or even tending off to infinity (i.e. $x = \infty$ might be the optimum of $\zeta(x)$)



Strengths and Limitations

+ Can be an incredibly sample efficient way of performing **global** optimization in certain settings; even producing exponential convergence rates in some settings
+ Does not require derivatives (though can also condition GP on them if available) and naturally deals with noisy evaluations
+ Provides characterization of function, not just a point estimate for optimum
- Expensive: should be avoided unless f itself expensive
- Generally scales poorly in dimensionality
- Model misspecification can wreck performance: often require some degree of inference over hyperparameters
- Many common problems have highly non–Gaussian noise and so often need to make transformations and/or approximations

Further Reading

- Tutorial from UAI 2018: `https://youtu.be/C5nqEHpdyoE`
- Accessible tutorial paper: Bobak Shahriari et al. “Taking the human out of the loop: A review of Bayesian optimization”. In: **Proceedings of the IEEE** 104.1 (2015), pp. 148–175
- Open source Python library:

  `https://sheffieldml.github.io/GPyOpt/`