Probability Density Functions

A random variable X on a discrete space is well-defined if

$$\sum_{x \in \mathcal{X}} P(X = x) = 1. \tag{1}$$

If the state space $\mathcal{X}$ is not discrete, for e.g. $\mathcal{X} = \mathbb{R}$ or $\mathbb{R}^n$, then a continuous random variable X is well-defined if there exists a probability density function (pdf) $f_X(x) \geq 0$ such that

$$\int_{\mathcal{X}} f_X(x)\,\mathrm{d}x = 1. \tag{2}$$

Its cumulative distribution function (cdf)

$$P(X \leq a) = \int_{-\infty}^{a} f_X(x)\,\mathrm{d}x \tag{3}$$

is a function of a, and is also denoted by $F(a)$.

Joint Distributions

A multivariate random variable $\boldsymbol{X} = (X_1, \ldots, X_n)$ with state space $\mathcal{X}_1, \ldots, \mathcal{X}_n$ is a joint distribution if

$$\sum_{x_1 \in \mathcal{X}_1, \ldots, x_n \in \mathcal{X}_n} P(X_1 = x_1, \ldots, X_n = x_n) = 1, \tag{4}$$

for discrete random variables. For continuous random variables, there exists a density function $f_{X_1,\ldots,X_n}(x_1, \ldots, x_n) \geq 0$ such that

$$\int_{x_1 \in \mathcal{X}_1, \ldots, x_n \in \mathcal{X}_n} f_{X_1,\ldots,X_n}(x_1, \ldots, x_n) \, \mathrm{d}\mathbf{x} = 1. \tag{5}$$

Marginal Distributions

With the joint distribution probabilities, one can derive the distribution of each individual X_i, or a subset of them. These distributions are known as marginal distributions.

For discrete random variables, the multivariate random variable $(X_1, \ldots, X_{n-1})$ has the probability distribution

$$P(X_1 = x_1, \ldots, X_{n-1} = x_{n-1}) = \sum_{x_n \in \mathcal{X}_n} P(X_1 = x_1, \ldots, X_n = x_n), \quad (6)$$

while the random variable X_1 has the density function

$$P(X_1 = x_1) = \sum_{x_2 \in \mathcal{X}_2, \ldots, x_n \in \mathcal{X}_n} P(X_1 = x_1, \ldots, X_n = x_n). \quad (7)$$

For continuous random variables, the random variable X_1 has the density function

$$f_{X_1}(x_1) = \int_{x_2 \in \mathcal{X}_2, \ldots, x_n \in \mathcal{X}_n} f_{X_1, \ldots, X_n}(x_1, \ldots, x_n) \, \mathrm{d}x_2 \ldots \mathrm{d}x_n. \quad (8)$$

Conditional Distributions

Given a joint discrete distribution (X, Y), the conditional probability function of X given Y is given by

$$P(X = x \mid Y = y) = \frac{P(X = x, Y = y)}{P(Y = y)}. \tag{9}$$

When (X, Y) is continuous, the probability density function, $f_{X|Y}(x \mid y)$, of X given Y has the expression

$$f_{X|Y}(x \mid y) = \frac{f_{X,Y}(x, y)}{f_Y(y)}. \tag{10}$$

Thus, the conditional distributions can be computed from the joint distributions and the marginal distributions.

Gaussian processes for regression and optimization

Conditional Gaussian distributions

Let $x \in \mathbb{R}^{n+p}$ be a Gaussian random vector which we will partition as

$$x = \begin{bmatrix} x_a \\ x_b \end{bmatrix},$$

where $x_a \in \mathbb{R}^n$ and $x_b \in \mathbb{R}^p$. Then the means and covariances can be represented as

$$\mu = \begin{bmatrix} \mu_a \\ \mu_b \end{bmatrix}, \qquad \Sigma = \begin{bmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{bmatrix}. \tag{11}$$

Here, Σ_{aa} is $n \times n$, Σ_{ab} is $n \times p$, Σ_{ba} is $p \times n$ and Σ_{bb} is $p \times p$.

We will denote the precision matrix Σ^{-1} as

$$\Lambda = \Sigma^{-1} = \begin{bmatrix} \Lambda_{aa} & \Lambda_{ab} \\ \Lambda_{ba} & \Lambda_{bb} \end{bmatrix}.$$

Then we can expand the exponent of the Gaussian distribution as

$$\begin{aligned} &-\frac{1}{2}\left\langle (x-\mu), \Sigma^{-1}(x-\mu)\right\rangle \\ &\quad = -\frac{1}{2}\left\langle (x_a-\mu_a), \Lambda_{aa}(x_a-\mu_a)\right\rangle - \frac{1}{2}\left\langle (x_a-\mu_a), \Lambda_{ab}(x_b-\mu_b)\right\rangle \\ &\qquad -\frac{1}{2}\left\langle (x_b-\mu_b), \Lambda_{ba}(x_a-\mu_a)\right\rangle - \frac{1}{2}\left\langle (x_b-\mu_b), \Lambda_{bb}(x_b-\mu_b)\right\rangle . \end{aligned} \tag{12}$$

- To find the conditional distribution, we set x_b to be a constant in (12) and renormalize. The resulting distribution is still Gaussian, so all that remains is to find the conditional mean and conditional covariance.
- Using the fact that Λ_{aa} is symmetric and $\Lambda_{ba}^T = \Lambda_{ab}$, (12) can be rewritten as

$$-\frac{1}{2}\langle x_a, \Lambda_{aa} x_a\rangle + \langle x_a, \Lambda_{aa}\mu_a - \Lambda_{ab}(x_b - \mu_b)\rangle + \text{const}, \qquad (13)$$

  where the constant term does not contain x_a.

We know that the exponent of the conditional distribution must take the form

$$
\begin{aligned}
&-\frac{1}{2}\left\langle \left(x_a-\mu_{a|b}\right),\, \Sigma_{a|b}^{-1}\left(x_a-\mu_{a|b}\right)\right\rangle \\
&\qquad = -\frac{1}{2}\left\langle x_a,\, \Sigma_{a|b}^{-1}x_a\right\rangle + \left\langle x_a,\, \Sigma_{a|b}^{-1}\mu_{a|b}\right\rangle + \text{const},
\end{aligned}
\tag{14}
$$

so comparing (13) and (14), we must have

$$
\begin{aligned}
\Sigma_{a|b} &= \Lambda_{aa}^{-1}, \\
\mu_{a|b} &= \mu_a - \Lambda_{aa}^{-1}\Lambda_{ab}(x_b-\mu_b).
\end{aligned}
$$

HW problem

Given a partitioned matrix, the following formula holds

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}^{-1} = \begin{bmatrix} M & -MBD^{-1} \\ -D^{-1}CM & D^{-1} + D^{-1}CMBD^{-1} \end{bmatrix},$$

where $M = (A - BD^{-1}C)^{-1}$.
Finally, using this formula and definition of the precision matrix, we can express the conditional mean and variance as

$$\mu_{a|b} = \mu_a + \Sigma_{ab}\Sigma_{bb}^{-1}(x_b - \mu_b),$$
$$\Sigma_{a|b} = \Sigma_{aa} - \Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba}.$$

Marginal Gaussian distributions

Given $p(x_a, x_b)$ which is normally distributed with mean and covariance as in (11), the marginal distribution

$$p(x_a) = \int p(x_a, x_b)\, \mathrm{d}x_b$$

is also Gaussian with

$$\mathbb{E}[x_a] = \mu_a,$$
$$\mathrm{Cov}[x_a] = \Sigma_{aa}.$$

Gaussian processes

Definition

A stochastic process $\{X_t; t \in T\}$ is a Gaussian process if for any finite set of indices $\{t_1, \ldots, t_n\}$ of T, $(X_{t_1}, \ldots, X_{t_n})$ is a multivariate normal random variable.

- If T is an infinite set, eg. a subset of $\mathbb{R}^d$, then this is an infinite-dimensional generalization of multivariate normal random variables.
- A Gaussian process is completely determined by its
  - Mean function $\mu(\cdot) : T \to \mathbb{R}$
  - Covariance function or kernel $k(\cdot, \cdot) : T \times T \to \mathbb{R}$
- The samples of a Gaussian process are paths if $T = \mathbb{R}$, or surfaces if $T = \mathbb{R}^d$, $d > 1$, and their smoothness is dependent on the kernel.

Examples of kernels

- Radial basis function (RBF or "Gaussian"):

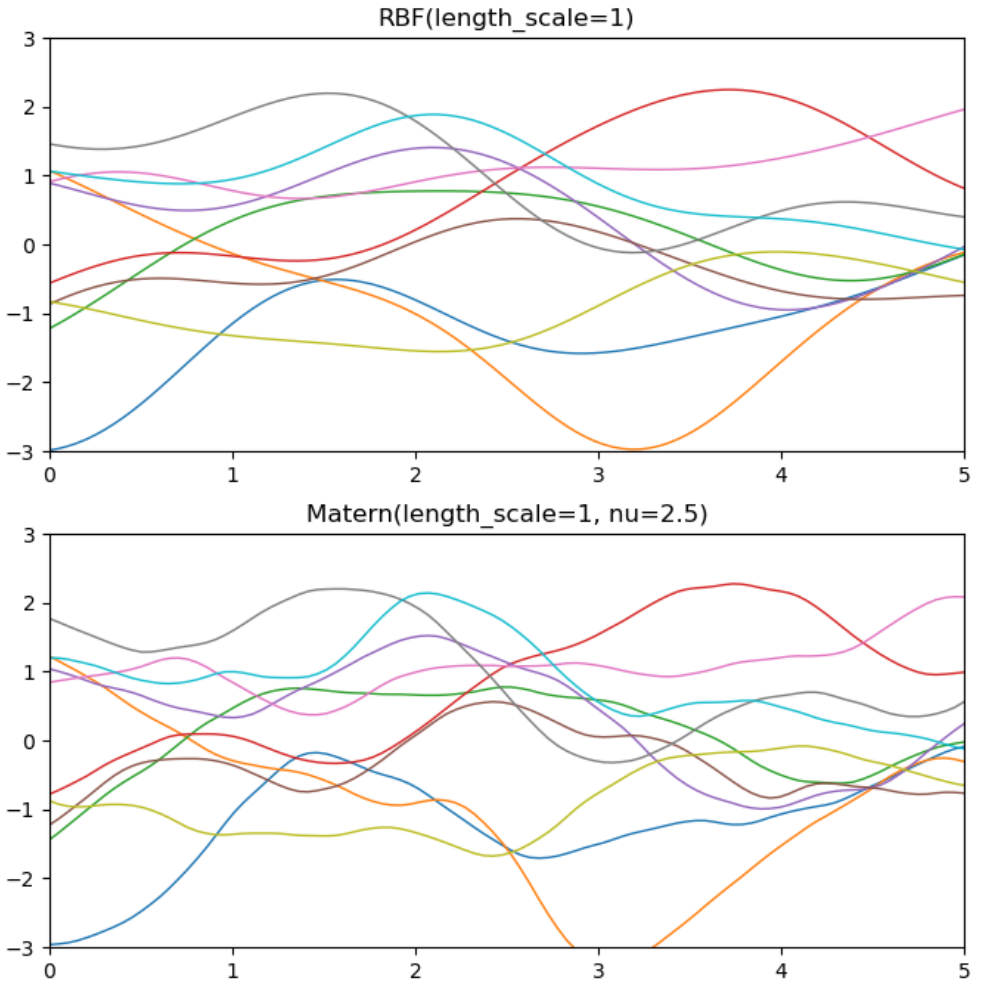
$$k(x,y) = e^{-\frac{\|x-y\|^2}{2\sigma^2}}, \quad x,y \in \mathbb{R}^d$$

- Matérn $\frac{5}{2}$ ($\frac{5}{2} = \nu + \frac{1}{2}$, $\nu = 2$):

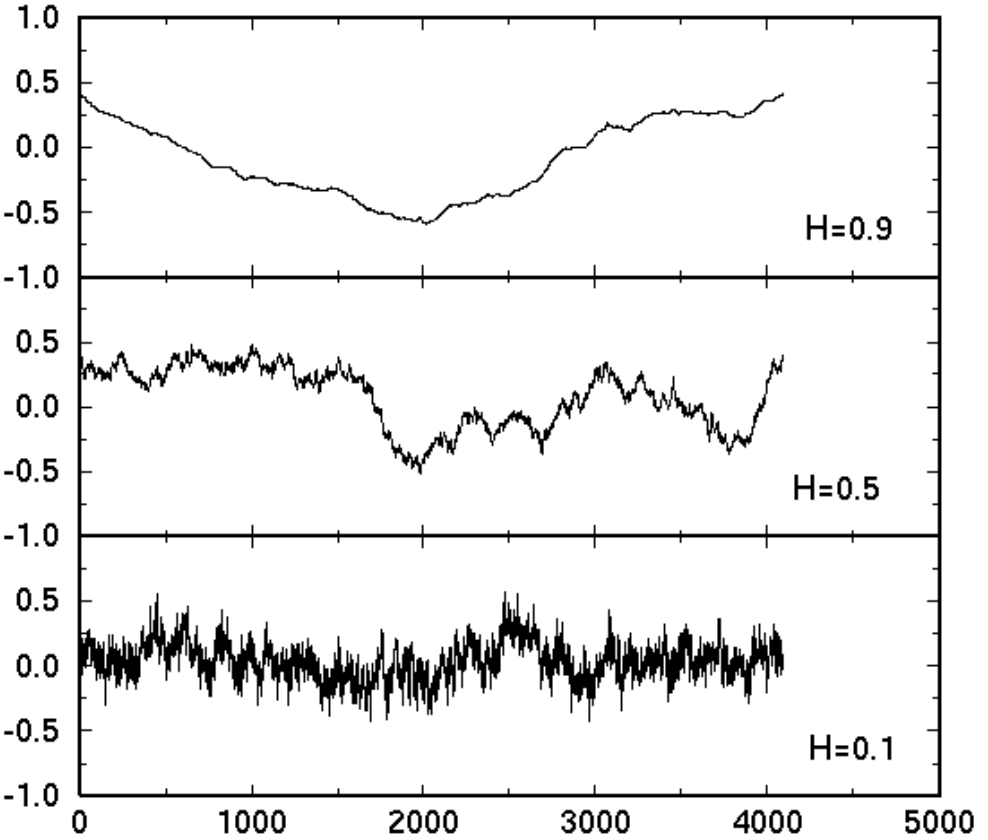
$$k(x,y) = \left(1 + \sqrt{5}\,\|x-y\| + \frac{5}{3}\,\|x-y\|^2\right) e^{-\sqrt{5}\|x-y\|}, \quad x,y \in \mathbb{R}^d$$

- Fractional Brownian motion, with Hurst parameter $H \in (0,1)$:

$$k(x,y) = \frac{1}{2}\left(x^{2H} + y^{2H} - |x-y|^{2H}\right), \quad x,y \in \mathbb{R}^+$$


RBF(length_scale=1)
Matern(length_scale=1, nu=2.5)
3
2
1
0
−1
−2
−3
0
1
2
3
4
5

Fractional Brownian motion



Gaussian processes for regression

- Recall that we model

$$y = f(x) + \epsilon, \tag{15}$$

  where we assume $\epsilon \sim \mathcal{N}(0, \tau^2)$ and is independent between samples.

- Previously, we have been assuming that the hypothesis function f is parametric, i.e. it can be described by a fixed number of parameters, e.g. $f(x) = \langle \theta, x \rangle$, which are to be learnt.

Parametric vs non-parametric methods

- In contrast, non-parametric methods have a flexible number of parameters that grows with the data.
- Furthermore, for non-parameteric algorithms, e.g. K-nearest neighbours (kNN), data points, or a subset of them, are kept and used in the prediction phase, resulting in a "memory-based" approach.
- In parametric algorithms, once data is used to learn the parameters, it can be discarded as only the learnt parameters are used for prediction.

- Returning to (15), we have

$$p(y|f(x)) \sim \mathcal{N}(f(x), \tau^2).$$

- Now instead of modeling f from a parameterized family of functions, we enforce a prior Gaussian distribution, with kernel K:

$$p(f(\cdot)) \sim \mathcal{N}(0, K).$$

Conditioned on input values $x^{(1)}, \ldots, x^{(n)}$, we compute the marginal distribution of y by

$$p(y) = \int p(y|f)p(f)df.$$

We know this is normally distributed with mean

$$\mathbb{E}[y] = \mathbb{E}[f] + \mathbb{E}[\epsilon] = 0$$

and covariance C^n with entries

$$\begin{aligned} C_{ij}^n = \mathbb{E}\left[y^{(i)}y^{(j)}\right] &= \mathbb{E}\left[\left(f\left(x^{(i)}\right) + \epsilon^{(i)}\right)\left(f\left(x^{(j)}\right) + \epsilon^{(j)}\right)\right] \\ &= \mathbb{E}\left[f\left(x^{(i)}\right) f\left(x^{(j)}\right)\right] + \mathbb{E}\left[\epsilon^{(i)}\epsilon^{(j)}\right] \\ &= K(x^{(i)}, x^{(j)}) + \tau^2\delta_{ij}. \end{aligned} \tag{16}$$

Prediction

- Given $\left(x^{(1)}, t^{(1)}\right), \ldots, \left(x^{(n)}, t^{(n)}\right)$, and a new input point $x^{(n+1)}$, we predict $y^{(n+1)}$ by computing the posterior distribution

$$p\left(y^{(n+1)} \,\middle|\, y^{(1)} = t^{(1)}, \ldots, y^{(n)} = t^{(n)}\right).$$

- The joint distribution over $y^{(1)}, \ldots, y^{(n)}, y^{(n+1)}$ has mean 0 and covariance C^{n+1} given by (16), which can be partitioned as

$$C^{n+1} = \begin{bmatrix} C^n & k \\ k^T & c \end{bmatrix},$$

where $k = \left[K\left(x^{(1)}, x^{(n+1)}\right), \ldots, K\left(x^{(n)}, x^{(n+1)}\right)\right]^T$ and $c = K\left(x^{(n+1)}, x^{(n+1)}\right) + \tau^2$.

Hence

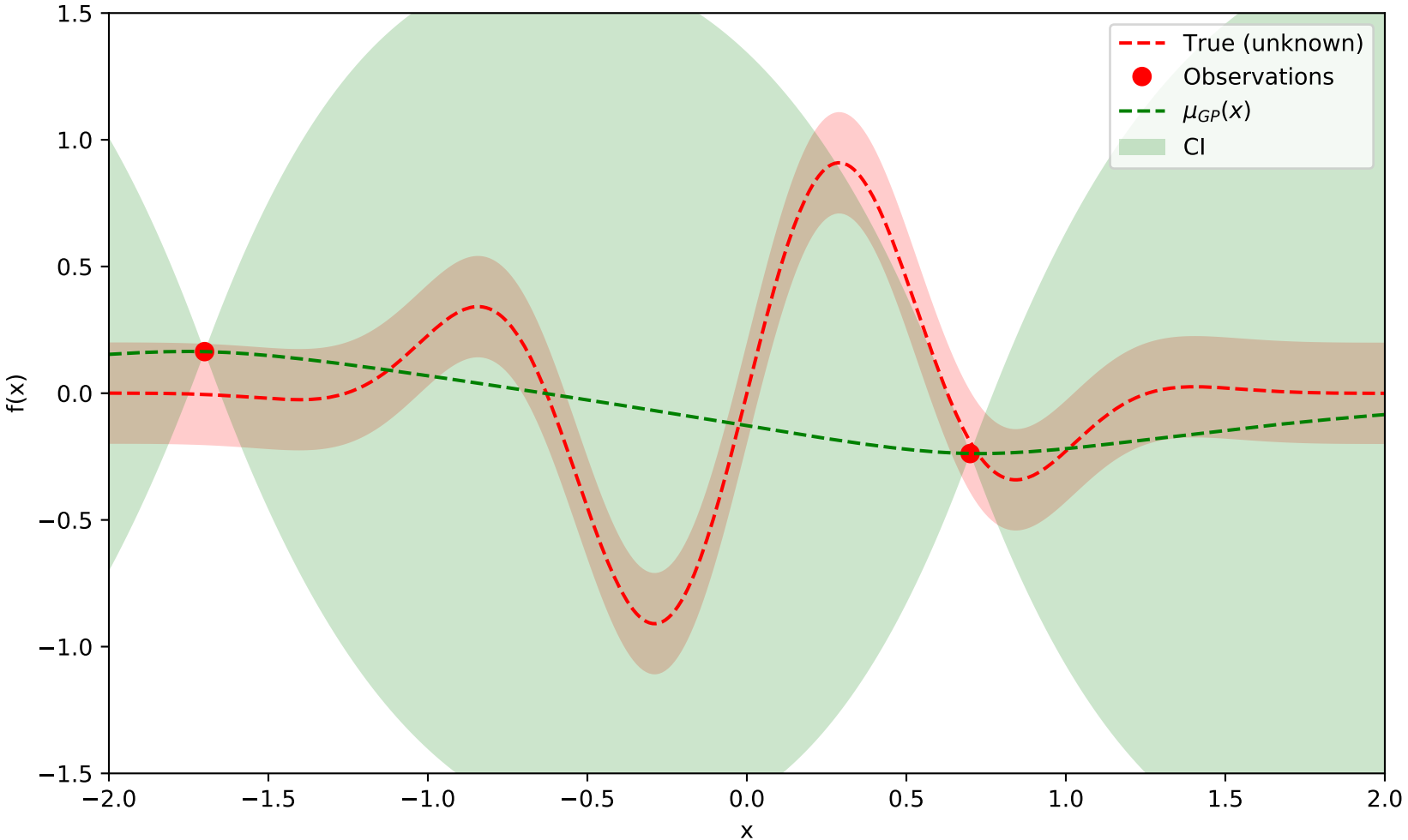
$$p\left(y^{(n+1)} \middle| y^{(1)} = t^{(1)}, \ldots, y^{(n)} = t^{(n)}\right) \sim \mathcal{N}\left(\mu, \sigma^2\right),$$

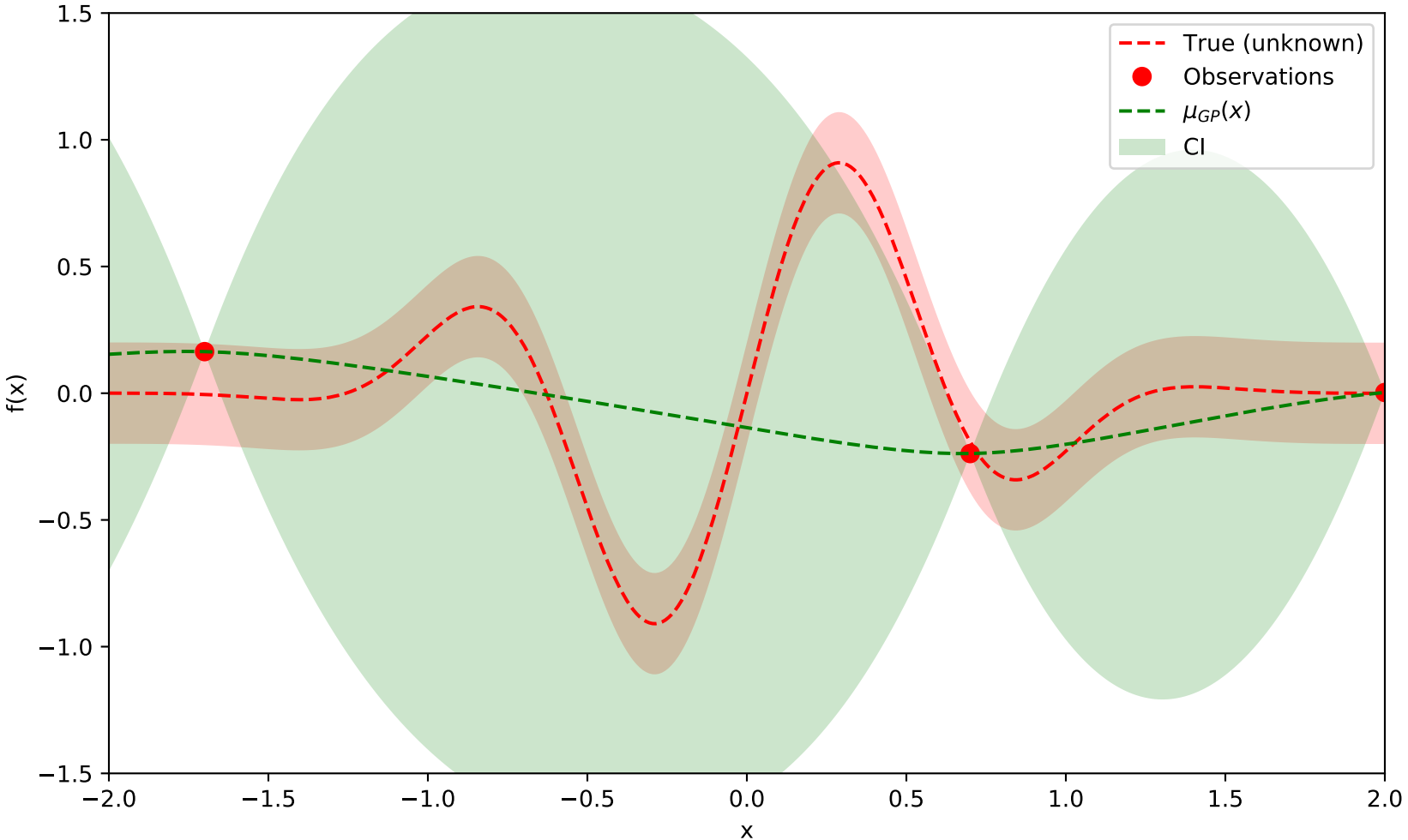
where

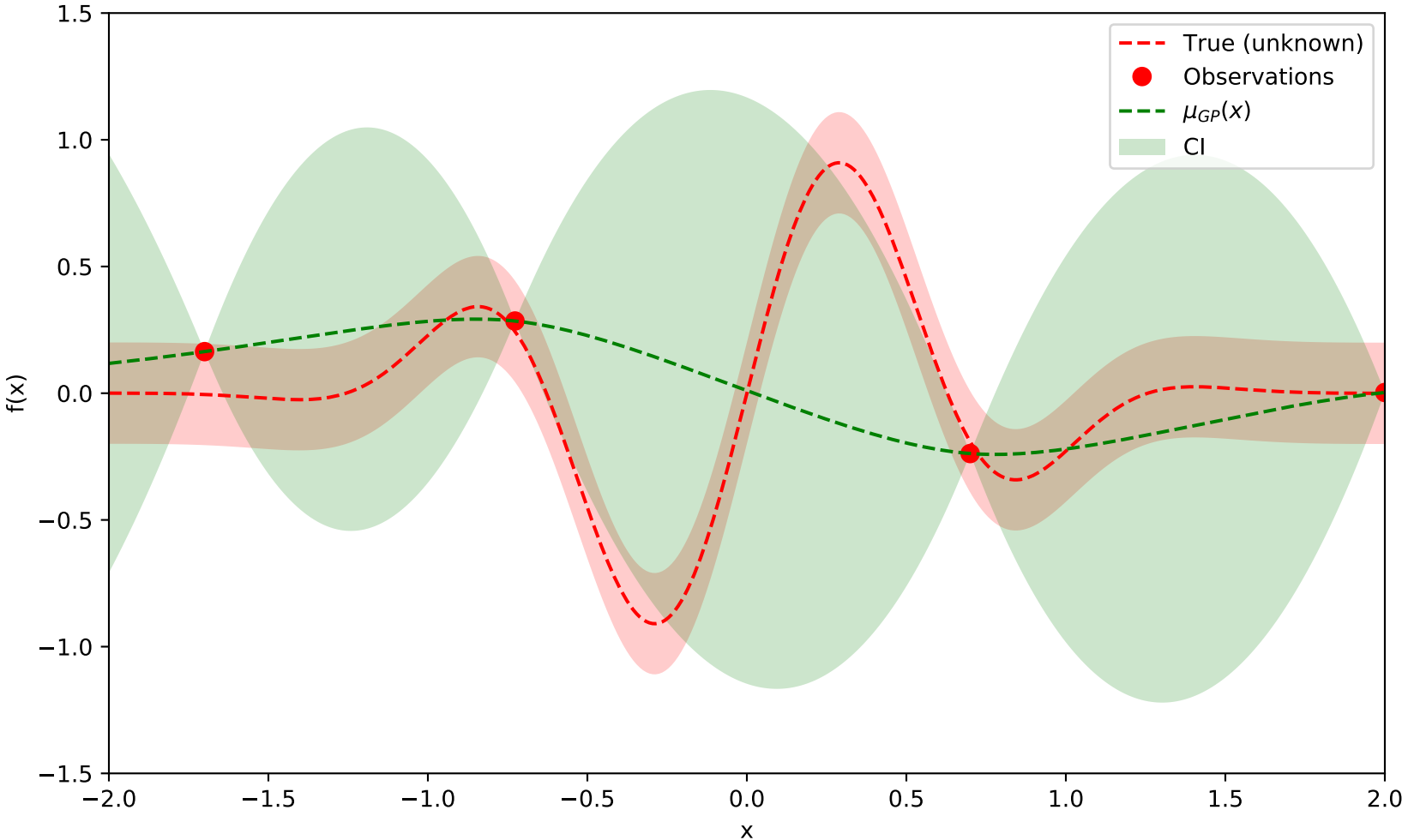
$$\mu = k^T \left(C^n\right)^{-1} \mathbf{t}_n,$$
$$\sigma^2 = c - k^T \left(C^n\right)^{-1} k,$$

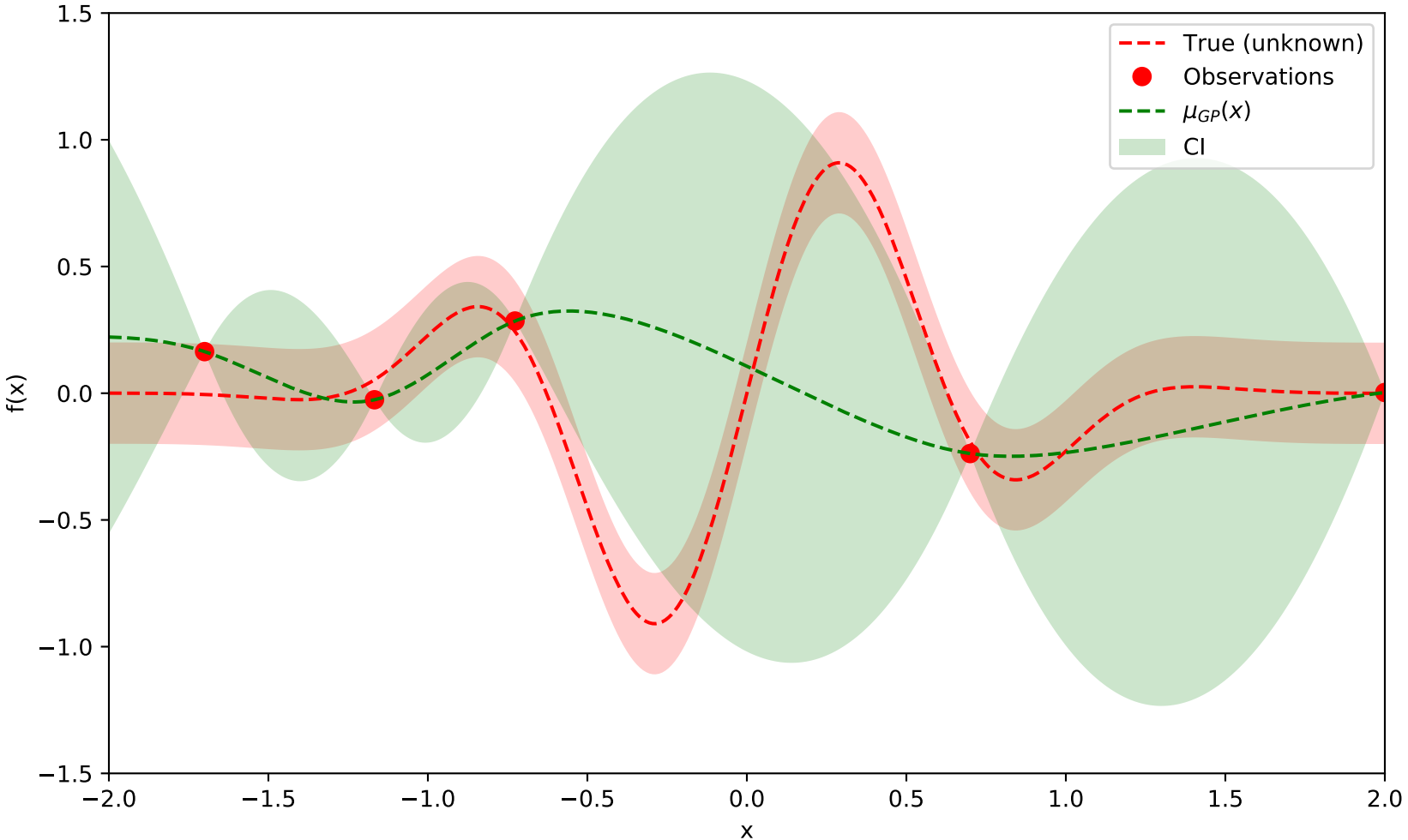
and we denote $\mathbf{t_n} = \left[t^{(1)}, \ldots, t^{(n)}\right]^T$.

Note that μ and σ can be viewed as functions of $x^{(n+1)}$.


True (unknown)
Observations
$\mu_{GP}(x)$
CI
f(x)
x
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0


True (unknown)
Observations
$\mu_{GP}(x)$
CI
f(x)
x
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0


True (unknown)
Observations
$\mu_{GP}(x)$
CI
f(x)
x
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0


True (unknown)
Observations
$\mu_{GP}(x)$
CI
f(x)
x
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

Prediction (summary)

Given $x^{(1)}, \ldots, x^{(n+1)}$,

$$\text{prior distribution} \xrightarrow{\left\{t^{(i)}\right\}_{i=1}^{n}} \text{posterior distribution},$$

i.e.

$$p\left(y^{(1)}, \ldots, y^{(n+1)}\right) \sim \mathcal{N}\left(0, C^{n+1}\right) \longrightarrow p\left(y^{(n+1)} \,\middle|\, y^{(1)} = t^{(1)}, \ldots, y^{(n)} = t^{(n)}\right) \sim \mathcal{N}\left(\mu, \sigma^2\right).$$

Hence, the prior distribution is entirely determined by the kernel K, whereas the the posterior distribution is continually being updated by the observations $t^{(i)}$.

Bayesian optimization with Gaussian processes

- Why?
  - In many machine learning problems, the objective function f is a black-box function which does not have an analytic expression (i.e. one cannot take derivatives), or has one that is too costly to compute.
  - Moreover, f may be expensive to evaluate, or its domain may be high-dimensional, which makes a grid-search over its domain prohibitively time-consuming (curse of dimensionality).
  - Eg. Hyper-parameter tuning in deep learning models

- Bayesian optimization techniques attempt to find the global optimum of f in as few steps as possible.
- How?
  - Define a *surrogate model* which approximates the objective function f, eg. a Gaussian process.
  - Use an *acquisition function* to direct sampling to areas where one will have an increased probability of finding the optimum.

Optimization algorithm

In the start, select a kernel to model the objective function and choose an acquisition function $A(x)$. Now assume we are at time n where the n^{th} data-point $x^{(n)}$ and the corresponding value $y^{(n)}$ has just been sampled.

1. Update the posterior distribution with $y^{(n)}$.
2. Find the next sampling point $x^{(n+1)}$ by optimizing

$$x^{(n+1)} = \arg\max_x A\left(x \,\middle|\, x^{(1)}, \ldots, x^{(n)}\right)$$

3. Obtain a (possibly noisy) sample $y^{(n+1)} = f\left(x^{(n+1)}\right) + \epsilon^{(n+1)}$.

Common acquisition functions

Let f^* denote the maximum value for f found so far, and let $\gamma_x = \frac{\mu_x - f^*}{\sigma_x}$.

1. Probability of Improvement:

$$A(x, f^*) = P(f_x > f^*) = \mathbf{\Phi}(\gamma_x),$$

2. Expected Improvement:

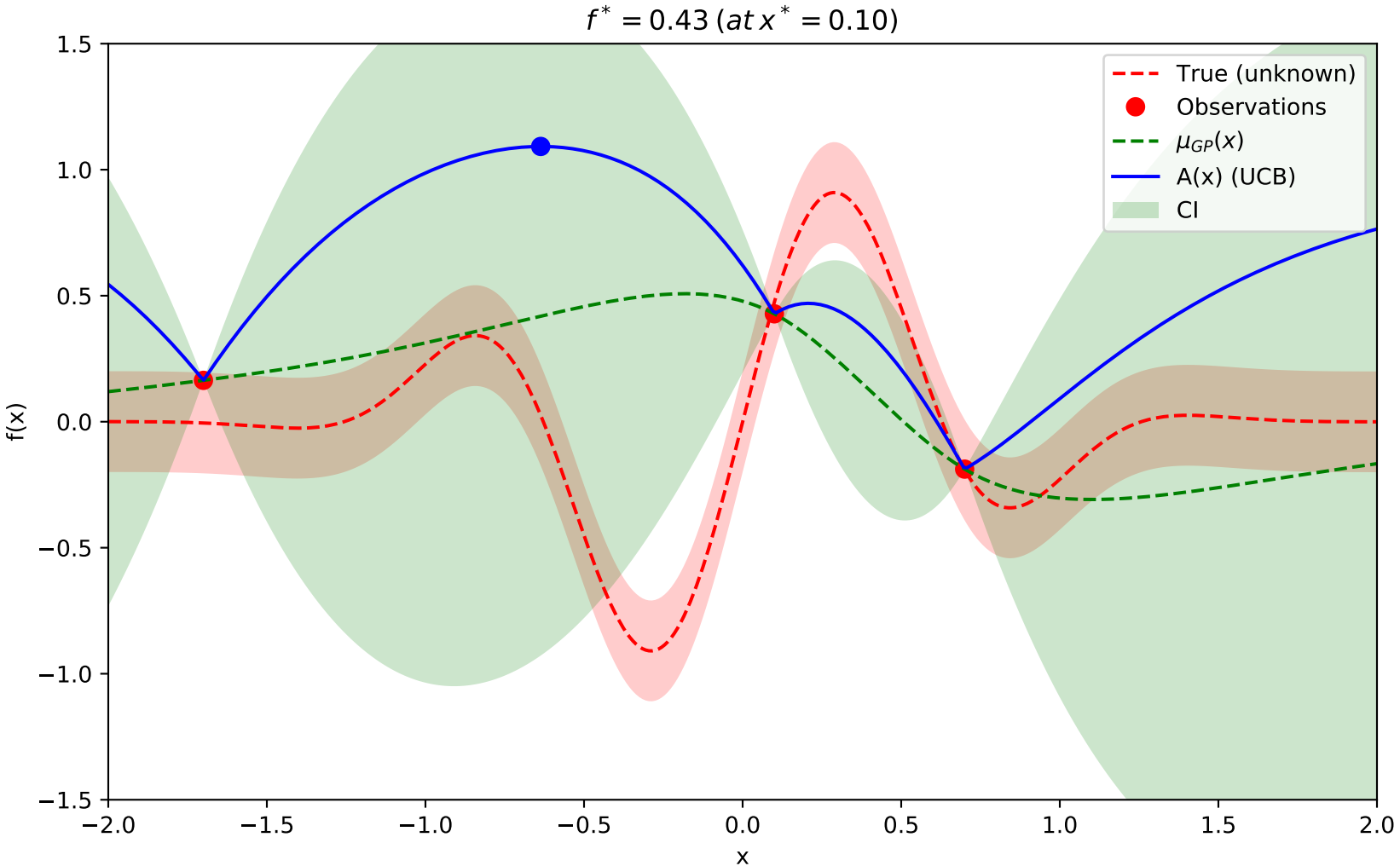
$$A(x, f^*) = \mathbb{E}[\max\{f_x - f^*, 0\}] = \sigma_x[\gamma_x \mathbf{\Phi}(\gamma_x) + \phi(\gamma_x)]$$

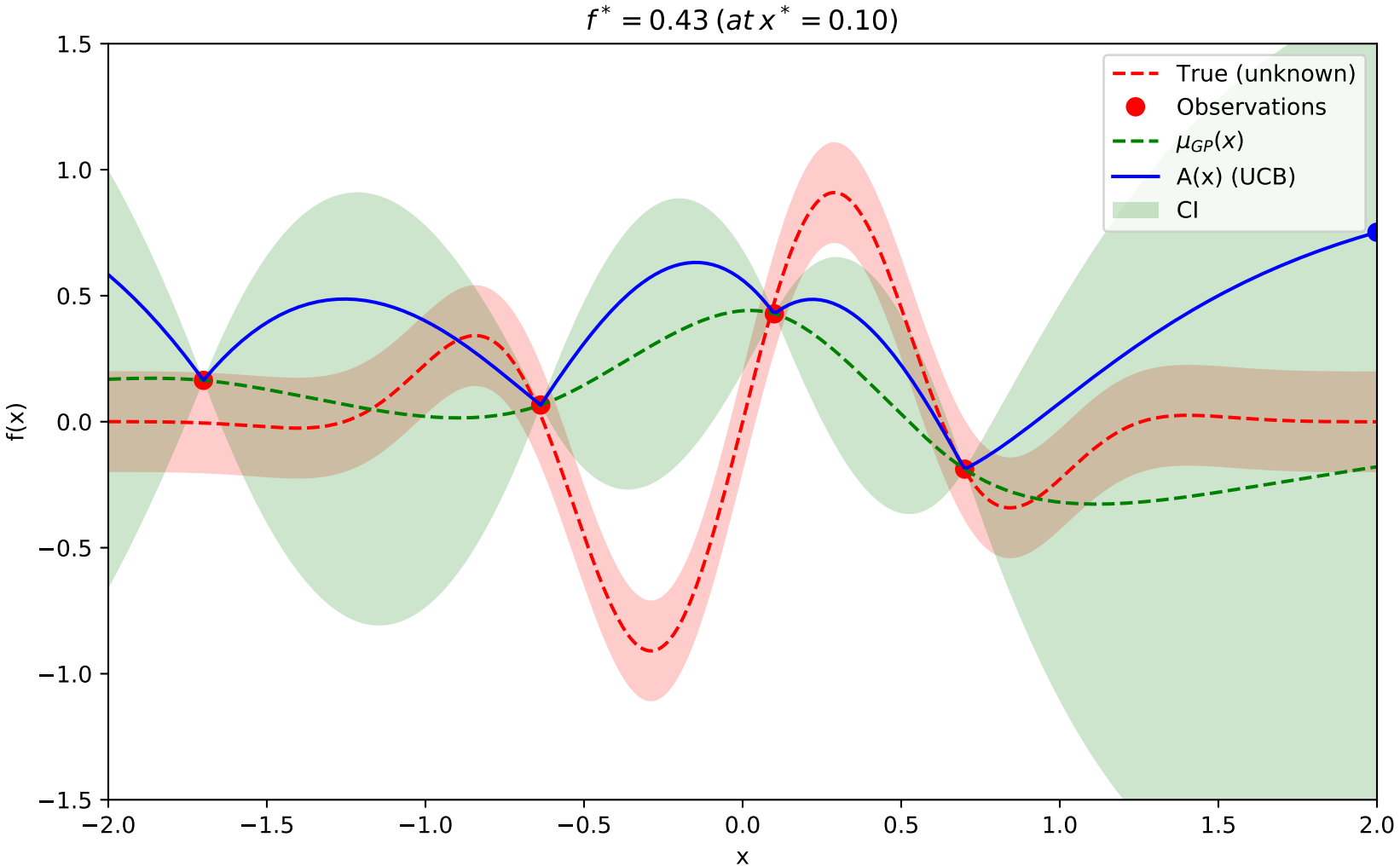
3. Upper Confidence Bound (UCB):

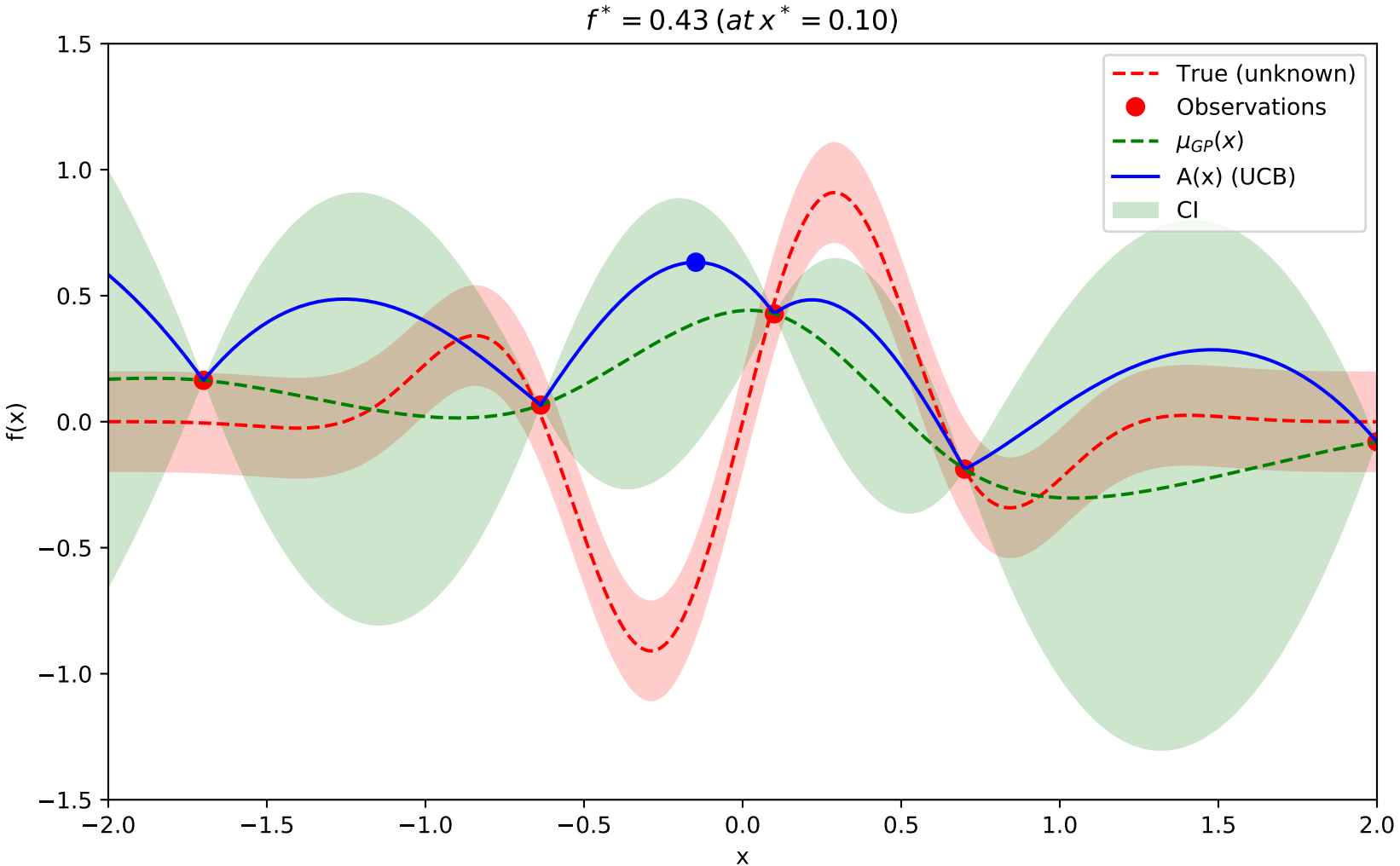
$$A(x) = \mu_x + \kappa\sigma_x$$

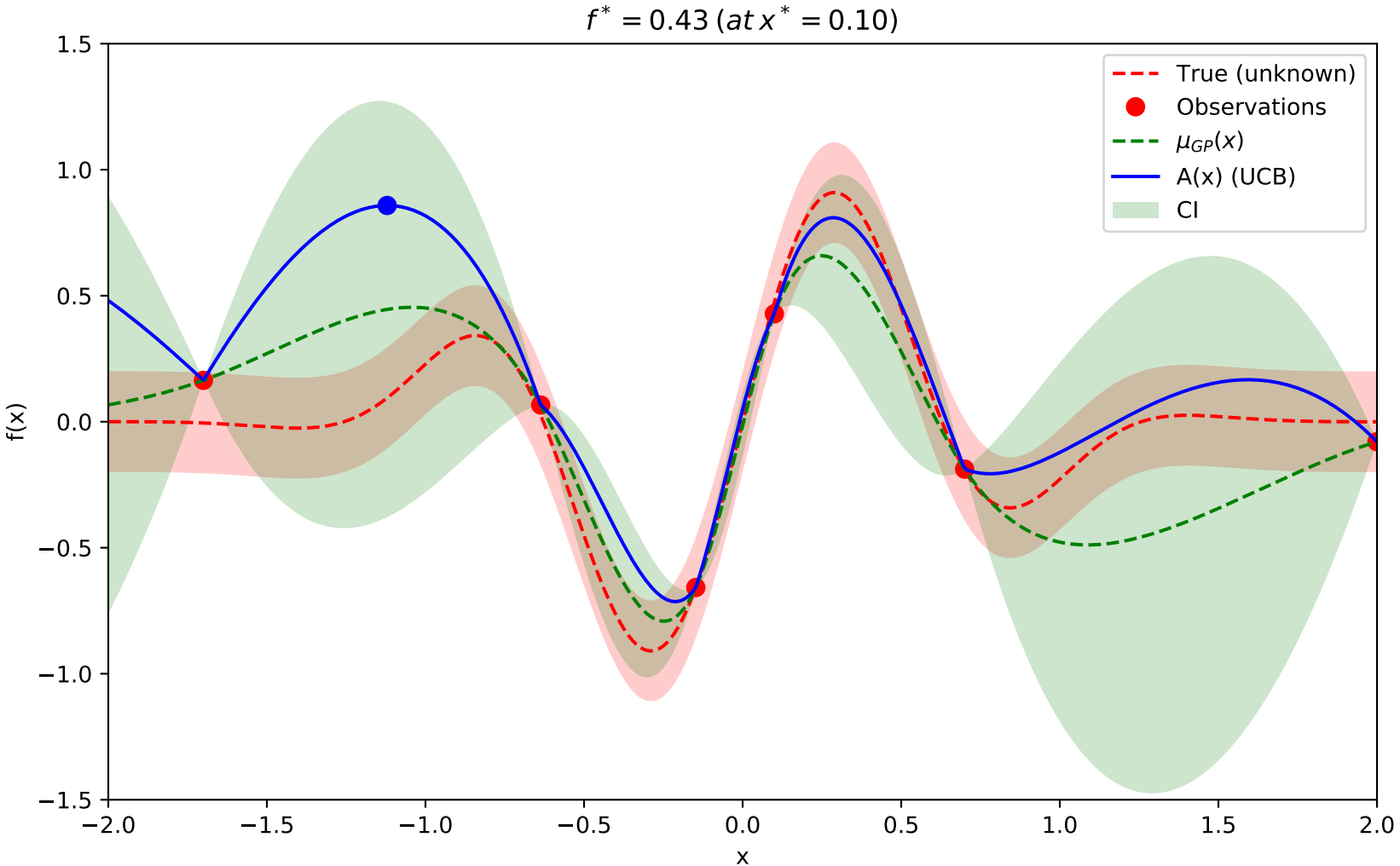
($\mathbf{\Phi}$ and ϕ denote the cdf and pdf of the standard normal distribution, and note that $f_x \sim \mathcal{N}(\mu_x, \sigma_x^2)$ denotes the predicted value of f at x)

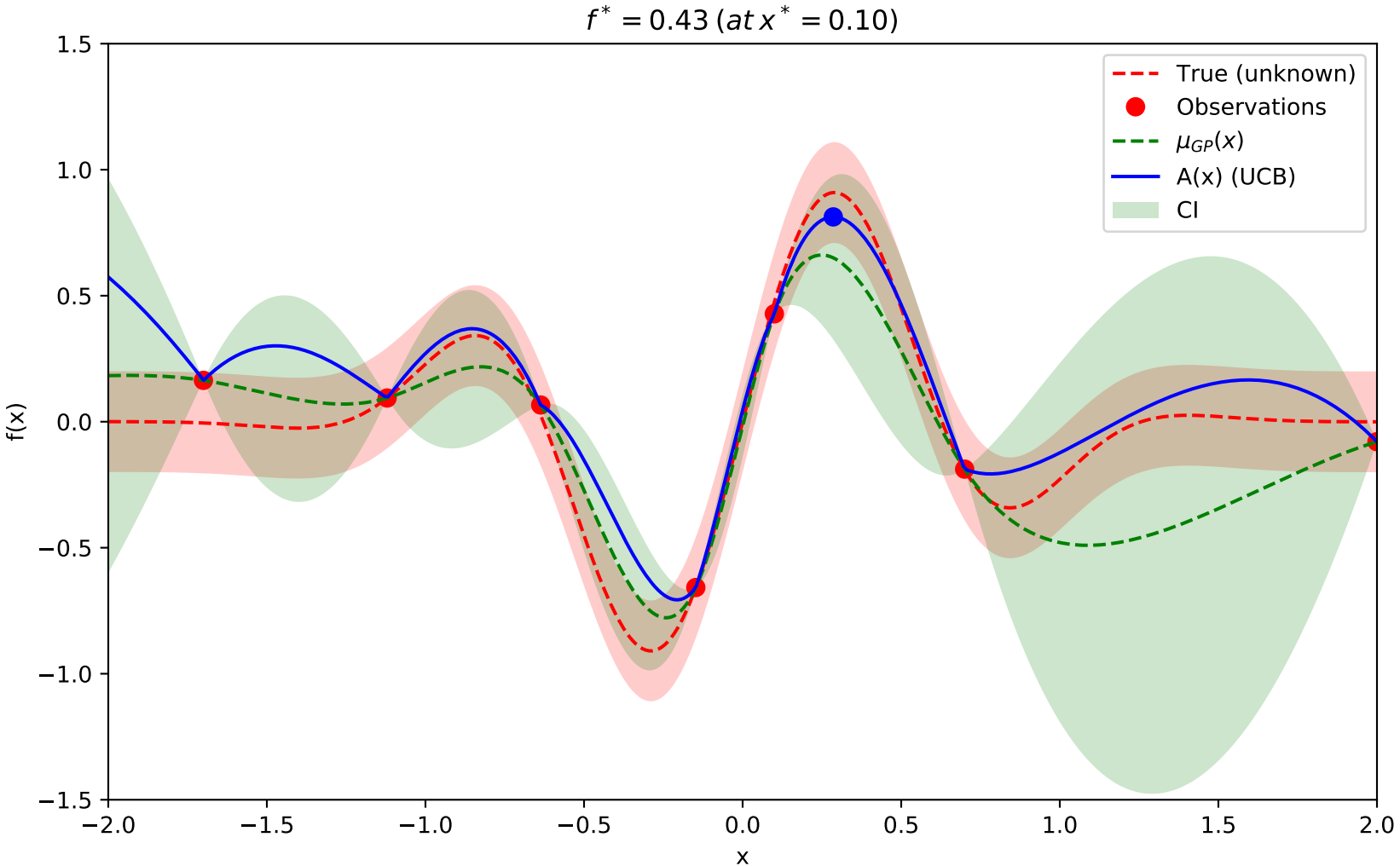
- The acquisition functions are relatively simple functions of μ_x and σ_x.
- Thus, Gaussian process regression replaces the original intractable objective function f with a tractable acquisition function which can be optimized by conventional methods, eg. gradient descent.

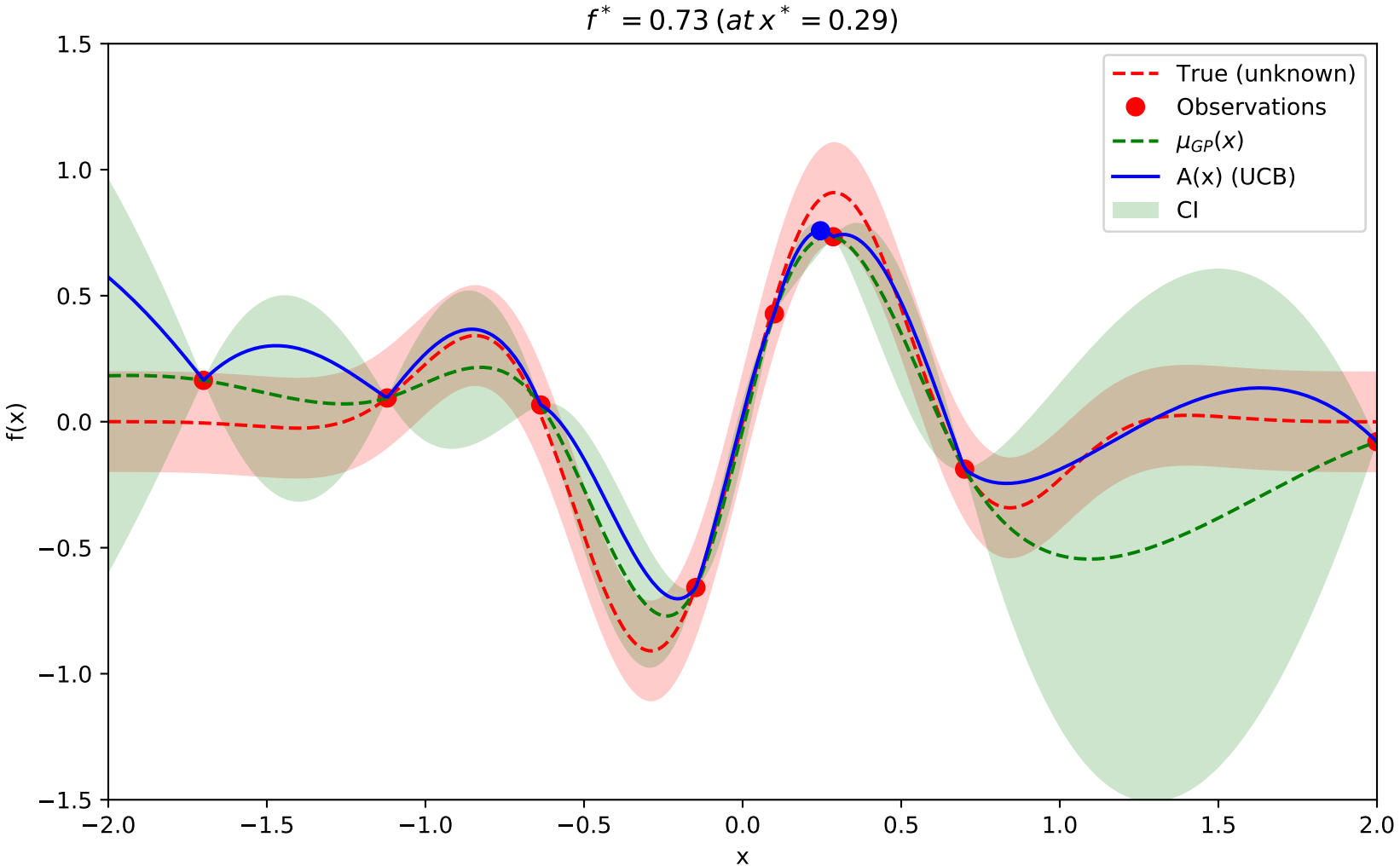

$f^* = 0.43\ (at\ x^* = 0.10)$
True (unknown)
Observations
$\mu_{GP}(x)$
A(x) (UCB)
CI
f(x)
x
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0


$f^* = 0.43\,(at\,x^* = 0.10)$
True (unknown)
Observations
$\mu_{GP}(x)$
A(x) (UCB)
CI
f(x)
x
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0


$f^* = 0.43\,(at\,x^* = 0.10)$
True (unknown)
Observations
$\mu_{GP}(x)$
A(x) (UCB)
CI
f(x)
x
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0


$f^* = 0.43\,(at\,x^* = 0.10)$
True (unknown)
Observations
$\mu_{GP}(x)$
A(x) (UCB)
CI
f(x)
x
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0


$f^* = 0.43\,(at\,x^* = 0.10)$
True (unknown)
Observations
$\mu_{GP}(x)$
A(x) (UCB)
CI
f(x)
x
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0


$f^* = 0.73\,(at\,x^* = 0.29)$
True (unknown)
Observations
$\mu_{GP}(x)$
A(x) (UCB)
CI
f(x)
x
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

Code

```
from scipy.optimize import minimize
from scipy.stats import norm
from sklearn.gaussian_process import GaussianProcessRegressor
from sklearn.gaussian_process.kernels import RBF, Matern

gp = GaussianProcessRegressor(kernel=RBF(length_scale=1.0))
gp.fit(xi, yi)
mu_x, sigma_x = gp.predict(x, return_std=True)
```