Introduction to Bayesian Optimisation

Rui-Yang Zhang

Contents

Preface

Notes on the introduction to Bayesian optimisation. Several classes of common and standard acquisition functions are introduced. A recurring numerical example is presented for most of the introduced acquisition functions, and the codes are implemented using the BoTorch package in Python (Balandat et al., 2020).

Chapter 1

Introduction

1.1 Motivations of Bayesian Optimisation

Optimisation is a fundamental task in many modern-day disciplines and is particularly essential in the field of statistics and machine learning. A large proportion of the questions of interest in statistics and machine learning can be phrased as an optimisation problem - given some cost function f defined on some input space X, we wish to find a value $x \in X$ that minimises f. Depending on the function f and our knowledge of it, different approaches can be taken.

For example, the most standard optimisation assumes that f is smooth and we know its analytic expression as well as its gradient ∇f. Also, the input space X is the full Euclidean space. Then, optimisation of f can be done using (stochastic) **gradient descent**, where we iteratively move our values in the directions of the steepest descent characterised by the gradient at the current position. Provided with further assumptions on f, such as convexity or Lipschitz continuity, we can characterise the quality of the algorithm and its speed of convergence (to an optimum).

We could have a constrained input space and a certain structure of the cost function f that could be beneficial to us. For example, if the cost function is linear, and the input space is limited by a set of linear constraints (such as linear inequalities), we can phrase the optimisation problem as a linear programming problem, and tackle it using devices such as the **simplex algorithm**. Theoretical understanding of such linear programming algorithms and amendments for large and hard problems are abundant and well-established in the mathematical programming literature.

However, there is another class of problems - we do not know the analytic expression of f, we think it is probably continuous, and it is extremely costly to do point-wise evaluations of f. This class of problems, although sounding unnecessarily hard and unrealistic at first, is increasingly more present in modern-day machine learning. These functions f could be a performance measure of a large model (say a massive, deep neural network with non-trivial architecture or a large language model) and the input space is the set of all possible values of the parameters of the model - which could be of thousands or even millions in dimension in total. In such a case, any of the above methods like gradient descent and linear programming will fail instantly as the fundamental assumptions of such algorithms are never going to be met. In addition, we should refrain from doing a grid search or any other method that involves a lot of evaluations, as computing f in this type of case will involve re-training the model with the full dataset and it could be extremely costly. This type of optimisation is not only present in machine learning but in many real-world applications too. For example, we may be interested in understanding some remote areas of the Earth (say the sea floor) and our detection method is highly costly (in price and time) so we do not wish to blindly make too many detections and observations unnecessarily. For such problems, the common approach is via **Bayesian optimisation**, which is the central topic of this note.

1.2 Bayesian Optimisation Framework

From the top level, Bayesian optimisation is a sequential/iterative decision-making process that recursively conducts the following to find an optimum for the function f:

1. Find the next evaluation point x according to a policy.
2. Evaluate the function at the point, $y = f(x)$, and add (x, y) to our dataset $\mathcal{D}$.
3. Update our surrogate model $g \approx f$ using the updated dataset $\mathcal{D}$.

There are two key ingredients of the above procedure - the policy and the surrogate model. The surrogate model is a sufficiently large class of functions that is (relatively) cheap to work with. Almost always, for Bayesian optimisation, the surrogate model is chosen as a **Gaussian process** due to its nice conjugacy for model updates and its ability to quantify uncertainties. A short introduction to Gaussian processes will be presented in Chapter 2.

The choice of policy, however, is not as unanimous. The goal of the policy is to direct our search strategy for the optimum, and depending on the questions of interest, we could be focusing on different things - leading to a different choice of policy. A theme in the design of search policy is the **exploration-exploitation tradeoff**, which is the problem of spending our limited (computational) resources on having a better grasp of the problem - i.e. exploration - versus improving on our existing results - i.e. exploitation. Naturally, over-emphasis on either one of the two is not desirable, and careful deliberation is needed. The rest of the notes will be devoted to discussing the various existing search policies and their similarities and differences.

More often than not, the policy of choice is phrase using an **acquisition function**. We denote the input space of our objective function f as X, and consider the search space - the space that we are allowed to make evaluations - as S. Here we will assume that $S = X$, and will use X to denote both for simplicity of notation. However, one should note that the search space and the input space need not be identical, and in many real-life applications, we are interested in the case where $S \subsetneq X$. The acquisition function α is a function defined on the entire search space S, and it captures the benefits (further gain for maximisation and further loss for minimisation) of making a further evaluation at that particular point. Then, our BO step 1 is to find $\arg\max_{x \in S} \alpha(x)$ as our next evaluation point.

Before moving on to describe some common examples of acquisition functions (and how to find their arg max exactly or approximately), let us pause for a bit and consider what we are doing with the BO loops. As mentioned earlier, BO is doing sequential decision-making, and in particular, for each iteration, we are making the move that maximises our gain *within one more move*. This means, using decision theory terms, we are doing an **one-step lookahead**, which is a **myopic**, or greedy, policy. Certainly, we could make it non-myopic by considering doing a multiple-step lookahead and picking the choices of these steps that maximise our overall gain. However, there are two immediate problems: (1) it is significantly more costly to find the best n-moves, as it often involves solving some Bellman equation which gets exponentially harder as n increases; (2) our understanding of the objective function via the acquisition function is incomplete, and making too many predictions in sequence using a bad model is not a good idea. Therefore, although myopic policies are not ideal in general, in the case of Bayesian optimisation, it becomes beneficial.

For the rest of this notes, we will consider a maximisation problem for our BO by default. The conversion of a minimisation problem $\min f$ can be transformed trivially to a maximisation problem as $\max -f$.

Chapter 2

Introduction to Gaussian Processes

In this chapter, we will introduce the basic notions and relevant results about Gaussian processes, which is the fundamental model that we will use for modelling ocean currents. We will motivate Gaussian processes and illustrate it as a natural extension of the multivariate Gaussian in Section 2.1. We will then describe various linear algebra results and their implications in Gaussian random vectors in Section 2.2, which serve as the theoretical building blocks for Gaussian process regressions in Section 2.3. The main degree of freedom of a Gaussian process is its covariance function, and we will spend Section 2.4 investigating it in detail. One should note that a large portion of the material of this chapter is based on Williams and Rasmussen (2006).

2.1 From Gaussian Distribution To Gaussian Process

A **stochastic process** is a sequence of random variables $\{X_t\}_t$ indexed by t. The index usually represents the time steps of the process, but it is not always the case. In the case of the **Gaussian process** (GP), and in particular the applications of GP that we consider here, the index is not related to time but to space. This change will become clear soon.

A univariate Gaussian distribution $N(\mu, \sigma^2)$ with mean μ and variance σ^2 is a common object of interest. It is one-dimensional, and we can generalise it to make it finite-dimensional, which we denote by $N_d(\mu, \Sigma)$ and call it a multivariate Gaussian, where the **mean vector** μ here is now a d-vector and **covariance matrix** Σ is a $d \times d$ matrix that is symmetric and positive semi-definite. A matrix Σ is symmetric if $\Sigma^T = \Sigma$, and it is positive semi-definite if $x^T \Sigma x \geq 0$ for all d-vectors x.

For simplicity, we will let the means and the mean vectors we consider here zero unless stated otherwise. Consider a multivariate Gaussian with highly correlated dimensions. Normally, when we plot a sample from a d-dimensional Gaussian, we will plot it in the d-dimensional space. Here, we will do something different, and plot the values of each dimension in the same plot, sequentially. In the following plots, we can see how a sample might look like in this format with $d = 3, 10, 100$.

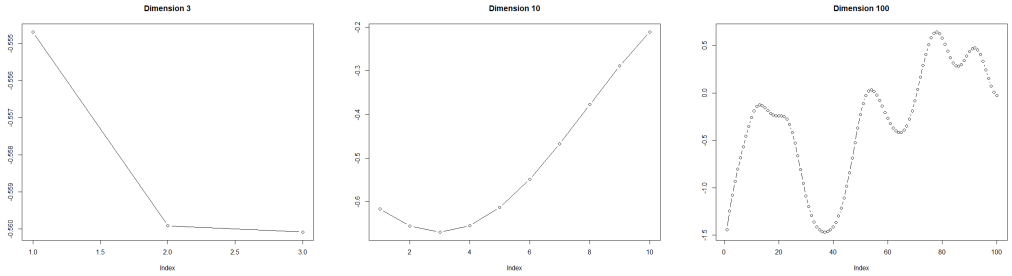

Figure 1: Sequentially plotted samples from d-dimensional Gaussian with $d = 3, 10, 100$.

As shown in Figure 1, the highly correlated Gaussian plotted sequentially makes it look very much like a (smooth) function. This inspires us to consider the infinite-dimensional extension of a multivariate Gaussian and use that to approximate a function. This is the idea of **Gaussian process**.

Formally, the **Gaussian process** (GP) $y = \{y(x)\}_{x\in\mathcal{X}}$ is a stochastic process indexed by space $x \in \mathcal{X}$ with a mean function $\mu(\cdot)$ and a covariance function $k(\cdot,\cdot)$ such that $\mathbb{E}[y(x)] = \mu(x)$ and $\mathrm{Cov}(y(x), y(x')) = k(x, x')$. We will denote such a process as

$$y(\cdot) \sim GP(\mu(\cdot), k(\cdot,\cdot)).$$

A key feature of a GP is that any finite-dimensional segment (i.e. isolating finitely many points from the full process) is a multivariate Gaussian distribution.

The covariance function needs to satisfy some requirements for it to be a suitable covariance for a GP. The requirement is essentially an extension of the requirement of covariance matrices for a multivariate Gaussian. The covariance function k of a GP needs to be **symmetric** (so $k(x, y) = k(y, x)$ for any x, y) and **positive semi-definite**, in the sense that for any $x_1, x_2, \ldots, x_n$, the matrix K formed by setting $K_{ij} = k(x_i, x_j)$ needs to be a positive semi-definite matrix. The covariance function is a way to ensure the dependency/similarity of points across indices.

The two degrees of freedom of a GP are its mean function and its covariance function, and we have assumed the mean is zero. It is not hard to assume that the covariance function characterises the GP to a very large extent. The detailed ways of how properties of the covariance function imply the properties of the GP, as well as some common covariance functions, will be discussed in Section 2.4.

A Gaussian process defined on the real line $\mathbb{R}$ will have (uncountably) infinite points, meaning that if we wish to plot it numerically, we would need to generate all infinitely many points, which is not feasible. Instead, the common practice in such situations is to do a simple discretisation. Assume that we are interested in a small region of the real line, say $[a, b]$ with $a < b$, we will break the interval down into smaller equal-length pieces and use the endpoints of those pieces to approximate the full trajectories. To be more precise, if we wish to break the interval down into n pieces, we would have

$$a = x_1 < x_2 < \cdots < x_n = b$$

where $x_{i+1} - x_i = (b - a)/(n - 1)$ for $i = 1, 2, \ldots, n - 1$. Then, using the property of GP, the distribution of any finite points will follow a multivariate Gaussian distribution. If we denote the covariance matrix of the points $x = (x_1, x_2, \ldots, x_n)$ by K where $K_{ij} = k(x_i, x_j)$ and k is the covariance function of the GP, then we have

$$y(x) = \begin{bmatrix} y(x_1) \\ y(x_2) \\ \vdots \\ y(x_n) \end{bmatrix} \sim N_n(\mu(x), K)$$

so generating those points would be straightforward. The gaps between the points will be extrapolated by a straight line. We can also the variance of the multivariate Gaussian to draw a confidence region of the generated process.

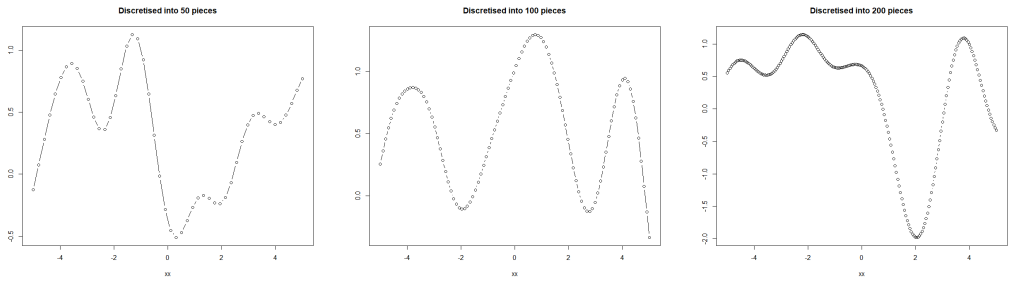

Figure 2: Gaussian Processes generated with different number of segments.

2.2 Gaussian Properties and Linear Algebra

In this section, we will look at several key properties of Gaussian random variables and Gaussian processes that will play a fundamental role in the rest of this note. Some derivations will be included.

Consider we have a multivariate Gaussian distribution $N_d(m, \Sigma)$ with mean vector m and covariance matrix Σ. The probability density function is given by

$$p(x; m, \Sigma) = (2\pi)^{-d/2}|\Sigma|^{-1/2} \exp\left[-\frac{1}{2}(x-m)^T\Sigma^{-1}(x-m)\right].$$

Next, consider two multivariate Gaussian $X \sim N_{d_1}(m_x, A)$ and $Y \sim N_{d_2}(m_y, B)$ where d_1 and d_2 may not be the same. The joint distribution of X and Y is given by

$$\begin{bmatrix} X \\ Y \end{bmatrix} \sim N_{d_1+d_2}\left(\begin{bmatrix} m_x \\ m_y \end{bmatrix}, \begin{bmatrix} A & C \\ C^T & B \end{bmatrix}\right)$$

where C represents the covariance matrix between X and Y. If X and Y are independent, then C is zero. With this joint distribution, we would be interested in the marginal distributions and the conditional distributions. We have, for marginals,

$$\begin{aligned} p(x) &= \int p(x,y)dy, \qquad X \sim N_{d_1}(m_x, A) \\ p(y) &= \int p(x,y)dx, \qquad Y \sim N_{d_2}(m_y, B). \end{aligned}$$

For conditionals, we have

$$\begin{aligned} p(x|y) &= p(x,y)/p(y), \qquad X|Y = y \sim N(m_x + CB^{-1}(y-m_y), A - CB^{-1}C^T) \\ p(y|x) &= p(x,y)/p(x), \qquad Y|X = x \sim N(m_y + C^TA^{-1}(x-m_x), B - C^TA^{-1}C). \end{aligned}$$

The derivation of the marginal distributions is easy to see, which we omit. For conditionals, it is more complicated and we will establish it below. One identity that we will use is the formula for the inverse of the block matrix, which is easy to verify.

Proposition 2.1 (Inverse of Block Matrix)**.** *For block matrix*

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

where A, B, C, D are matrices on their own, the inverse

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}^{-1} = \begin{bmatrix} (A-BD^{-1}C)^{-1} & -(A-BD^{-1}C)^{-1}BD^{-1} \\ -D^{-1}C(A-BD^{-1}C)^{-1} & D^{-1} + D^{-1}C(A-BD^{-1}C)^{-1}BD^{-1} \end{bmatrix}.$$

Using the above result, we can derive the conditional distribution. We will just derive the case of $X|Y = y$, and the distribution of $Y|X = x$ is similar.

Proposition 2.2. *For joint distribution of X_1 and X_2 where X_1 is of dimension n_1, X_2 is of dimension n_2 and the joint is of dimension $n = n_1 + n_2$, it is given by*

$$\begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \sim N(\mu, \Sigma) = N\left(\begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix} \right),$$

we have the conditional distribution

$$X_1|X_2 = x_2 \sim N(\mu_{1|2}, \Sigma_{1|2}).$$

where $\mu_{1|2} = \mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(X_2 - \mu_2)$ and $\Sigma_{1|2} = \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}$.

Proof. Using the definition of density, we have

$$\begin{aligned} p(x_1|x_2) = \frac{p(x_1, x_2)}{p(x_2)} &= \frac{(2\pi)^{-n/2}|\Sigma|^{-1/2} \exp\left[-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right]}{(2\pi)^{-n_2/2}|\Sigma_{22}|^{-1/2} \exp\left[-\frac{1}{2}(x_2-\mu_2)^T\Sigma_{22}^{-1}(x_2-\mu_2)\right]} \\ &= (2\pi)^{-n_1/2}|\Sigma|^{-1/2}|\Sigma_{22}|^{1/2} \exp\left[-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu) + \frac{1}{2}(x_2-\mu_2)^T\Sigma_{22}^{-1}(x_2-\mu_2)\right]. \end{aligned}$$

Following the inverse of block matrix formula of Proposition 2.1, we have the following if we focus just on the exponential

$$\begin{aligned} & -\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu) + \frac{1}{2}(x_2-\mu_2)^T\Sigma_{22}^{-1}(x_2-\mu_2) \\ = & -\frac{1}{2}(x-\mu)^T \\ & \begin{bmatrix} (\Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21})^{-1} & -(\Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21})^{-1}\Sigma_{12}\Sigma_{22}^{-1} \\ -\Sigma_{22}\Sigma_{21}(\Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21})^{-1} & \Sigma_{22}^{-1} + \Sigma_{22}^{-1}\Sigma_{21}(\Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21})^{-1}\Sigma_{12}\Sigma_{22}^{-1} \end{bmatrix} \\ & (x-\mu) + \frac{1}{2}(x_2-\mu_2)^T\Sigma_{22}^{-1}(x_2-\mu_2) \\ = & -\frac{1}{2}(x_1-\mu_1)^T(\Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21})^{-1}(x_1-\mu_1) \\ & + (x_1-\mu_1)^T(\Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21})^{-1}\Sigma_{12}\Sigma_{22}^{-1}(x_2-\mu_2) \\ & - \frac{1}{2}(x_2-\mu_2)^T[\Sigma_{22}^{-1} + \Sigma_{22}^{-1}\Sigma_{21}(\Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21})^{-1}\Sigma_{12}\Sigma_{22}^{-1}](x_2-\mu_2) \\ & + \frac{1}{2}(x_2-\mu_2)^T\Sigma_{22}^{-1}(x_2-\mu_2) \\ = & -\frac{1}{2}(x_1-\mu_1)^T(\Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21})^{-1}(x_1-\mu_1) \\ & + (x_1-\mu_1)^T(\Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21})^{-1}\Sigma_{12}\Sigma_{22}^{-1}(x_2-\mu_2) \\ & - \frac{1}{2}(x_2-\mu_2)^T\Sigma_{22}^{-1}\Sigma_{21}(\Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21})^{-1}\Sigma_{12}\Sigma_{22}^{-1}(x_2-\mu_2). \end{aligned}$$

If we rearrange them, the above term would become

$$-\frac{1}{2}[x_1 - \mu_1 - \Sigma_{12}\Sigma_{22}^{-1}(x_2-\mu_2)]^T(\Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21})^{-1}[x_1 - \mu_1 - \Sigma_{12}\Sigma_{22}^{-1}(x_2-\mu_2)]$$

which yields the mean and covariance of the conditional distribution, as desired. □

Another nice property of Gaussian distributions is that a linear combination of Gaussian is still Gaussian. For example, for multivariate X, $AX + B$ is a multivariate distribution too where A is a matrix and B is a vector with the right dimension. This can be formally established using the moment-generating function, which we will omit.

Proposition 2.3. *For multivariate Gaussian $X \sim N(\mu, \Sigma)$, given matrix A and vector B with the right dimension, we have*

$$AX + B \sim N(A\mu + B, B\Sigma B^T).$$

A consequence of the above result is that the sum of two multivariate Gaussian with the same dimension is also a multivariate Gaussian.

Proposition 2.4. *For independent multivariate Gaussian $X \sim N(\mu_1, \Sigma_1)$ and $Y \sim N(\mu_2, \Sigma_2)$ with the same dimension, we have*

$$X + Y \sim N(\mu_1 + \mu_2, \Sigma_1 + \Sigma_2)$$

The above result can be obtained by first getting the joint distribution of X and Y, then considering the linear transformation with $A = I$ and $B = 0$.

Another result that we will state but not derive is the product of two Gaussian densities is proportionate to a Gaussian density.

Proposition 2.5. *Let $X \sim N(a, A)$ with density $p(x)$ and $Y \sim N(b, B)$ with density $p(y)$, we have*

$$p(x)p(y) \propto p(z)$$

where $p(z)$ is the density of $Z \sim N(c, C)$ and

$$C = (A^{-1} + B^{-1})^{-1}, \qquad c = C(A^{-1}a + B^{-1}b).$$

2.3 Gaussian Process Regression

The results mentioned in the previous section will play an important role in this section where we consider doing the task of **regressions** using Gaussian processes. In the field of **supervised learning** (where we have both data and their label, as opposed to **unsupervised learning** where we only have data and not labels), the two main tasks are regression and **classification**. They are essentially the same problem, but regression deals with continuous labels, and classification deals with discrete labels. One should also know that Gaussian process regression is used in many areas, and it is often called **kriging** in the spatial statistics literature.

2.3.1 Linear Regression

The problem of regression studies the relationship between covariates and the response using data. For example, in many scientific disciplines, we are interested in understanding the relationship between various factors (called **covariates**) and some key metric (called **response**), e.g. the relationship between biological measurements (such as weight, height, and blood pressure) and the hazard rate of a disease (such as diabetes). This has been heavily studied throughout the history of statistics and machine learning. One of the most basic types of regression, which is the topic here, is linear regression where we assume a linear relationship between the covariates and the response. We will also take a Bayesian approach.

If we denote the covariates by a vector x and the response by y (which we assume to be one-dimensional here, although it can be generalised), the linear regression assumes the following model:

$$f(x) = x^T w, \qquad y = f(x) + \varepsilon \tag{1}$$

where w is the weight vector for each of the covariates, $f(x)$ is the (latent) true function representing the relationship between x and y, while ε represents the noise vector, and we assume it is a Gaussian random variable here with mean zero and variance σ_n^2, i.e. $\varepsilon \sim N(0, \sigma_n^2)$ where n is the size of the data set. Normally, one of the covariates in x will represent the bias of the model, and that element in x is usually put as 1.

The Equation (1) represents the general formula, and for each data point, indexed by i, with response y_i and covariates x_i. The likelihood of the model for observing a data point (x_i, y_i) is then

$$p(y_i|w, x_i) = \frac{1}{\sqrt{2\pi}\sigma_n} \exp\left[-\frac{(y_i - x_i^T w)^2}{2\sigma_n^2}\right] \tag{2}$$

due to the Gaussianity of the noise ε.

We assume the variance of the noise is known, which can usually be estimated using an empirical variance estimator. This assumption means that the only unknowns of our linear regression problem are the values of the weight vector w, which we aim to infer using data. In the Frequentist setting, the likelihood described in Equation (2) is all we need - we just multiply the likelihoods for all the data points, and find the w values that maximise the multiplied quantity. In the Bayesian setting, we need more than that. First, we need to impose a prior distribution to the weight vector w, and then compute the posterior by multiplying the prior and the likelihood, then normalise, using the Bayes formula. For **prior** of weight vector w, we will use

$$w \sim N(0, \Sigma_p). \tag{3}$$

Some further comments on the prior choices will be stated later on.

Combining the prior of Equation (3) and the likelihood of Equation (2), we have the **posterior** distribution as follows using Proposition 2.5

$$\begin{aligned} p(w|x, y) &\propto p(w)p(y|w, x) \\ &\propto \exp\left[-\frac{1}{2\sigma_n^2}(y - x^T w)^T(y - x^T w)\right] \exp\left[-\frac{1}{2}w^T\Sigma_p^{-1}w\right] \\ &\propto \exp\left[-\frac{1}{2}(w - \bar{w})^T\left(\frac{1}{\sigma_n^2}xx^T - \Sigma_p^{-1}\right)(w - \bar{w})\right] \end{aligned}$$

where $\bar{w} = \sigma_n^{-2}(\sigma_n^{-2}xx^T + \Sigma_p^{-1})^{-1}xy$, making the posterior the density of

$$w|x, y \sim N(\bar{w}, A^{-1}), \qquad A := \sigma_n^{-2}xx^T + \Sigma_p^{-1} \tag{4}$$

which is a very nice conjugacy. An estimator for the weights given to the data will be some summary statistics of the posterior distribution, such as the mean or the mode. In this case, due to the geometry of the Gaussian distribution, the mean and the mode are identical and are identical to the maximum likelihood estimator when we formulate the problem as a **ridge regression** in the frequentist literature. If we use a different prior for w, such as the slab-and-spike prior, we could recover the **lasso regression** maximum likelihood estimator.

If we are fed with a new data point without the response to the model, we are essentially hoping to make a prediction using the weight vector $w|x, y$ we have inferred and the observed covariates x_*. The **prediction** distribution f_* will have the distribution

$$p(f_*|x_*, x, y) = \int p(f_*|x_*, w)p(w|x, y)dw, \qquad f_* \sim N\left(\frac{1}{\sigma_n^2}x_*^T A^{-1}xy, x_*^T A^{-1}x_*\right) \tag{5}$$

using again the result of Proposition 2.5.

In general, there are two main goals of regressions: (1) understanding the relationship between covariates and response, (2) predicting the response for new data points. The first point is to make analyses and comments based on our updated knowledge of weights w, while the second point is to exploit the predictive distribution of Equation (5). In our case, the relationship between the covariates and the response is forced to be linear by construction, which is like 'fitting a straight line'. In general, as we expand the class of functions for the possible relationships $f(x)$ (such as in the case of GLM), we would be 'fitting a curve'. In the next part, we will study how one can do curve-fitting using GPs.

2.3.2 Regression using GPs

Linear regression is fitting a linear function $f(x) = x^T w$ using the observed covariates and the response variable. A Gaussian process, as we have described in Section 2.1, can be used to approximate a lot of functions. This inspires us to use a GP to model the relationship $f(x)$. An important assumption that we will make here is to assume the noise ε is always Gaussian, which allows us to leverage the nice conjugacy of Gaussian random variables. This will become clear soon.

First, we will consider the simple problem of regression with exact observation, meaning that the observed data points $(x_i, y_i)_i$ contain no noise, so we get what we see. In this case, we will use x to denote all the observed covariates and f to denote all the observed responses. In this case, extending what we have described at the end of Section 2.1, we can have the joint distribution of f and f^* where f^* is the points that we use to approximate the GP trajectory discretised at points x^* as

$$\begin{bmatrix} f \\ f^* \end{bmatrix} \sim N\left(0, \begin{bmatrix} K(x,x) & K(x,x^*) \\ K(x^*,x) & K(x^*,x^*) \end{bmatrix}\right)$$

where K represents the covariance matrix constructed by measuring the covariance between any two points using the covariance function k of the GP. As our observations do not have noise, the covariance matrix is exactly as it is. Consequently, using Proposition 2.2, we know that

$$f^*|x, x^*, f \sim N(K(x^*,x)K(x,x)^{-1}f, K(x^*,x^*) - K(x^*,x)K(x,x)^{-1}K(x,x^*)$$

which essentially collapses the process at the observed points x, f due to the absence of noise in observations. This gives us the regression curve after observing the data (x, f).

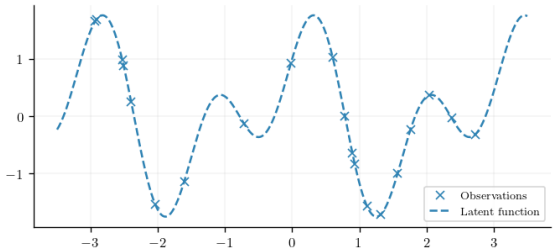

Figure 3: Latent Function of Gaussian Process Regression with 20 data points and no noise, using GPJax package of Pinder and Dodd (2022).

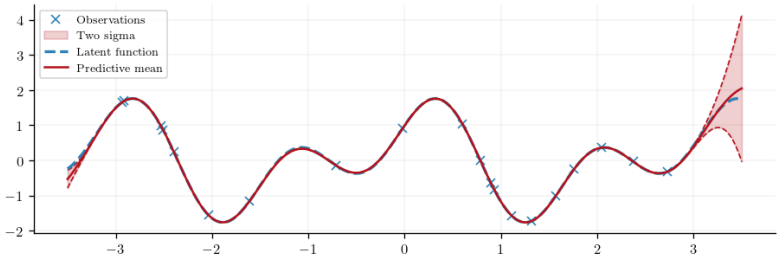

Figure 4: Posterior of Gaussian Process Regression with 20 data points and no noise, using GPJax package of Pinder and Dodd (2022).

A straight-forward and probably necessary extension is to consider the case where we make not exact but noisy observations where the noise is

$$\varepsilon \sim N(0, \sigma_n^2).$$

In this case, if we observe (x_p, y_p) and (x_q, y_q), their covariance will be

$$\text{Cov}(y_p, y_q) = k(x_p, x_q) + \sigma_n^2 \delta_{pq}$$

where $\delta_{ab} = 1$ when $a = b$ and is zero otherwise. We can then generalise it to get

$$\text{Cov}(y) = K(x, x) + \sigma_n^2 I$$

and

$$\begin{bmatrix} y \\ f^* \end{bmatrix} \sim N\left(0, \begin{bmatrix} K(x,x) + \sigma_n^2 I & K(x, x^*) \\ K(x^*, x) & K(x^*, x^*) \end{bmatrix}\right).$$

This then leads to the following conditional distribution, which is the regression curve after the observations

$$\begin{aligned} f^*|x, x^*, f &\sim N(\bar{f}^*, \text{Cov}(f^*)) \\ \bar{f}^* &:= K(x^*, x)[K(x,x) + \sigma_n^2 I]^{-1} f, \\ \text{Cov}(f^*) &:= K(x^*, x^*) - K(x^*, x)[K(x,x) + \sigma_n^2 I]^{-1} K(x, x^*). \end{aligned}$$

Another quantity of interest is the **marginal likelihood** of observing the data $p(y|x)$ given the model. Notice that here we are not specifying the dependencies of parameters in any of our expressions explicitly, we can certainly imagine the existence of some hyperparameters in our kernel function, as well as the variance of the observation noise being unknown. Knowing the marginal likelihood $p(y|x)$ is helpful for estimating these quantities of interest, and an expression for the marginal likelihood is

$$\begin{aligned} f|X \sim N(0, K), \quad y|f &\sim N(f, \sigma_n^2 I) \\ p(y|x) &= \int p(y|x, f) p(f|y) df \\ \log p(y|x) &= -\frac{1}{2} y^T (K + \sigma_n^2 I)^{-1} y - \frac{1}{2} \log |K + \sigma_n^2 I| - \frac{n}{2} \log 2\pi \end{aligned} \tag{6}$$

when we set a Gaussian prior to $f|X$ as this leads to $y \sim N(0, K + \sigma_n^2 I)$.

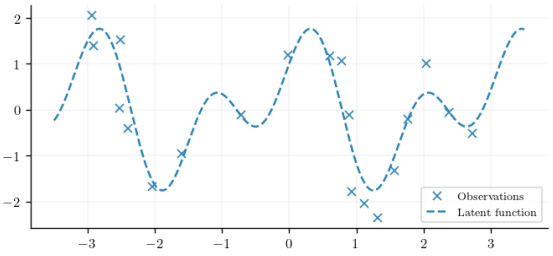

Figure 5: Latent Function of Gaussian Process Regression with 20 data points and noise, using GPJax package of Pinder and Dodd (2022).

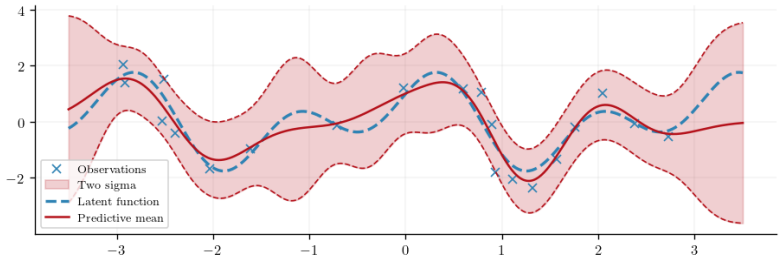

Figure 6: Posterior of Gaussian Process Regression with 20 data points and noise, using GPJax package of Pinder and Dodd (2022).

2.4 Covariance Function

In this section, we will explore more concretely the covariance functions of a Gaussian process. The two degrees of freedom of a Gaussian process are the mean function and the covariance function, and we often set the mean function to be zero, so the only real variability of a GP is the covariance function. Different choices of the covariance function will certainly lead to very different GPs. In this section, we will first outline some general properties and definitions related to covariance functions, then study a few commonly used covariance functions in detail.

2.4.1 Definitions and General Properties

As mentioned in Section 2.1, a GP $y(\cdot)$ is a stochastic process with the mean function $\mu(\cdot)$ and the covariance function $k(\cdot,\cdot)$, such that any finite points of the GP will form a multivariate Gaussian distribution. Because of this requirement, the covariance function needs to be (1) **symmetric**, i.e. $k(a,b) = k(b,a)$ (2) **positive semi-definite**, i.e.

$$\int k(x,x')f(x)f(x')d\mu(x)d\mu(x') \geq 0$$

for all $f \in L_2(X,\mu)$ where μ is some base measure and X is the support of the GP. Such a covariance function k will also be called a **kernel**, due to its link with the theory of integral operators.

For a set of points $x = \{x_i\}_{i=1}^n$, we can compute its **Gram matrix** K using the kernel k such that $K \in \mathbb{R}^{n\times n}$ and $K_{ij} = k(x_i, x_j)$. The Gram matrix in the context of GP will be used as the covariance matrix for the joint distribution of the points $x = \{x_i\}_i$.

One can view the kernel as a way to measure the similarity between two points. Since we would wish two points x, x' close to each other to be very similar - so highly dependent - in order to achieve some degrees of smoothness and the regularities of the overall GP, the quantities $x - x'$ and $\|x - x'\|$ would be of major importance. A covariance function that can be defined as a function of $x - x'$ is called **stationary (in the wide sense)**, or wide-sense stationarity (WSS), as it will be invariant to translations in the input space/support. A covariance function that can be defined as a function of $\|x - x'\|$ is called **isotropic** as it will be invariant under all rigid motions.

Finally, we can compose existing kernels to get new kernels. The sum of kernels is a kernel, and the product of kernels is also a kernel. These results follow naturally by checking the positive definiteness of the constructed functions, and the proof for these results can be found in Section 4.2.4 of Williams and Rasmussen (2006).

2.4.2 Examples of Covariance Functions

Two examples of covariance functions will be introduced here - the **squared exponential** (SE) covariance function and the **Matérn class** covariance function.

The **squared exponential** (SE) covariance function k_{SE} is defined by

$$k_{\text{SE}}(x,x') = \exp\left[-\frac{\|x-x'\|^2}{2l^2}\right] =: \exp\left[-\frac{r^2}{2l^2}\right] = k_{\text{SE}}(r)$$

where we define $r := \|x - x'\|$ and $l > 0$ is the **length-scale** of the kernel. From the definition, it is straightforward to notice that the SE kernel is stationary and isotropic, and the value of the length-scale characterises the degree of similarity between two nearby points - the higher the l, the more dependent two nearby points become. One should realise by the expression of k_{SE} that it is more of an exponentiated quadratic than a squared exponential, therefore some authors will denote the same kernel as the **exponentiated quadratic** kernel.

Since the SE kernel is defined by an exponential function, it is therefore infinitely differentiable (or smooth).

The **Matérn** class kernels is defined by

$$\begin{aligned} k_{\text{Matérn}}(x, x') &= \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}\|x - x'\|}{l} \right)^{\nu} K_{\nu} \left(\frac{\sqrt{2\nu}\|x - x'\|}{l} \right) \\ &=: \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}r}{l} \right)^{\nu} K_{\nu} \left(\frac{\sqrt{2\nu}}{l} r \right) = k_{\text{Matérn}}(r) \end{aligned}$$

where we define $r := \|x - x'\|$, $l > 0$ is the **length-scale** of the kernel, Γ is the Gamma function, $\nu > 0$ is the smoothness parameter of the kernel, and K_ν is the modified Bessel function of the second kind. The smoothness parameter ν usually is chosen to be half integers, as $\nu = p + 1/2$ for non-negative integer p. For example, we have the following three examples of ν:

$$\begin{aligned} k^{\nu=1/2}_{\text{Matérn}}(r) &= \exp\left(-\frac{r}{l}\right) \\ k^{\nu=3/2}_{\text{Matérn}}(r) &= \left[1 + \frac{\sqrt{3}r}{l}\right] \exp\left(-\frac{\sqrt{3}r}{l}\right) \\ k^{\nu=5/2}_{\text{Matérn}}(r) &= \left[1 + \frac{\sqrt{5}r}{l} + \frac{5r^2}{3l^2}\right] \exp\left(-\frac{\sqrt{5}r}{l}\right) \end{aligned}$$

and we can also show that using the definition, as $\nu \to \infty$, $k_{\text{Matérn}}(r) \to k_{\text{SE}}(r)$.

2.4.3 Hyperparameters Tuning

In Section 2.3, we have focused on how one can do curve fitting using GPs with a fixed, predetermined kernel. However, as we have seen in earlier parts of this section, there are multiple choices for kernels, and each kernel also depends on tuning hyperparameters that will influence the GP. Therefore, in practice, one should really consider the problem of hyperparameter tuning, which we will study here.

Assuming that we have fixed the choice of kernel, we then need to worry about how to tune the hyperparameters. We will treat the hyperparameters as additional parameters of the overall GP model while fitting the GP during regression. If one wishes to do regression using maximum likelihood, then it is quite straightforward - just pick the values for the hyperparameters (and the weight vector for covariates) that maximise the joint likelihood function using all the data. The log-likelihood is then provided as

$$\log p(y|X) = -\frac{1}{2} y^T (K + \sigma_n^2 I)^{-1} y - \frac{1}{2} \log |K + \sigma_n^2 I| - \frac{n}{2} \log 2\pi$$

where X is the prior model, y is the vector of n observations and σ_n^2 is the observation noise, and K is the Gram matrix of the observations y. This form of log-likelihood assumes the observations are independent and identically distributed.

If one wishes to do regression using a Bayesian approach, then one would need to pose some prior on the weight vector, as well as the hyperparameters of the kernel, then compute the posterior distribution of all the parameters of interest and do the estimation using some summary statistics of the posterior, computed/estimated using conjugacy or Monte Carlo methods (such as MCMC).

Chapter 3

Common Acquisition Functions

To recall from the previous sections, in a Bayesian optimisation iteration, we will first find the next evaluation according to some policy (here as we are taking the acquisition function approach, we will be using the maximiser of the acquisition function as the next evaluation point), evaluate it and append it to the dataset, and refit our surrogate model.

When designing acquisition functions, it is the bare minimum that the maximisation of an acquisition function should be cheaper than evaluating a point from the objective function - or else the purpose of doing BO will be defeated. Notice that the maximisation of the acquisition function is a maximisation task embedded in one iteration of Bayesian optimisation, we will often denote this as the **inner-loop maximisation**.

The following derivations are done with the additional assumptions that the observations are **without noise** - i.e. we are making exact observations $f(x)$.

We divide the acquisition functions mentioned in this chapter into three main classes:

1. improvement-based (probability of improvement, expected improvement, knowledge gradient), Section 3.1
2. bandit-based (upper confidence bound, Thompson sampling), Section 3.2
3. information-theoretic (entropy search, predictive entropy search, max-value entropy search), Section 3.3.

With each acquisition choice, we will provide a numerical example of running Bayesian optimisation with that particular choice of acquisition function. The function that we wish to maximise is

$$f(x) = \sin(5x) + \cos(8x + 3)$$

over the closed interval $x \in [0, 2]$. We first make 3 randomly drawn observations. The observations here are all with additive Gaussian noise of standard deviation 0.1. Afterwards, we will run the Bayesian optimisation 6 times and will obtain 6 further observations (with noise).

3.1 Improvement-Based Acquisitions

3.1.1 Probability of Improvement

The first choice of an acquisition function is the **probability of improvement**.

Consider we have already evaluated n data points and our current dataset is $\mathcal{D}_n = \{(x_k, y_k)\}_{k=1}^n$. With $\mathcal{D}_n$, we have also fitted a surrogate model using GP with the prior model being $g \sim \mathrm{GP}(\mu, k)$ and the posterior after observing $\mathcal{D}_n$ as $g_n = g|\mathcal{D}_n \sim \mathrm{GP}(\mu_n, k_n)$. Also, we denote the largest (as we are maximising) observation value so far as $y_n^* = \max_{m \leq n} y_m$.

Given our current surrogate model g_n, the probability of improvement wishes to find the next evaluation point that beats our current best (i.e. y_n^*) with the highest probability. Therefore, we can define a utility function

$$u_n^{\text{PI}}(x) = \begin{cases} 1 & \text{if } g_n(x) \geq y_n^* \\ 0 & \text{else} \end{cases}$$

and a subsequent acquisition function

$$\begin{aligned} \alpha_n^{\text{PI}}(x) &= \mathbb{E}[u_n^{\text{PI}}(x)|\mathcal{D}_n] \\ &= \mathbb{P}[g_n(x) \geq y_n^*] \\ &= \Phi\left(\frac{y_n^* - \mu_n(x)}{k_n(x,x)}\right) \end{aligned} \tag{7}$$

where Φ is the CDF of standard normal, since our surrogate model $g_n \sim \text{GP}(\mu_n, k_n)$ evaluated at the point x collapses to a normal distribution and $g_n(x) \sim N(\mu_n(x), k_n(x,x))$. This acquisition function is not hard to compute point-wise, and it certainly captures some notions of improvements, thus it is a sensible choice - not a bad starting point.

3.1.2 Expected Improvement

Recall that when deriving the probability of improvement acquisition function, we defined a utility function that takes 1 when an improvement happens and 0 otherwise. This does not account for the magnitude of improvement - surely a 80% chance of improving by 10 is superior to a 81% chance of improving by 5. So, with the new utility function for the **expected improvement**, we will account for the magnitude of the improvement too. So we have a utility function

$$u_n^{\text{EI}}(x) = \begin{cases} g_n(x) - y_n^* & \text{if } g_n(x) \geq y_n^* \\ 0 & \text{else} \end{cases}$$

and a subsequent acquisition function

$$\begin{aligned} \alpha_n^{\text{EI}}(x) &= \mathbb{E}[u_n^{\text{EI}}(x)|\mathcal{D}_n] = \mathbb{E}_{g_n}[(X - y_n^*)1_{X \geq y_n^*}] \\ &= \int_{y_n^*}^{\infty} (x - y_n^*)\phi_n(x)dx = \int_{y_n^*}^{\infty} x\phi_n(x)dx - y_n^* \int_{y_n^*}^{\infty} \phi_n(x)dx \\ &= \mu_n(x)\Phi\left(\frac{y_n^* - \mu_n(x)}{k_n(x,x)}\right) + k_n(x,x)\phi\left(-\frac{y_n^* - \mu_n(x)}{k_n(x,x)}\right) - y_n^*\Phi\left(\frac{y_n^* - \mu_n(x)}{k_n(x,x)}\right) \\ &= [\mu_n(x) - y_n^*]\Phi\left(\frac{y_n^* - \mu_n(x)}{k_n(x,x)}\right) + k_n(x,x)\phi\left(-\frac{y_n^* - \mu_n(x)}{k_n(x,x)}\right) \end{aligned} \tag{8}$$

where 1_A is the indicator function for event A, $\phi_n(x)$ denotes the density of g_n and ϕ denotes the density of standard normal.

It is still feasible to compute the above acquisition point-wise, so it is still a sensible choice. Referencing the exploration-exploitation trade-off, one can view the two terms of the expected improvement acquisition function as the exploitation term (the first term of Equation 8) and the exploration term (the second term of Equation 8). The first term favours the locations that give a high chance and high magnitude of improvement, while the second term favours the locations with high uncertainties.

3.1.3 Knowledge Gradient

One of the shared assumptions in the probability of improvement and the expected improvement is that when choosing the output point, we are restricted to the points that we have evaluated so far, i.e. $y_n^* = \max_{m \leq n} y_m$. This is certainly not necessary, as we could set the points of consideration as the entire input space, and output the point with the largest mean, so $y_n^* = \max_{x \in X} \mu_n(x)$.

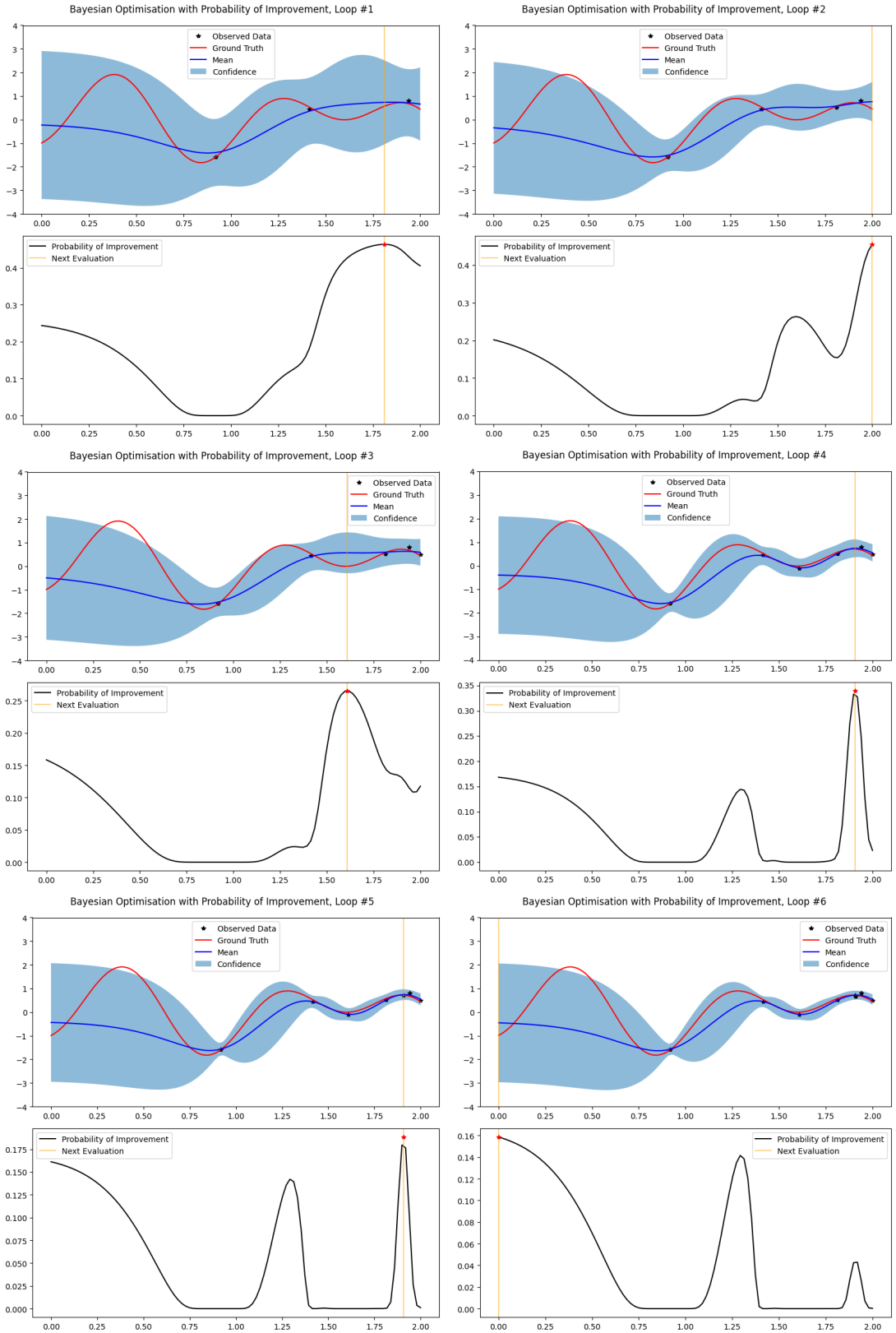

Figure 7: Probability of Improvement

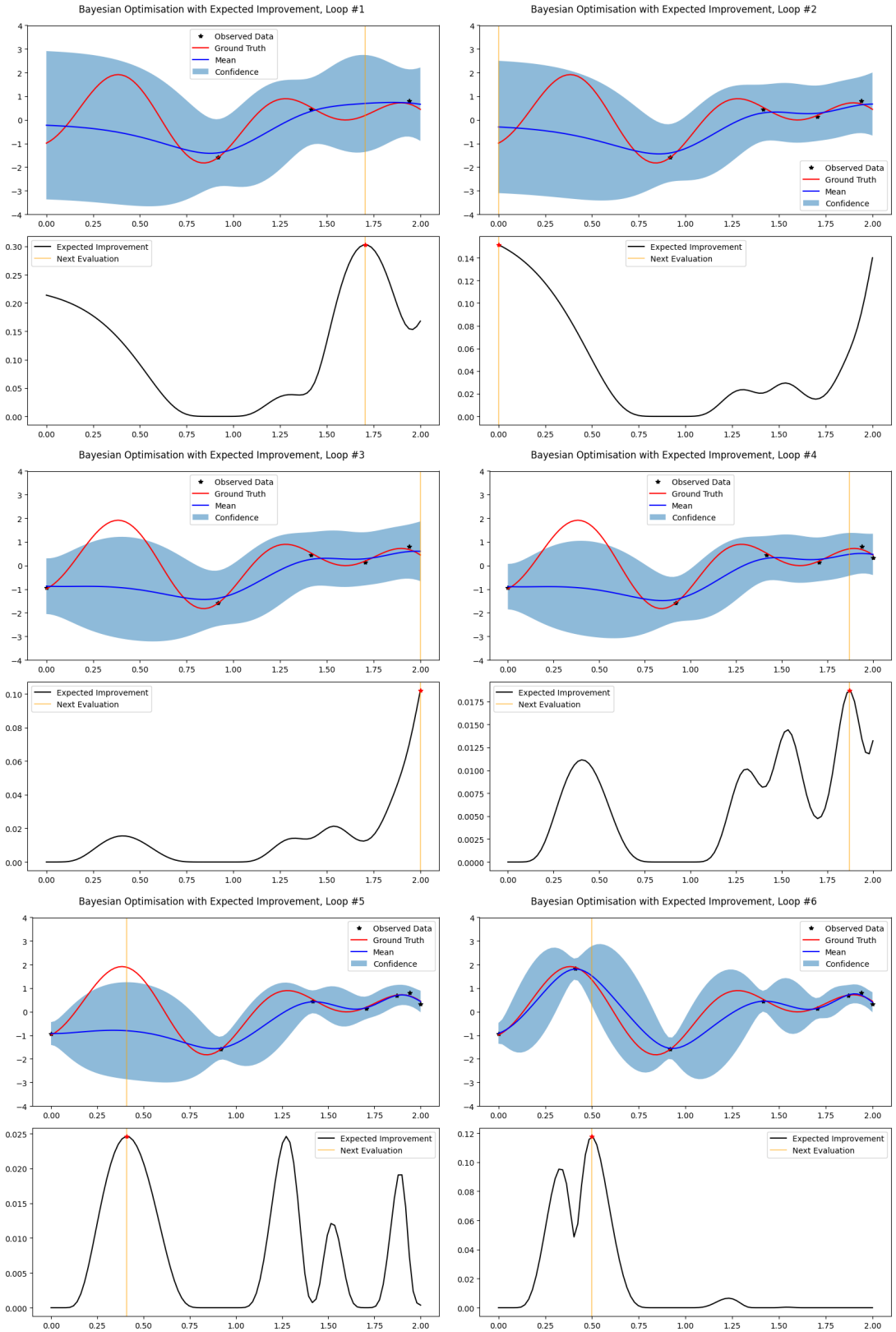

Figure 8: Expected Improvement

As a consequence of this adjustment, we will incorporate the new evaluation into our surrogate model, figure out the updated mean function μ_{n+1}, and compare the maximum of that with our current best y_n^*. This is the **knowledge gradient**.

If we denote $\mu_{n+1}^*(x)$ as the largest mean of the surrogate model after incorporating the new value $(x, f(x))$, i.e. $\mu_{n+1}^*(x) = \max_{y \in X} \mu_{n+1}^x(y)$ with $\mu_{n+1}^x(\cdot)$ being the mean function of the updated surrogate $g|\mathcal{D}_n \cup \{(x, f(x)\}$, then the improvement we will make after observing x to our overall result of the BO iteration is

$$u_n^{\mathrm{KG}}(x) = \mu_{n+1}^*(x) - \mu_n^*,$$

which is the utility function for the knowledge gradient. Notice that we will not know $\mu_{n+1}^*(x)$ - since it depends on evaluating at x - at time n without doing evaluations, we will need to maximise the expected value of the utility function, which is the knowledge gradient acquisition function

$$\alpha_n^{\mathrm{KG}}(x) = \mathbb{E}[u_n^{\mathrm{KG}}(x)|\mathcal{D}_n] = \mathbb{E}_{g_n}[\mu_{n+1}^*(x) - \mu_n^*]. \tag{9}$$

A major obstacle with the knowledge gradient is the computation (and maximisation) of the acquisition function α_n^{KG} as there lacks a general closed-formed expression. In the discrete domain (i.e. the size of the input space X is finite), the knowledge gradient acquisition function can be computed explicitly. In the general domain, except for certain nice GP models, we do not have a nice expression and therefore must resort to numerical approximations.

add details of numerical approximations

3.2 Bandit-Based Acquisitions

The **multi-armed bandit** (MAB) is a sequential decision problem. The problem assumes that we have N arms that we can pull which then provides a reward instantly after paying some price for pulling the arm, and we can choose which arm to toggle at each time step. The goal is to devise a policy of pulling the arms so that we obtain a maximum overall reward at a terminal time point. The rewards generated by an arm are not (always) deterministic, but it is commonly assumed to follow a fixed probability distribution. If the rewards are deterministic, the problem will be easy - just pull each of the N arms once, get the rewards, and only pull the one that yields maximum rewards afterwards.

The immediate application of a multi-armed bandit is the slot machine in casinos, where each arm is a button of the machine, and we wish to maximise our overall reward (gains in cash). There are many other problems that can be phrased as an MAB problem, such as the patient allocation of a clinical trial in the response-adaptive randomisation setting. In general, the MAB is a special case of an **reinforcement learning** problem where an agent is interacting with an environment and obtaining feedback which in turn guides the followed-up behaviours of the agent.

It is not too much of a stretch to notice the similarity between an MAB and a Bayesian optimisation problem - both are making decisions sequentially, both obtain a reward after the action and use that to decide on what to do afterwards, and both have an objective function that we wish to optimise. There are of course differences, such as the MAB has only N arms thus finite N options at each time, whereas the BO can evaluate at any point in the input space X, which is often infinite, such as $\mathbb{R}$. Nevertheless, there are sufficient motivations for us to borrow ideas from MAB and apply them to BO, as the two acquisition functions below will illustrate.

3.2.1 Upper Confidence Bound

The idea of **upper confidence bound** is very simple, and it explicitly encodes the exploration-exploitation trade-off. For each point x in the search space $S = X$, we know two things about it from the surrogate model $g_n = g|\mathcal{D}_n$: its mean $\mu_n(x)$ and its variance $k_n(x, x)$(thus standard deviation $\sqrt{k_n(x, x)}$). We want to exploit places with high mean (which could promise gains) and

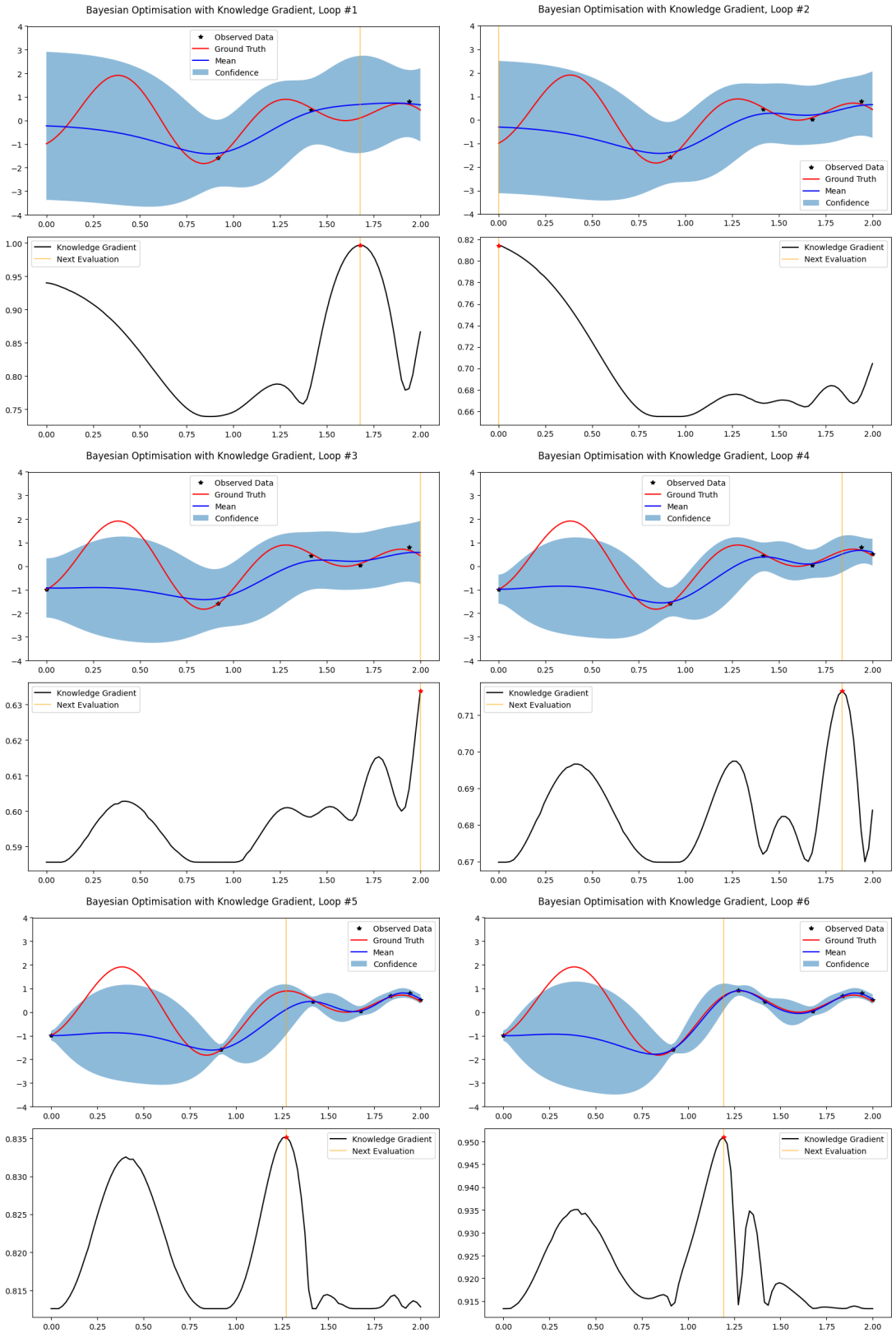

Figure 9: Knowledge Gradient

explore places with high uncertainties (which could lead to surprising gains). Thus, we have the following upper confidence bound acquisition function

$$\alpha_n^{\text{UCB}}(x) = \mu_n(x) + \beta\sqrt{k_n(x,x)} \tag{10}$$

where $\beta = \Phi^{-1}(c)$ is the tuning parameter balancing exploration-exploitation, obtained via the inverse standard normal CDF Φ^{-1} at confidence level $c \in (0,1)$. For example, $\Phi^{-1}(0.5) = 0.5$.

Many places treat β as a mere tuning parameter, but it is actually more informative than that and is indeed allowing our acquisition function to be about upper confidence bound. For a point x, its distribution under our surrogate model g_n is $N(\mu_n(x), k_n(x,x))$, so its $100c\%$-quantile is given by $\mu_n(x) + \Phi^{-1}(c)\sqrt{k_n(x,x)}$, precisely that of $\alpha_n^{\text{UCB}}(x)$.

show example

3.2.2 Thompson Sampling

The idea of Thompson sampling (applied to BO) goes like this: given a surrogate model g_n, we build a probability distribution $p(x^*|\mathcal{D}_n)$ that describes our knowledge about the location of global maximum x^*, and draw a sample from it as our next evaluation point. So, ideally, the distribution $p(x^*|\mathcal{D}_n)$ will put a high probability in regions of the input space where it thinks the global maximum will lie, and a low probability in regions where it thinks otherwise.

Constructing such a probability distribution of the location of the global maximum is the key challenge of this approach. The most common way to tackle it is via empirical distribution. One will draw a sample function g_n^i from g_n, find its maximum $x_i^* = \arg\max_{x\in X} g_n^i(x)$, and treat it as a sample from $p(x^*|\mathcal{D}_n)$. This is repeated M times, and we get $x_1^*, x_2^*, \ldots, x_M^*$, which forms an empirical distribution $\hat{p}(x^*|\mathcal{D}_n)$ that approximates $p(x^*|\mathcal{D}_n)$. The next evaluation point is therefore drawn from $\hat{p}(x^*|\mathcal{D}_n)$. Depending on the size of M, each iteration of BO using Thompson sampling could be costly.

show example

3.3 Information-Theoretic Acquisitions

Both classes of approaches of Sections 3.1 and 3.2 design acquisition functions with the goal of maximising the magnitude of improvement in our maximiser of the objective function. Although some interests in uncertainties exist via the exploration components of some of the acquisition functions, none treats reducing the uncertainties (thus increasing the information) of the overall maximiser as the goal. There could be benefits in taking that approach, and the information-theoretic acquisition functions take precisely that path.

For a random variable A with density function $p(a)$, its **(differential) entropy** is defined as

$$H(A) = -\mathbb{E}[\log p(A)] = -\int p(a)\log p(a)da.$$

Similarly, for two random variables A, B with joint density $p(a,b)$, we can define their **joint entropy** as the entropy of the bivariate random variable, i.e.

$$H(A,B) = -\mathbb{E}[\log p(A,B)] = -\iint p(a,b)\log p(a,b)dadb.$$

If A and B are independent, we have $p(a,b) = p(a)p(b)$ for any a, b, and we would get

$$\begin{aligned}
H(A,B) &= -\iint p(a)p(b)[\log p(a) + \log p(b)]dadb \\
&= -\iint p(a)p(b)\log p(a)dadb - \iint p(a)p(b)\log p(b)dadb \\
&= -\int p(a)\log p(a)da - \int p(b)\log p(b)db \\
&= H(A) + H(B).
\end{aligned}$$

When A, B are dependent, so $p(a|b) \neq p(a)$, we uses $p(a,b) = p(a|b)p(b)$ to get

$$\begin{aligned} H(A,B) &= -\iint p(a|b)p(b)[\log p(a|b) + \log p(b)]dadb \\ &= -\iint p(a|b)p(b)\log p(a|b)dadb - \iint p(a|b)p(b)\log p(b)dadb \\ &= -\iint p(a,b)\log p(a|b)dadb - \int p(b)\log p(b)db \\ &=: H(A|B) + H(B). \end{aligned}$$

where we define the **conditional (differential) entropy** $H(A|B)$ as

$$H(A|B) = -\mathbb{E}_{A,B}[\log p(A|B)] = -\iint p(a|b)\log p(a|b)dadb = H(A,B) - H(B).$$

When A and B are strongly correlated, so learning about B is almost like learning about A, the joint entropy $H(A,B)$ will be close to $H(B)$ and the conditional entropy $H(A|B)$ will be small.

Closely linked to conditional entropy is the notion of **mutual information** between two random variables A, B, which is defined as

$$MI(A,B) = \iint p(a,b)\log\frac{p(a,b)}{p(a)p(b)}dadb.$$

We can realise first that

$$MI(A,B) = H(A) + H(B) - H(A,B) = H(A) - H(A|B) = H(B) - H(B|A)$$

and second that

$$\begin{aligned} MI(A,B) &= \iint p(a,b)\log p(a|b)dadb - \iint p(a,b)\log p(a)dadb \\ &= \int_B\int_A p(a|b)\log p(a|b)dap(b)db + H(A) \\ &= H(A) - \int_B\left[-\int_A p(a|b)\log p(a|b)da\right]p(b)db \\ &= H(A) - \mathbb{E}_B[H(A|B)] = H(B) - \mathbb{E}_A[H(B|A)] \end{aligned}$$

where the last equality follows from symmetry.

To illustrate the above concepts, we will consider the bivariate normal distribution

$$\begin{bmatrix} A \\ B \end{bmatrix} \sim N_2\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix}\right).$$

Here $\rho \in [-1,1]$ is the correlation parameter. When $\rho = 0$, A and B are independent. When ρ is large in magnitude, A and B are strongly correlated. It can be derived that the entropy of a normal distribution $X \sim N(\mu, \sigma^2)$ is

$$H(X) = \frac{1}{2} + \frac{1}{2}\log(2\pi\sigma^2),$$

while the entropy of a bivariate normal $\boldsymbol{X}$ with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ is

$$H(\boldsymbol{X}) = 1 + \log(2\pi) + \frac{1}{2}\log|\boldsymbol{\Sigma}|$$

where $|\cdot|$ is the determinant of a matrix. In our case, we have

$$H(A,B) = 1 + \log(2\pi) + \frac{1}{2}\log(1-\rho^2)$$

and

$$H(A) = H(B) = \frac{1}{2} + \frac{1}{2}\log(2\pi).$$

This means the conditional entropy becomes

$$H(A|B) = H(B|A) = \frac{1}{2} + \frac{1}{2}\log(2\pi) + \frac{1}{2}\log(1-\rho^2)$$

and the mutual information becomes

$$MI(A,B) = -\frac{1}{2}\log(1-\rho^2).$$

It is now very easy to see how these quantities change as we change the correlation ρ. For tiny (in magnitude) ρ, we have relatively independent variables, and the mutual information is close to zero. For large (in magnitude) ρ, we have relatively dependent variables, and the mutual information is very large.

Finally, we will define the **relative entropy** of two random variables A, B with densities p_A and p_B respectively. The relative entropy, also known as the **Kullback-Leibler divergence** or KL divergence, is defined as

$$\mathrm{KL}(p_A \| p_B) = \mathbb{E}_A[\log p_A/p_B] = \int p_A(x) \log \frac{p_A(x)}{p_B(x)} dx.$$

Using this concept, we can also interpret the mutual information as the KL divergence between the independent coupling $A \otimes B$ and the actual joint distribution. This also makes immediate sense of why the mutual information is close to zero when ρ is tiny in magnitude.

These notions of information and entropies will be used in the following acquisition functions.

3.3.1 Entropy Search and Predictive Entropy Search

The main idea of information-theoretic search strategies is that one wishes to consider the moves that maximise our knowledge (and minimise the uncertainty) about the target object. The target object could be about the input space, i.e. the input space location of the maximiser $x^* = \arg\max_{x \in X} f(x)$, or about the output space, i.e. the maximum of the objective function $f(x^*)$. We will first look at the input space versions, and consider the output space versions below.

Consider our surrogate model g_n. We wish to minimise our uncertainty about the maximiser x^* using a new observation y_x which is $f(x)$ when the observation is exact. We use $H(x^*|\mathcal{D}_n)$ to denote the entropy of the distribution of x^* given data $\mathcal{D}_n$. With a new observation y_x evaluated at x, the entropy of the maximiser becomes $H(x^*|y_x, \mathcal{D}_n)$.

Quite naturally, we can define the utility function of **entropy search** as the difference in the two entropy above, i.e.

$$u_n^{\mathrm{ES}}(x) = H(x^*|\mathcal{D}_n) - H(x^*|y_x, \mathcal{D}_n)$$

and the subsequent acquisition function of entropy search becomes

$$\alpha_n^{\mathrm{ES}}(x) = H(x^*|\mathcal{D}_n) - \mathbb{E}_x[H(x^*|y_x, \mathcal{D}_n)|\mathcal{D}_n] = MI(x^*, y_x|\mathcal{D}_n) \tag{11}$$

where the last equality follows from the definition of mutual information.

Before discussing how one might actually compute the above acquisition function, we realise that by the symmetry of mutual information, we have

$$MI(x^*, y_x|\mathcal{D}_n) = H(y_x|\mathcal{D}_n) - \mathbb{E}_{x^*}[H(y_x|x^*, \mathcal{D}_n)|\mathcal{D}_n] =: \alpha_n^{\mathrm{PES}}(x) \tag{12}$$

which becomes the acquisition function of **predictive entropy search**. This acquisition function could often be easier to find than entropy search.

add details of numerical approximations

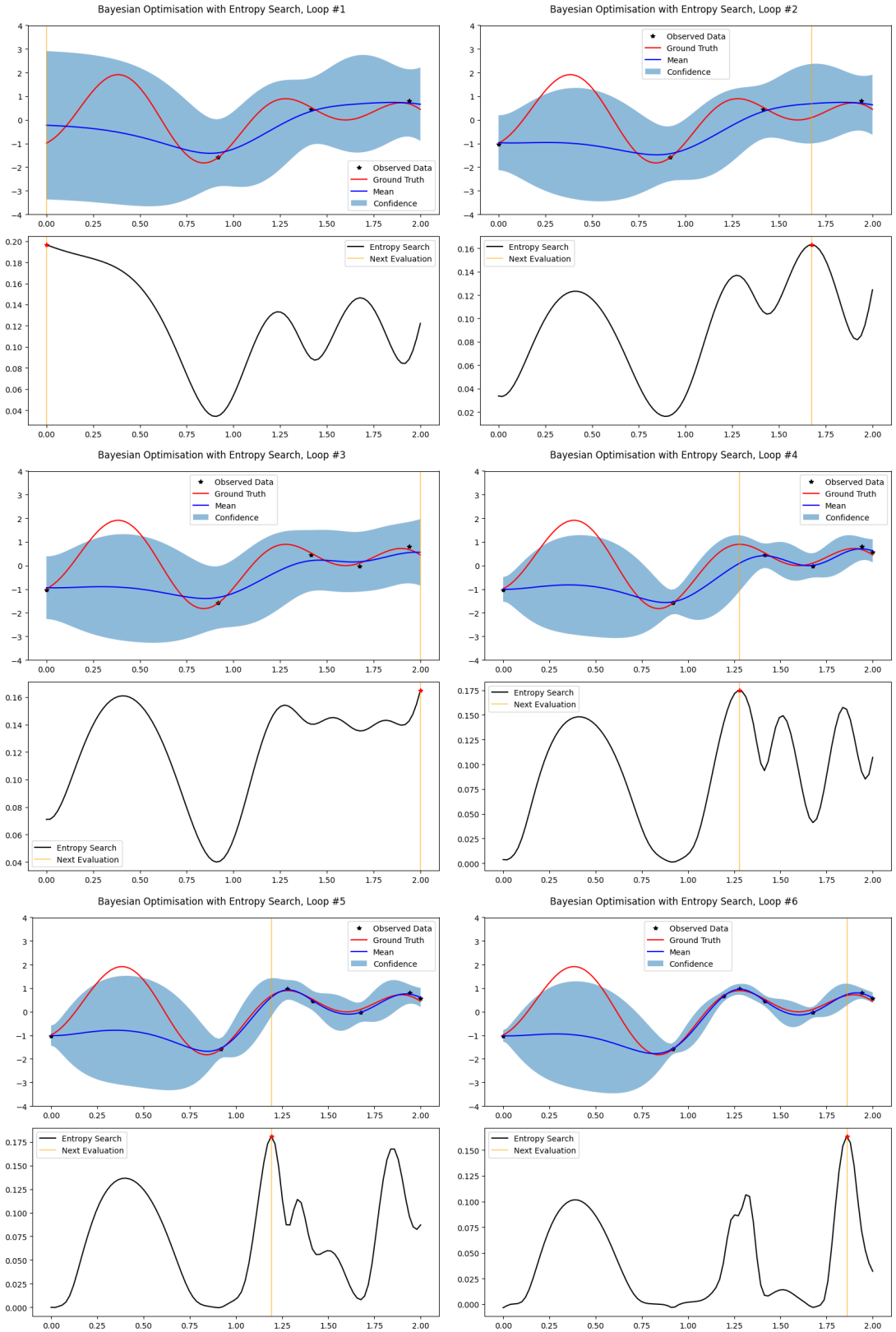

Figure 10: Entropy Search

3.3.2 Max-Value Entropy Search

As explained previously, we can also consider doing an entropy search in the output space. This makes sense - sometimes when maximising, we do not care about the maximiser but only the maximum value of the objective function.

Instead of comparing the entropy of the distribution of the maximiser, we will consider the entropy of the distribution of the global maximum $f^* = \max f(x)$. This gives us the following acquisition function for **max-value entropy search**, which uses the formulation of predictive entropy search,

$$\alpha_n^{\text{MES}}(x) = MI(y_x, f^* | \mathcal{D}_n) = H(y_x|\mathcal{D}_n) - \mathbb{E}_{f^*}[H(y_x|f^*, \mathcal{D}_n)|\mathcal{D}_n]. \tag{13}$$

add details of numerical approximations

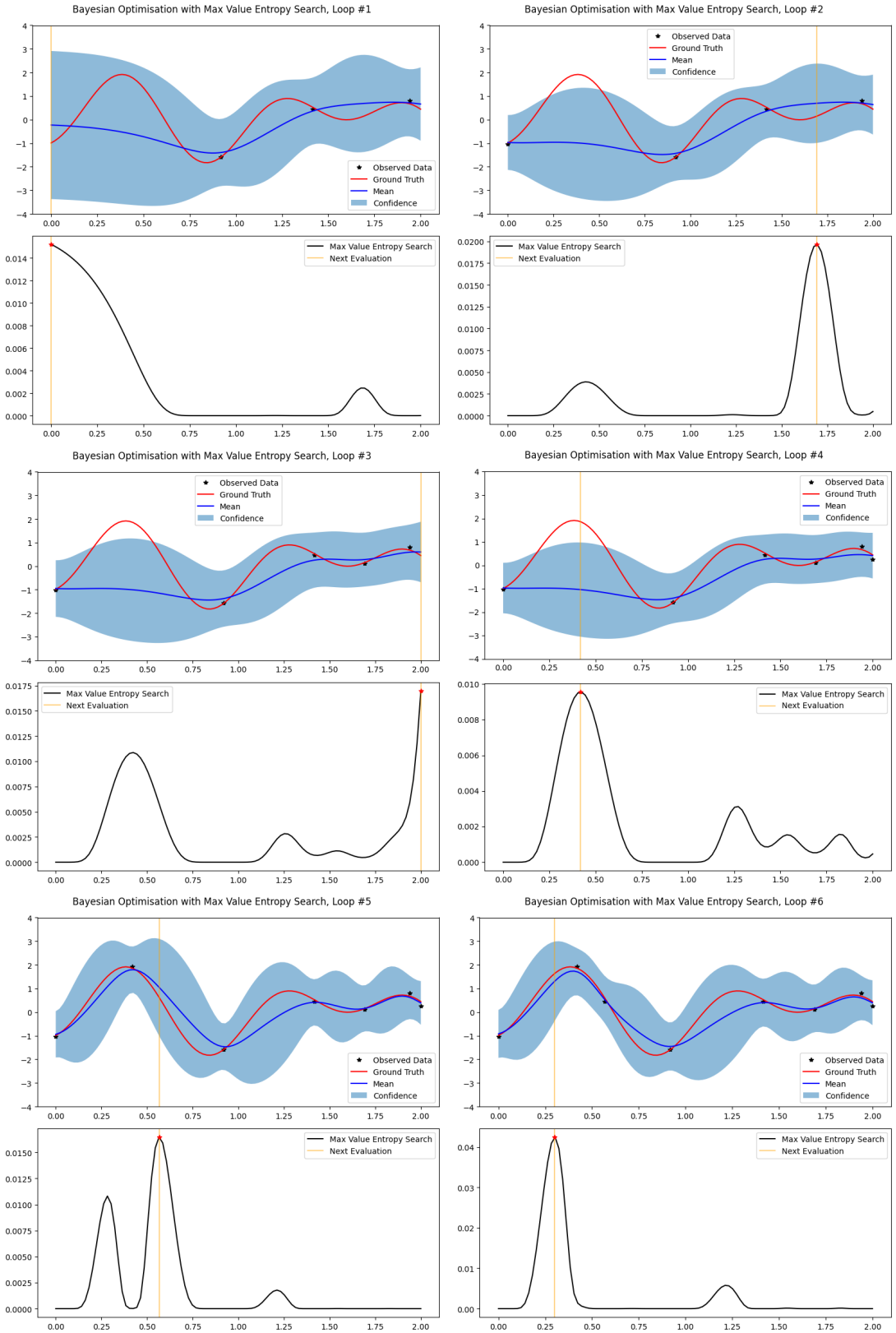

Figure 11: Max Value Entropy Search

Bibliography

Balandat, M., Karrer, B., Jiang, D. R., Daulton, S., Letham, B., Wilson, A. G. and Bakshy, E. (2020). BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization, *Advances in Neural Information Processing Systems 33*.
URL: *http://arxiv.org/abs/1910.06403*

Pinder, T. and Dodd, D. (2022). GPJax: A Gaussian Process Framework in JAX, *Journal of Open Source Software* **7**(75): 4455.

Williams, C. K. and Rasmussen, C. E. (2006). *Gaussian Processes for Machine Learning*, Vol. 2, MIT press Cambridge, MA.