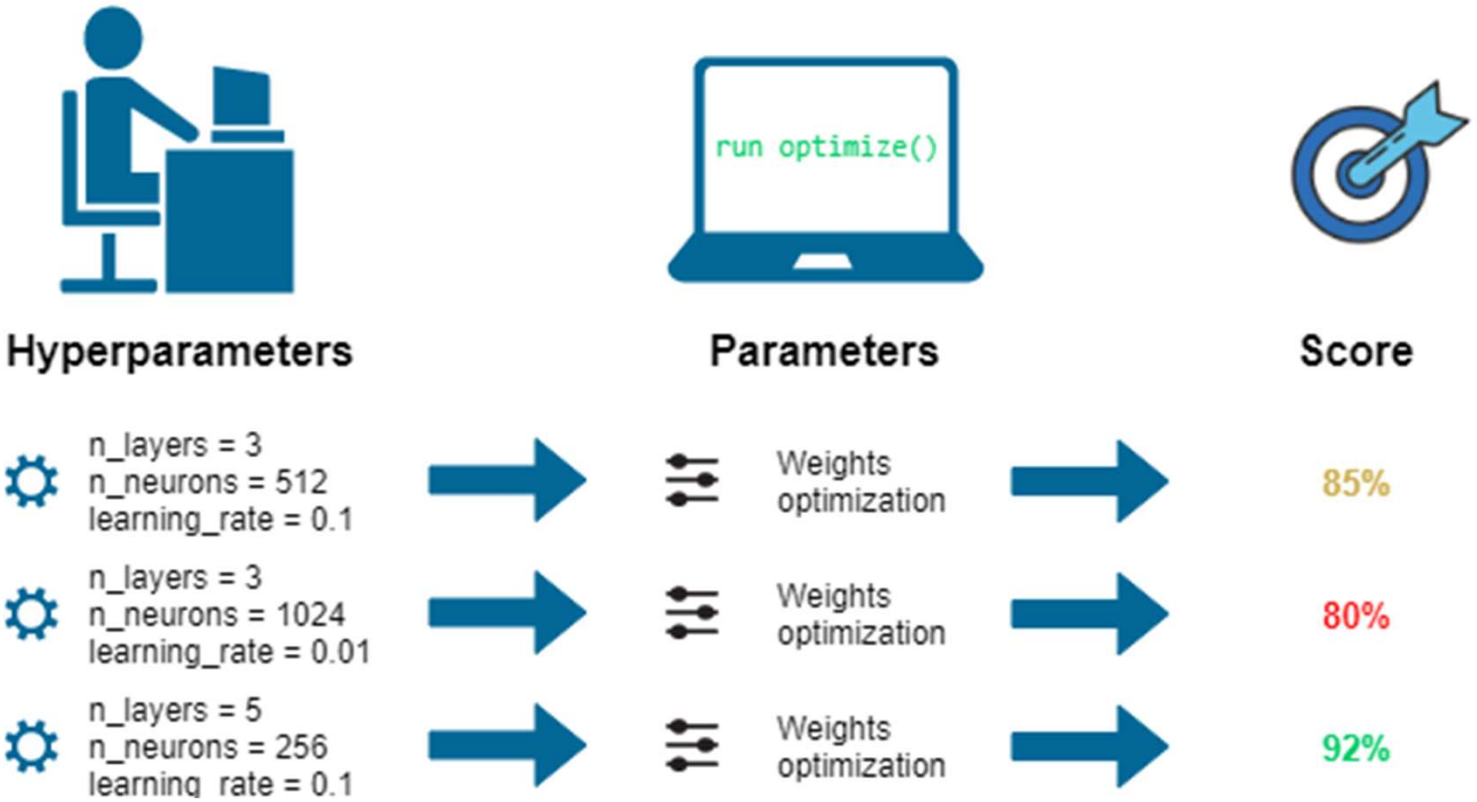
Recent Advances in Bayesian Optimization

Dr Vu Nguyen
vu@robots.ox.ac.uk
University of Oxford

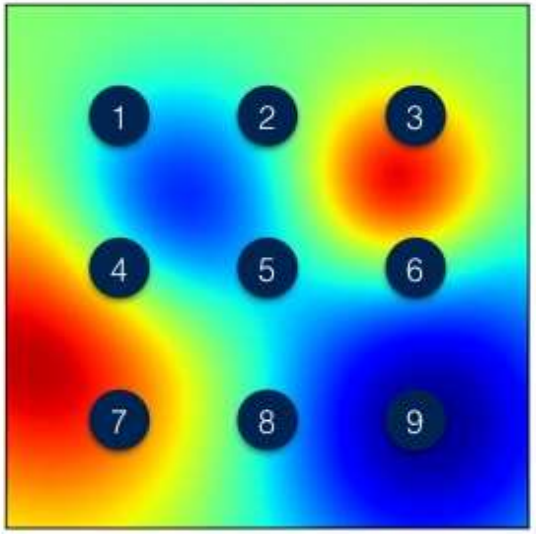
Hyperparameters Optimization

- ML algorithm's performances depend on hyper-parameters.
- Finding the best hyperparameters for the highest performance

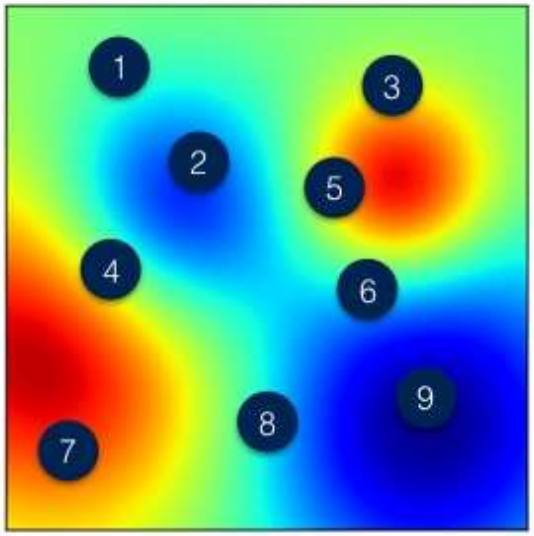


Traditional Hyper-parameter Tuning

- Grid Search:
  - Create a list of values for each parameter.
  - Consider all possible combinations of these values.
  - Exhaustively evaluate the model and choose the best parameter.

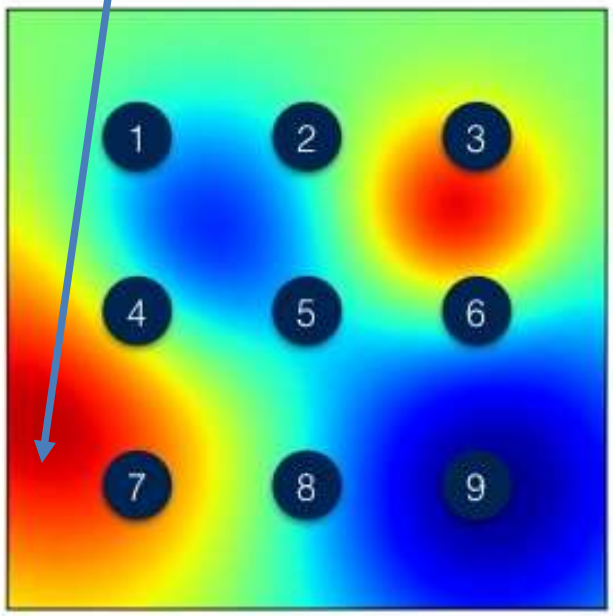


- Random Search:
  - Randomly select a parameter to evaluate.
  - Select the best parameter.



Grid vs Random vs Bayesian Optimization

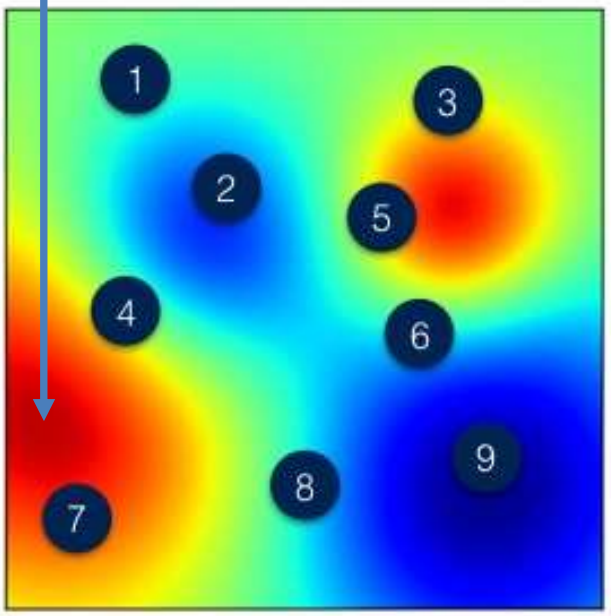
Missed
optimum location



Grid Search

FAILED

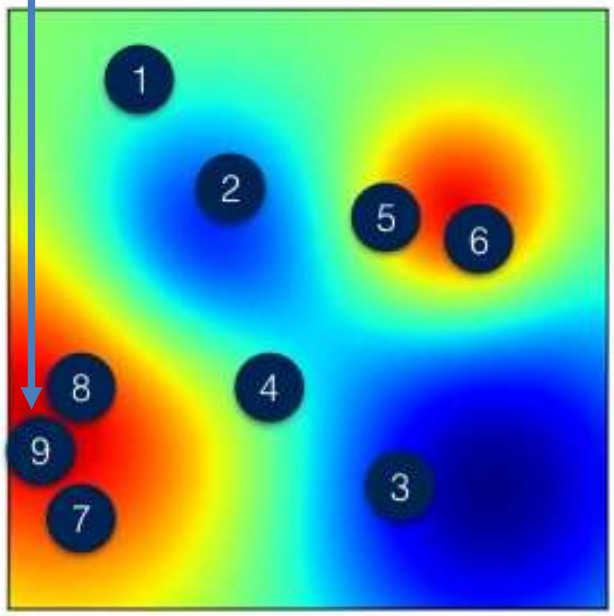
Missed
optimum location



Random Search

FAILED

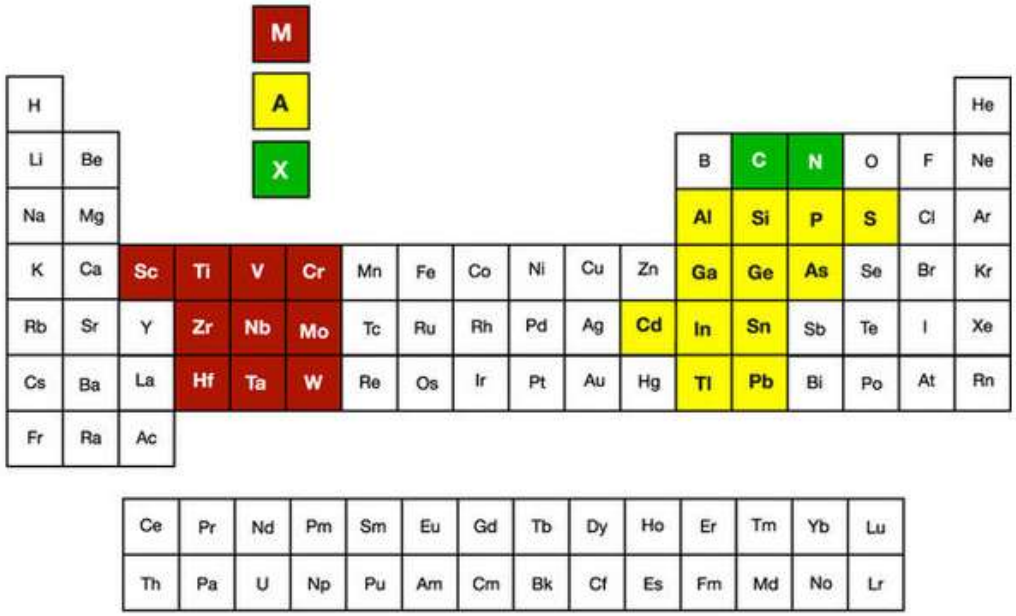
Found
optimum location



Bayesian Optimization

Another Example: Alloy Development

- Alloy composition: $X = [\%\, Al, \%\, Co, \%Fe, \%\mathrm{Cu}, \%\mathrm{C} \ldots]$
- Strength: $\mathbf{y}$
- Goal: find the best composition X for the highest strength y.

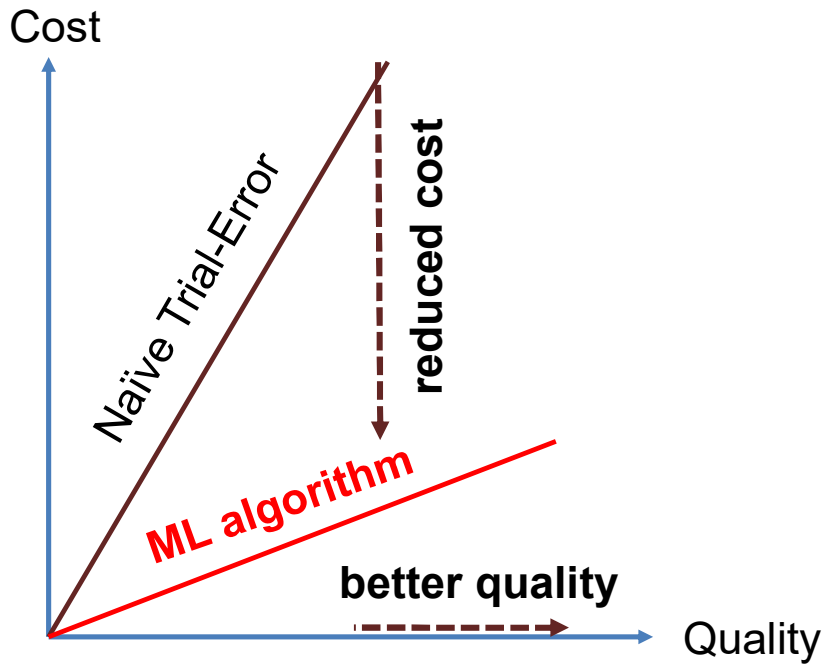


Trial-Error Approach

- Trial-error approach is typically used for alloy development using expert knowledge

The Problem is Expensive Cost and Time

- 1 alloy testing = **1 day** and $100
- 100 experiments = **3 months** and $10,000
- Even with 100 experiments, trial-error still **can not** get the optimum solution



2 hours Tutorial on Bayesian Optimization

- Goal: introduce the Bayesian Optimization - **techniques**, **applications** and future **research directions**.
  - Broad summary of recent advances in
    - Batch Bayesian Optimization
    - High dimensional Bayes Opt
    - Mixed Categorical-Continuous Bayes Opt
  - A **released package MiniBO**.
  - 30 papers are surveyed and organized in this talk, but they are by no means to complete.
- Tutorial website: vu-nguyen.org/BOTutorial_ACML20.html

Agenda

- Hyperparameter Tuning and Experimental Design as Black-Boxes
- Part I: Bayesian Optimization [1 hour]
- Part II: Recent Advances in Bayesian Optimization [1 hour]
  - Batch Bayesian Optimization
  - High dimensional Bayes Opt
  - Mixed Categorical-Continuous Bayes Opt
- Future Research Directions in Bayesian Optimization

Part I: Bayesian Optimization

Agenda

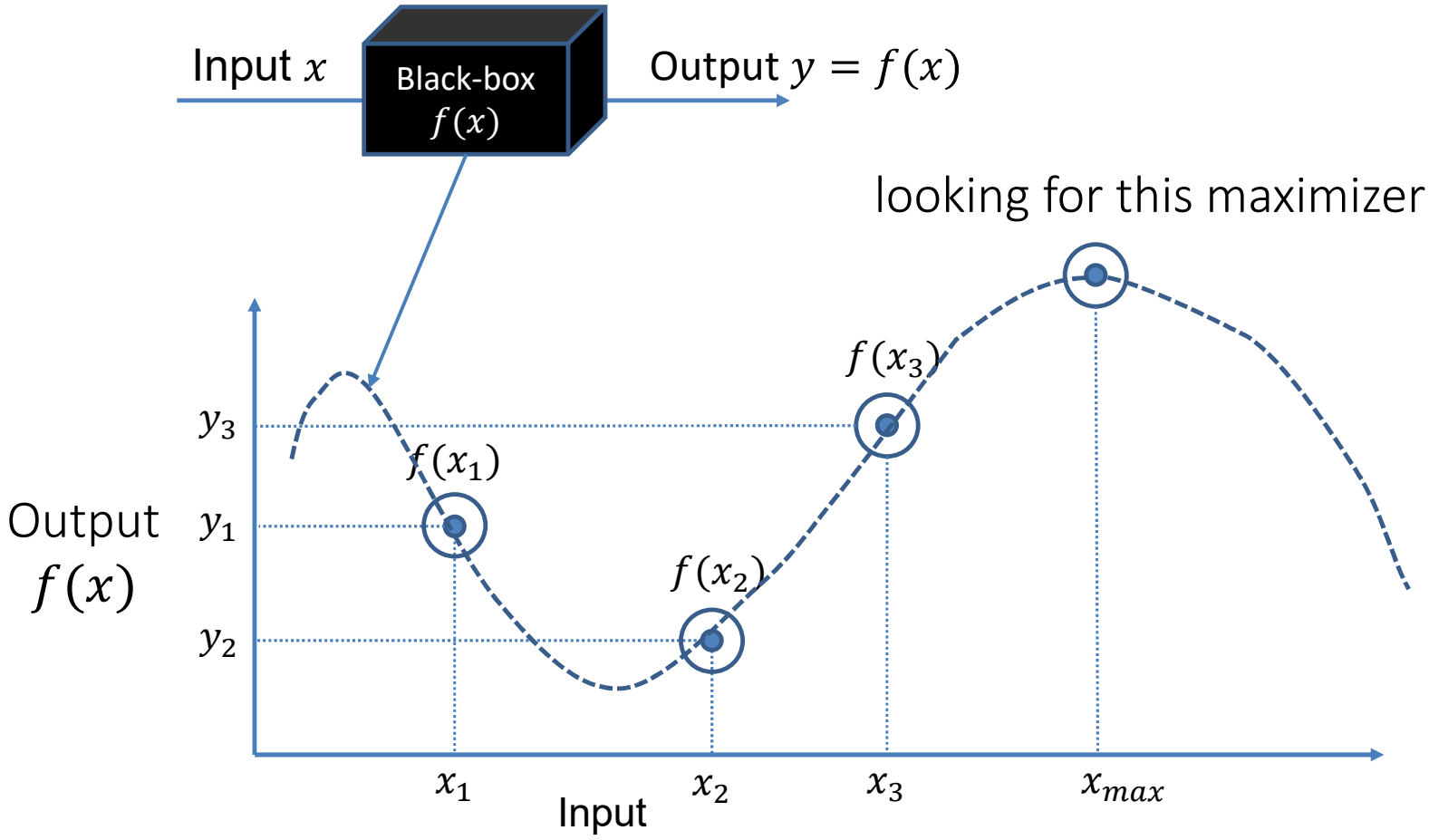
- Hyperparameter Tuning and Experimental Design as Black-Boxes
- Part I: Bayesian Optimization
- Part II: Recent Advances in Bayesian Optimization
  - Batch Bayesian Optimization
  - High dimensional Bayes Opt
  - Mixed Categorical-Continuous Bayes Opt
- Research Directions in Bayesian Optimization

Outline Part 1: Bayesian Optimization

- Bayesian Optimization
- Gaussian Processes
- Acquisition Functions
- Applications
- Demo Mini BayesOpt

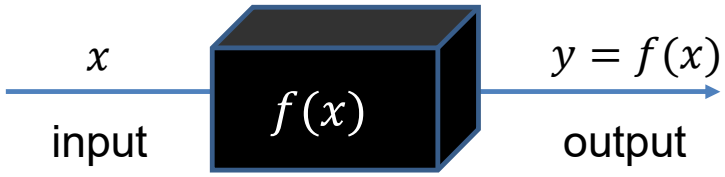
Black-box Optimization

The relationship from x to y is through the black-box.



Properties of Black-box Function

$$f: X \in \mathcal{R}^d \rightarrow Y \in \mathcal{R}$$



Function form is not known $\cancel{y = ax + b}$

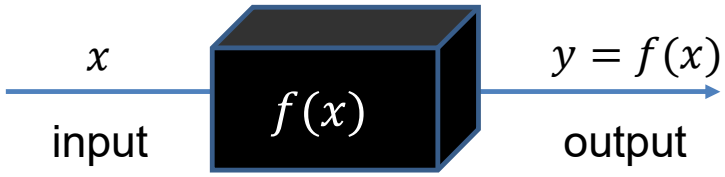
No derivative form $\cancel{\frac{\partial f}{\partial x} = \cdots}$

Expensive to evaluate (in time and cost)

Nothing is known about the function, except a few evaluations $y = f(x)$

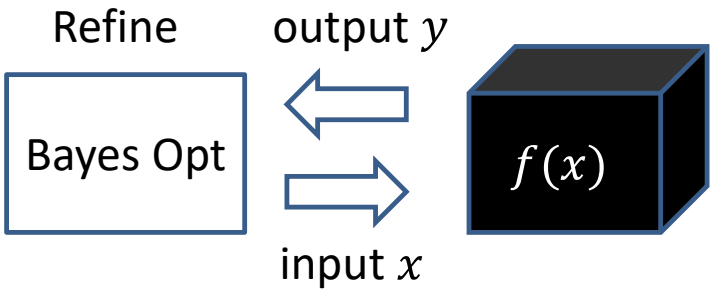
Bayesian Optimization

- The goal is to optimize the black-box function $\boldsymbol{x}^* = \underset{\boldsymbol{x} \in \mathcal{X}}{\operatorname{argmax}} f(\boldsymbol{x})$.



- Bayes opt makes a series of evaluations $\mathbf{x}_1, x_2, \ldots, x_T$ such that the maximum of f is found in the fewest iterations.

Bayesian Optimization Overview

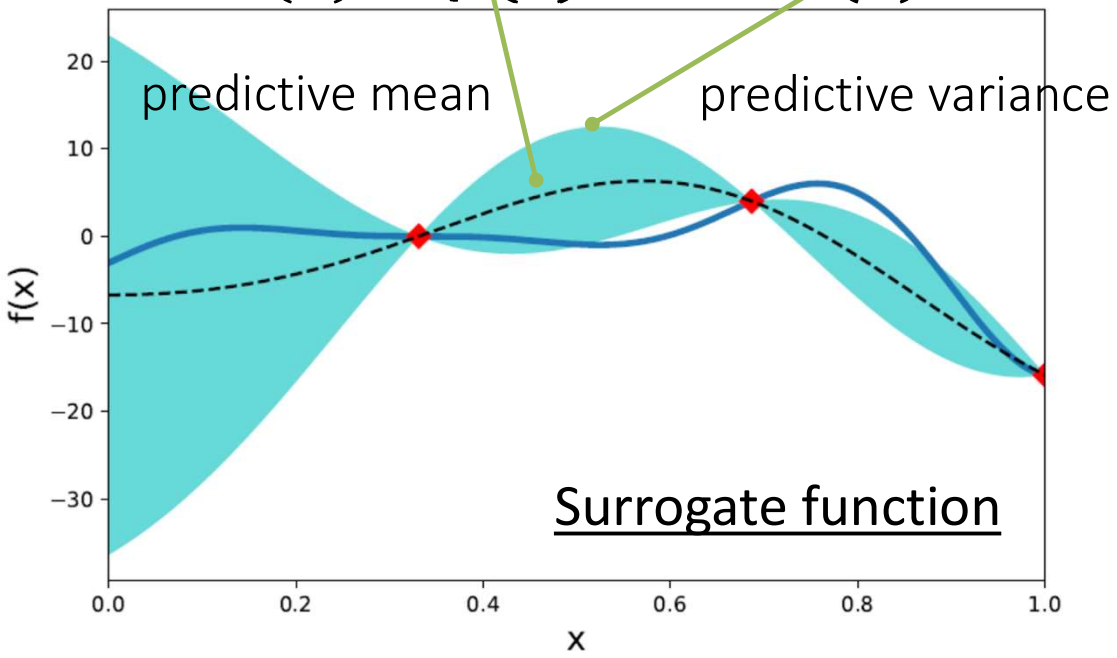


- Make a series of evaluations $x_1, x_2, \ldots x_T$

Acquisition function $\alpha(x) = \boldsymbol{\mu(x)} + \kappa \times \sigma(x)$

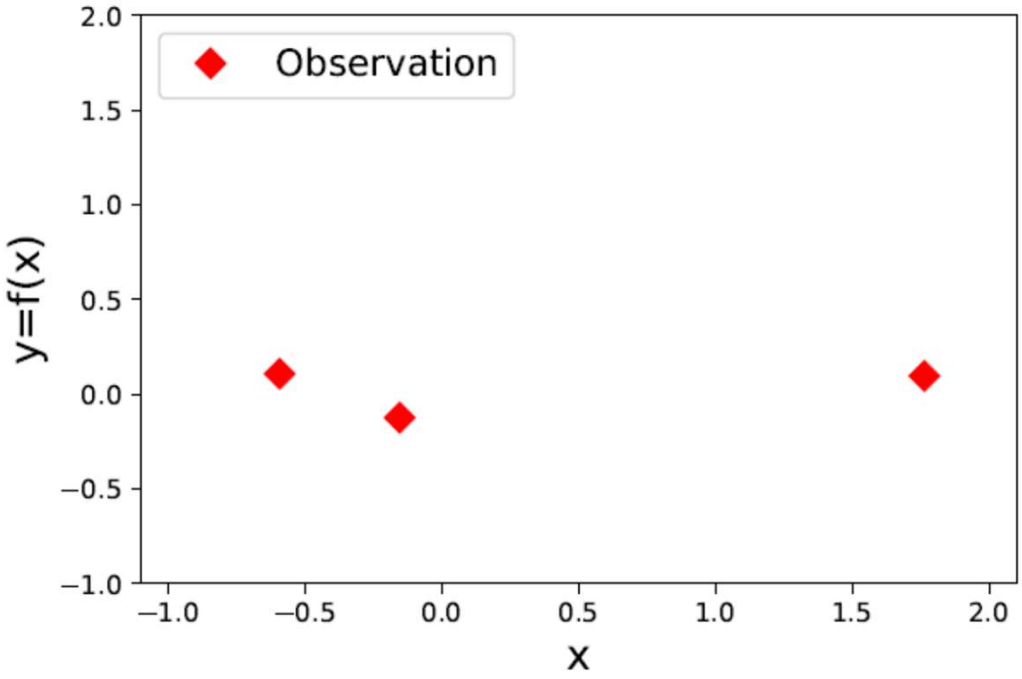
exploit

explore



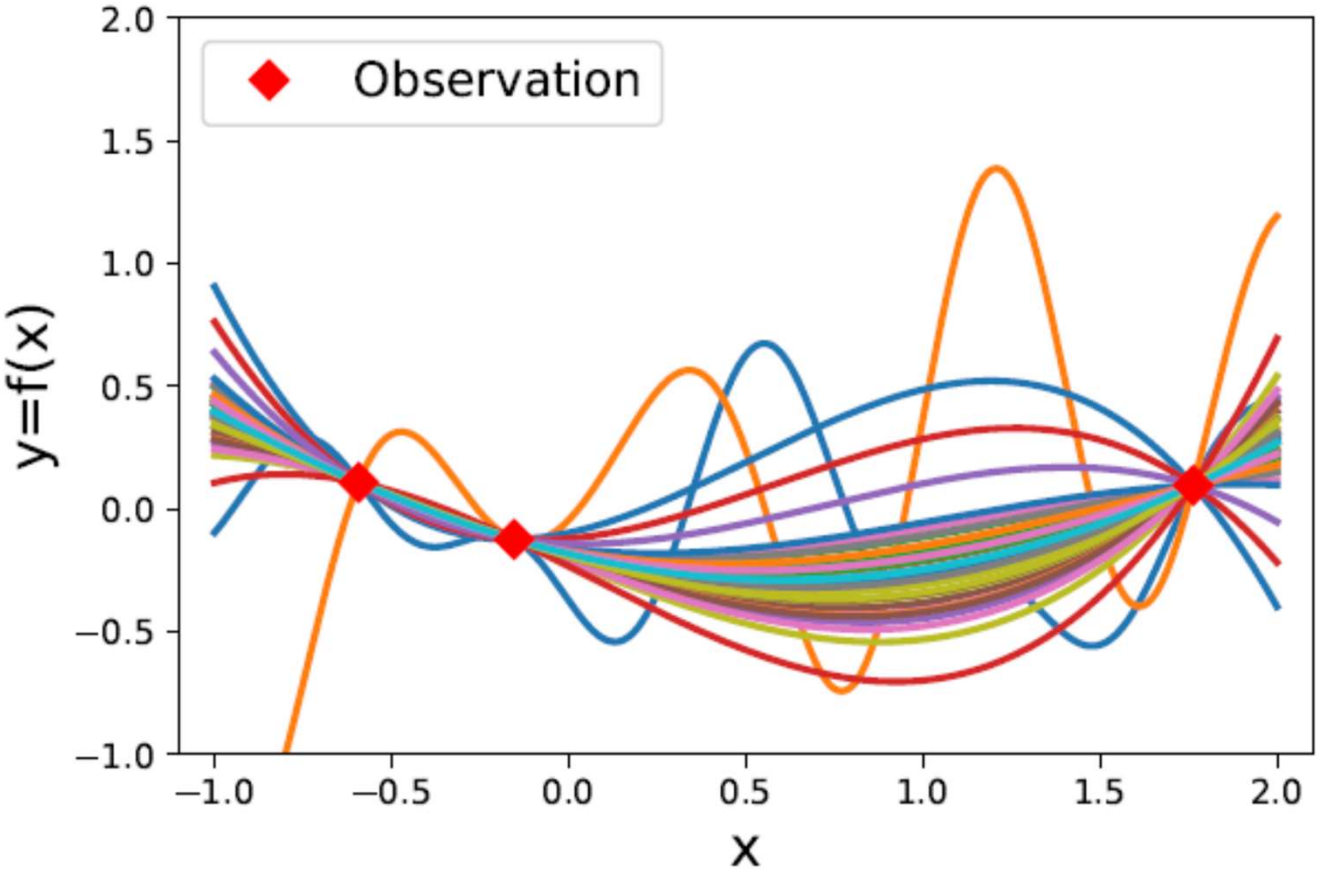
Motivation

- Given the observations from black-box function.
- Our goal is to find the global maximizer.
- Where should we evaluate next?



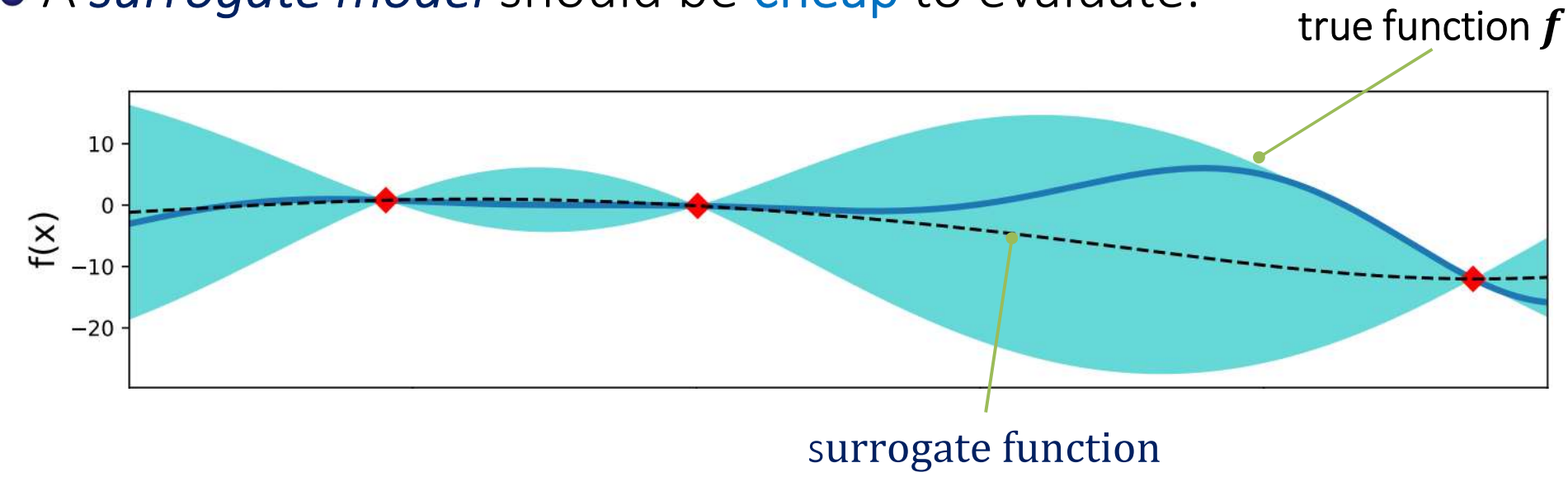
Motivation

- Each line represents our belief about the underlying function given three observations.



Bayesian Optimization uses Surrogate Models

- We define a *surrogate model* to learn and update such belief.
- A *surrogate model* mimics the behaviour of the *true function f* as closely as possible.
- A *surrogate model* should be cheap to evaluate.



Bayesian Optimization

1. Choose a surrogate model (prior) over the possible spaces of *f*.

2. Combine the prior and the likelihood (from the new evaluation) to get the posterior for the surrogate.

3. Use the posterior to build the acquisition function to select the next evaluation.

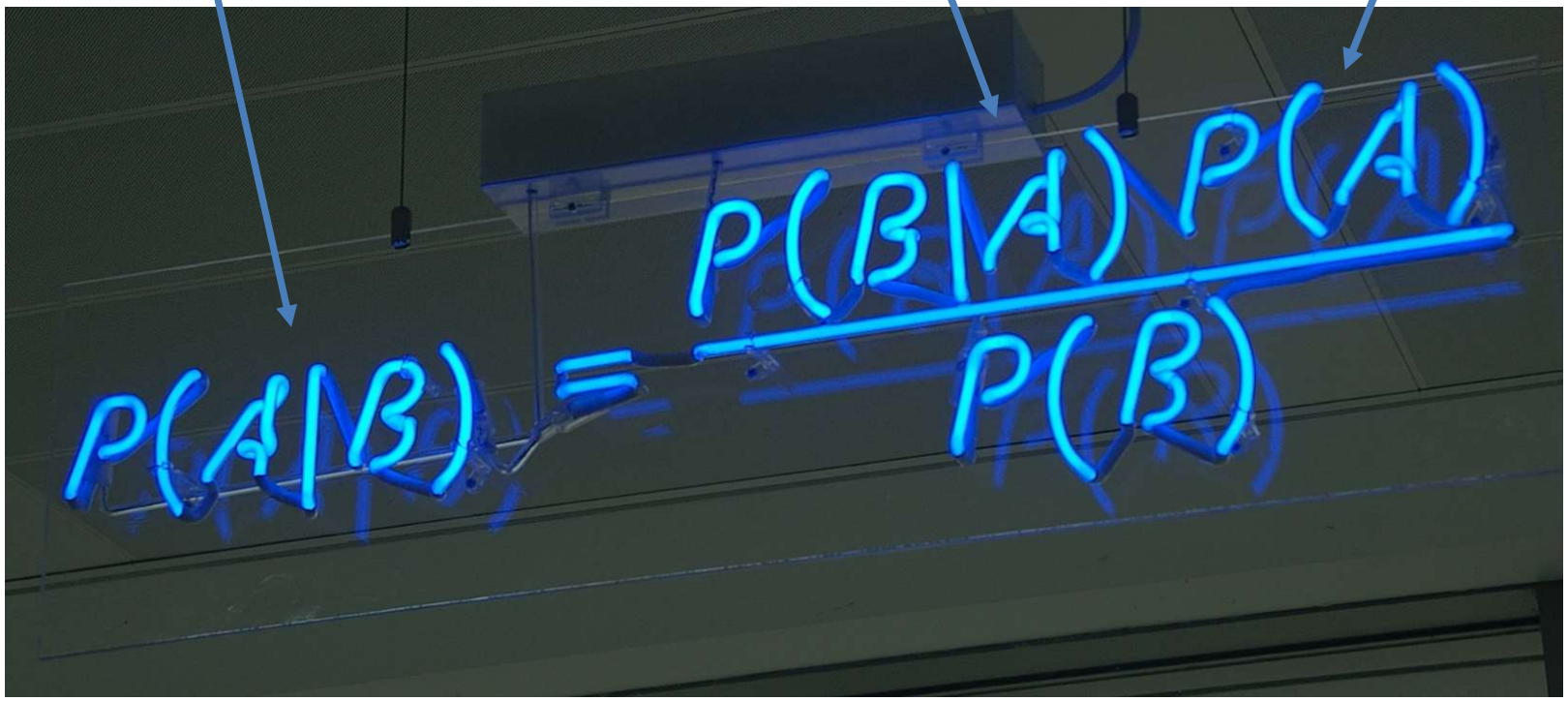
4. Augment the data and repeat steps 2 and 3.

Why Bayesian?

update surrogate model
given new data

new data

surrogate model



Why Bayesian?

Update surrogate model
given new data

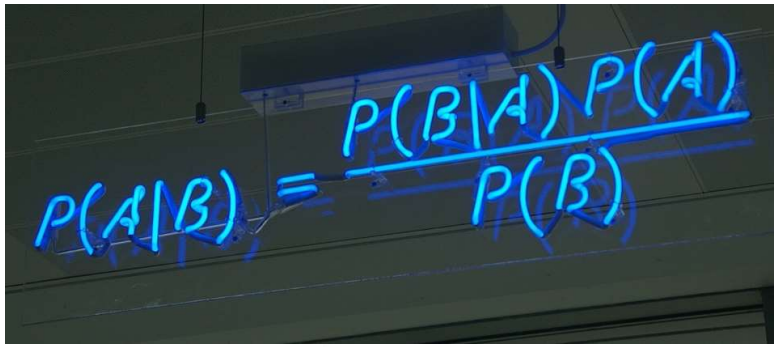


Illustration of Bayesian Optimization (3 points)

- Given 3 initial observations

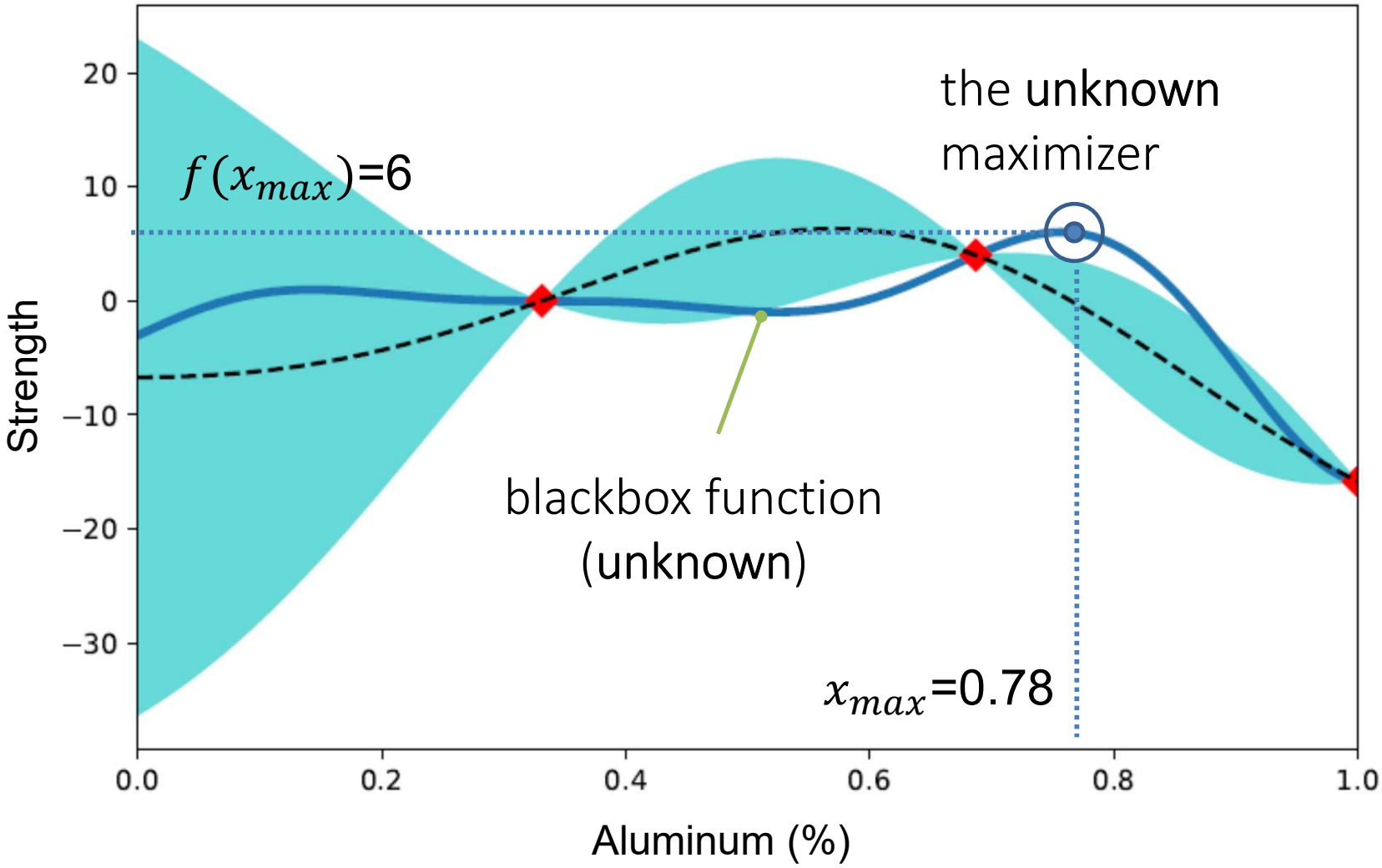


Illustration of Bayesian Optimization (3 points)

- Given 3 initial observations

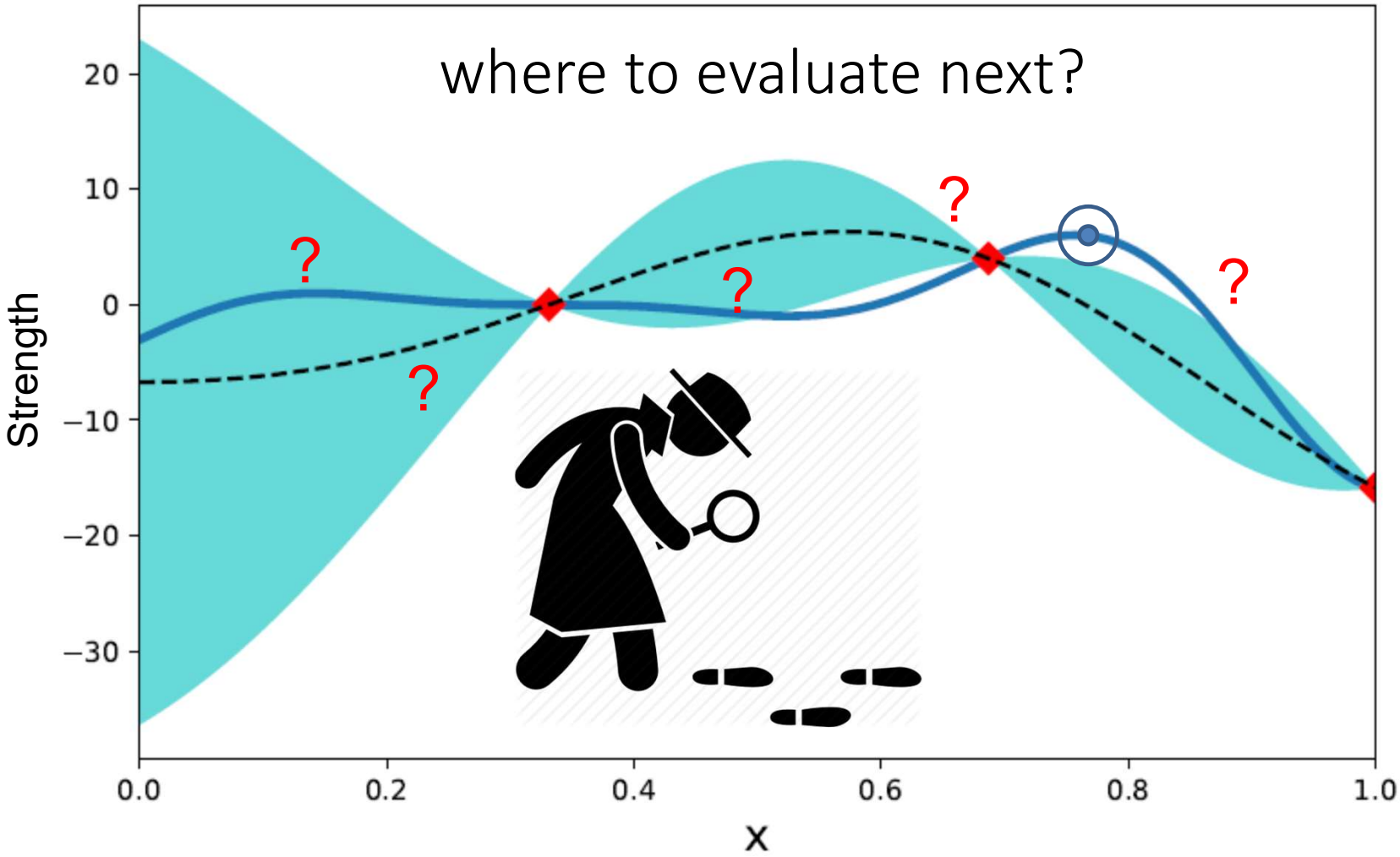


Illustration of Bayesian Optimization (4 points)

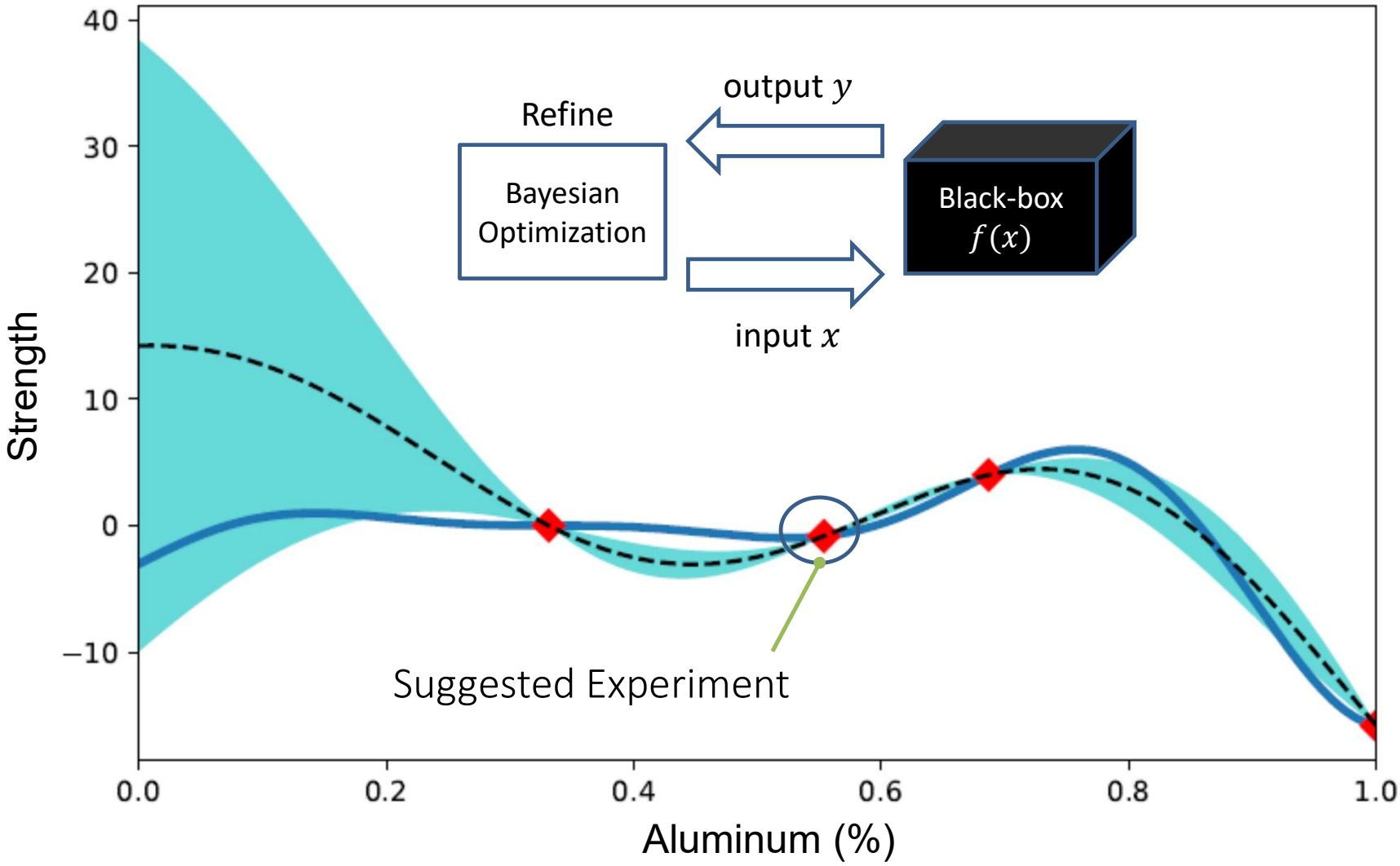


Illustration of Bayesian Optimization (5 points)

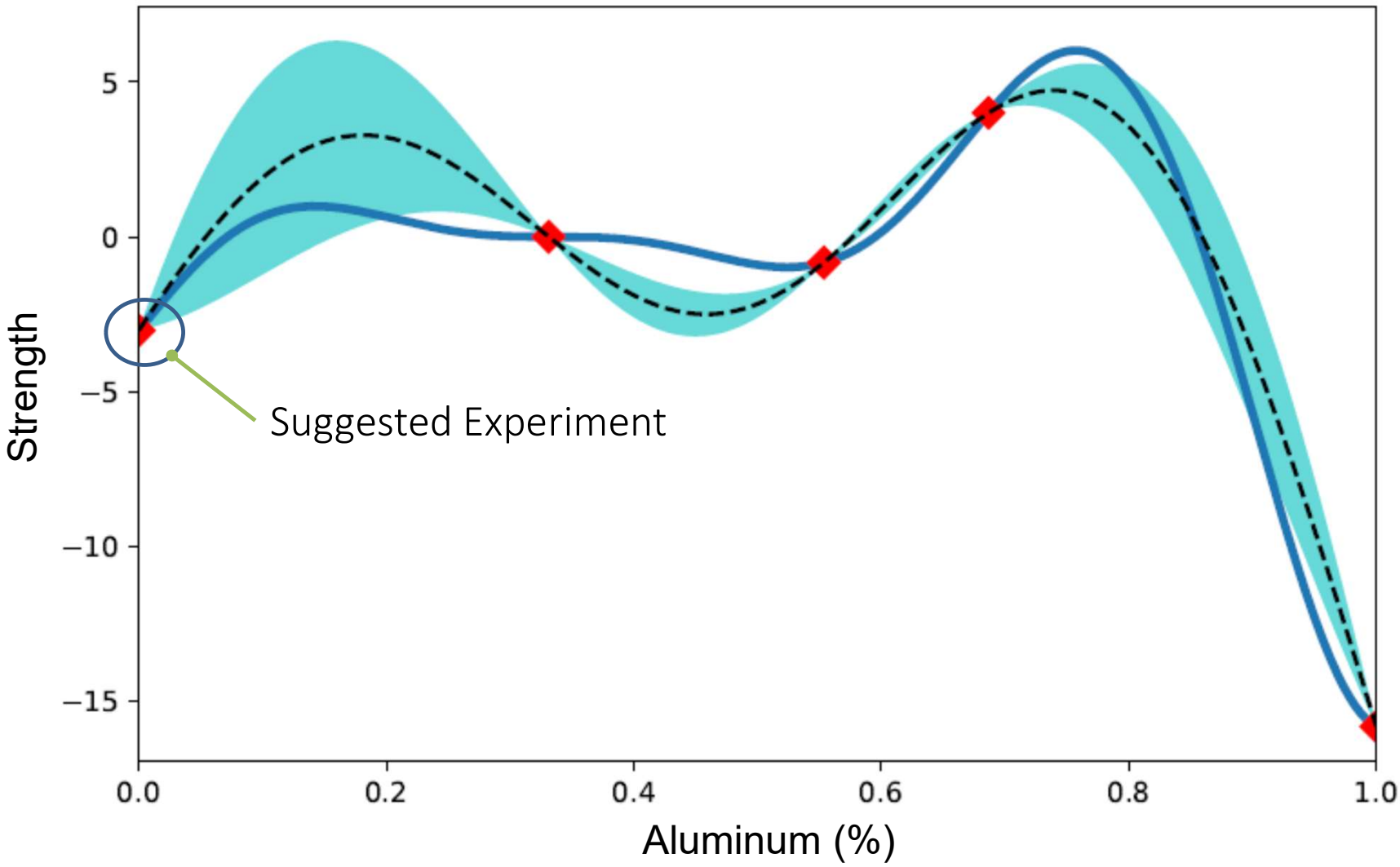


Illustration of Bayesian Optimization (6 points)

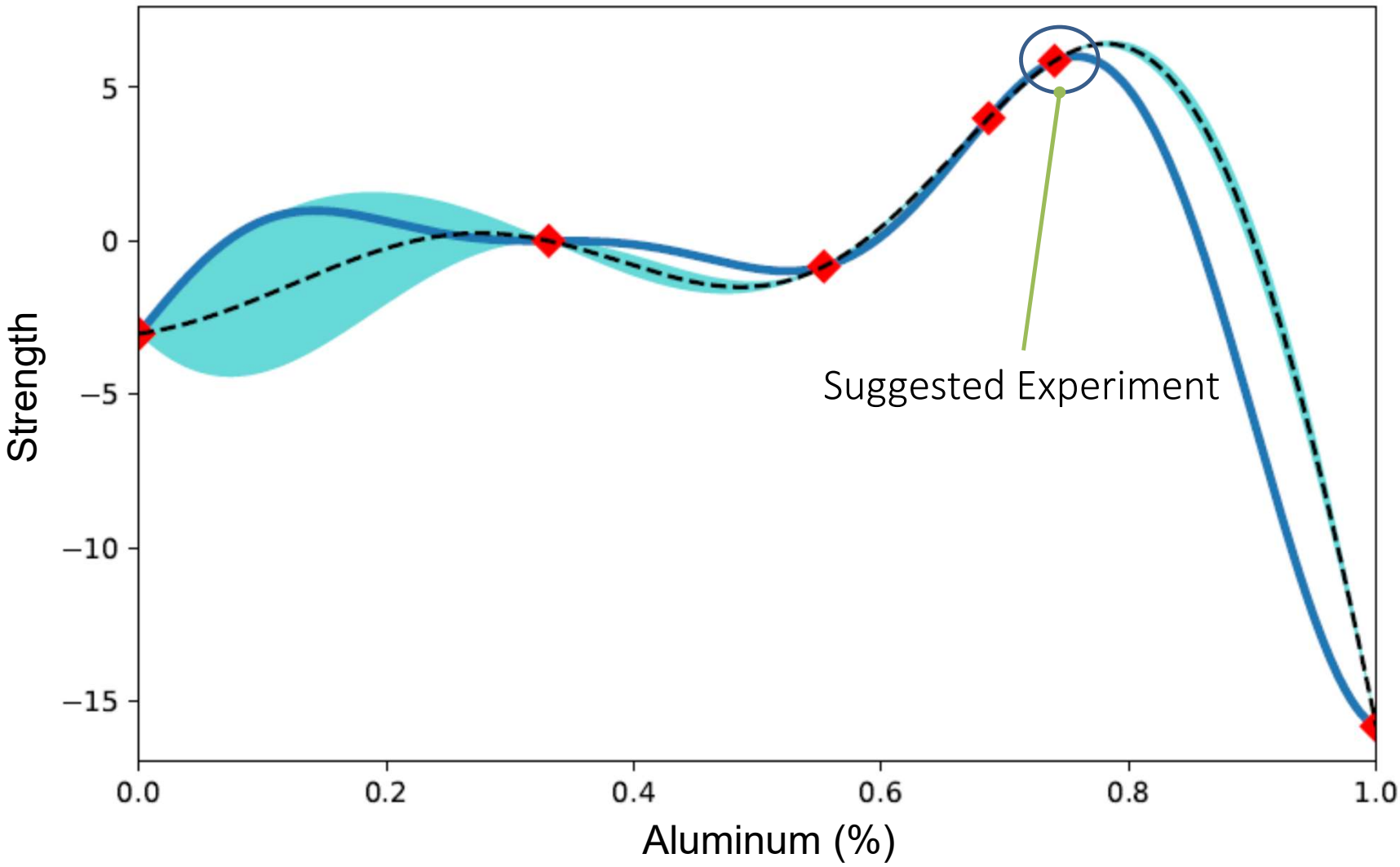


Illustration of Bayesian Optimization (7 points)

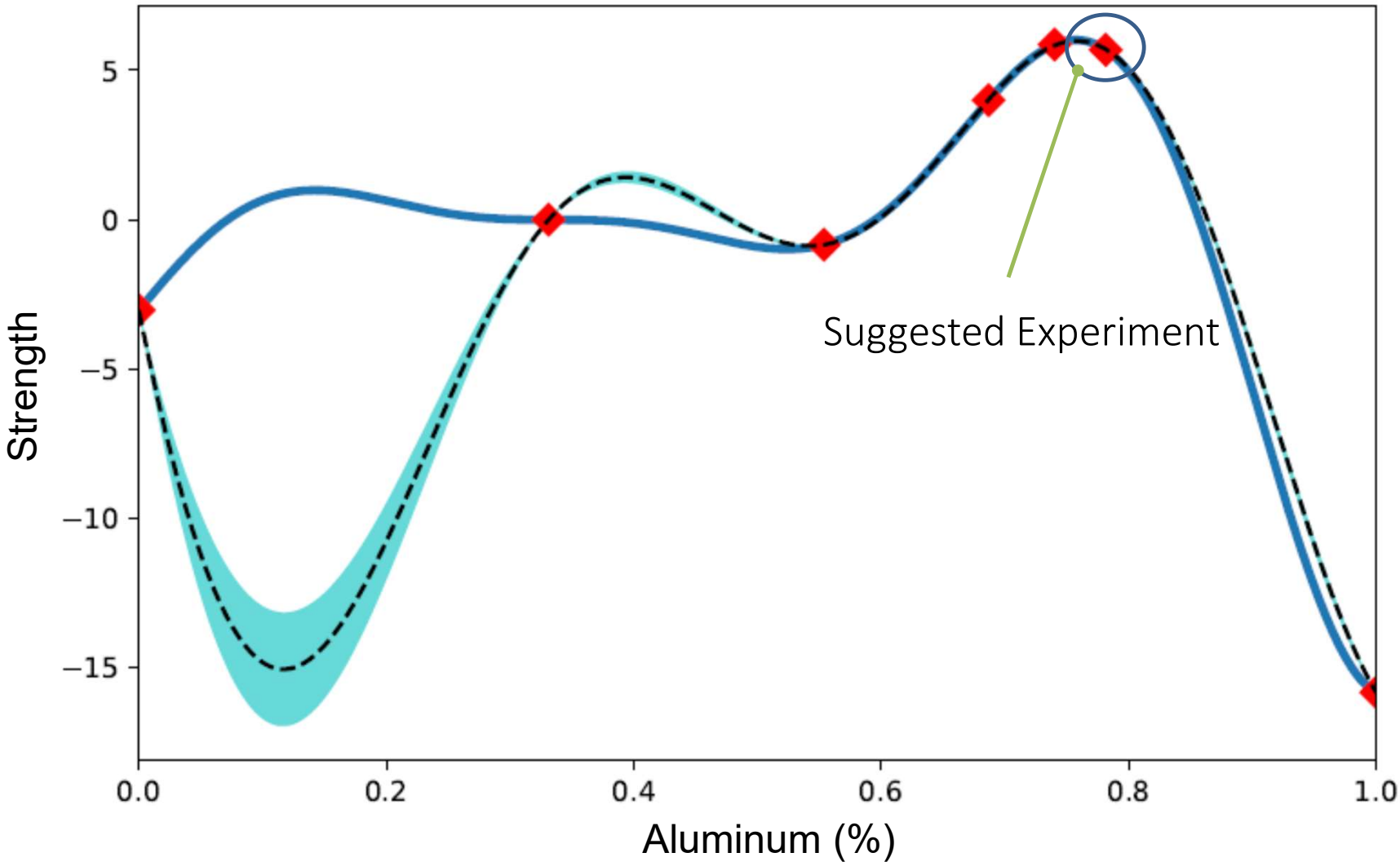
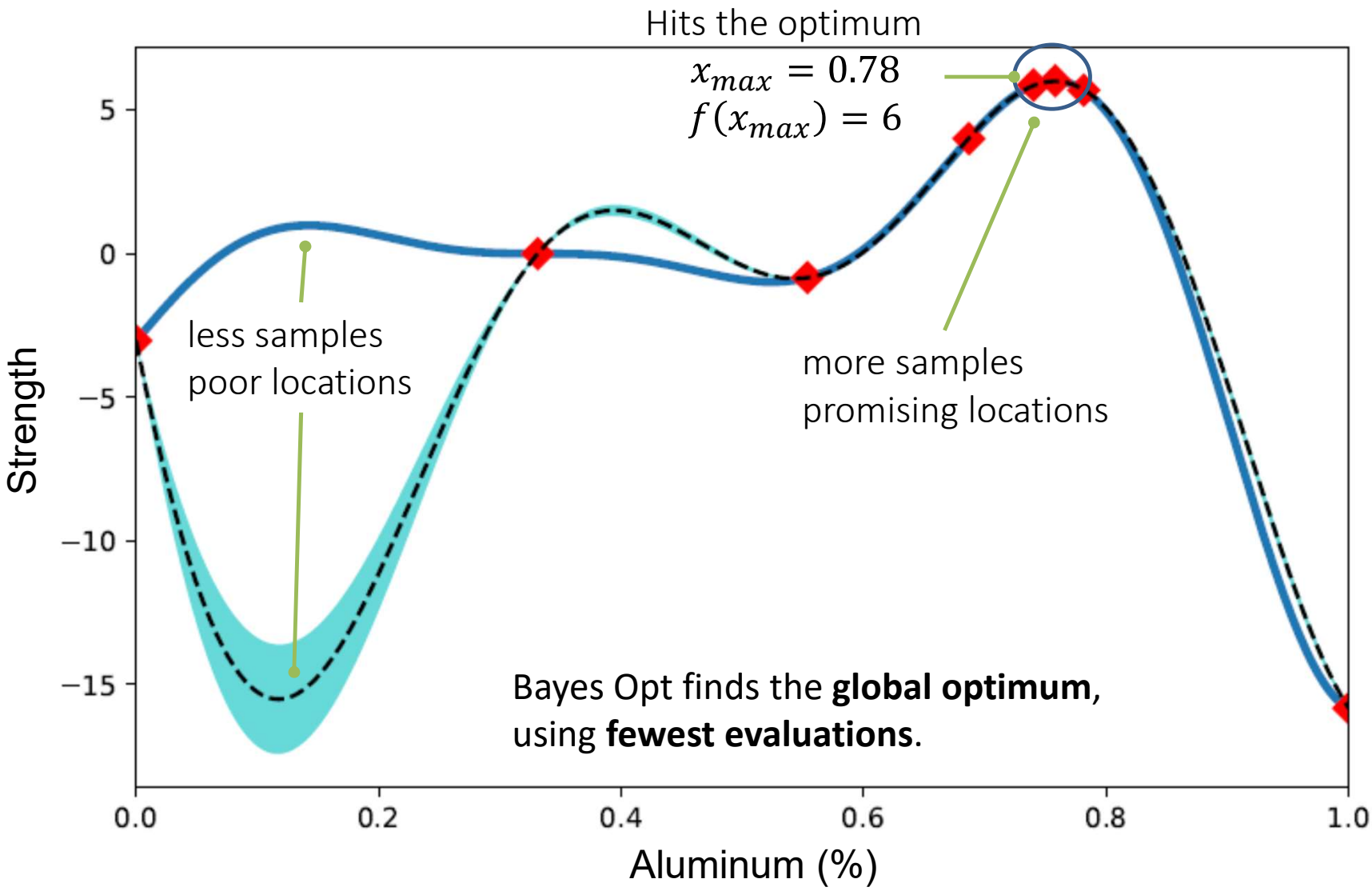


Illustration of Bayesian Optimization (8 points)



Surrogate Models Requirements

- Requirement 1: mimic the behaviour of the true function *f*.
  - We can use non-linear regression models.
- Requirement 2: uncertainty for exploration
  - We can use any non-linear models which can provide the uncertainty.

Surrogate Models for Bayesian Optimization

- Gaussian Process
- Random Forest (F. Hutter et al 2011)
- Support Vector Regression
- Student-t Process (A. Shah et al NIPS 2013 Workshop)
- Deep Neural Network (J. Snoek et al ICML 2015)
- Bayesian Neural Network (JT. Springenberg at al NIPS 2016)

Outline Part 1: Bayesian Optimization

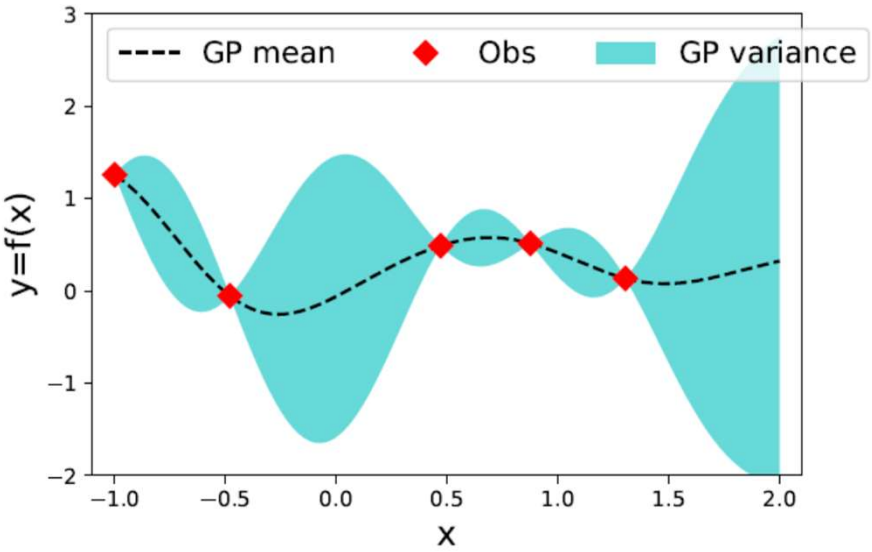
- Bayesian Optimization
- Gaussian Processes
- Acquisition Functions
- Applications
- Demo Mini BayesOpt

Gaussian Process

- Gaussian process (GP) is a distribution on functions.
- GP is characterized by the mean function $m: X \to R$

and a positive definite covariance function K: $X \times X \to R$

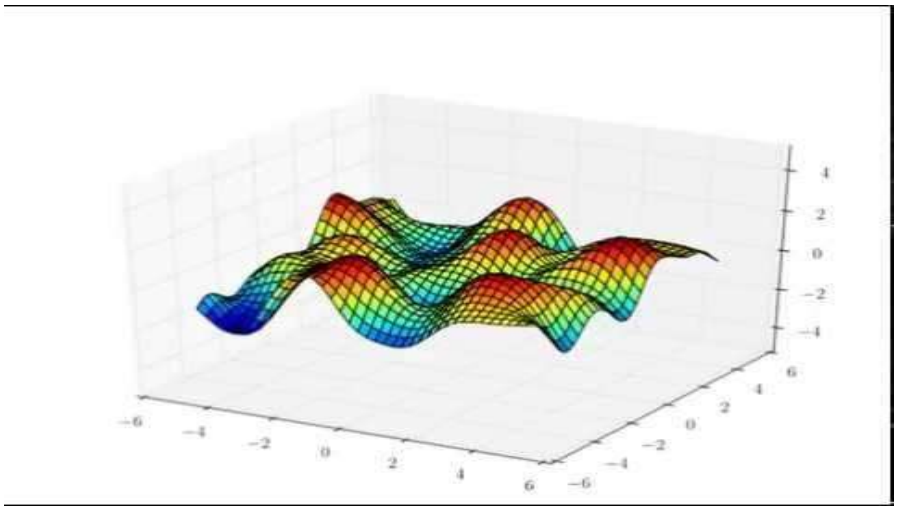
- Similar input (high covariance) should have similar output.
- We can compute predictive mean and variance in closed-form.

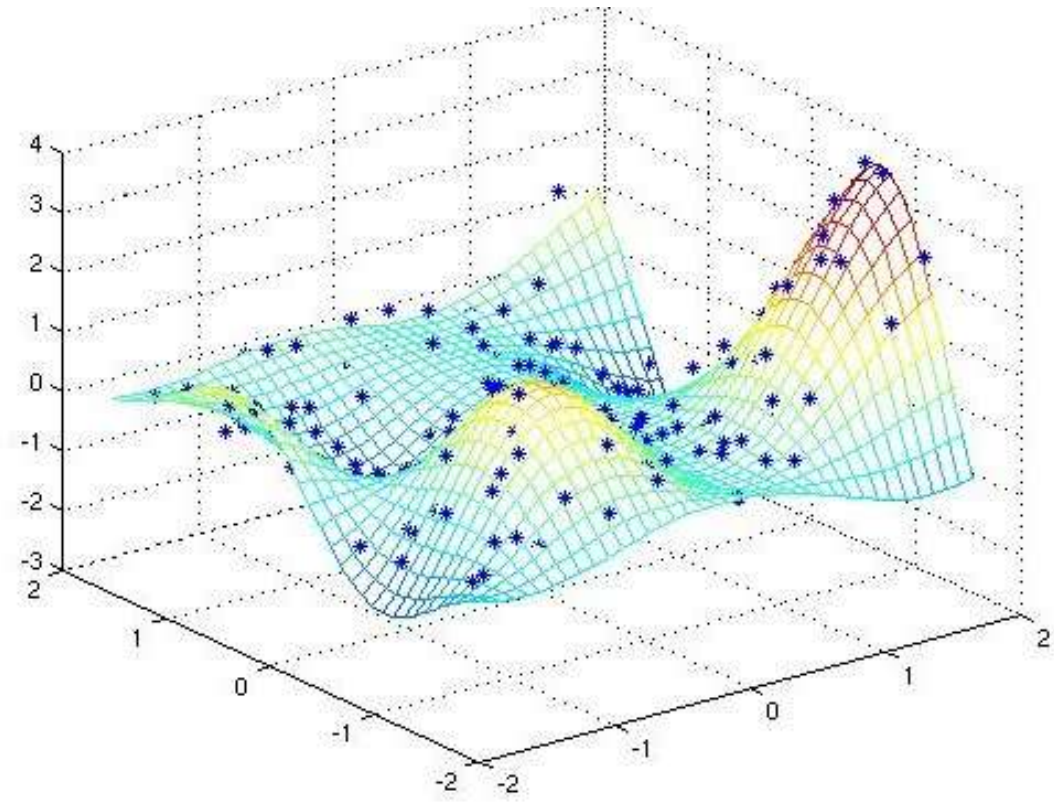


Rasmussen, C. E. Gaussian processes for machine learning, 2006.

Examples of Gaussian Process

- Fitting 2D using Gaussian Process





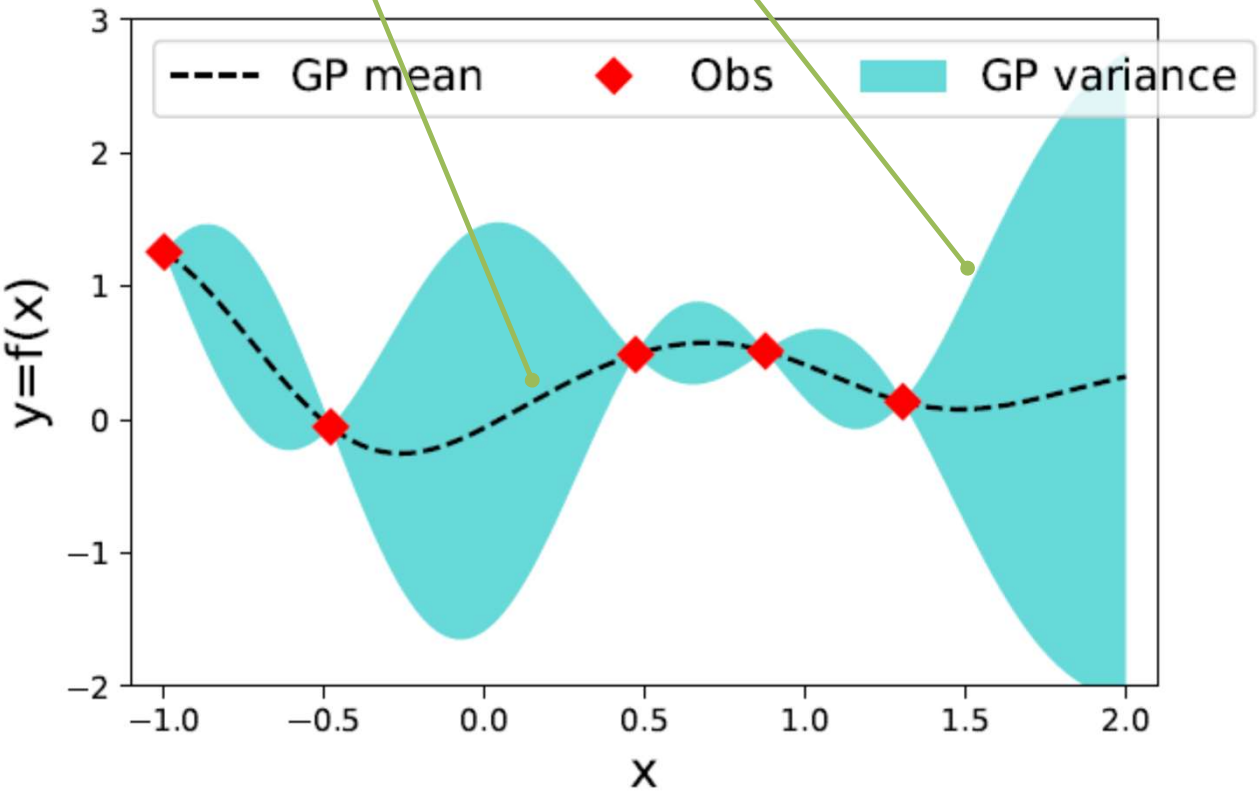
The Benefits of using Gaussian Process

- Uncertainty
- Closed-form
- Limited data
- Non-linear

$$p(y_* \mid \boldsymbol{y}) \sim \mathcal{N}\left(\underbrace{K_* K^{-1} \boldsymbol{y}}_{\mu(x)}, \underbrace{K_{**} - K_* K^{-1} K_*^T}_{\sigma(x)}\right)$$

uncertainty

belief about function value



Gaussian Process

- The joint distribution follows a Multivariate Gaussian

Training points

Testing points

$$\begin{bmatrix} \boldsymbol{y} \\ y_* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K & K_*^T \\ K_* & K_{**} \end{bmatrix}\right)$$

- The covariance matrices are

$$K_* = \begin{bmatrix} k(x_*, x_1) & \cdots & k(x_*, x_N) \end{bmatrix}$$

$$K_{**} = k(x_*, x_*)$$

$$K = \begin{bmatrix} k(x_1, x_1) & \cdots & k(x_1, x_N) \\ k(x_2, x_1) & \cdots & k(x_2, x_N) \\ \vdots & \ddots & \vdots \\ k(x_N, x_1) & \cdots & k(x_N, x_N) \end{bmatrix}$$

$$k(x, x') = \sigma_f^2 \exp\left[\frac{-(x - x')^2}{2l^2}\right]$$

Gaussian Process

- The joint distribution

$$\begin{bmatrix} \boldsymbol{y} \\ y_* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K & K_*^T \\ K_* & K_{**} \end{bmatrix}\right)$$

linear algebra

- The predictive probability

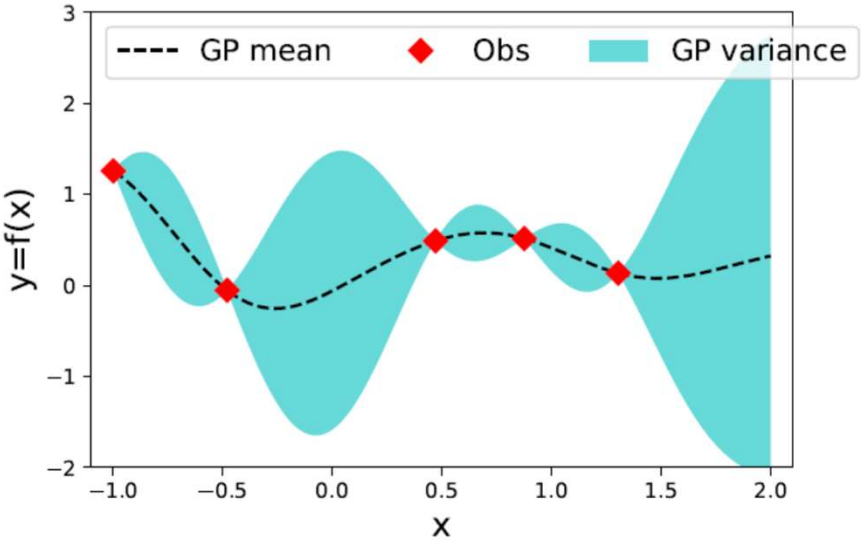
Testing points

Training points

$$p(y_* \mid \boldsymbol{y}) \sim \mathcal{N}\left(\underbrace{K_* K^{-1} \boldsymbol{y}}_{\mu(x)}, \underbrace{K_{**} - K_* K^{-1} K_*^T}_{\sigma(x)}\right)$$

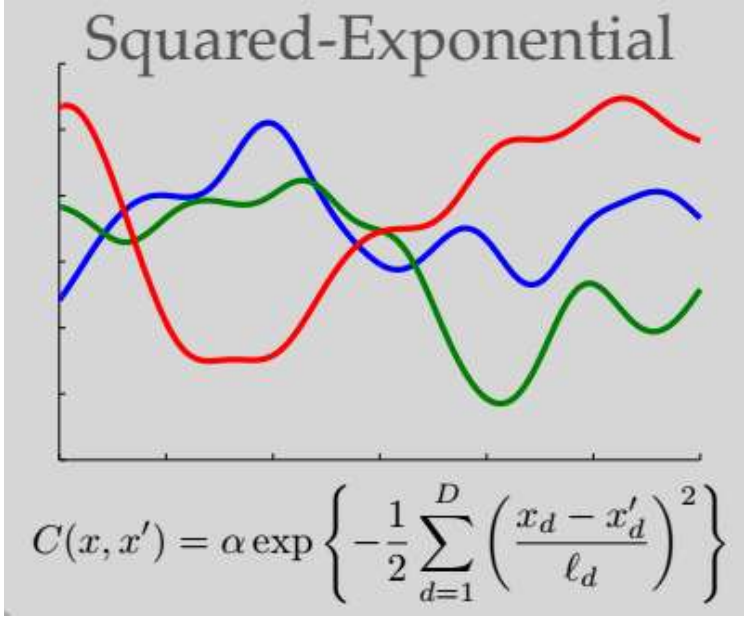
predictive mean

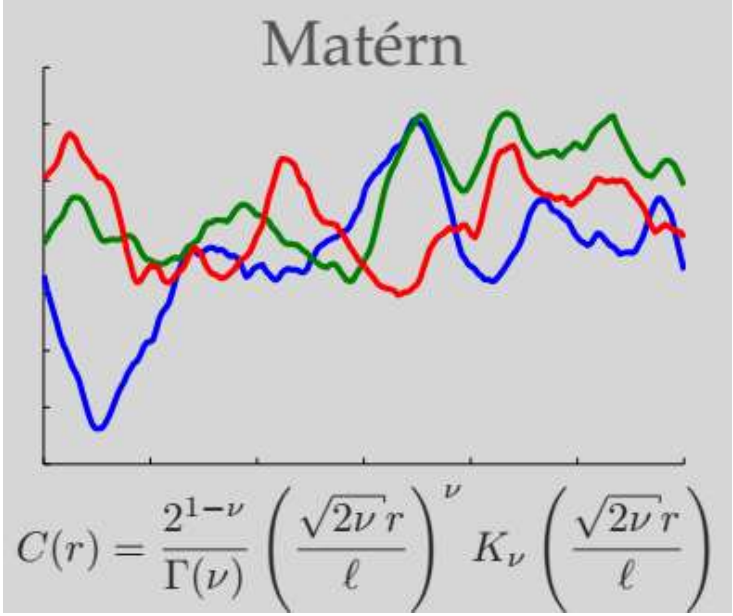
predictive variance



Examples of GP Covariance functions

- Two commonly used covariance functions in Bayes Opt

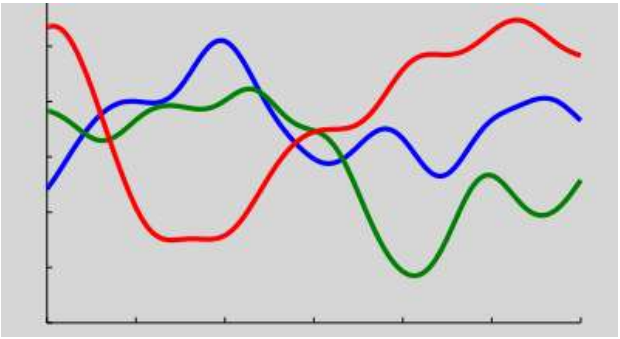


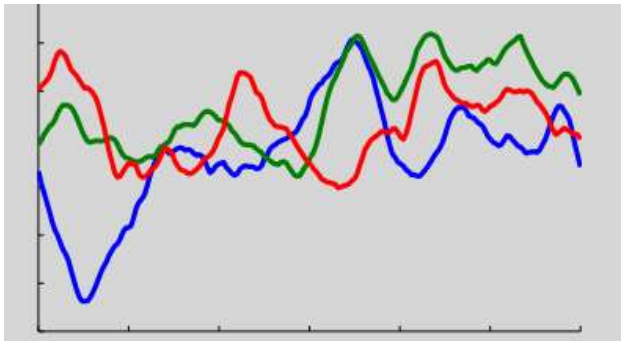


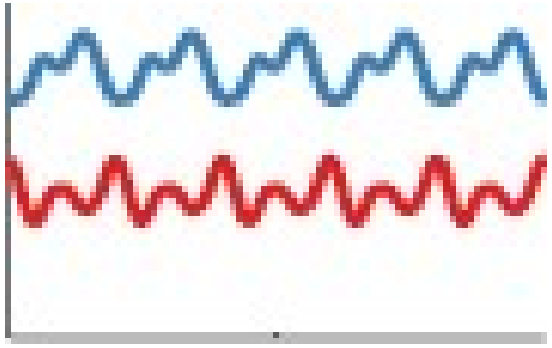
Hyper-parameter Treatments in GP

- This turns out to be crucial to good performance.

- Covariance function selection (using prior knowledge)
  - SE kernel, Matern kernel, periodic kernel…etc







- Gaussian process hyper-parameters.
  - E.g., length scale parameter of SE kernel.

Treatment for GP Hyper-parameters

- Minimize negative log marginal likelihood

$$\mathcal{L} = -\log p(\mathbf{y}|\boldsymbol{\theta}) = \frac{1}{2}\log\det\mathbf{C}(\boldsymbol{\theta}) + \frac{1}{2}\mathbf{y}^{\top}\mathbf{C}^{-1}(\boldsymbol{\theta})\mathbf{y} + \frac{N}{2}\log(2\pi)$$

where θ denotes for the hyper-parameters and noise level; $C = K + \sigma^2 I$.

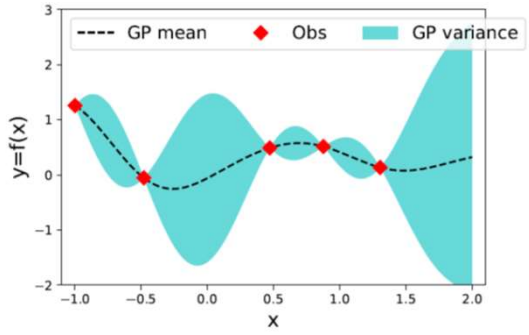
- Optimize $\mathcal{L}$ using gradient descent.

Outline Part 1: Bayesian Optimization

- Bayesian Optimization
- Gaussian Processes
- Acquisition Functions
  - Upper Confidence Bound (UCB) and Expected Improvement (EI)
  - Thompson Sampling (TS)
  - Optimization toolbox for acquisition function.
- Applications
- Demo Mini BayesOpt

Bayesian Optimization Algorithm

Get initial data



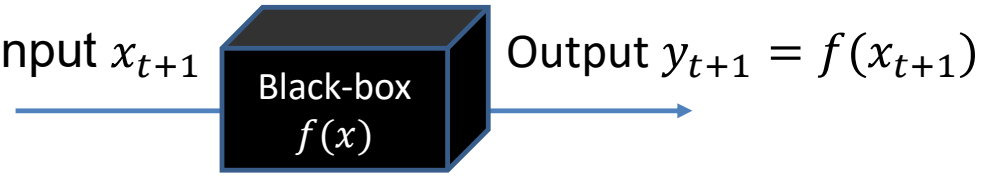
Repeat

1. Fit a GP model to the data $(x_i, y_i)_{i=1}^{t}$

2. Define the acquisition function $\alpha(x)$ from the GP model

3. Select the next query $x_{t+1} = \arg\max\ \alpha(x)$

4. Evaluate the black-box to get the score



Acquisition Function α is Built from a GP

- Based on a GP surrogate above, BO defines an acquisition function $\alpha(x)$ to select a point for evaluation.

instead of $x_t = argmax_{x \in X} f(x)$  $x_t = argmax_{x \in X} \alpha(x)$

unsolvable!

solvable!

- Optimizing the acquisition function α is cheaper without using black-box evaluation.

Acquisition Function

Acquisition function balances the **explore-exploit**.

- Explore: seek places with high uncertainty.

- Exploit: seek places in the locality of where you are already doing well at.

Common Acquisition Functions

- Expected Improvement [Mokus, 1972]
- Probability of Improvement [Krushner, 1997]
- GP Upper Confidence Bound [Srinivas, 2010]
- Predictive Entropy Search [Hernández-Lobato, 2014]
- ….

Balancing the exploration-exploitation in different ways.

Outline Part 1: Bayesian Optimization

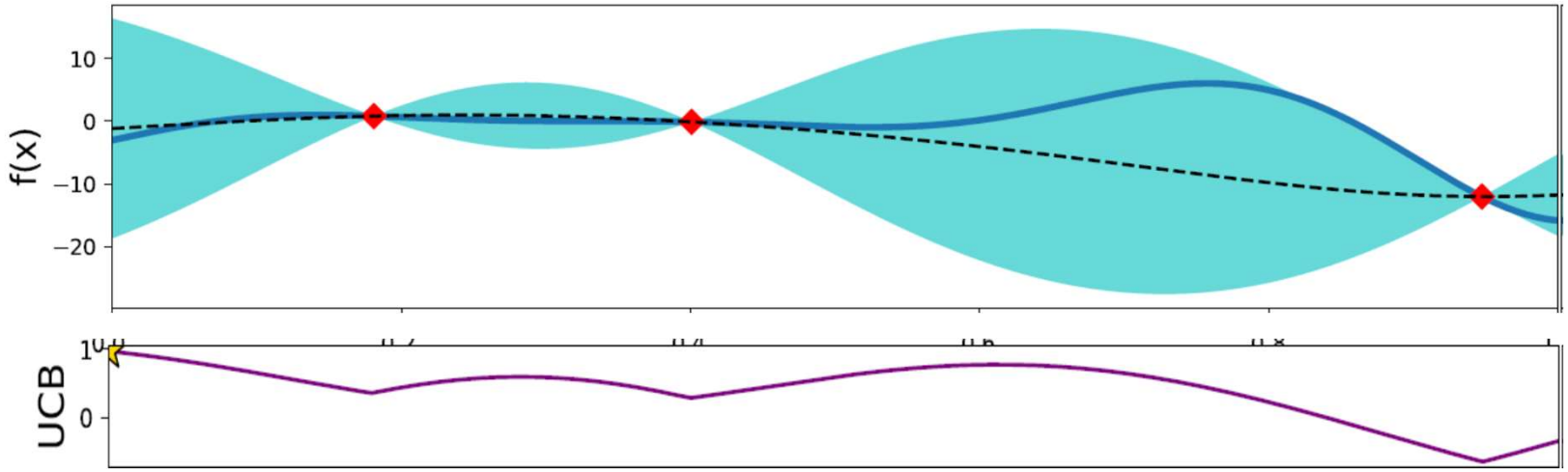
- Bayesian Optimization
- Gaussian Processes
- Acquisition Functions
  - Upper Confidence Bound (UCB) and Expected Improvement (EI)
  - Thompson Sampling (TS)
  - Optimization toolbox for acquisition function.
- Applications
- Demo Mini BayesOpt

Upper Confidence Bound (UCB)

- Upper Confidence Bound (UCB) is one of the most used acquisition function.

$$\alpha^{GP-UCB}(x)=\mu(x)+\sqrt{\beta} \times \sigma(x)$$

GP-UCB simply a combination of mean and variance functions .
It encourages high mean $\mu(\quad)$ and high variance $\sigma(\quad)$



Expected Improvement (EI)

- Define the improvement function over the incumbent y^{max}

$$I(x) = \max\{0, f(x) - y^{max}\}$$

where y^{max} is the best value so far.

- The expected improvement is defined as $E[I(x)]$.

- We get the closed-form solution for the EI as

$$\alpha^{EI}(x) = E[I(x)] = \sigma_t(x)\phi(z) + [\mu_t(x) - y^{max}]\Phi(x)$$

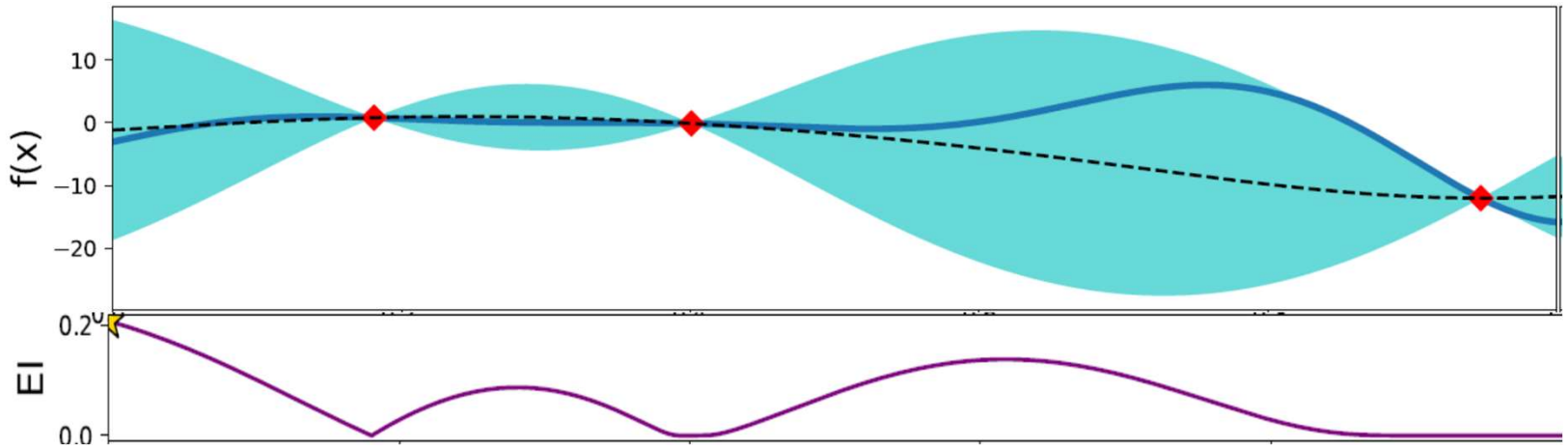
where $z = z(x) = \frac{\mu_t(x) - y^{max}}{\sigma_t(x)}$, $\phi()$ is the normal p.d.f. and $\Phi()$ is the normal c.d.f.

Expected Improvement

- We get the closed-form solution as

$$\alpha^{EI}(x) = E[I(x)] = \sigma_t(x)\phi(z) + [\mu_t(x) - \xi]\Phi(x)$$

where $z = \frac{\mu_t(x)-\xi}{\sigma_t(x)}$, ϕ is the normal p.d.f. and Φ is the normal c.d.f.



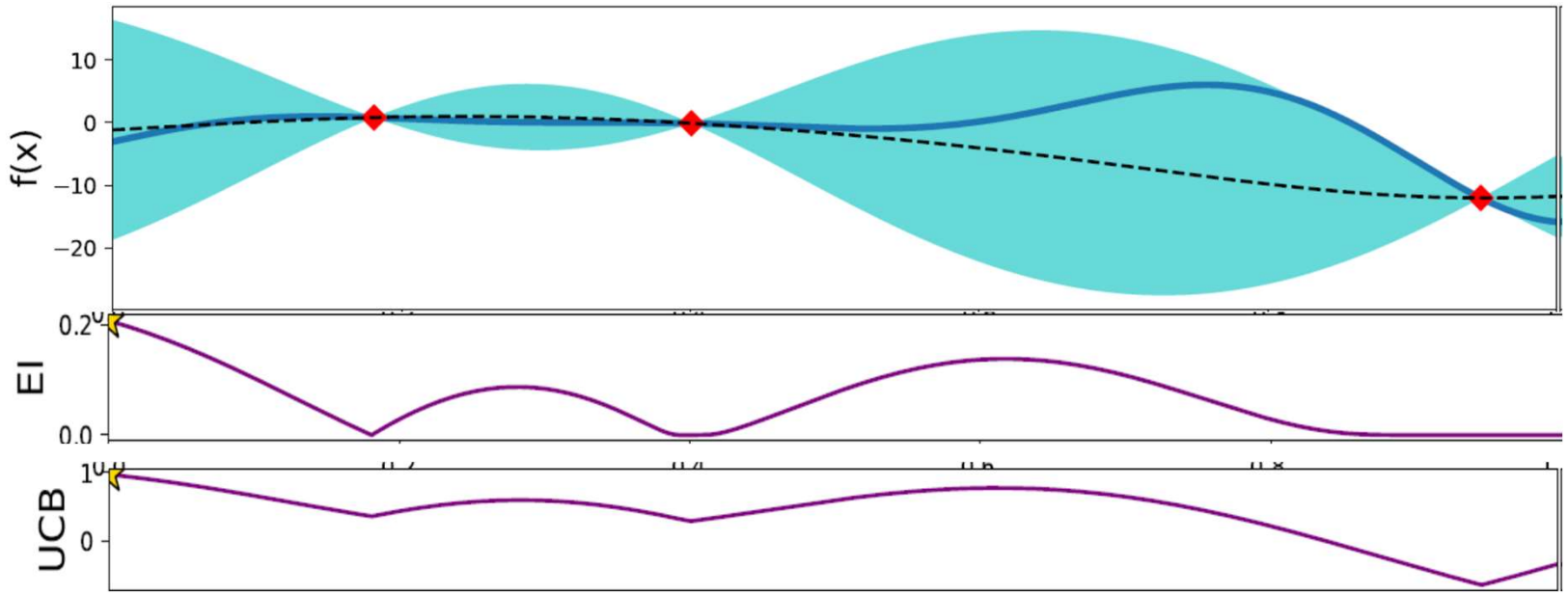
- High mean and high variance => High value for EI.

Remark on Expected Improvement vs UCB

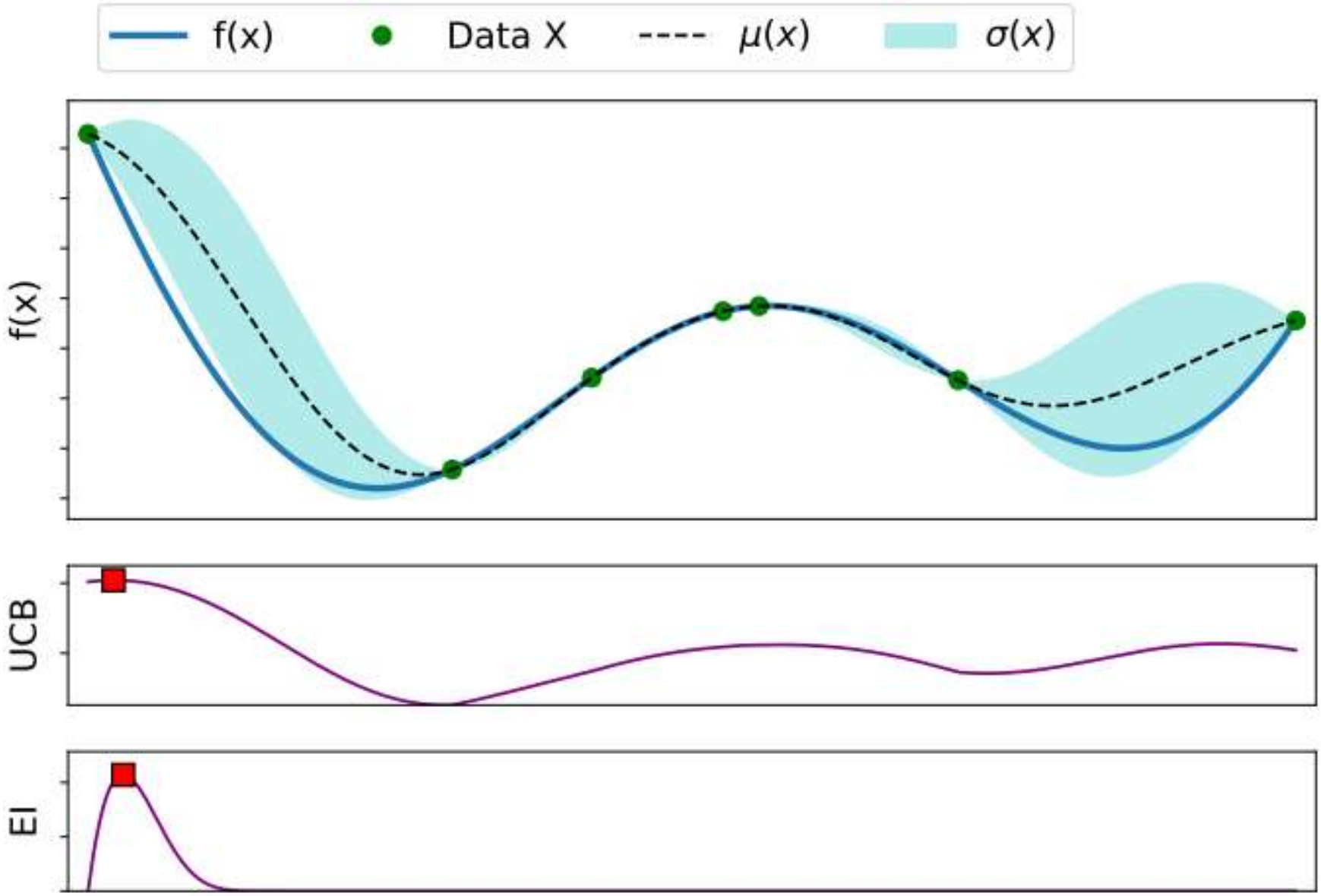
- Both EI and UCB encourages high mean and high variance. They may and may not give the same suggestion.

$$\alpha^{EI}(x) = E[I(x)] = \sigma_t(x)\phi(z) + [\mu_t(x) - \xi]\Phi(x)$$

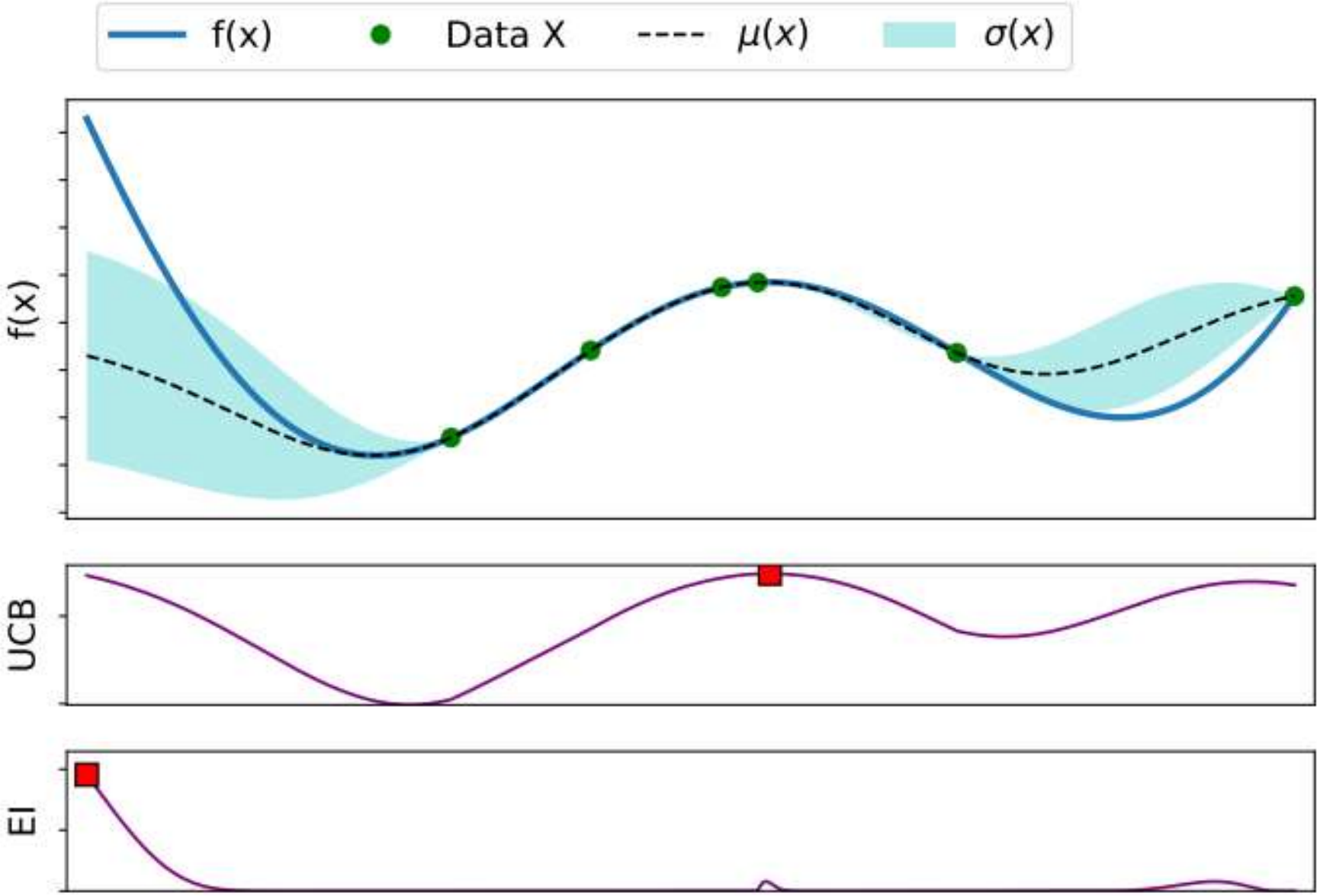
$$\alpha^{GP-UCB}(x) = \mu(x) + \sqrt{\beta} \times \sigma(x)$$



EI and UCB agree with each other



EI and UCB can disagree with each other



Different of Acquisition Functions (3 points)

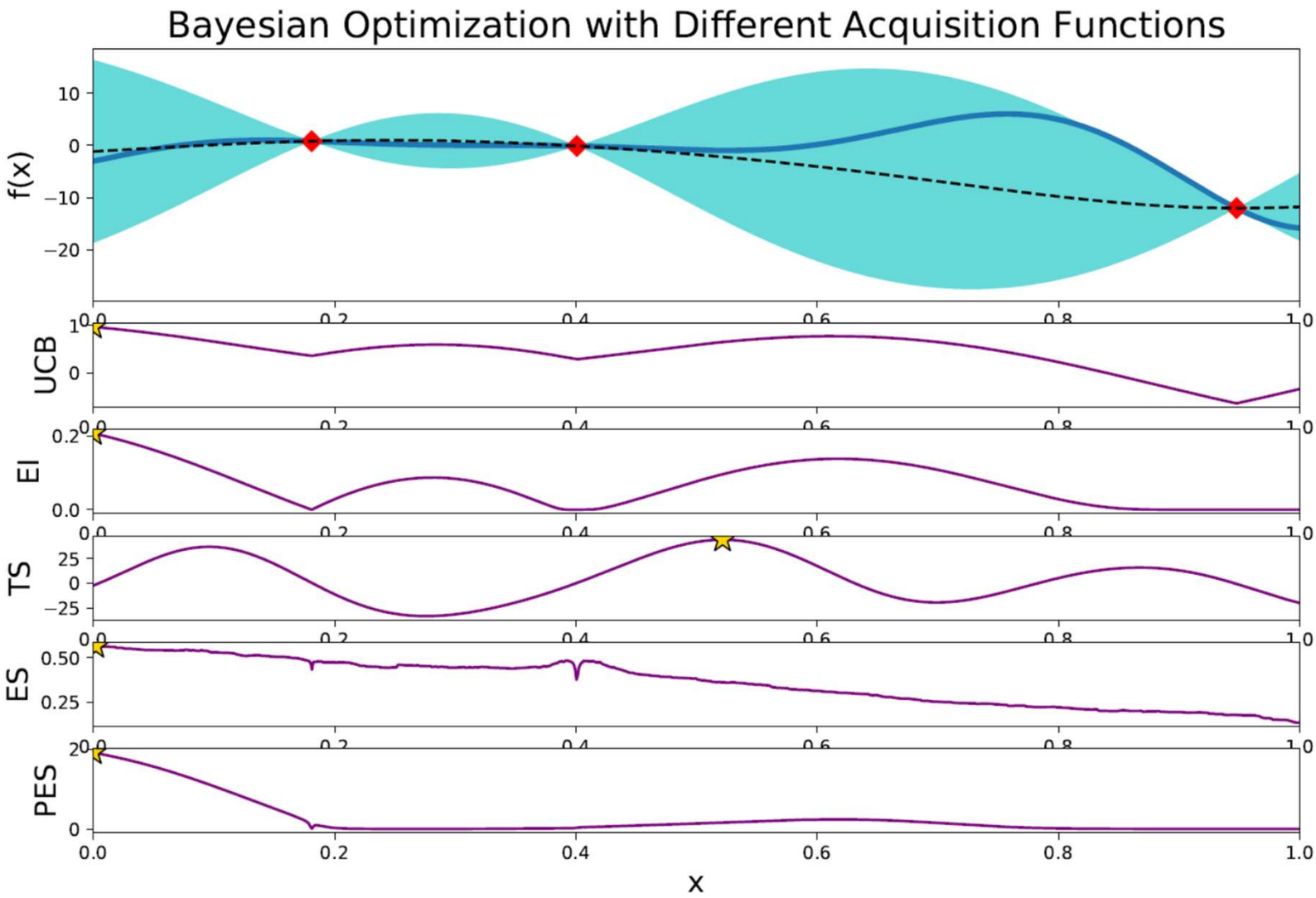
Srinivas et al 2010

Jones et al 1998

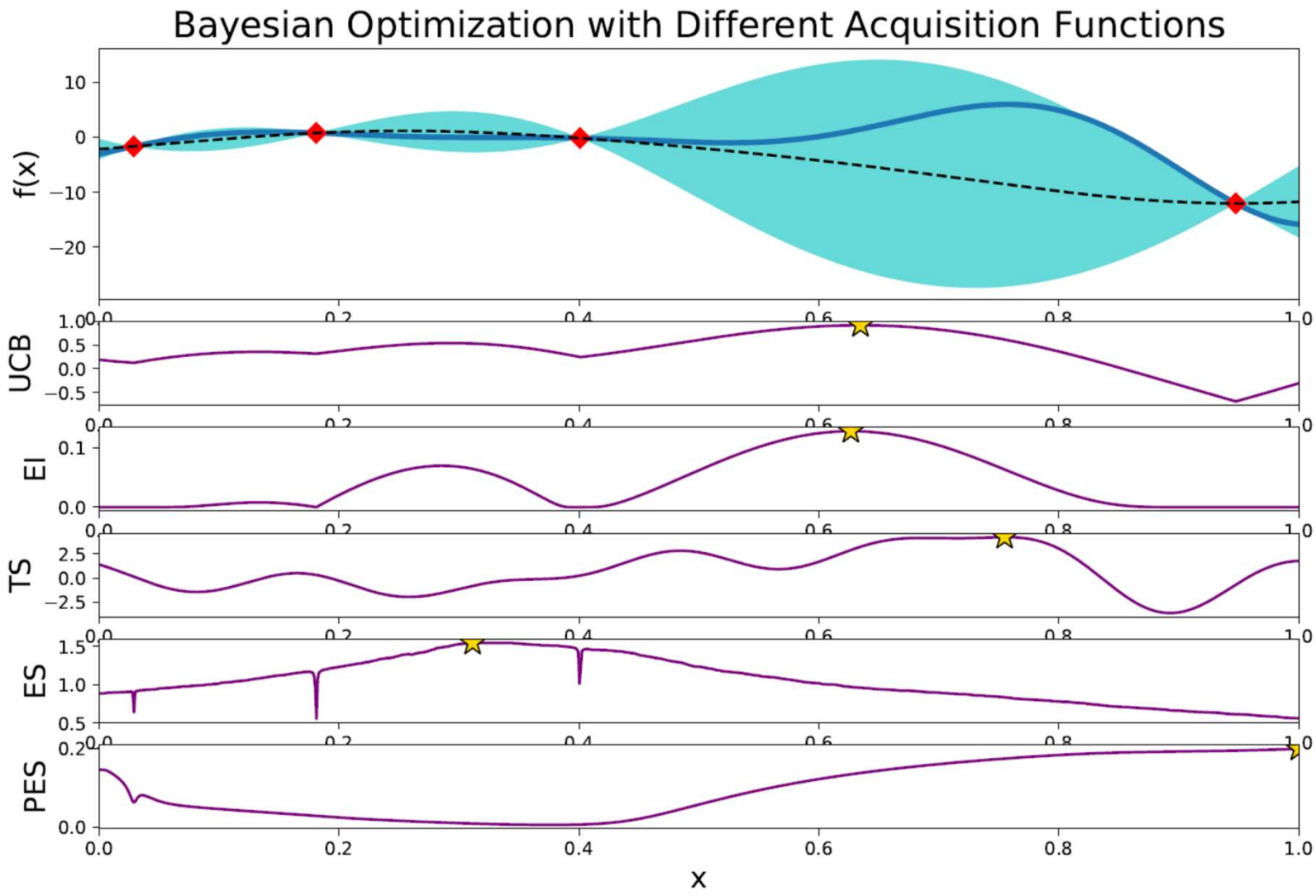
Hernandez-Lobato et al 2017

Hennig et al 2012

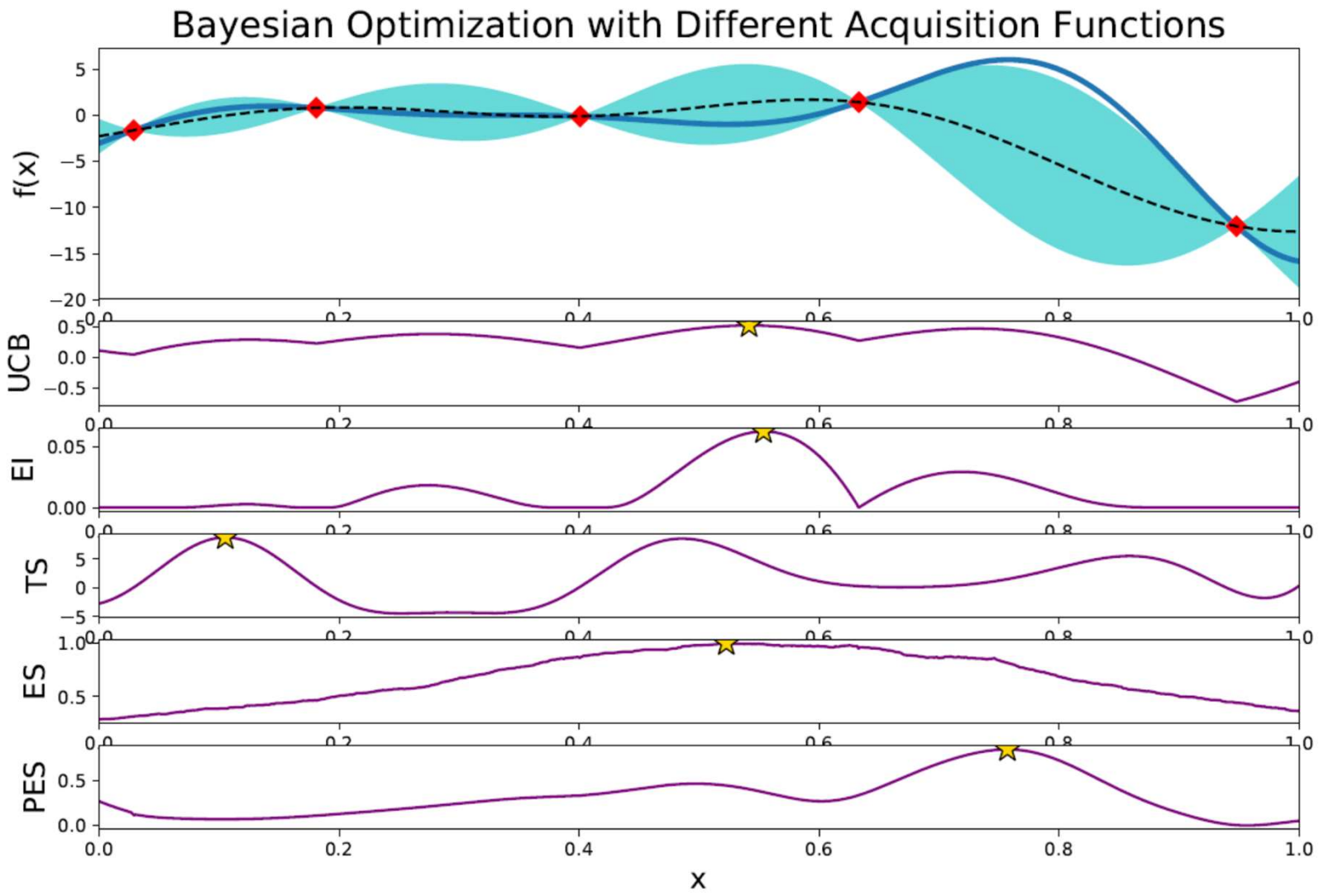
Hernandez-Lobato et al 2014



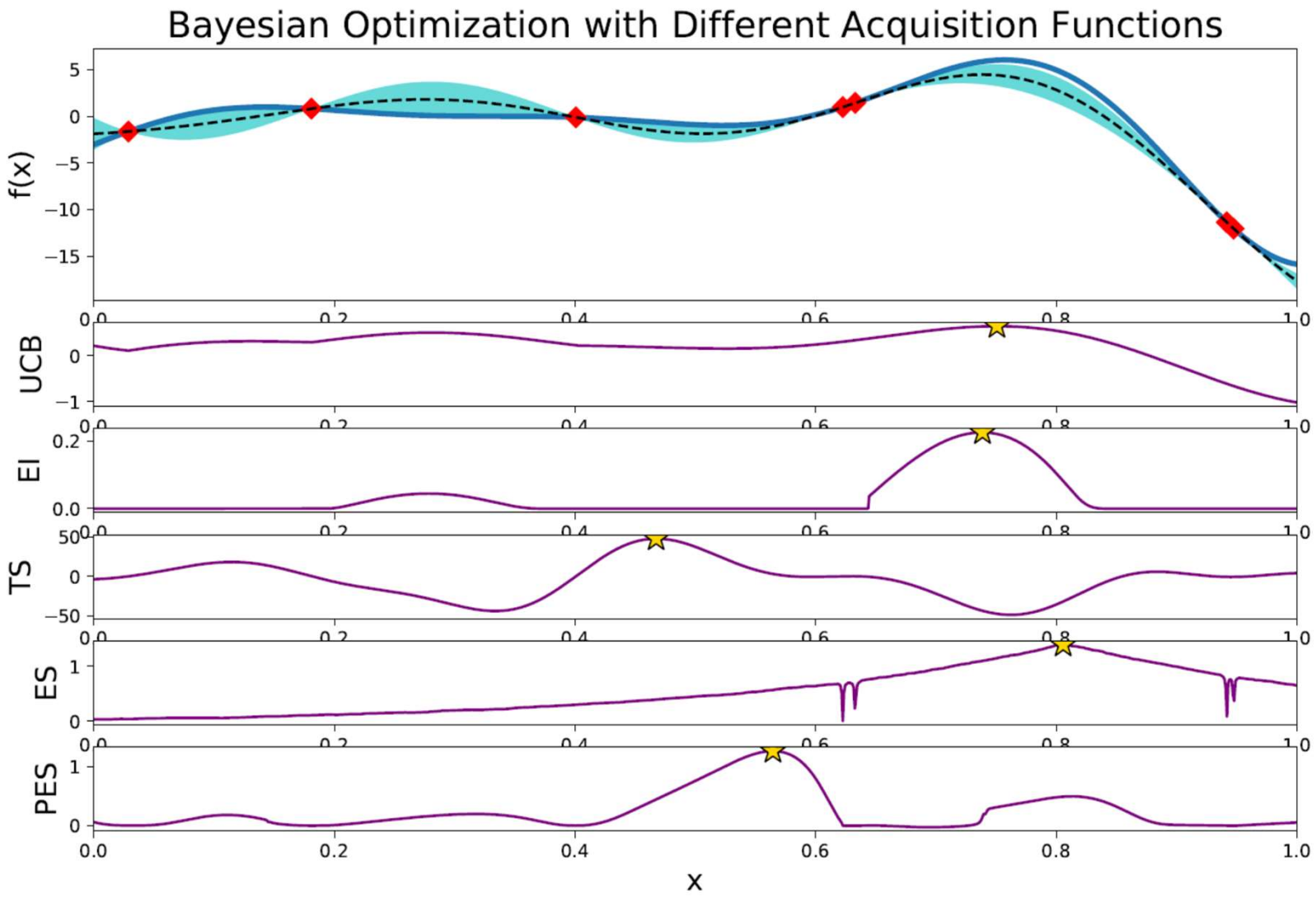
Different of Acquisition Functions (4 observations)



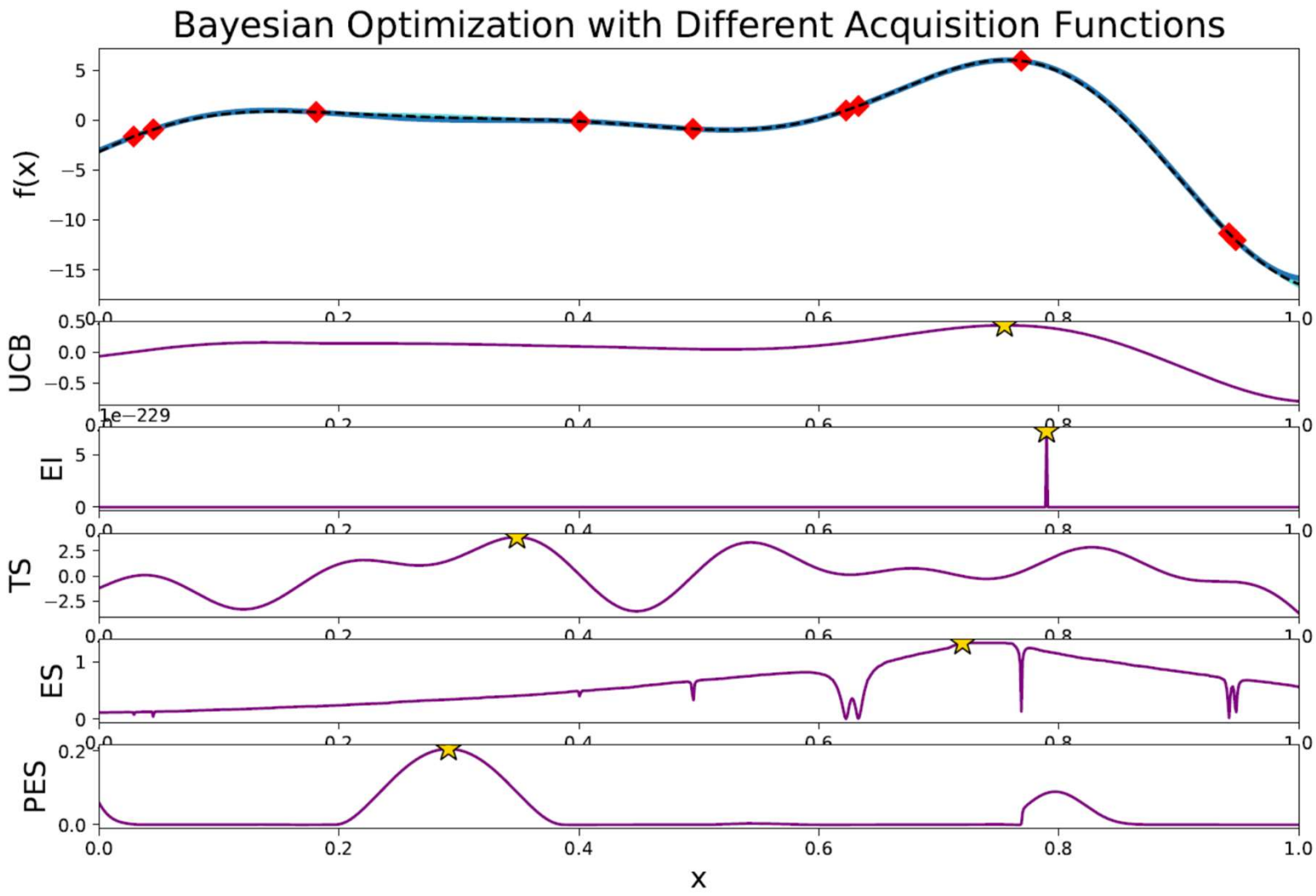
Different of Acquisition Functions (5 points)



Different of Acquisition Functions (7 points)



Different of Acquisition Functions (10 points)

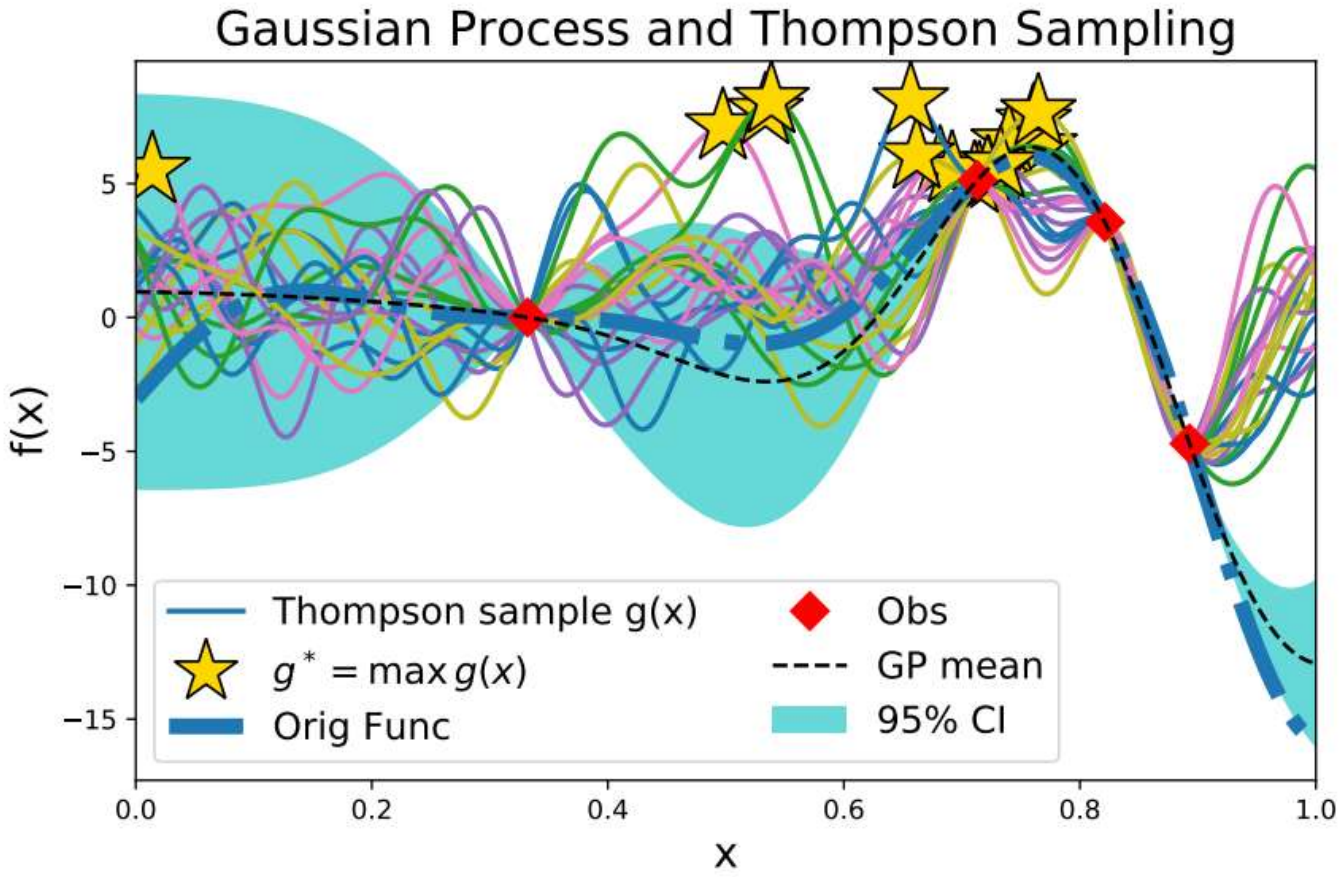


Outline Part 1: Bayesian Optimization

- Bayesian Optimization
- Gaussian Processes
- Acquisition Functions
  - Upper Confidence Bound (UCB) and Expected Improvement (EI)
  - Thompson Sampling (TS)
  - Optimization toolbox for acquisition function.
- Applications
- Demo Mini BayesOpt

Thompson Sampling for GP

- For GPs, f is an infinite-dimensional object so sampling it is not simple.



Connection of Gaussian Process to Linear Model

- Bayesian Linear Regression using feature ϕ and GP with kernel k are equivalent.

- Let $k(x, x') = \phi(x)^T \phi(x)$, the linear model $g(x) = \phi(x)^T w$

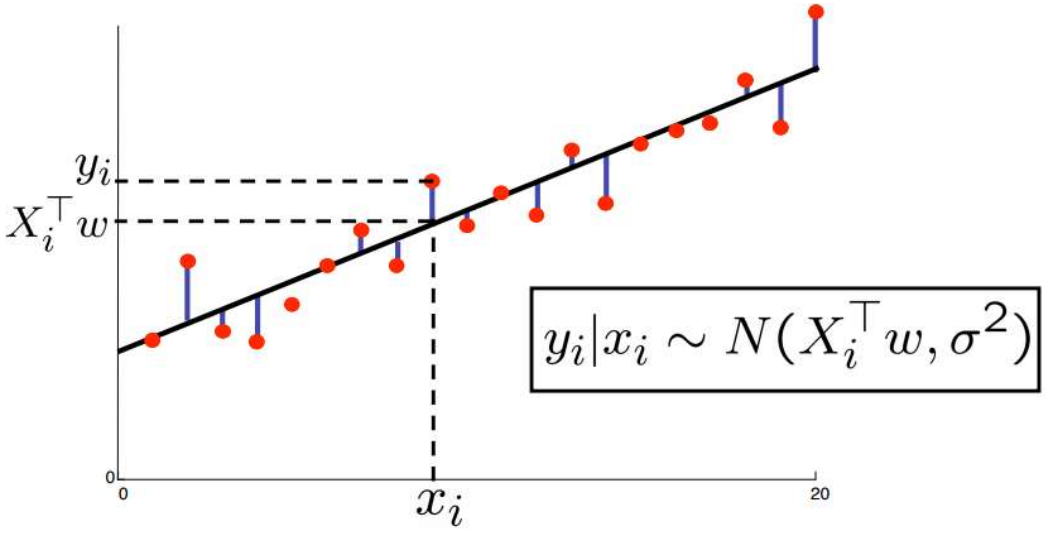
is an approximate sample from $p(f|D)$ where

$$w \sim \mathcal{N}\left([\Phi^T\Phi + \sigma^2 I]^{-1}\Phi^T y, \sigma^2[\Phi^T\Phi + \sigma^2 I]^{-1}\right)$$

- Borrowing the idea of Thompson Sampling for Bayesian Linear Regression, then do similarly for Gaussian Process.

Connection of GP to Bayesian Linear Model

- Bayesian Linear Regression
- Transform space $x \rightarrow \phi(x)$



Linear function $f(x) = \phi(x)^T \mathbf{w}$

Prior distribution $\mathbf{w} \sim \mathcal{N}(0, \mathbf{I})$

Likelihood $y_i \sim \mathcal{N}\left(f(x_i), \sigma^2\right)$

Posterior $p(w) \sim \mathcal{N}(m, V)$

Posterior of Bayesian Linear Regression

Posterior $p(w) \sim \mathcal{N}(m, V)$ where $m = \frac{1}{\sigma^2}\Phi V \boldsymbol{y}$

$V = \sigma^2 \left(\Phi^T \Phi + \sigma^2 \mathbf{I}\right)^{-1}$

Bishop book page 153

Predictive distribution $p(x \mid \dots) \sim \mathcal{N}(\mu_n(x), \sigma_n(x))$

Bishop book page 156

$$\mu_n(\boldsymbol{x}) = \phi(\boldsymbol{x})^T m = \phi(\boldsymbol{x})^T \Phi \left(\Phi^T \Phi + \sigma^2 \mathbf{I}\right)^{-1} \boldsymbol{y}$$

$$\sigma_n(\boldsymbol{x}) = \phi(\boldsymbol{x})^T V \phi(\boldsymbol{x}) + \sigma^2 = \phi(\boldsymbol{x})^T \sigma^2 \left(\Phi^T \Phi + \sigma^2 \mathbf{I}\right)^{-1} \phi(\boldsymbol{x}) + \sigma^2$$

Woodbury matrix-inversion

$$\sigma_n(\boldsymbol{x}) = \phi(\boldsymbol{x})^T \phi(\boldsymbol{x}) - \phi(\boldsymbol{x})^T \Phi^T \left(\Phi^T \Phi + \sigma^2 \mathbf{I}\right)^{-1} \Phi \phi(\boldsymbol{x}) + \sigma^2$$

Equivalence between GP vs BLR

- Bayesian Linear Regression

$$\mu_n(\boldsymbol{x}) = \underbrace{\phi(\boldsymbol{x})^T\Phi}_{K_*}\underbrace{\left(\Phi^T\Phi+\sigma^2\mathbf{I}\right)^{-1}}_{K^{-1}}\boldsymbol{y}$$

$$\sigma_n(\boldsymbol{x}) = \underbrace{\phi(\boldsymbol{x})^T\phi(\boldsymbol{x})}_{K_{**}} - \underbrace{\phi(\boldsymbol{x})^T\Phi^T}_{K_*}\underbrace{\left(\Phi^T\Phi+\sigma^2\mathbf{I}\right)^{-1}}_{K^{-1}}\underbrace{\Phi\phi(\boldsymbol{x})}_{K_*^T}$$

Equivalent !

- Posterior Predictive distribution for GP

$$p(y_* \mid \boldsymbol{y}) \sim \mathcal{N}\left(\underbrace{K_*K^{-1}\boldsymbol{y}}_{\mu(x)}, \underbrace{K_{**}-K_*K^{-1}K_*^T}_{\sigma(x)}\right)$$

Thompson Sampling for BLR

- Randomly draw w from the **posterior** $w \mid X \sim \mathcal{N}(\mu_n, \Sigma_n)$
- Find the optimum using w $$x_* = \underset{x \in \mathcal{X}}{\operatorname{argmax}} x^T w$$

Approximate kernel by Random Fourier feature

- Transform the data space $x \in R^d$ to feature space $\phi(x) \in R^M$

- Using Bochner theorem, we approximate

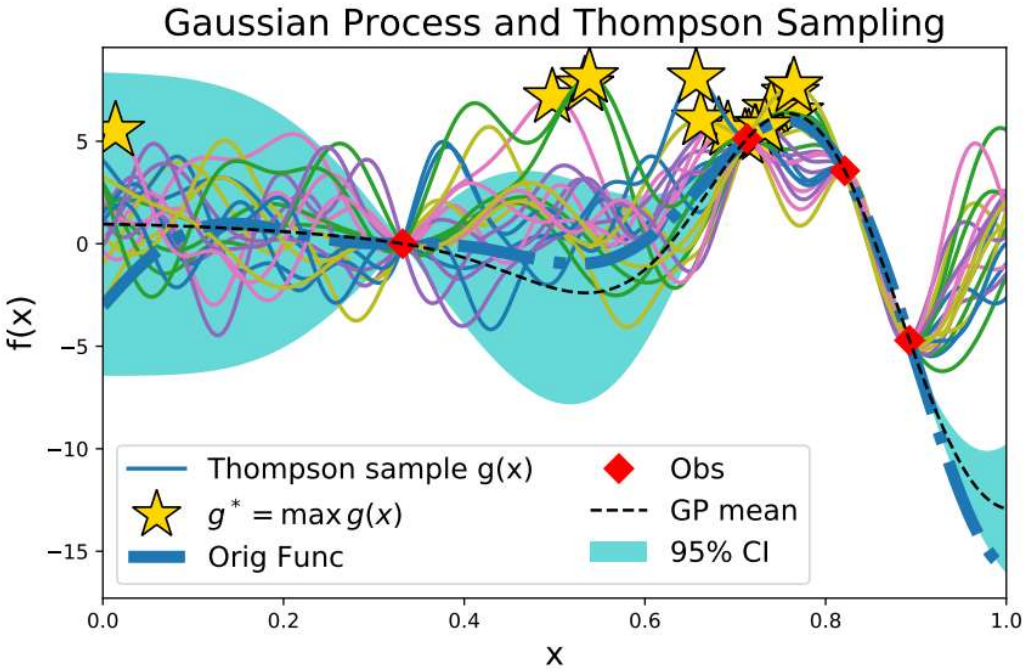
$$k(x, x') = E\left[\cos(\omega^T x + b], \sin(\omega^T x + b)\right]$$

where $b \sim U[0, 2\pi]$, $\omega \sim \mathcal{N}(0, \sigma_l^2 I)$, and σ_l^2 is the length-scale of the SE kernel.

Rahimi, A. and Recht, B., Random features for large-scale kernel machines. NeurIPS, 2008.

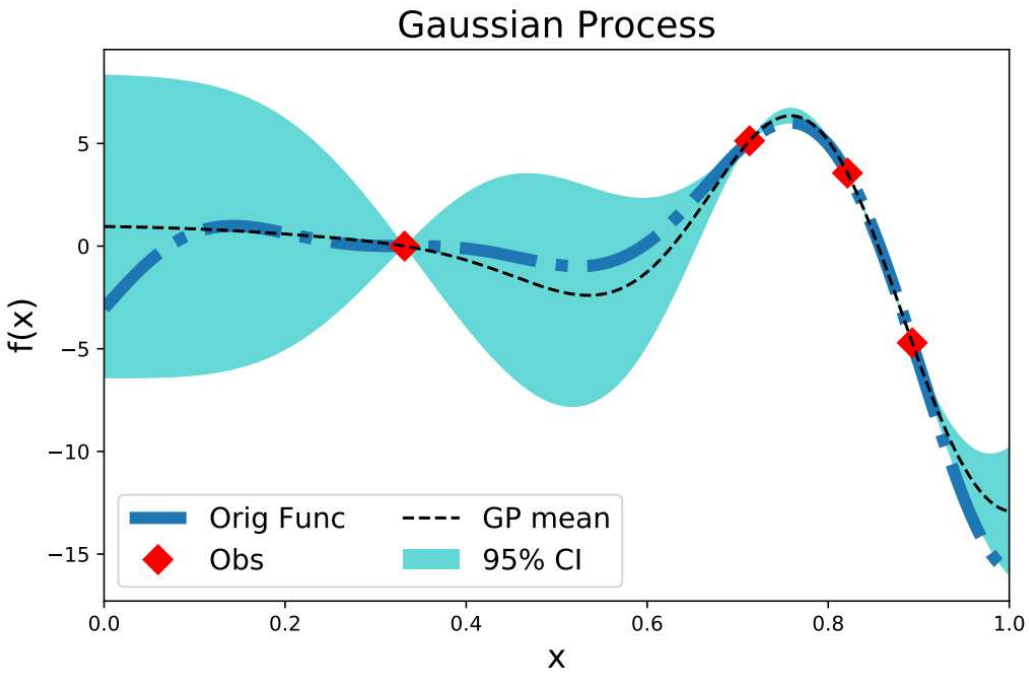
Thompson Sampling from a Gaussian process

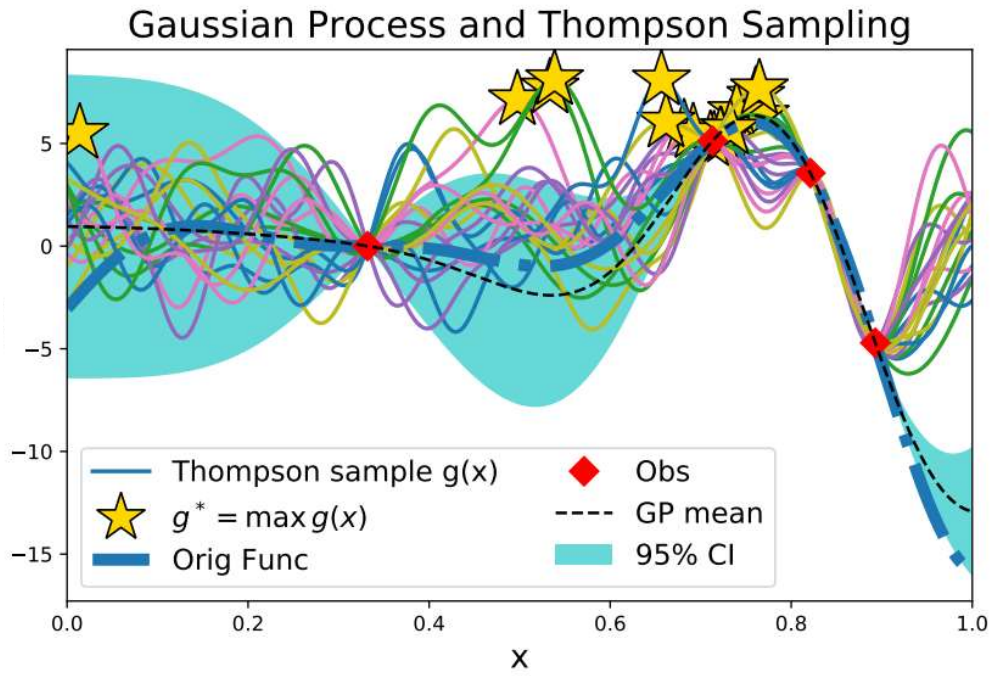
- $\omega_m \sim \mathcal{N}(0, \sigma_l^2 I_{d\times d}), \forall m \leq M$ and $W = [\omega_1, ..., \omega_m] \in R^{M\times d}$
- $\phi(x) = \sqrt{\frac{2\alpha}{M}} \{\cos(Wx + b), \sin(Wx + b)\}$, $\Phi = [\phi(x_i)]_{i=1}^N, \forall x_i \in D_t$
- $g(x) = \phi(x)^T [\Phi^T \Phi + \sigma^2 I]^{-1} \Phi^T y$



Thompson Sampling to Sample the Optimum Locations

- Thompson Sampling draws samples $g()$ from GP.
- Each yellow stars x^* is the maximizer of the sampled function $g()$
- We consider x^* as the perceived optimal samples



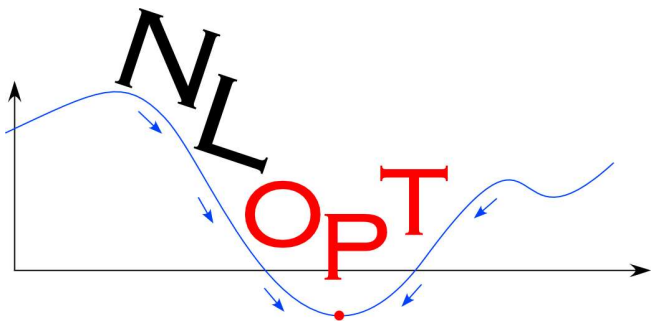


Bayesian Optimization Algorithm

Repeat

1. Fit a GP model to the data $(x_i, y_i)_{i=1}^{t}$
2. Define the acquisition function $\alpha(x)$ from the GP model
3. Select the next query $x_{t+1} = \arg\max\ \alpha(x)$
4. Evaluate the black-box to get the score $y_{t+1} = f(x_{t+1})$

After defining $\boldsymbol{\alpha}()$, the optimization is done using popular toolboxes.
Optimizing $\boldsymbol{\alpha}()$ is easier and cheaper than $\boldsymbol{f}()$.



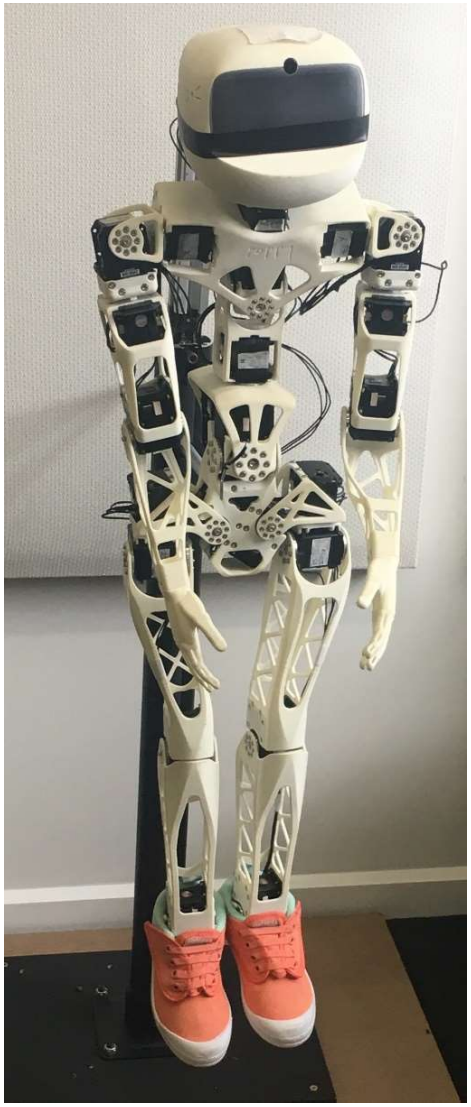
Outline Part 1: Bayesian Optimization

- Bayesian Optimization
- Gaussian Processes
- Acquisition Functions
  - Upper Confidence Bound (UCB) and Expected Improvement (EI)
  - Thompson Sampling (TS)
  - Optimization toolbox for acquisition function.
- Applications
- Demo Mini BayesOpt

Applications

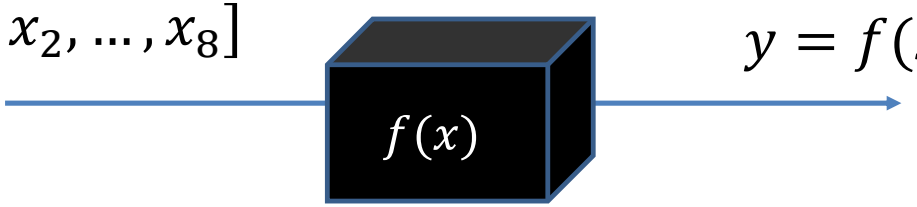
- Robot control design
- Alloy design
- Heat-treatment design
- Machine learning hyper-parameter tuning.
- ….

Applications – Robot Design



Tuning 8 parameters (x) to maximize the speed (y) of bipedal robot.

Physical configurations
$x = [x_1, x_2, \ldots, x_8]$

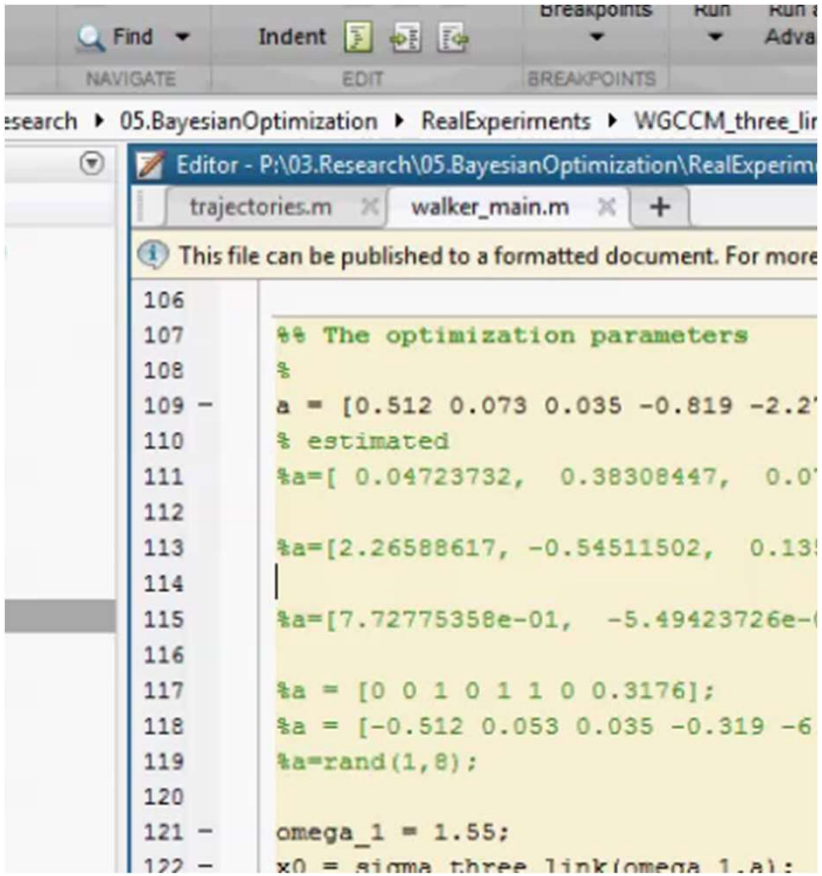


Walking speed
$y = f(x)$

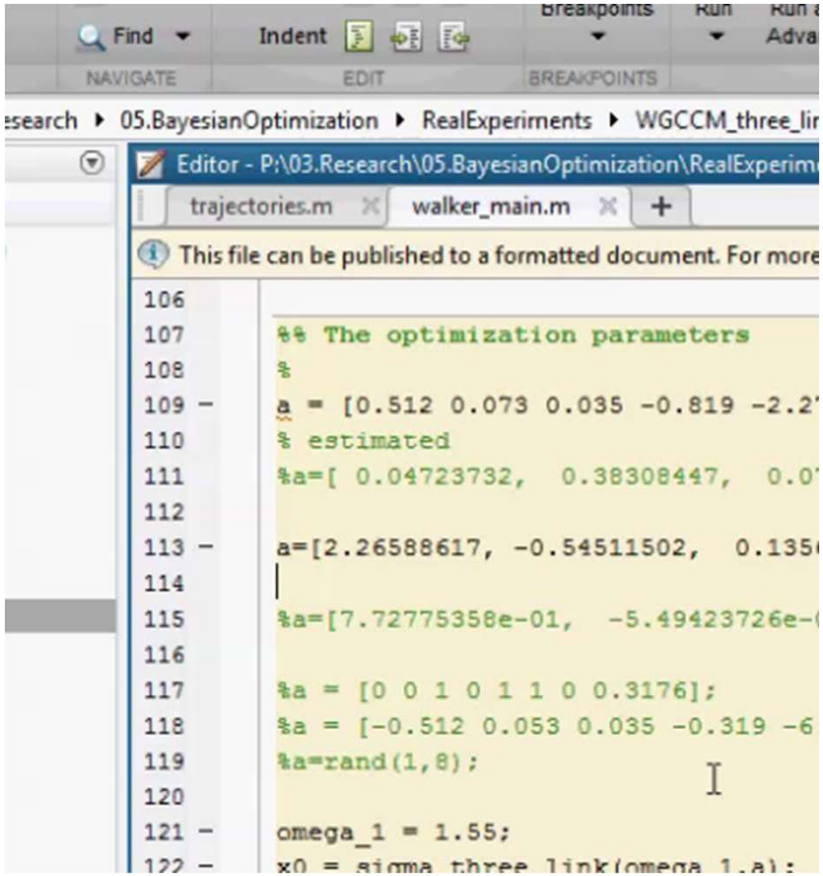
Vu Nguyen et al, *ACML 2017*.

Applications – Robot Design

- Before tuning



- After tuning with Bayesian optimization



Bayesian Optimization for Tuning Deep Learning

- Deep learning offers breakthrough in image recognition, speed recognition and self-driving cars.

- The DL performance critically depends on the hyperparameters.

- Hyperparameters include
  1. #layers,
  2. #node per layer,
  3. mini-batch size,
  4. learning rate…

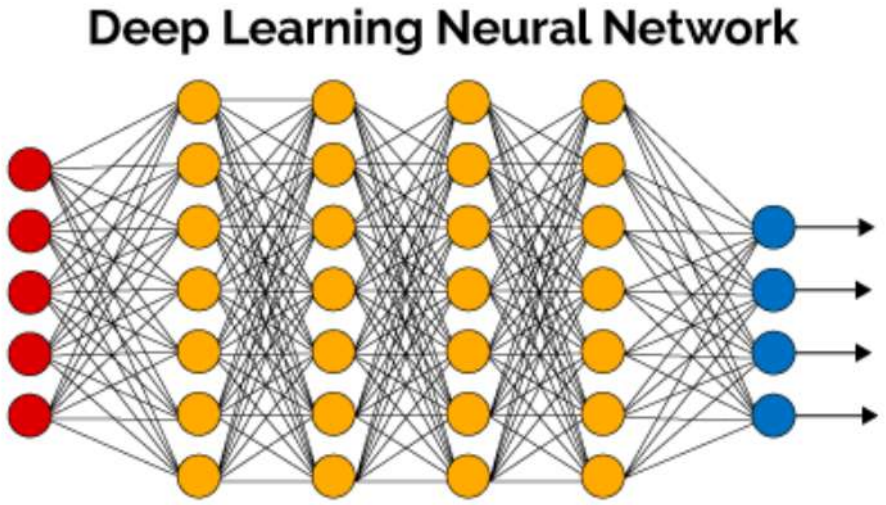
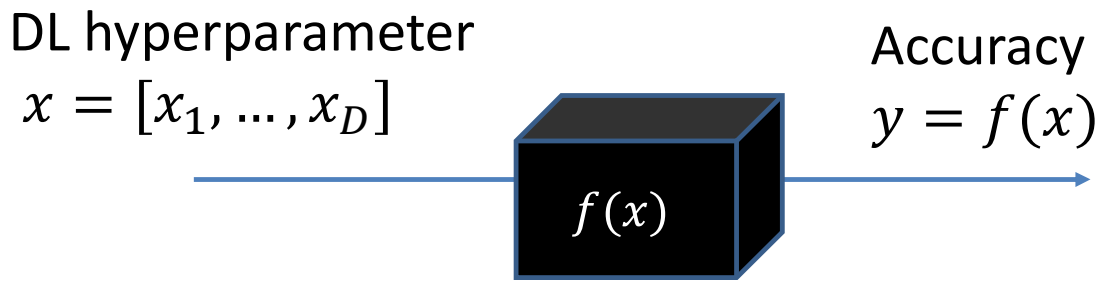
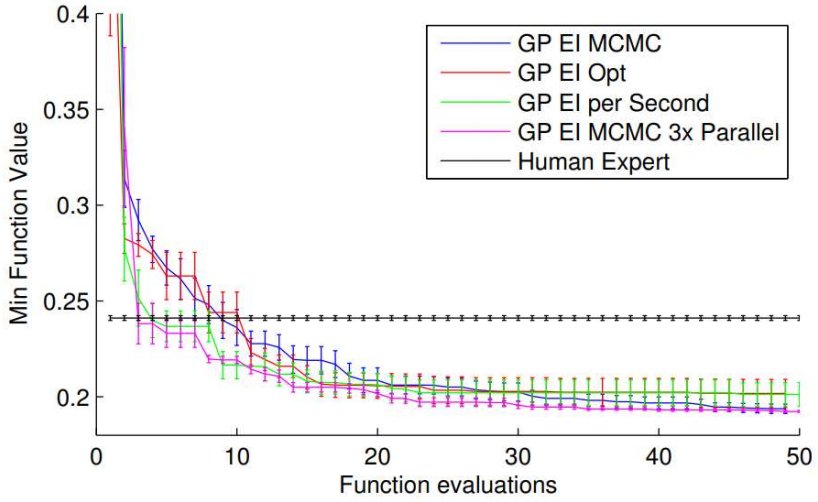


Photo from hackernoon.com

Bayesian Optimization for Tuning Deep Learning



- Bayes Opt is better than human expert tuning for deep learning.



Snoek, J. et al. Practical Bayesian optimization of machine learning algorithms. NIPS 2012.

BO for Tuning Deep Reinforcement Learning

- Win-rate from 50% to 66%

Bayesian Optimization in AlphaGo

Yutian Chen, Aja Huang, Ziyu Wang, Ioannis Antonoglou, Julian Schrittwieser, David Silver & Nando de Freitas

DeepMind, London, UK
yutianc@google.com

Abstract

During the development of AlphaGo, its many hyper-parameters were tuned with Bayesian optimization multiple times. This automatic tuning process resulted in substantial improvements in playing strength. For example, prior to the match with Lee Sedol, we tuned the latest AlphaGo agent and this improved its win-rate from 50% to 66.5% in self-play games. This tuned version was deployed in the final match. Of course, since we tuned AlphaGo many times during its development cycle, the compounded contribution was even higher than this percentage. It is our hope that this brief case study will be of interest to Go fans, and also provide Bayesian optimization practitioners with some insights and inspiration.

Short Summary

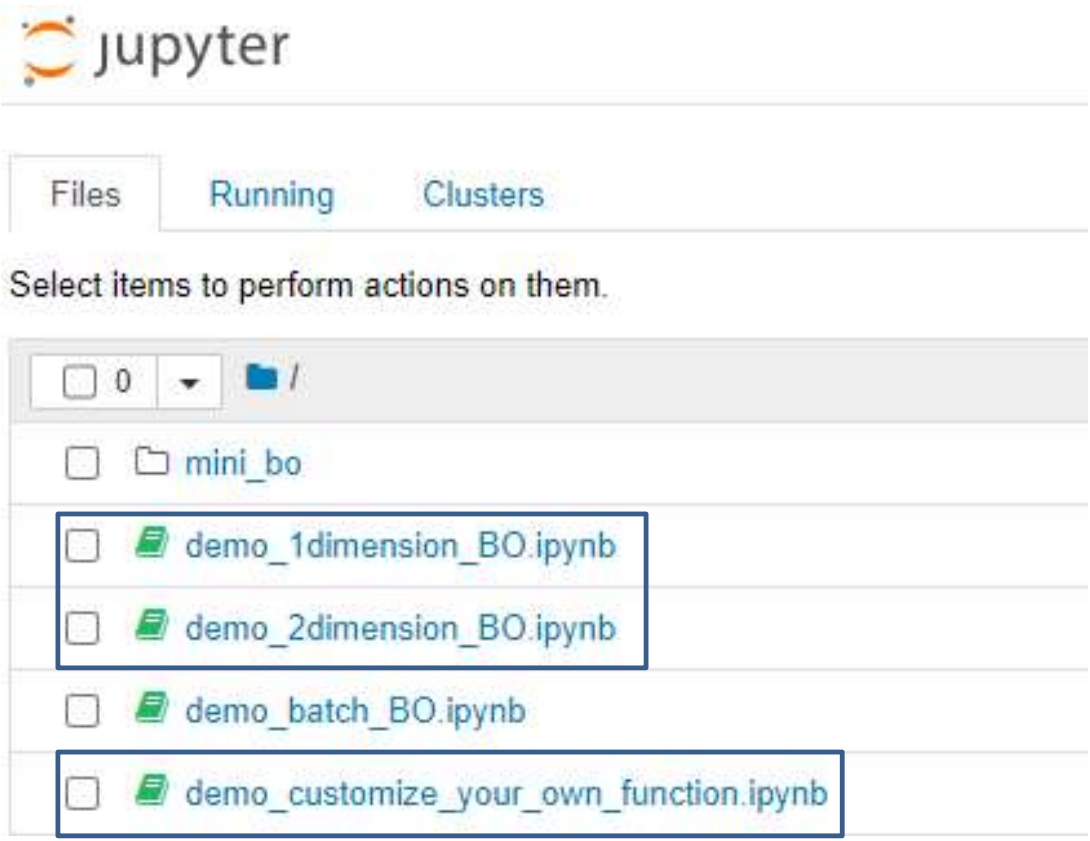
- Bayesian optimization is essential for hyper-parameters tuning of the black-box functions (e.g., machine learning algorithm and experimental design).
- Bayesian optimization has both theoretical guarantee and empirical success that it performs better than random search and grid search (especially for high dimensions).
- Bayesian optimization is an active research direction.

Library for Bayesian Optimization

Package	License	URL	Language	Model
SMAC	Academic non-commercial license.	http://www.cs.ubc.ca/labs/beta/Projects/SMAC	Java	Random forest
Hyperopt	BSD	https://github.com/hyperopt/hyperopt	Python	Tree Parzen estimator
Spearmint	Academic non-commercial license.	https://github.com/HIPS/Spearmint	Python	Gaussian process
Bayesopt	GPL	http://rmcantin.bitbucket.org/html	C++	Gaussian process
PyBO	BSD	https://github.com/mwhoffman/pybo	Python	Gaussian process
MOE	Apache 2.0	https://github.com/Yelp/MOE	Python / C++	Gaussian process

MiniBayesOpt

- Package: vu-nguyen.org/BOTutorial_ACML20
- Github repository: MiniBayesOpt



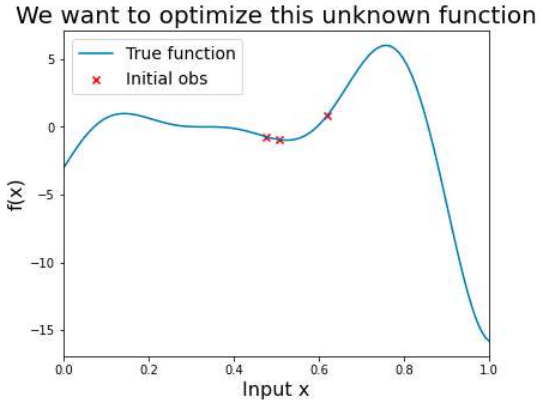
https://github.com/ntienvu/minibo

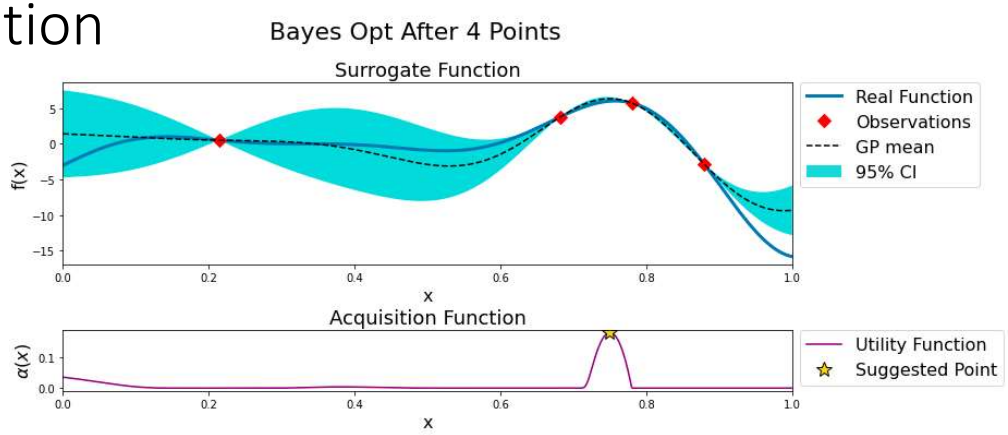
MiniBayesOpt

https://github.com/ntienvu/minibo

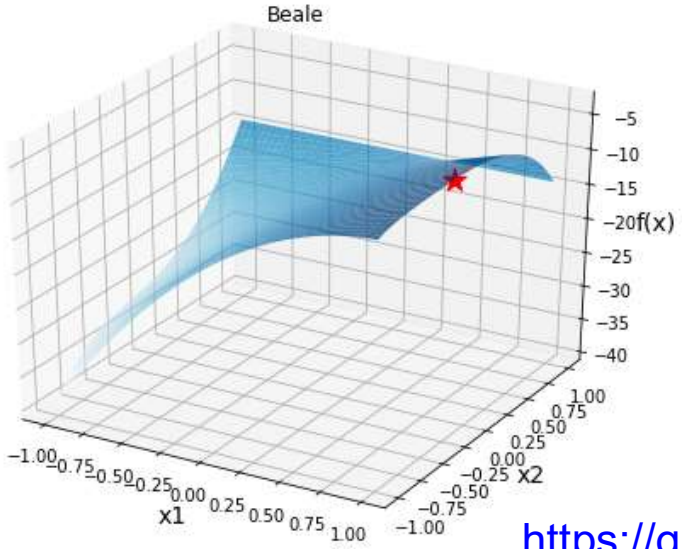
Demo in 1D and 2D

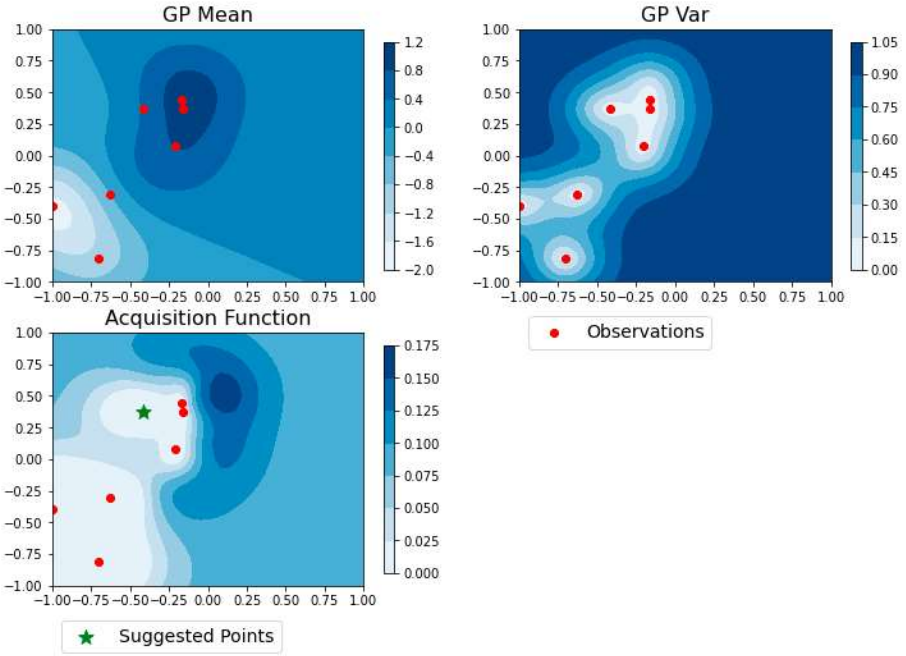
- Illustration on 1d optimization





- Illustration on 2d optimization





https://github.com/ntienvu/minibo

Demo using your own function

- Customize your own black-box function

```
class YourFunction:
    def __init__(self):
        self.bounds = np.asarray([          # define the search range for each
                                  [-5,10], # variable 1: min=-5 ; max=10
                                   [0,15] # variable 2: min=0; max=15
                                  ])

        self.input_dim = self.bounds.shape[0] # this is 2

        # do we want to maximize the function or minimize ?
        self.ismax=-1  # set -1 if we want to minimize

        self.name='MiniBO'          # set the name of your function

    # evaluate y=f(X)
    def evaluate_single_fx(self,X): # this is actually a Branin function
        X = np.reshape(X,self.input_dim)
        x1,x2=X[0],X[1]

        a,b=1,5.1/(4*np.pi**2)
        c,r,s=5/np.pi,6,10
        t=1/(8*np.pi)
        y=a*(x2-b*x1*x1+c*x1-r)**2+s*(1-t)*np.cos(x1)+s

        return y*self.ismax # return the (-1) * fx for minimization problem
```

ranges of each variables

$x_1 \in [-5,10]$

$x_2 \in [0,15]$

if **maximize,** set it **= 1**

if minimize, set it = -1

given the input x

return the output $y = f(x)$

(defined by yourself)

https://github.com/ntienvu/minibo

How to use Bayesian Optimization

- Run for multiple iterations

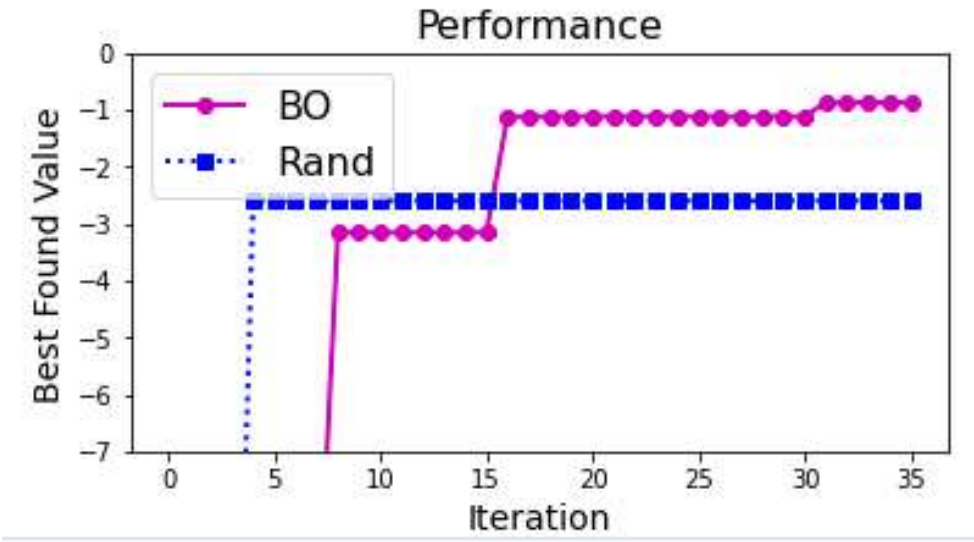
```
In [6]: NN=15*myfunction.input_dim
        for index in range(0,NN):

            bo.select_next_point()

            print(tabulate([[ index,np.round(bo.X_ori[-1],3), np.round(bo
                       headers=['Iter','Selected x', 'Output y=f(x)', 'Be
        print("============================================================
        idxMax=np.argmax(bo.Y_ori)
        print(tabulate([[ np.round(bo.X_ori[idxMax],3), np.round(bo.Y_ori
                    headers=['Best found x', 'Best found f(x)']))
```

```
estimated lengthscale [0.0353949]
  Iter  Selected x         Output y=f(x)    Best Observed Value
------  ---------------  ---------------  ---------------------
     0  [-0.862  5.967]          -18.501                -18.501
  Iter  Selected x         Output y=f(x)    Best Observed Value
------  ---------------  ---------------  ---------------------
     1  [-1.818  7.607]          -10.586                -10.586
  Iter  Selected x         Output y=f(x)    Best Observed Value
------  ---------------  ---------------  ---------------------
     2  [-2.858 10.062]           -3.158                 -3.158
  Iter  Selected x         Output y=f(x)    Best Observed Value
------  ---------------  ---------------  ---------------------
     3  [-4.881 10.904]          -46.923                 -3.158
  Iter  Selected x         Output y=f(x)    Best Observed Value
------  ---------------  ---------------  ---------------------
     4  [-1.63   9.679]           -9.986                 -3.158
  Iter  Selected x       Output y=f(x)    Best Observed Value
```

- Compare with Random search



https://github.com/ntienvu/minibo

Question and Answer

Reference

- V. Nguyen, S. Gupta, S. Rana, C. Li, S. Venkatesh. Regret for Expected Improvement over the Best-Observed Value and Stopping Condition. ACML, 2017.
- Hernandez-Lobato, J. M., Hoffman, M. W., & Ghahramani, Z. Predictive entropy search for efficient global optimization of black-box functions. NeurIPS, 2014.
- Hennig, P., & Schuler, C. J. Entropy search for information-efficient global optimization. Journal of Machine Learning Research, 13, 1809–1837, 2012.
- Rasmussen, C. E. Gaussian processes for machine learning, 2006.
- Mockus, J., Tiesis, V., & Zilinskas, A. The application of bayesian methods for seeking the extremum. Towards global optimization, 2(117-129), 2, 1978.
- Snoek, J., Larochelle, H., & Adams, R. P. Practical bayesian optimization of machine learning algorithms. NeurIPS, 2012.
- V. Nguyen, M. A. Osborne. Knowing The What But Not The Where in Bayesian Optimization. ICML, 2020.
- Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., & De Freitas, N. (2015). Taking the human out of the loop: A review of Bayesian optimization. Proceedings of the IEEE, 104(1), 148-175.
- Brochu, E., Cora, V. M., & De Freitas, N. (2010). A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. arXiv preprint arXiv:1012.2599.

Reference

- Jones, D. R., Perttunen, C. D., & Stuckman, B. E. Lipschitzian optimization without the lipschitz constant. Journal of Optimization Theory and Applications, 79(1), 157–181, 1993.
- Kushner, H. J. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. Journal of Basic Engineering, 86(1), 97-106, 1964.
- Rahimi, A. and Recht, B., Random features for large-scale kernel machines. NeurIPS, 2008.
- Li, C., de Celis Leal, D.R., Rana, S., Gupta, S., Sutti, A., Greenhill, S., Slezak, T., Height, M. and Venkatesh, S., 2017. Rapid Bayesian optimisation for synthesis of short polymer fiber materials. *Scientific reports*, *7*(1), p.5683.
- F. Hutter, H. H. Hoos, and K. Leyton-Brown, ‘‘Sequential model-based optimization for general algorithm configuration,’’ Learning and Intelligent Optimization, Berlin, Germany: Springer-Verlag, 2011, pp. 507–523.
- J. Snoek et al., ‘‘Scalable Bayesian optimization using deep neural networks,’’ in ICML 2015.
- J. Parker-Holder, V. Nguyen, SJ. Roberts. Provably Efficient Online Hyperparameter Optimization with Population-Based Bandits. NeurIPS, 2020.
- Bishop, Christopher M. Pattern recognition and machine learning. springer, 2006.