Tutorial on Bayesian Optimization

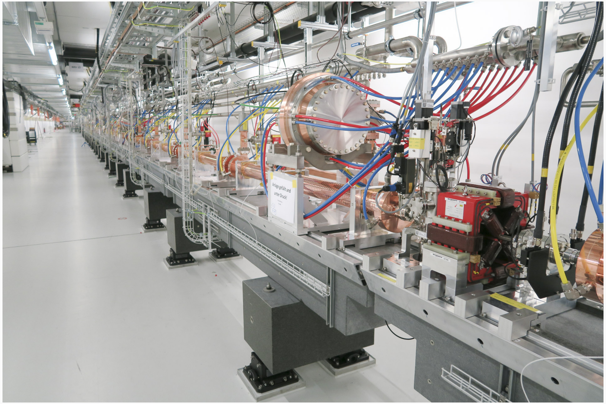
Johannes Kirschner and Mojmír Mutný

February 26th, 2019

ICFA ML Workshop, PSI

ETH zürich

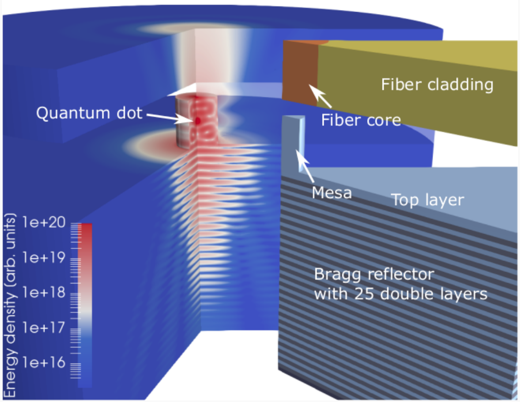
Motivating Application: Parameter Tuning of Accelerator

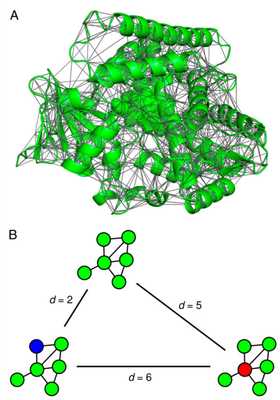


Maximize (photon) signal, minimize losses, . . .

[McIntire et al., 2016, Kirschner et al., 2019]

Motivating Application: Experimental Design

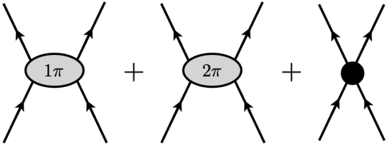




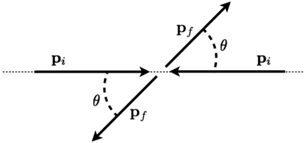
Optimize design parameters, e.g. nano materials, molecules,...

[Schneider et al., 2018, Romero et al., 2013]

Motivating Application: Fitting Physical Models



χEFT model for the NN interaction



Experimental data on NN scattering

Optimize model parameters to fit observational data

[Ekström et al., 2019]

Optimizing Black-Box Functions

Maximize a Black box function:

$$\mathcal{X} \longrightarrow f \rightsquigarrow f(x) + \epsilon$$

Optimizing Black-Box Functions

Maximize a Black box function:

$$\mathcal{X} \longrightarrow f \rightsquigarrow f(x) + \epsilon$$

- Parameter space $\mathcal{X} \subset \mathbb{R}^d$, can also be combinatorial
- Little assumptions on f: non-convex, multiple local optima, ...
- No analytical formula for f
- No access to gradients

Optimizing Black-Box Functions

Maximize a Black box function:

$$\mathcal{X} \longrightarrow f \rightsquigarrow f(x) + \epsilon$$

- Parameter space $\mathcal{X} \subset \mathbb{R}^d$, can also be combinatorial
- Little assumptions on f: non-convex, multiple local optima, ...
- No analytical formula for f
- No access to gradients

Only get (noisy) evaluations $y = f(x) + \epsilon$

- Evaluations of f are 'expensive'

Bayesian Optimization: Overview

Prior data set: $\mathcal{D}_0$

Bayesian Optimization: Overview

Prior data set: $\mathcal{D}_0$

For each step $t = 1, 2, 3, \ldots, T$,

`Step 1:` Build *probabilistic model* $\hat{f}_t$ of the objective using data $\mathcal{D}_{t-1}$

- Gaussian process regression (**Part I**)

Bayesian Optimization: Overview

Prior data set: $\mathcal{D}_0$

For each step $t = 1, 2, 3, \dots, T$,

`Step 1:` Build *probabilistic model* $\hat{f}_t$ of the objective using data $\mathcal{D}_{t-1}$

▷ Gaussian process regression (**Part I**)

`Step 2:` Reduce model *uncertainty about maximizers*

▷ Search guided by acquisition function $x_t = \arg\max_{x \in \mathcal{X}} \alpha(x|\hat{f}_t)$ (**Part II**)

Bayesian Optimization: Overview

Prior data set: $\mathcal{D}_0$

For each step $t = 1, 2, 3, \ldots, T$,

`Step 1:` Build *probabilistic model* $\hat{f}_t$ of the objective using data $\mathcal{D}_{t-1}$

- ▷ Gaussian process regression (**Part I**)

`Step 2:` Reduce model *uncertainty about maximizers*

- ▷ Search guided by acquisition function $x_t = \arg\max_{x \in \mathcal{X}} \alpha(x|\hat{f}_t)$ (**Part II**)

`Step 3:` Observe (noisy) measurement $y_t = f(x_t) + \epsilon$

- ▷ Augment data $\mathcal{D}_t = \mathcal{D}_{t-1} \cup \{(x_t, y_t)\}$

Bayesian Optimization: Overview

Prior data set: $\mathcal{D}_0$

For each step $t = 1, 2, 3, \ldots, T$,

`Step 1:` Build *probabilistic model* $\hat{f}_t$ of the objective using data $\mathcal{D}_{t-1}$

▷ Gaussian process regression (**Part I**)

`Step 2:` Reduce model *uncertainty about maximizers*

▷ Search guided by acquisition function $x_t = \arg\max_{x \in \mathcal{X}} \alpha(x|\hat{f}_t)$ (**Part II**)

`Step 3:` Observe (noisy) measurement $y_t = f(x_t) + \epsilon$

▷ Augment data $\mathcal{D}_t = \mathcal{D}_{t-1} \cup \{(x_t, y_t)\}$

At final time T: *Use model to find best predicted setting*.

Part I: Gaussian Process Regression

Bayesian Statistics

Prior: Distribution $\mathcal{P}(f)$ over f

- "prior belief"

Bayesian Statistics

Prior: Distribution $\mathcal{P}(f)$ over f

- "prior belief"

Data likelihood: $\mathcal{P}(D_t|f)$

- e.g. $y \sim f(x) + \mathcal{N}(0, 1)$

Bayesian Statistics

Prior: Distribution $\mathcal{P}(f)$ over f

- ▷ "prior belief"

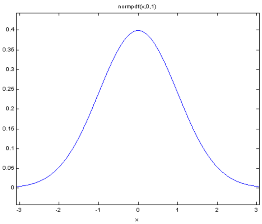
Data likelihood: $\mathcal{P}(D_t|f)$

- ▷ e.g. $y \sim f(x) + \mathcal{N}(0, 1)$

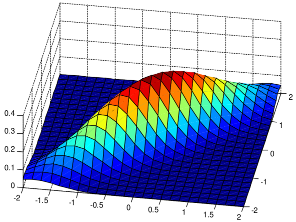
Posterior distribution: $\mathcal{P}(f|D_t) = \dfrac{\mathcal{P}(D_t|f)\mathcal{P}(f)}{\mathcal{P}(D_t)}$

- ▷ Bayes' theorem
- ▷ The posterior distribution captures our belief in f after seeing the data.

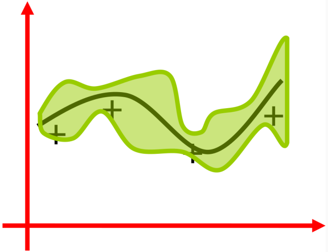
Gaussian Processes



Normal dist.
(1-D Gaussian)



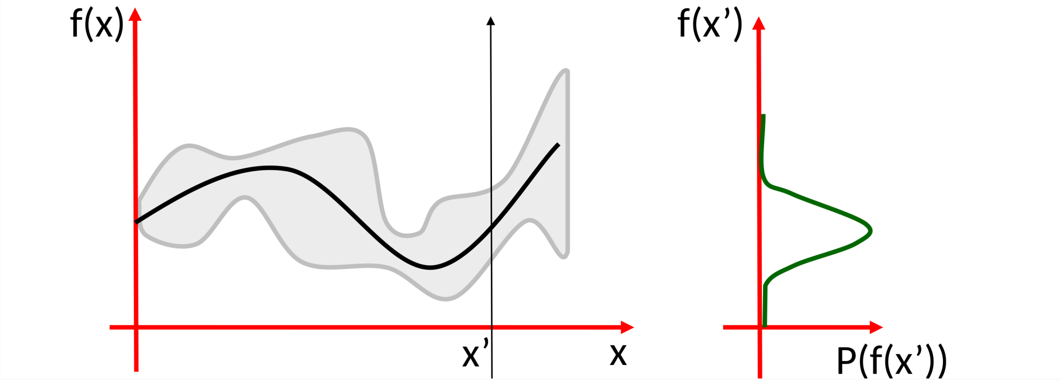
Multivariate normal
(n-D Gaussian)



Gaussian process
(∞-D Gaussian)

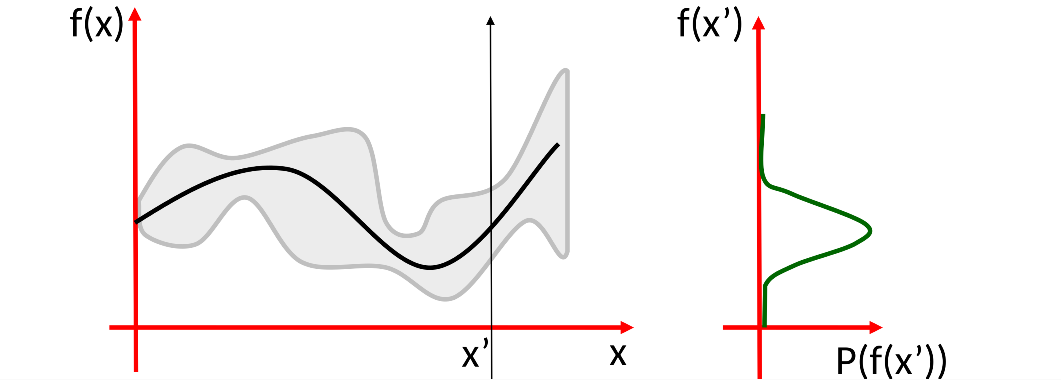
- ▷ **Gaussian process (GP)** $=$ normal distribution over *functions*

Gaussian Processes



- **Gaussian process (GP)** = normal distribution over *functions*
- Finite marginals $f(x_1), \ldots, f(x_n)$ are multivariate Gaussians

Gaussian Processes



- **Gaussian process (GP)** = normal distribution over *functions*
- Finite marginals $f(x_1), \ldots, f(x_n)$ are multivariate Gaussians
- Parameterized by covariance function (kernel) $k(x, x') = Cov(f(x), f(x'))$

Gaussian Process on Finite Domain

Finite domain: $\mathcal{X} = \{x_1, \ldots, x_n\}$

Gaussian Process on Finite Domain

Finite domain: $\mathcal{X} = \{x_1, \ldots, x_n\}$

Definition: f is a Gaussian process with mean $\mu(x)$ and kernel $k(x, x')$

- if $f(x_1), \ldots, f(x_n)$ is multivariate normal $\mathcal{N}(m, K)$ with

Gaussian Process on Finite Domain

Finite domain: $\mathcal{X} = \{x_1, \ldots, x_n\}$

Definition: f is a Gaussian process with mean $\mu(x)$ and kernel $k(x, x')$

- ▷ if $f(x_1), \ldots, f(x_n)$ is multivariate normal $\mathcal{N}(m, K)$ with
- ▷ mean $m = [\mu(x_1), \ldots \mu(x_n)]$,

Gaussian Process on Finite Domain

Finite domain: $\mathcal{X} = \{x_1, \ldots, x_n\}$

Definition: f is a Gaussian process with mean $\mu(x)$ and kernel $k(x, x')$

- ▷ if $f(x_1), \ldots, f(x_n)$ is multivariate normal $\mathcal{N}(m, K)$ with
- ▷ mean $m = [\mu(x_1), \ldots \mu(x_n)]$,
- ▷ covariance $K = [k(x_i, x_j)]_{i,j=1,\ldots,n}$.

Gaussian Process on Finite Domain

Finite domain: $\mathcal{X} = \{x_1, \dots, x_n\}$

Definition: f is a Gaussian process with mean $\mu(x)$ and kernel $k(x, x')$

- $\triangleright$ if $f(x_1), \dots, f(x_n)$ is multivariate normal $\mathcal{N}(m, K)$ with
- $\triangleright$ mean $m = [\mu(x_1), \dots \mu(x_n)]$,
- $\triangleright$ covariance $K = [k(x_i, x_j)]_{i,j=1,\dots,n}$.

Denote $f \sim GP(m, k)$.

Gaussian Process on Continuous Domain

Continuous domain: $\mathcal{X} \subset \mathbb{R}^d$

Gaussian Process on Continuous Domain

Continuous domain: $\mathcal{X} \subset \mathbb{R}^d$

Definition: f is a Gaussian process with mean $\mu(x)$ and kernel $k(x, x')$ if

- for **any finite subset** $\{x_1, \ldots, x_n\} \subset \mathcal{X}$,

Gaussian Process on Continuous Domain

Continuous domain: $\mathcal{X} \subset \mathbb{R}^d$

Definition: f is a Gaussian process with mean $\mu(x)$ and kernel $k(x, x')$ if

- ▷ for **any finite subset** $\{x_1, \dots, x_n\} \subset \mathcal{X}$,
- ▷ $f(x_1), \dots, f(x_n)$ is multivariate normal $\mathcal{N}(m, K)$ with
- ▷ mean $m = [\mu(x_1), \dots \mu(x_n)]$,
- ▷ covariance $K = [k(x_i, x_j)]_{i,j=1,\dots,n}$.

Gaussian Process on Continuous Domain

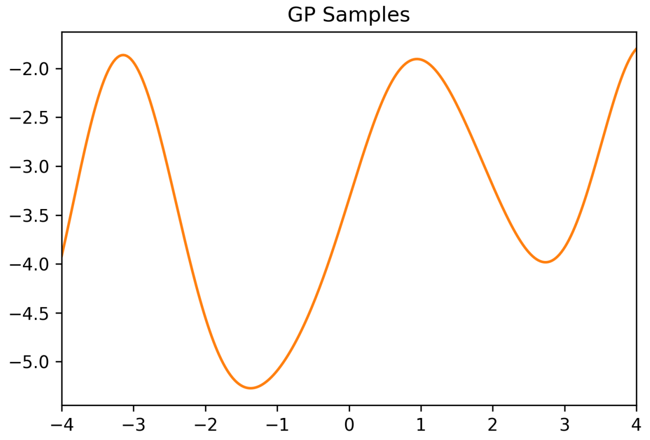
Continuous domain: $\mathcal{X} \subset \mathbb{R}^d$

Definition: f is a Gaussian process with mean $\mu(x)$ and kernel $k(x, x')$ if

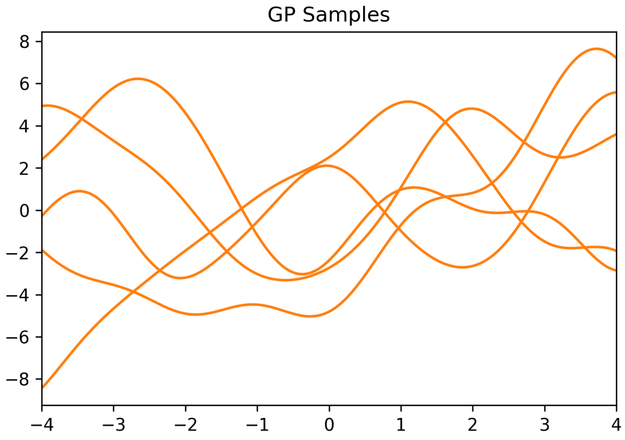
- ▷ for **any finite subset** $\{x_1, \ldots, x_n\} \subset \mathcal{X}$,
- ▷ $f(x_1), \ldots, f(x_n)$ is multivariate normal $\mathcal{N}(m, K)$ with
- ▷ mean $m = [\mu(x_1), \ldots \mu(x_n)]$,
- ▷ covariance $K = [k(x_i, x_j)]_{i,j=1,\ldots,n}$.

In practice we always evaluate/sample the GP on finite (grid) domains.

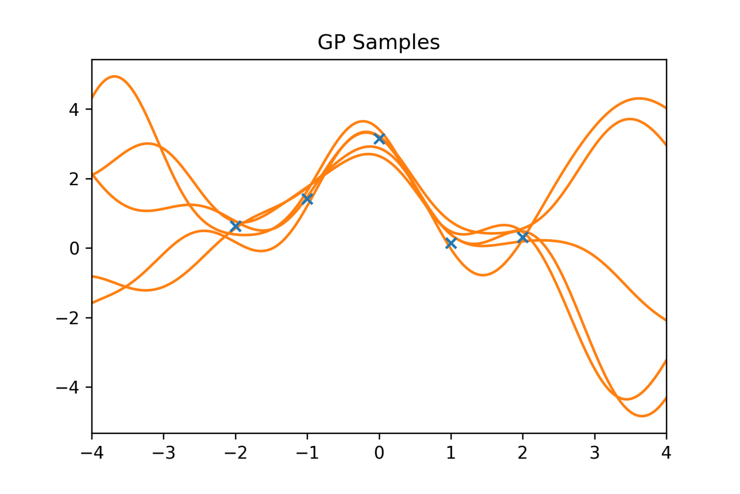
Samples from a Gaussian Process



Samples from a Gaussian Process



Samples from a Gaussian Process



Gaussian Process Regression

Prior: GP prior $\mathcal{P}(f) = GP(\mu, k)$ over f

- ▷ "prior belief" with prior mean μ and kernel k

Gaussian Process Regression

Prior: GP prior $\mathcal{P}(f) = GP(\mu, k)$ over f

- ▷ "prior belief" with prior mean μ and kernel k

Gaussian likelihood: iid Gaussian noise:

- ▷ $\mathcal{P}(\{y_1, \ldots, y_m\} | f(x_1), \ldots, f(x_n)) = \prod_i \mathcal{N}(f(x_i), \rho^2)$
- ▷ e.g. $y \sim f(x) + \mathcal{N}(0, \rho^2)$

Gaussian Process Regression

Prior: GP prior $\mathcal{P}(f) = GP(\mu, k)$ over f

- "prior belief" with prior mean μ and kernel k

Gaussian likelihood: iid Gaussian noise:

- $\mathcal{P}(\{y_1, \dots, y_m\} | f(x_1), \dots, f(x_n)) = \prod_i \mathcal{N}(f(x_i), \rho^2)$
- e.g. $y \sim f(x) + \mathcal{N}(0, \rho^2)$

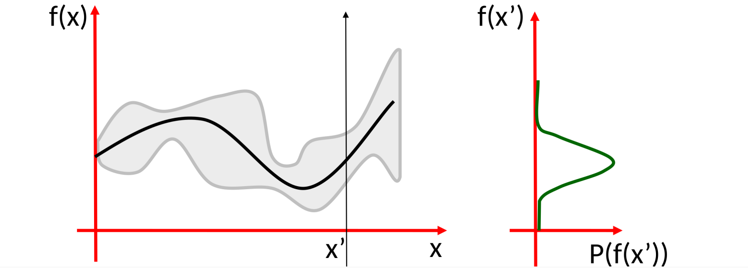
Posterior distribution: $\mathcal{P}(f | D_t) = GP(\mu_n, k_n)$

- Posterior distributions is a again a GP!
- Closed form updates exist.
- Excellent book (free pdf): [Rasmussen, 2004, Chapter 2]

Marginals

Posterior distribution: $\mathcal{P}(f|D_t) = GP(\mu_n, k_n)$

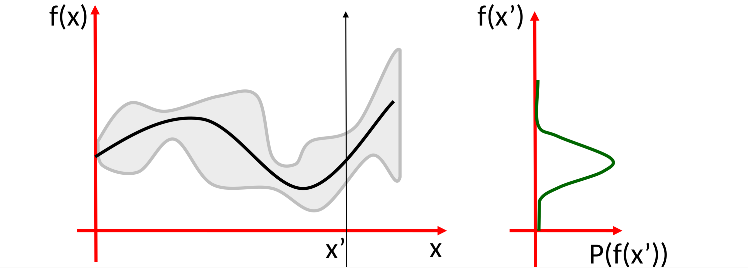
- *Remember:* Finite marginals are Gaussians!
- Marginal posterior distribution at any point x is $\mathcal{N}(\mu_n(x), k_n(x, x))$



Marginals

Posterior distribution: $\mathcal{P}(f|D_t) = GP(\mu_n, k_n)$

- *Remember:* Finite marginals are Gaussians!
- Marginal posterior distribution at any point x is $\mathcal{N}(\mu_n(x), k_n(x,x))$



Posterior variance $\sigma_n(x)^2 = k_n(x,x)$ quantifies uncertainty

Kernel Functions

Kernel k needs to satisfy some technical assumptions:

- symmetric
- positive semidefinite.

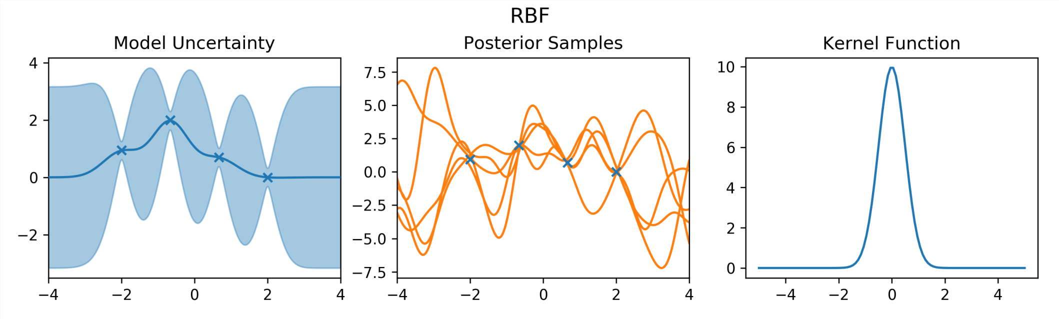
Kernel Functions

Kernel k needs to satisfy some technical assumptions:

- symmetric
- positive semidefinite.

Kernels are similarity measures between points and encodes smoothness.

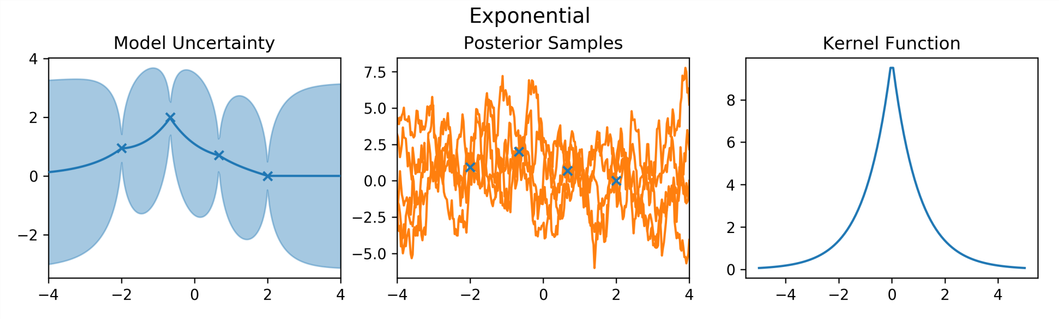
Kernel Functions: Squared Exponential (RBF)



Squared exponential kernel: $k(x, x') = \exp(-\|x - x'\|^2 / l^2)$

▷ l is called lengthscale (or bandwidth)

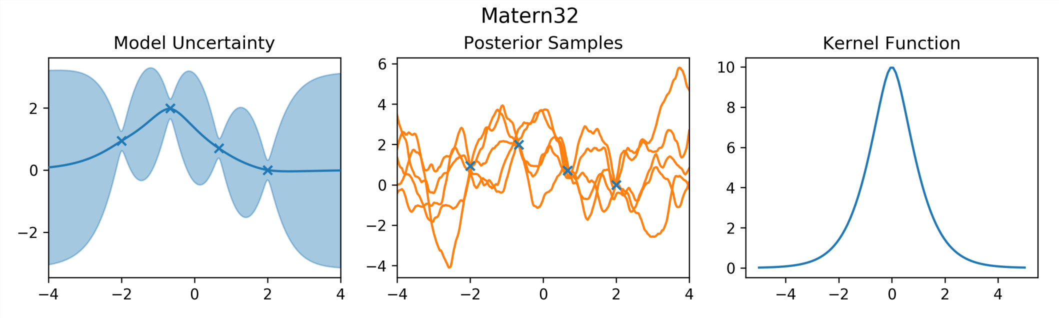
Kernel Functions: Exponential



Exponential kernel: $k(x, x') = \exp(-\|x - x'\|/l^2)$

- l is called lengthscale (or bandwidth)

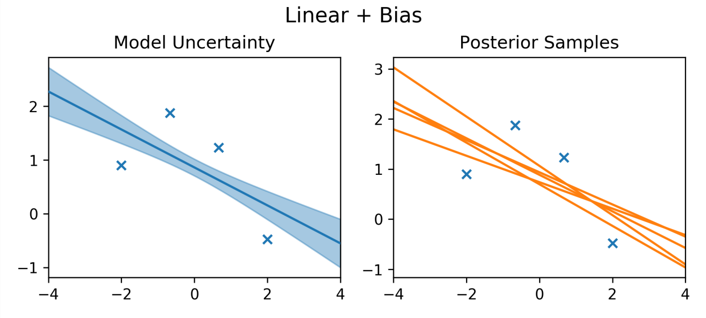
Kernel Functions: Matern



Matern32 kernel: $k(x, x') = \left(1 + \frac{\sqrt{3}\|x - x\|}{l}\right) \exp\left(-\frac{\sqrt{3}\|x - x'\|}{l}\right)$

- l is called lengthscale (or bandwidth)
- Matern52, etc: Family of kernels with increasing smoothness

Kernel Functions: Linear



Linear kernel: $k(x, x') = x^\top x'$

- ▷ Recovers (Bayesian) linear regression!

Feature kernel: $k(x, x') = \Phi(x)^\top \Phi(x')$

- ▷ E.g. polynomials $\Phi(x) = [1, x, x^2]$

Kernel Parameters I

Noise variance

- ▷ Easy to measure
- ▷ Slightly larger value increases robustness

Kernel Parameters I

Noise variance

- Easy to measure
- Slightly larger value increases robustness

Kernel

- Smoothness of function
- RBF smooth functions
- Matern32, Matern52, less smooth, often work well in pratice
- Can also combine kernels, e.g. RBF $+$ 5$\cdot$Matern32
- Each kernel has its own hyper-parameters

Kernel Parameters II

Normalizes objective (y-values)

Prior variance

- Expected range of objective values
- Keep fixed (to 1) and normalize data

Kernel Parameters II

Normalizes objective (y-values)

Prior variance

- Expected range of objective values
- Keep fixed (to 1) and normalize data

Normalizes domain (x-values)

Lengthscale

- Smoothness of function
- If too large, might not model the objective well
- Can pick different lengthscales for different dimensions (ARD)
- Normalizes the domain

How to choose parameters?

Try and error

- ▷ Parameters usually more intuitive to tune

How to choose parameters?

Try and error

- Parameters usually more intuitive to tune

Point estimates

- Maximum a posteriori estimation: $\theta^* = \arg\max_\theta \mathcal{P}(D_t|\theta)\mathcal{P}(\theta)$
- Requires 'representative' initial data
- Might not work well with data collected while optimizing

How to choose parameters?

Try and error

- Parameters usually more intuitive to tune

Point estimates

- Maximum a posteriori estimation: $\theta^* = \arg\max_\theta \mathcal{P}(D_t|\theta)\mathcal{P}(\theta)$
- Requires 'representative' initial data
- Might not work well with data collected while optimizing

Bayesian approach

- Define 'reasonable' prior distribution $\mathcal{P}(\theta)$ over θ
- Marginalize predictions over posterior $\mathcal{P}(\theta|D_t)$
- More expensive to compute, no closed form
- Eliminates hyperparameters

Notebook Session: GP Regression using GPy

Part II: Bayesian Optimization

Optimization - recap

- Assume function $f(x)$ where $x \in \mathcal{X}$.

Optimization - recap

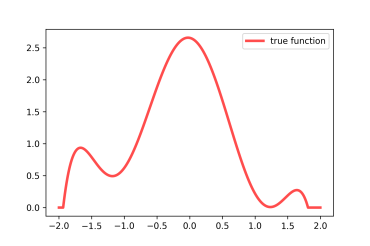
- Assume function $f(x)$ where $x \in \mathcal{X}$.
- Noisy zero-order oracle $\iff y = f(x) + \epsilon$

Optimization - recap

- Assume function $f(x)$ where $x \in \mathcal{X}$.
- Noisy zero-order oracle $\iff y = f(x) + \epsilon$
- Grid approach fails:

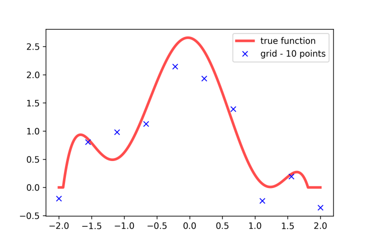
Optimization - recap

- Assume function $f(x)$ where $x \in \mathcal{X}$.
- Noisy zero-order oracle $\iff$ $y = f(x) + \epsilon$
- Grid approach fails:
  - due to noise



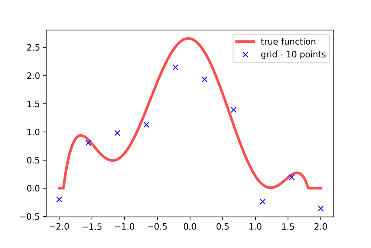
Optimization - recap

- Assume function $f(x)$ where $x \in \mathcal{X}$.
- Noisy zero-order oracle $\iff$ $y = f(x) + \epsilon$
- Grid approach fails:
    - due to noise



Optimization - recap

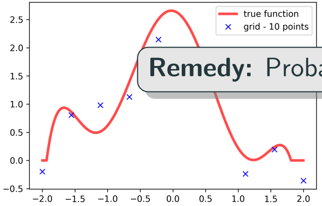
- Assume function $f(x)$ where $x \in \mathcal{X}$.
- Noisy zero-order oracle $\iff$ $y = f(x) + \epsilon$
- Grid approach fails:
  - due to noise



  - due to efficiency [to come]

Optimization - recap

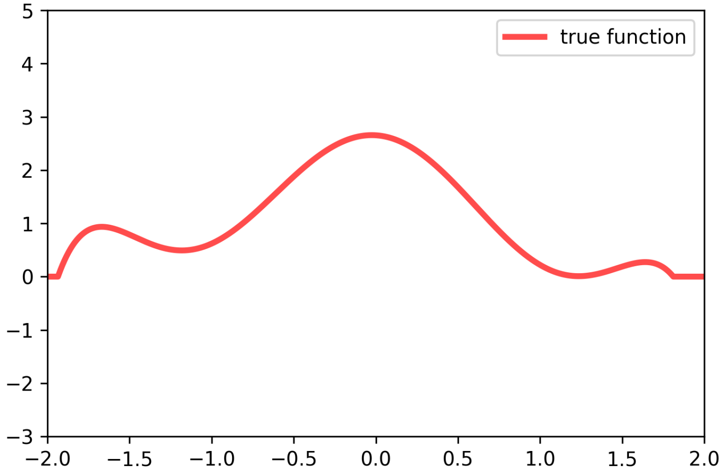
- Assume function $f(x)$ where $x \in \mathcal{X}$.
- Noisy zero-order oracle $\iff$ $y = f(x) + \epsilon$
- Grid approach fails:
  - due to noise



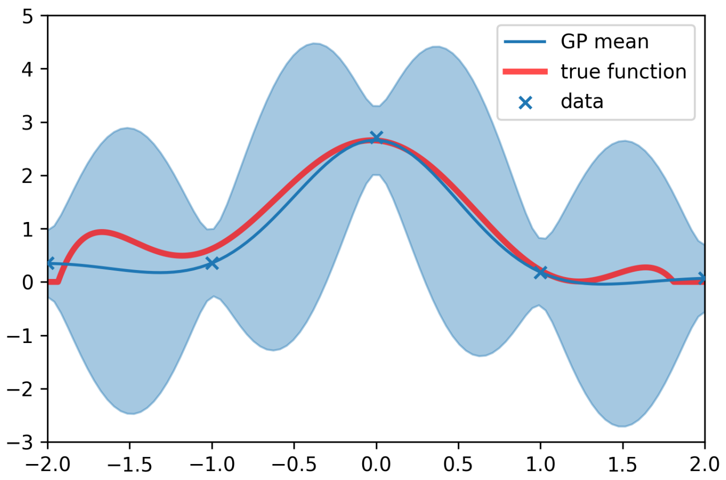
Remedy: Probabilistic model: Gaussian Processes!

  - due to efficiency [to come]

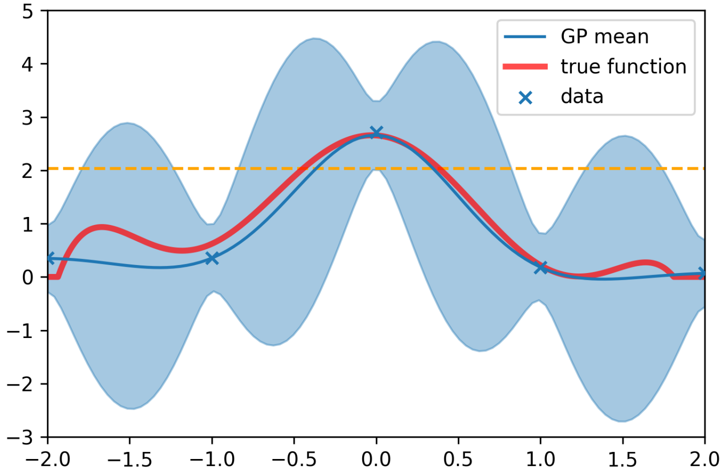
GPs: Avoid not necessary evaluations



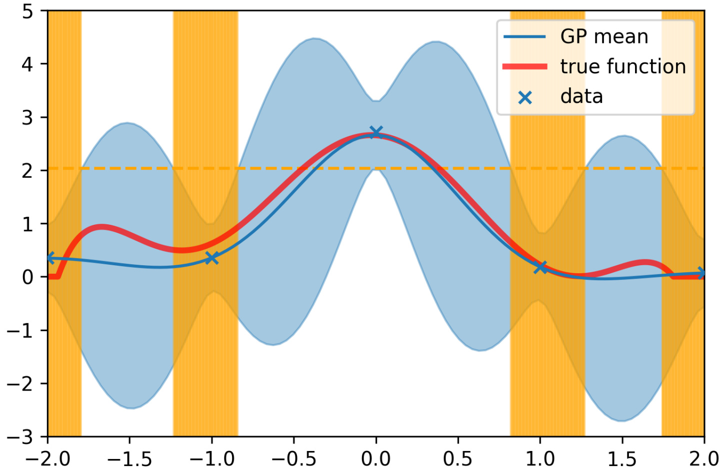
GPs: Avoid not necessary evaluations



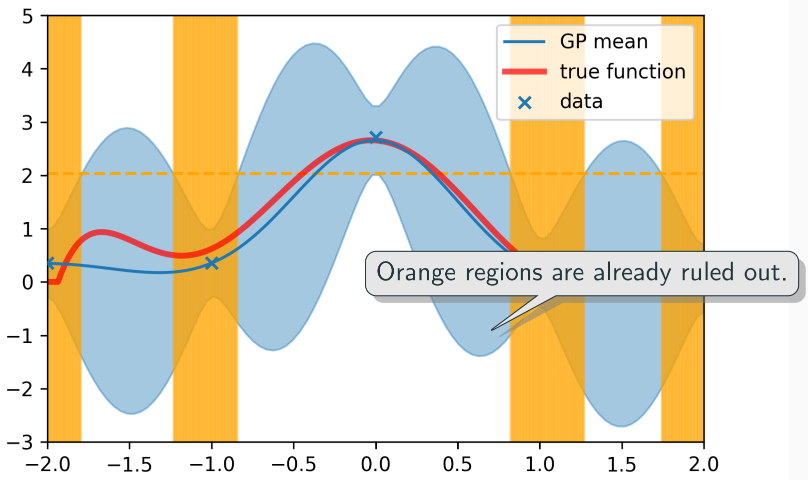
GPs: Avoid not necessary evaluations



GPs: Avoid not necessary evaluations



GPs: Avoid not necessary evaluations



Bayesian Optimization: Overview

Prior data set: $\mathcal{D}_0$

Bayesian Optimization: Overview

Prior data set: $\mathcal{D}_0$

For each step $t = 1, 2, 3, \ldots, T$,

`Step 1:` Build *probabilistic model* $\hat{f}_t$ of the objective using data $\mathcal{D}_{t-1}$

▷ Gaussian process regression (**Part I**)

Bayesian Optimization: Overview

Prior data set: $\mathcal{D}_0$

For each step $t = 1, 2, 3, \ldots, T$,

`Step 1:` Build *probabilistic model* $\hat{f}_t$ of the objective using data $\mathcal{D}_{t-1}$

▷ Gaussian process regression (**Part I**)

`Step 2:` Reduce model *uncertainty about maximizers*

▷ Search guided by acquisition function $x_t = \arg\max_{x \in \mathcal{X}} \alpha(x|\hat{f}_t)$ (**Part II**)

Bayesian Optimization: Overview

Prior data set: $\mathcal{D}_0$

For each step $t = 1, 2, 3, \ldots, T$,

`Step 1:` Build *probabilistic model* $\hat{f}_t$ of the objective using data $\mathcal{D}_{t-1}$

▷ Gaussian process regression (**Part I**)

`Step 2:` Reduce model *uncertainty about maximizers*

▷ Search guided by acquisition function $x_t = \arg\max_{x \in \mathcal{X}} \alpha(x|\hat{f}_t)$ (**Part II**)

`Step 3:` Observe (noisy) measurement $y_t = f(x_t) + \epsilon$

▷ Augment data $\mathcal{D}_t = \mathcal{D}_{t-1} \cup \{(x_t, y_t)\}$

Bayesian Optimization: Overview

Prior data set: $\mathcal{D}_0$

For each step $t = 1, 2, 3, \ldots, T$,

`Step 1:` Build *probabilistic model* $\hat{f}_t$ of the objective using data $\mathcal{D}_{t-1}$

▷ Gaussian process regression (**Part I**)

`Step 2:` Reduce model *uncertainty about maximizers*

▷ Search guided by acquisition function $x_t = \arg\max_{x \in \mathcal{X}} \alpha(x|\hat{f}_t)$ (**Part II**)

`Step 3:` Observe (noisy) measurement $y_t = f(x_t) + \epsilon$

▷ Augment data $\mathcal{D}_t = \mathcal{D}_{t-1} \cup \{(x_t, y_t)\}$

At final time T: *Use model to find best predicted setting*.

Upper Confidence Bound (UCB)

- $\mu_t \ldots$ *posterior mean* after seeing t points

Upper Confidence Bound (UCB)

- $\mu_t \ldots$ *posterior mean* after seeing t points
- $\sigma_t \ldots$ *posterior standard* deviation after seeing t points

Upper Confidence Bound (UCB)

- $\mu_t \ldots$ *posterior mean* after seeing t points
- $\sigma_t \ldots$ *posterior standard* deviation after seeing t points
- $\beta \in \mathbb{R}$ real parameter trading *exploration and exploitation* [see later]

Upper Confidence Bound (UCB)

- $\mu_t \ldots$ *posterior mean* after seeing t points
- $\sigma_t \ldots$ *posterior standard* deviation after seeing t points
- $\beta \in \mathbb{R}$ real parameter trading *exploration and exploitation* [see later]

$$\alpha_t(x) = \mu_t(x) + \beta\sigma_t(x)$$

- How to optimize $\alpha_t(x)$?
  - discretize search space $\mathcal{X}$

Upper Confidence Bound (UCB)

- $\mu_t \ldots$ *posterior mean* after seeing t points
- $\sigma_t \ldots$ *posterior standard* deviation after seeing t points
- $\beta \in \mathbb{R}$ real parameter trading *exploration and exploitation* [see later]

$$\alpha_t(x) = \mu_t(x) + \beta\sigma_t(x)$$

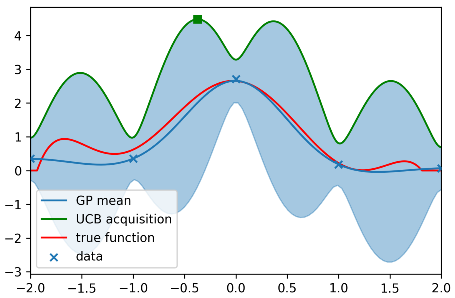
- How to optimize $\alpha_t(x)$?
  - discretize search space $\mathcal{X}$
  - first-order heuristics

Upper Confidence Bound (UCB)

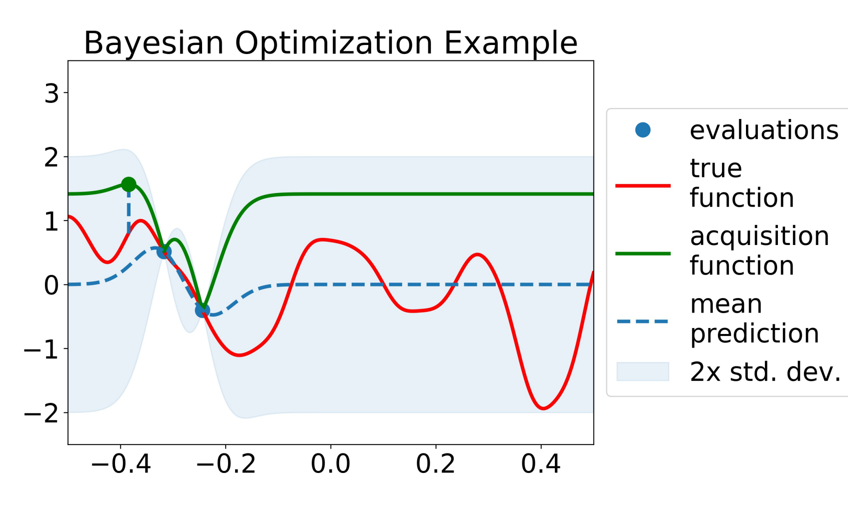
- $\mu_t \ldots$ *posterior mean* after seeing t points
- $\sigma_t \ldots$ *posterior standard* deviation after seeing t points
- $\beta \in \mathbb{R}$ real parameter trading *exploration and exploitation* [see later]

$$\alpha_t(x) = \mu_t(x) + \beta\sigma_t(x)$$

- How to optimize $\alpha_t(x)$?
  - discretize search space $\mathcal{X}$
  - first-order heuristics

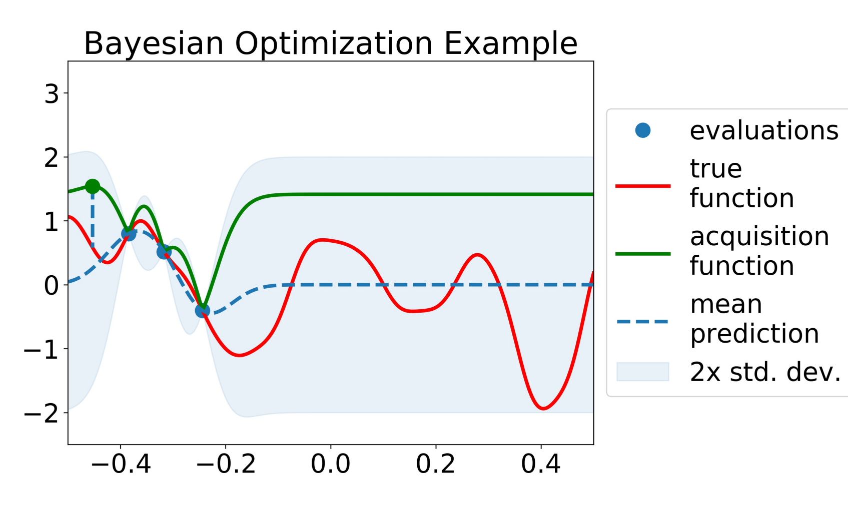


UCB: Example



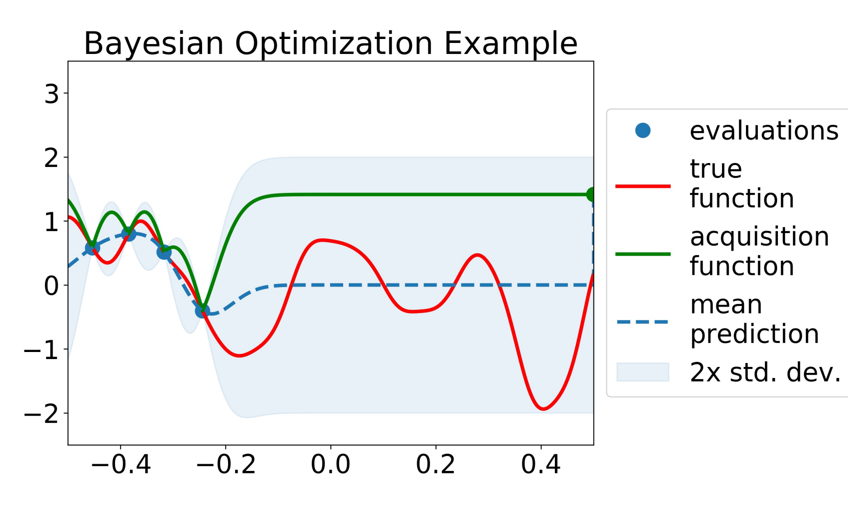
Bayesian Optimization Example
3
2
1
0
−1
−2
−0.4
−0.2
0.0
0.2
0.4
evaluations
true function
acquisition function
mean prediction
2x std. dev.

UCB: Example



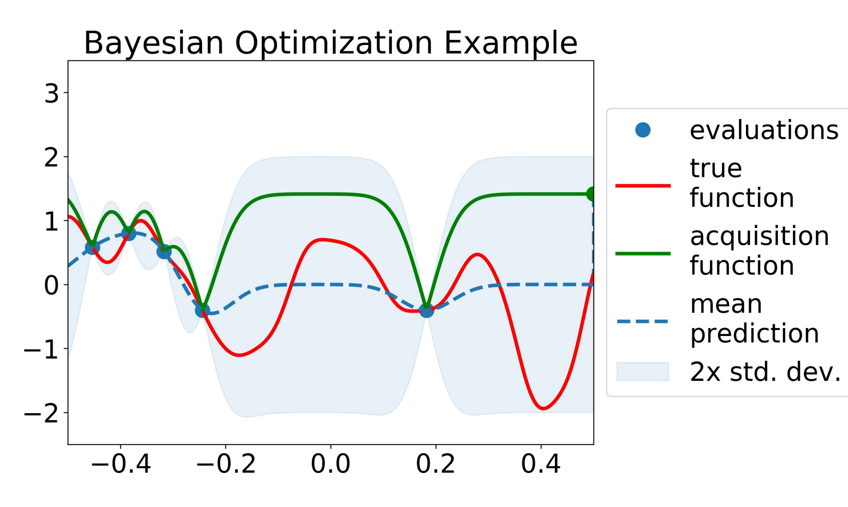
Bayesian Optimization Example
3
2
1
0
−1
−2
−0.4
−0.2
0.0
0.2
0.4
evaluations
true function
acquisition function
mean prediction
2x std. dev.

UCB: Example



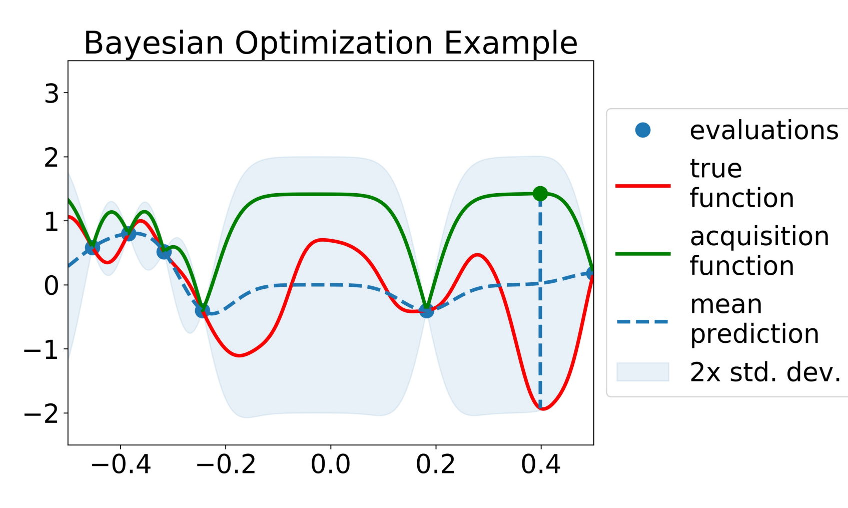
Bayesian Optimization Example
3
2
1
0
−1
−2
−0.4
−0.2
0.0
0.2
0.4
evaluations
true function
acquisition function
mean prediction
2x std. dev.

UCB: Example



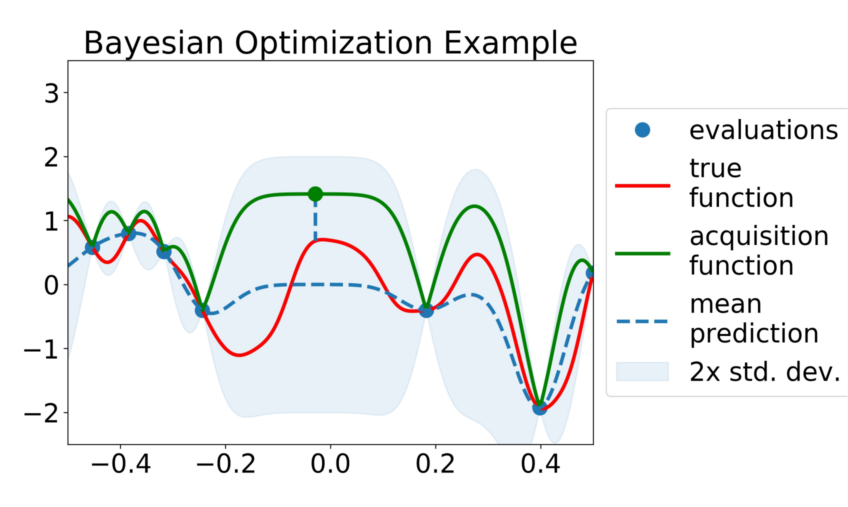
Bayesian Optimization Example
3
2
1
0
−1
−2
−0.4
−0.2
0.0
0.2
0.4
evaluations
true function
acquisition function
mean prediction
2x std. dev.

UCB: Example



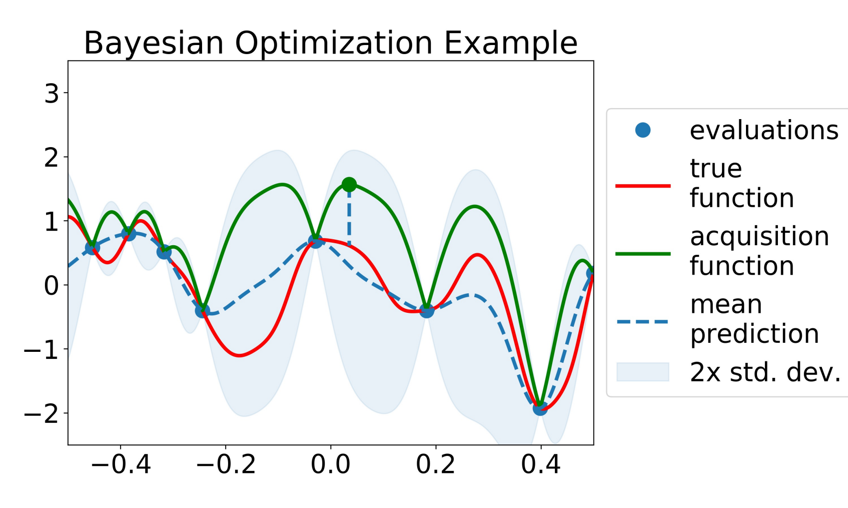
Bayesian Optimization Example
3
2
1
0
−1
−2
−0.4
−0.2
0.0
0.2
0.4
evaluations
true function
acquisition function
mean prediction
2x std. dev.

UCB: Example



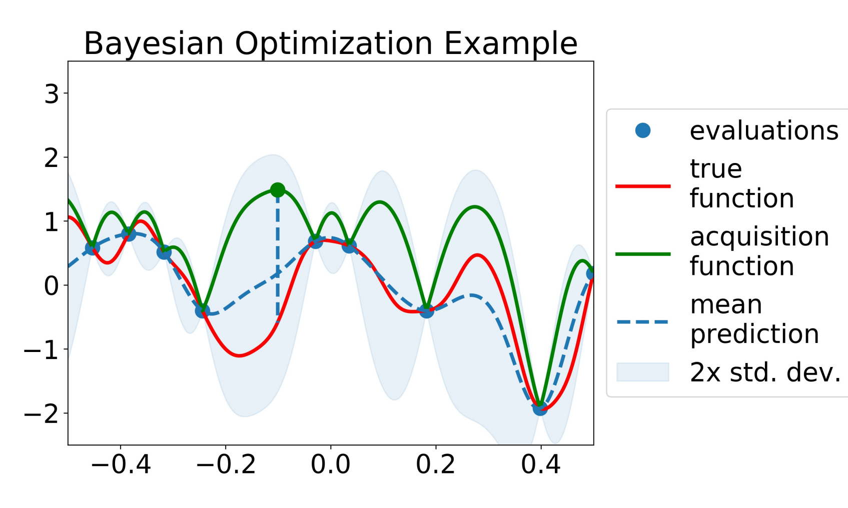
Bayesian Optimization Example
3
2
1
0
−1
−2
−0.4
−0.2
0.0
0.2
0.4
evaluations
true function
acquisition function
mean prediction
2x std. dev.

UCB: Example



Bayesian Optimization Example
3
2
1
0
−1
−2
−0.4
−0.2
0.0
0.2
0.4
evaluations
true function
acquisition function
mean prediction
2x std. dev.

UCB: Example



Bayesian Optimization Example
3
2
1
0
−1
−2
−0.4
−0.2
0.0
0.2
0.4
evaluations
true function
acquisition function
mean prediction
2x std. dev.

Consequence of β

- What is β?

Consequence of β

- What is β?
- β trades exploration and exploitation

Consequence of β

- What is β?
- β trades exploration and exploitation
- Theoretical value that ensure global convergence (right model assumption):

$$\beta_t = 2\log(\gamma_t + 1)$$

Consequence of β

- What is β?
- β trades exploration and exploitation
- Theoretical value that ensure global convergence (right model assumption):

$$\beta_t = 2\log(\gamma_t + 1)$$

where γ_t is maximum information gain, for RBF kernel $\gamma_t = C\log(T)^{d+1}$

Consequence of β

- What is β?
- β trades exploration and exploitation
- Theoretical value that ensure global convergence (right model assumption):

$$\beta_t = 2\log(\gamma_t + 1)$$

where γ_t is maximum information gain, for RBF kernel $\gamma_t = C\log(T)^{d+1}$

- (Very common) *heuristic* approach: $\beta \approx 2$.

Consequence of β

- What is β?
- β trades exploration and exploitation
- Theoretical value that ensure global convergence (right model assumption):

$$\beta_t = 2\log(\gamma_t + 1)$$

where γ_t is maximum information gain, for RBF kernel $\gamma_t = C\log(T)^{d+1}$

- (Very common) *heuristic* approach: $\beta \approx 2$.
- β too small $\implies$ gets stuck/hill climbing

Consequence of β

- What is β?
- β trades exploration and exploitation
- Theoretical value that ensure global convergence (right model assumption):

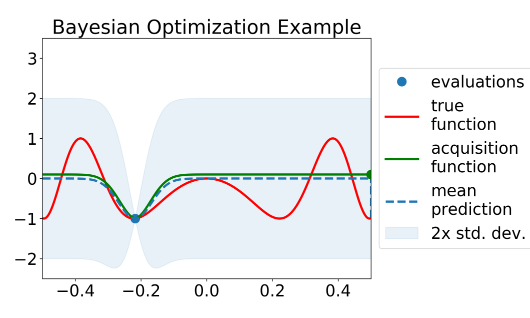
$$\beta_t = 2\log(\gamma_t + 1)$$

where γ_t is maximum information gain, for RBF kernel $\gamma_t = C\log(T)^{d+1}$

- (Very common) *heuristic* approach: $\beta \approx 2$.
- β too small $\implies$ gets stuck/hill climbing
- β too high $\implies$ incremental grid search

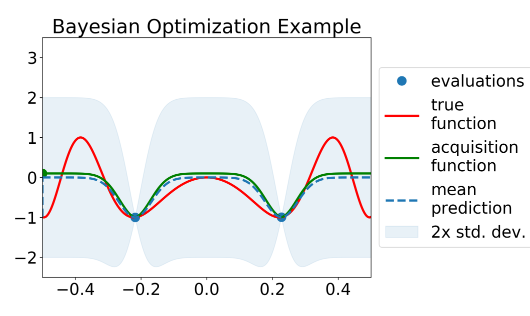
Consequence of β II

- Hill Climbing - β small



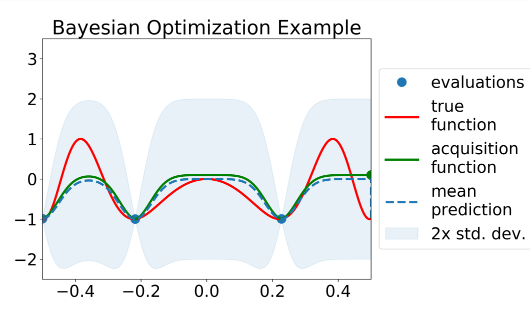
Consequence of β II

- Hill Climbing - β small



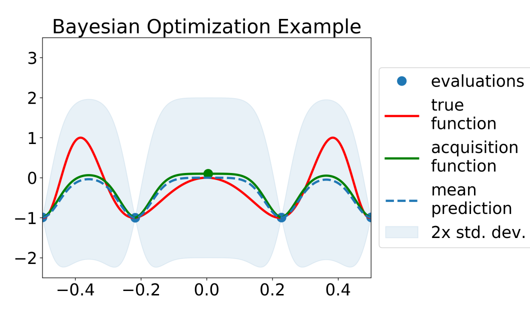
Consequence of β II

- Hill Climbing - β small



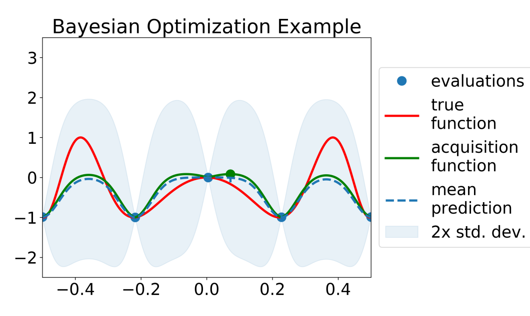
Consequence of β II

- Hill Climbing - β small



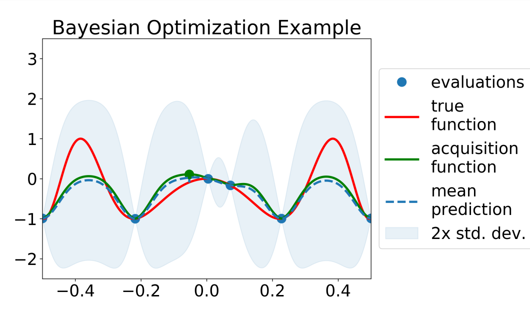
Consequence of β II

- Hill Climbing - β small



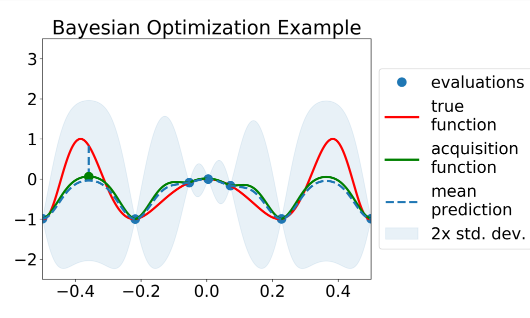
Consequence of β II

- Hill Climbing - β small



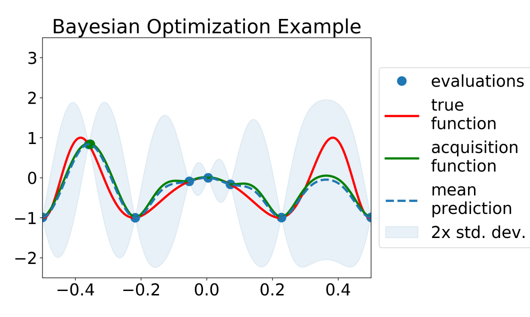
Consequence of β II

- Hill Climbing - β small



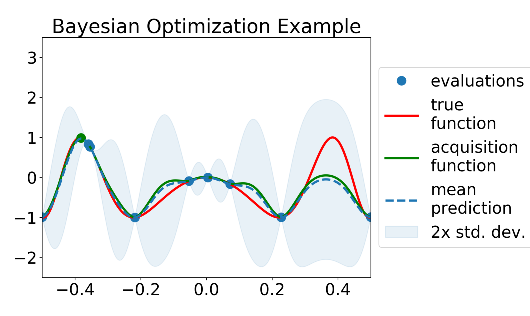
Consequence of β II

- Hill Climbing - β small



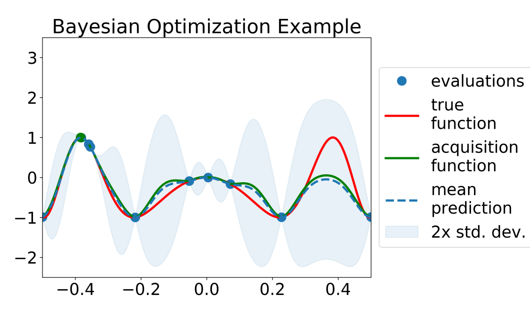
Consequence of β II

- Hill Climbing - β small



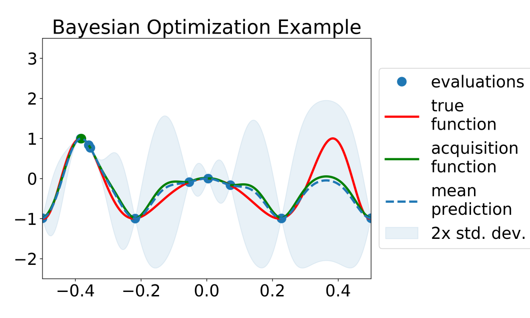
Consequence of β II

- Hill Climbing - β small



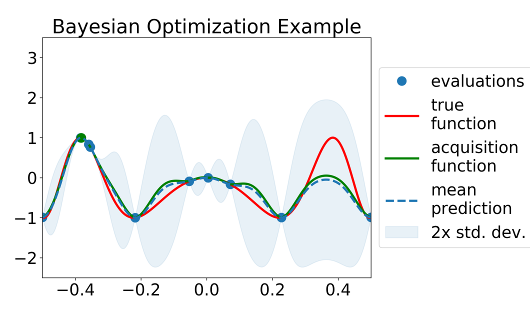
Consequence of β II

- Hill Climbing - β small

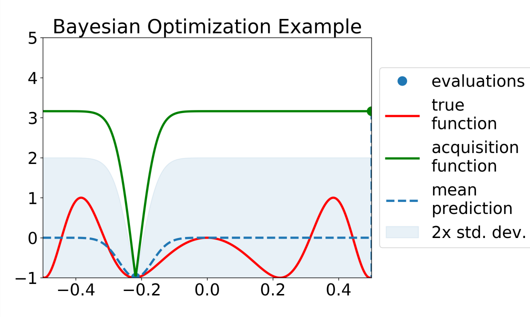


Consequence of β II

- Hill Climbing - β small

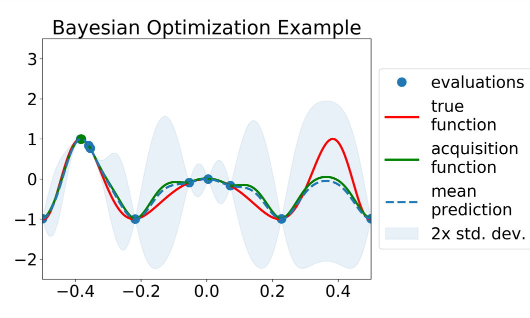


- Sequential Grid - β large

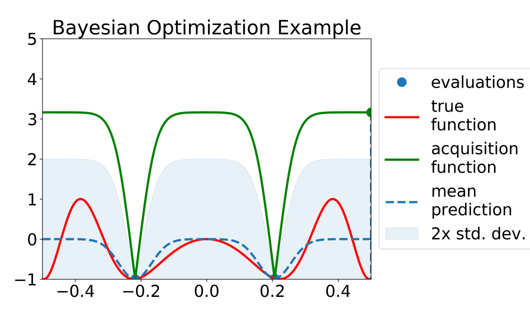


Consequence of β II

- Hill Climbing - β small

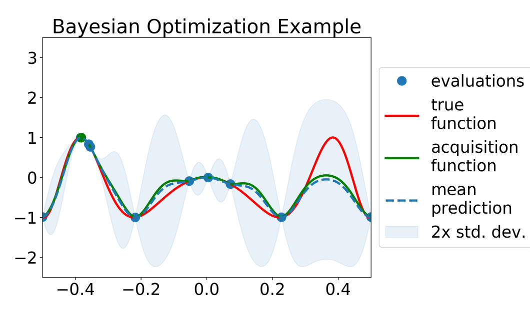


- Sequential Grid - β large

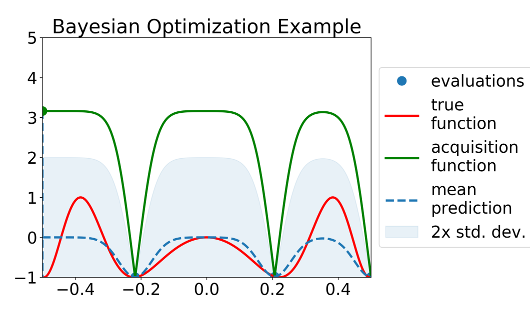


Consequence of β II

- Hill Climbing - β small

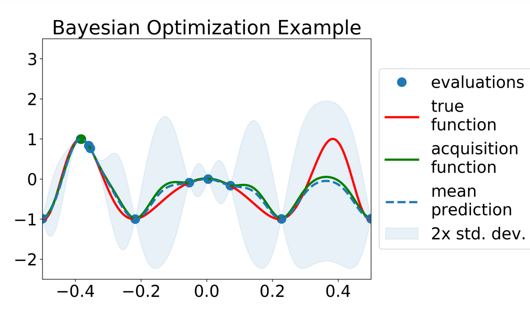


- Sequential Grid - β large

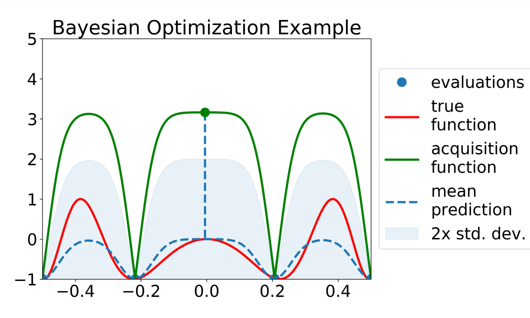


Consequence of β II

- Hill Climbing - β small

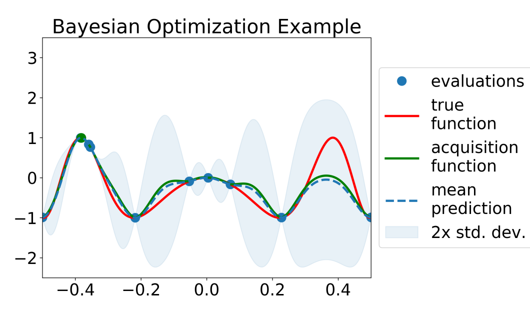


- Sequential Grid - β large

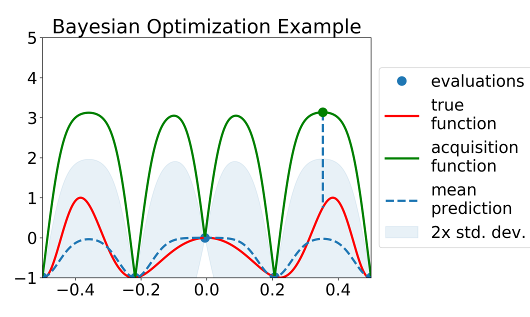


Consequence of β II

- Hill Climbing - β small

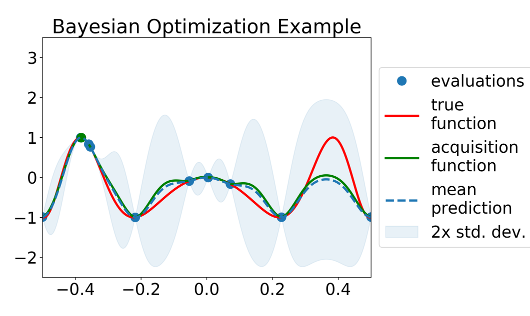


- Sequential Grid - β large

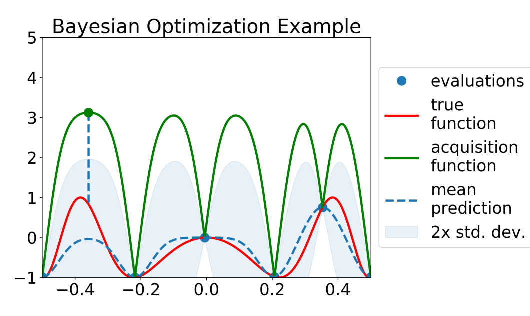


Consequence of β II

- Hill Climbing - β small

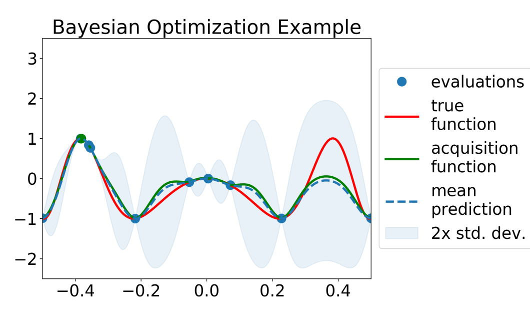


- Sequential Grid - β large

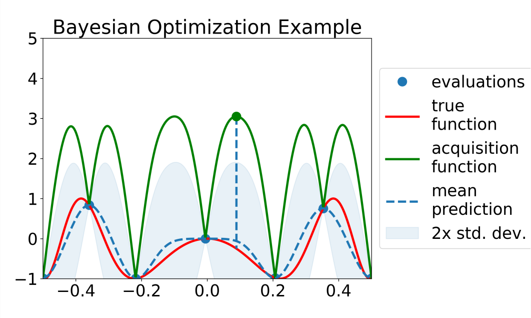


Consequence of β II

- Hill Climbing - β small



- Sequential Grid - β large



Other acquisition function

- *Thompson sampling*

Other acquisition function

- *Thompson sampling*
  - Sample a path $s \sim \mathrm{GP}(\mu_t, \sigma_t)$

Other acquisition function

- *Thompson sampling*
  - Sample a path $s \sim \mathrm{GP}(\mu_t, \sigma_t)$
  - Acquisition $\alpha_t(x) = s(x)$

Other acquisition function

- *Thompson sampling*
  - Sample a path $s \sim \mathrm{GP}(\mu_t, \sigma_t)$
  - Acquisition $\alpha_t(x) = s(x)$
  - Empirically works often better

Other acquisition function

- *Thompson sampling*
  - Sample a path $s \sim \text{GP}(\mu_t, \sigma_t)$
  - Acquisition $\alpha_t(x) = s(x)$
  - Empirically works often better
- *Expected Improvement* [Mockus, 1982]

Other acquisition function

- *Thompson sampling*
  - Sample a path $s \sim \mathrm{GP}(\mu_t, \sigma_t)$
  - Acquisition $\alpha_t(x) = s(x)$
  - Empirically works often better
- *Expected Improvement* [Mockus, 1982]
  - $\mu_t(x^+)$ is the best mean estimate

Other acquisition function

- *Thompson sampling*
  - Sample a path $s \sim \mathrm{GP}(\mu_t, \sigma_t)$
  - Acquisition $\alpha_t(x) = s(x)$
  - Empirically works often better
- *Expected Improvement* [Mockus, 1982]
  - $\mu_t(x^+)$ is the best mean estimate
  - $\alpha_t(x) = \mathbb{E}[\max(0, f(x) - f(x^+)|\mathcal{D}_t]$

Other acquisition function

- *Thompson sampling*
  - Sample a path $s \sim \mathrm{GP}(\mu_t, \sigma_t)$
  - Acquisition $\alpha_t(x) = s(x)$
  - Empirically works often better
- *Expected Improvement* [Mockus, 1982]
  - $\mu_t(x^+)$ is the best mean estimate
  - $\alpha_t(x) = \mathbb{E}[\max(0, f(x) - f(x^+)|\mathcal{D}_t]$
  - Analytical solution: $\alpha_t(x) = (\mu_t(x) - \mu(x^+))\Phi(Z) + \sigma(x)\phi(Z)$ where, $Z = \frac{\mu_t - \mu(x^+)}{\sigma_t(x)}$ and Φ, ϕ are cdf and pdf of standard normal.

Other acquisition function

- *Thompson sampling*
  - Sample a path $s \sim \mathrm{GP}(\mu_t, \sigma_t)$
  - Acquisition $\alpha_t(x) = s(x)$
  - Empirically works often better
- *Expected Improvement* [Mockus, 1982]
  - $\mu_t(x^+)$ is the best mean estimate
  - $\alpha_t(x) = \mathbb{E}[\max(0, f(x) - f(x^+)|\mathcal{D}_t]$
  - Analytical solution: $\alpha_t(x) = (\mu_t(x) - \mu(x^+))\Phi(Z) + \sigma(x)\phi(Z)$ where, $Z = \frac{\mu_t - \mu(x^+)}{\sigma_t(x)}$ and Φ, ϕ are cdf and pdf of standard normal.

Curse of dimensionality

- How do we apply this to multiple dimensions?

Curse of dimensionality

- How do we apply this to multiple dimensions?
- Naturally, $\alpha_t(x)$ can be defined in any $\mathcal{X} \subset \mathbb{R}^d$

Curse of dimensionality

- How do we apply this to multiple dimensions?
- Naturally, $\alpha_t(x)$ can be defined in any $\mathcal{X} \subset \mathbb{R}^d$
- Practically, $\alpha_t(x)$ cannot be optimized using a grid optimizer.

Curse of dimensionality

- How do we apply this to multiple dimensions?
- Naturally, $\alpha_t(x)$ can be defined in any $\mathcal{X} \subset \mathbb{R}^d$
- Practically, $\alpha_t(x)$ cannot be optimized using a grid optimizer.
- The size of the grid grows n^d and computational needs grow as $(n^d)^3$, where

n number of grid points in 1D.

Curse of dimensionality

- ▷ How do we apply this to multiple dimensions?
- ▷ Naturally, $\alpha_t(x)$ can be defined in any $\mathcal{X} \subset \mathbb{R}^d$
- ▷ Practically, $\alpha_t(x)$ cannot be optimized using a grid optimizer.
- ▷ The size of the grid grows n^d and computational needs grow as $(n^d)^3$, where n number of grid points in 1D.
- ▷ One can use a first-order heuristic to optimize the acquisition function locally.

Curse of dimensionality

- How do we apply this to multiple dimensions?
- Naturally, $\alpha_t(x)$ can be defined in any $\mathcal{X} \subset \mathbb{R}^d$
- Practically, $\alpha_t(x)$ cannot be optimized using a grid optimizer.
- The size of the grid grows n^d and computational needs grow as $(n^d)^3$, where n number of grid points in 1D.
- One can use a first-order heuristic to optimize the acquisition function locally.
- More advanced methods: Look [Mutný and Krause, 2018] or visit: **Talk of Johannes tomorrow on BO for SwissFEL.**

Curse of dimensionality

- How do we apply this to multiple dimensions?
- Naturally, $\alpha_t(x)$ can be defined in any $\mathcal{X} \subset \mathbb{R}^d$
- Practically, $\alpha_t(x)$ cannot be optimized using a grid optimizer.
- The size of the grid grows n^d and computational needs grow as $(n^d)^3$, where n number of grid points in 1D.
- One can use a first-order heuristic to optimize the acquisition function locally.
- More advanced methods: Look [Mutný and Krause, 2018] or visit: **Talk of Johannes tomorrow on BO for SwissFEL.**

Part II, Programming: Lets try it out.

Ekström, A., Forssén, C., Dimitrakakis, C., Dubhashi, D., Johansson, H., Muhammad, A., Salomonsson, H., and Schliep, A. (2019).
Bayesian optimization in ab initio nuclear physics.
arXiv preprint arXiv:1902.00941.

Kirschner, J., Mutný, M., Hiller, N., Ischebeck, R., and Krause, A. (2019).
Adaptive and safe bayesian optimization in high dimensions via one-dimensional subspaces.

McIntire, M., Cope, T., Ratner, D., and Ermon, S. (2016).
Bayesian optimization of fel performance at lcls.
Proceedings of IPAC2016.

Mockus, J. (1982).
The bayesian approach to global optimization.
System Modeling and Optimization, pages 473–481.

Mutný, M. and Krause, A. (2018).
Efficient high dimensional bayesian optimization with additivity and quadrature fourier features.
In *Neural and Information Processing Systems (NeurIPS)*.

Rasmussen, C. E. (2004).
Gaussian processes in machine learning.
In *Advanced lectures on machine learning*, pages 63–71. Springer.

Romero, P. A., Krause, A., and Arnold, F. H. (2013).
Navigating the protein fitness landscape with gaussian processes.
Proceedings of the National Academy of Sciences (PNAS), 110(3).

Schneider, P.-I., Santiago, X. G., Soltwisch, V., Hammerschmidt, M., Burger, S., and Rockstuhl, C. (2018).

Benchmarking five global optimization approaches for nano-optical shape optimization and parameter reconstruction.