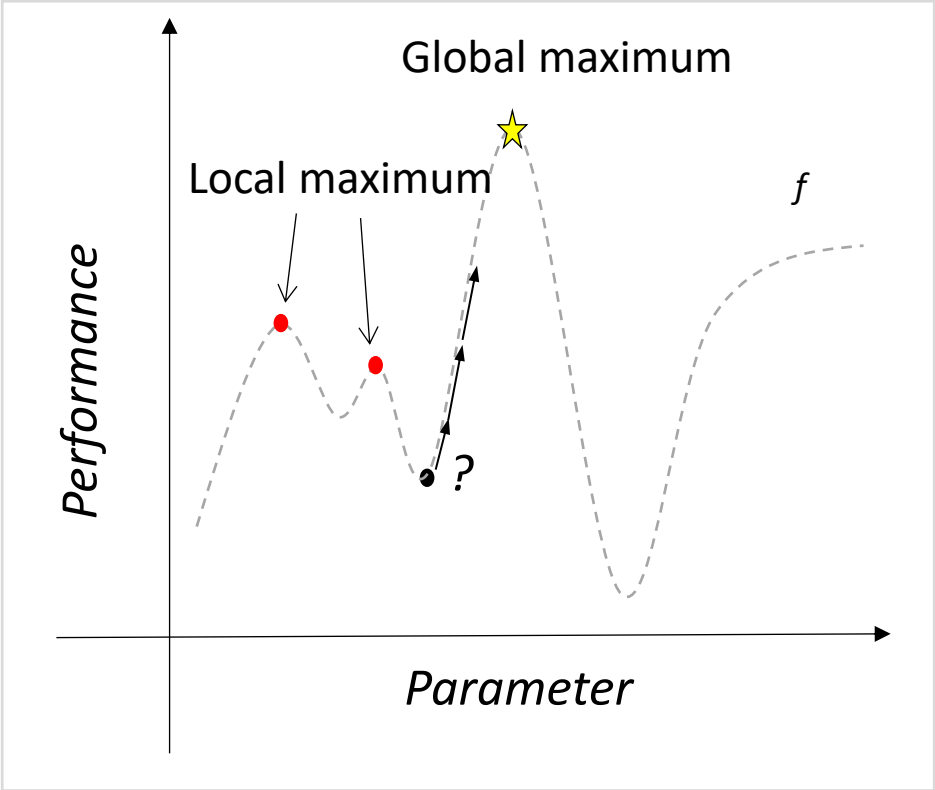
Bayesian Optimization

Felix Luong



- Gradient-based methods

(Newton's method, BFGS with multi-star, etc.)

➡ Unable to optimize **without gradient** ✗

- Derivative-free methods

(*DIRECT, Genetic Algorithm, Particle Swarm Optimization, etc.*)

➡ **Sample inefficient** due to costly evaluations ✗

➡ **Bayesian Optimization (BO)**

- ✓ Black-box optimization
- ✓ Sample efficiency
- ✓ Global optimization

Introduction

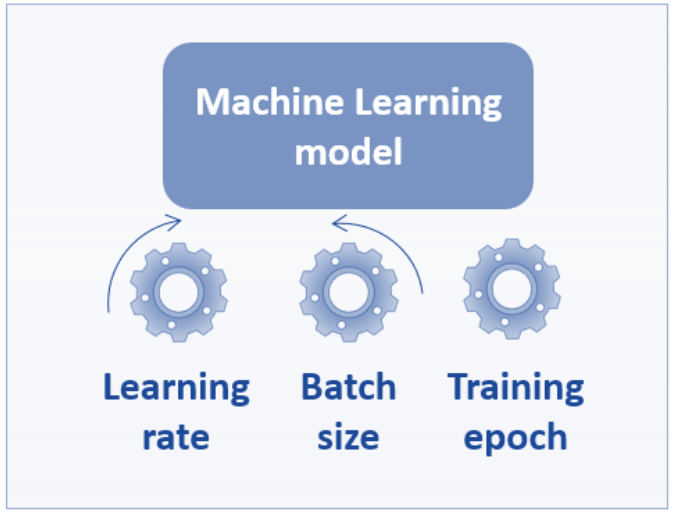
Automotive & Car design

Maximize driving safety
Minimize fuel usage

Heat treatment

Maximize alloy's strength
Minimize cost

Hyper-parameter tuning

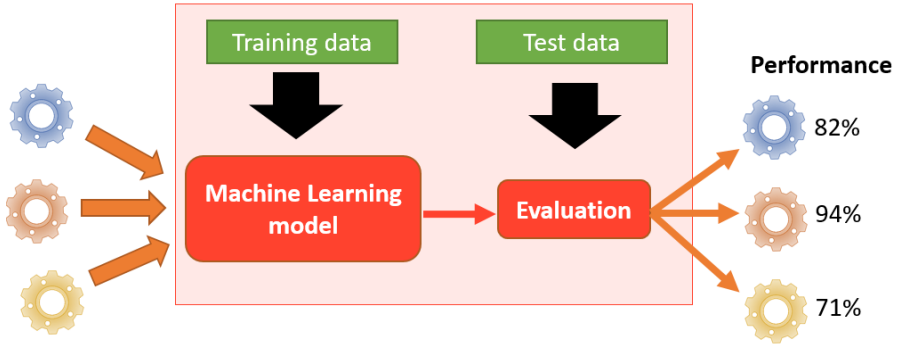


Maximize accuracy
Minimize training time

Images:
https://www.akka-technologies.com/sectors/automotive
https://www.reliance-foundry.com/blog/tempering-steel#gref

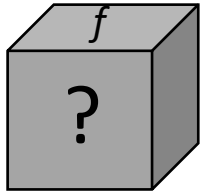
Introduction

Real-world optimizations are **costly** and **black-box**.



$$x = [x_1, \ldots, x_d]$$

Hyper-parameters



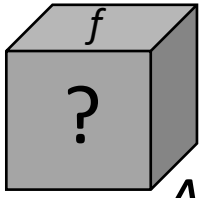
y

Accuracy

- Require training and testing the model.

$$x = [x_1, \ldots, x_d]$$

Temperature, time, etc.



y

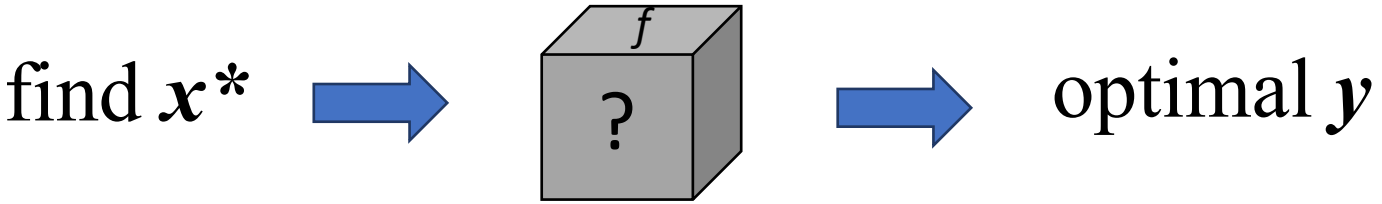
Alloy's strength

- Need real materials and time for experiments.

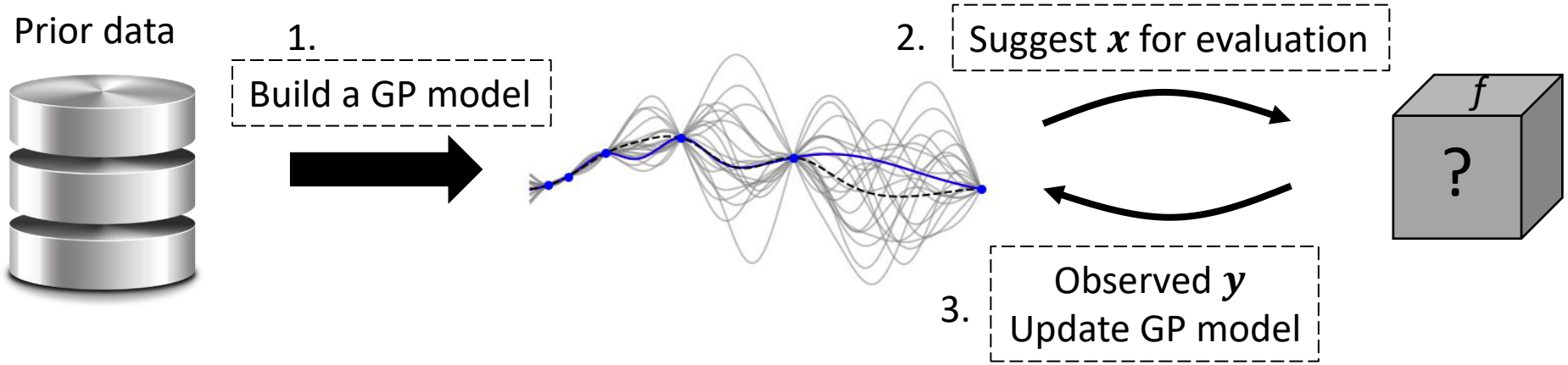
Images:
https://newatlas.com/pnnl-titanium-alloy/42677/

Introduction

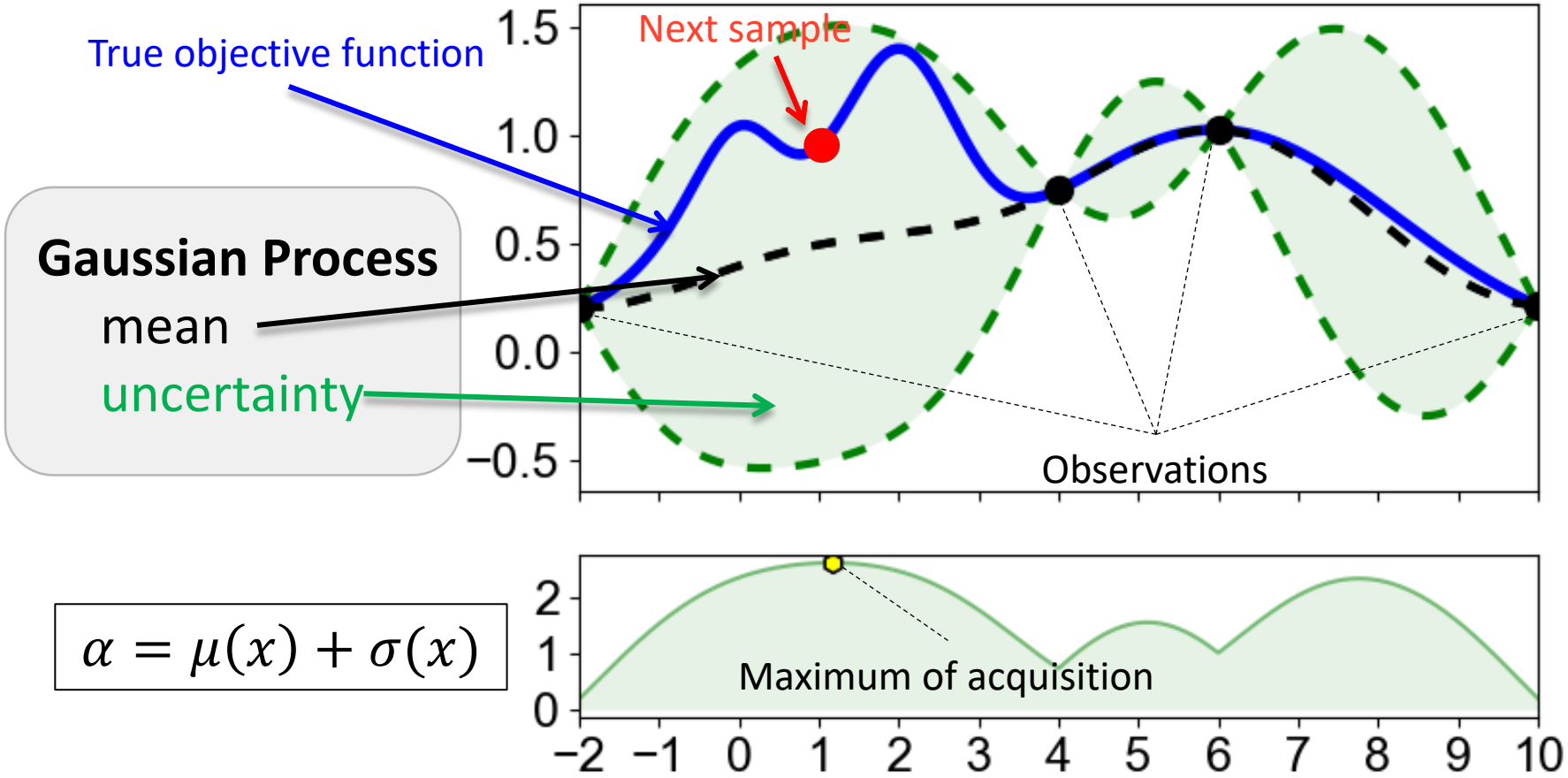
- BO is an efficient tools for automatically finding the optimal solution of expensive black-box functions.



- BO uses a Gaussian process to model the black-box objective function.

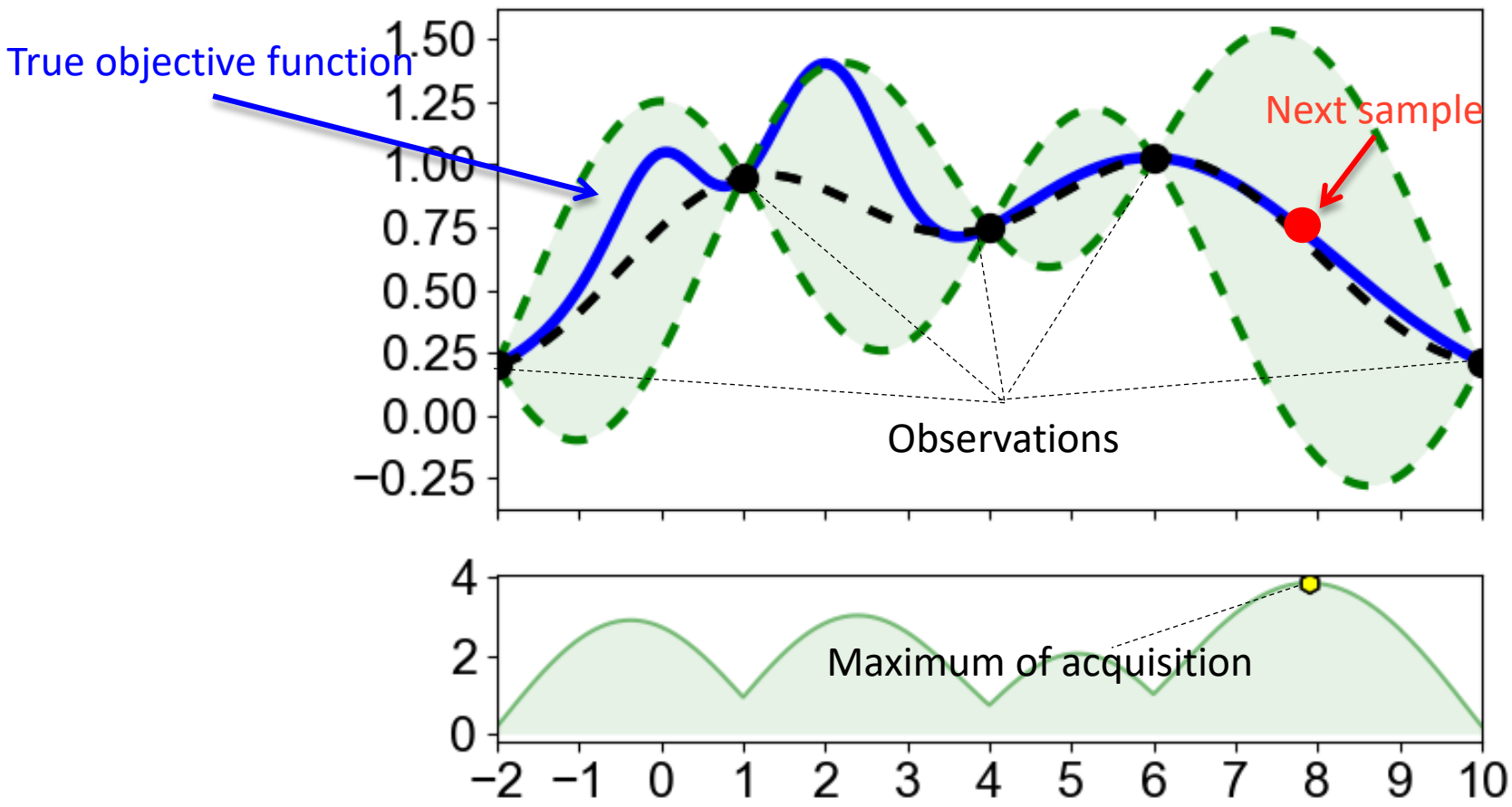


Background



$$\alpha = \mu(x) + \sigma(x)$$

Background


True objective function
Next sample
Observations
Maximum of acquisition
1.50
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
4
2
0
−2 −1 0 1 2 3 4 5 6 7 8 9 10

Common acquisition functions

- Probability of Improvement (PI)

$$\alpha_{\mathrm{PI}} = \mathrm{P}\big(\mathrm{f}(x) > f(x^{+})\big) = \boldsymbol{\Phi}\left(\frac{\mu(x) - f(x^{+}) - \xi}{\sigma(x)}\right)$$

- Expected improvement (EI)

$$\alpha_{\mathrm{EI}} = \begin{cases} z\,\boldsymbol{\Phi}\left(\frac{z}{\sigma(x)}\right) + \sigma(x)\boldsymbol{\phi}\left(\frac{z}{\sigma(x)}\right) & \text{if } \sigma(x) > 0 \\ 0 & \text{if } \sigma(x) = 0 \end{cases}$$

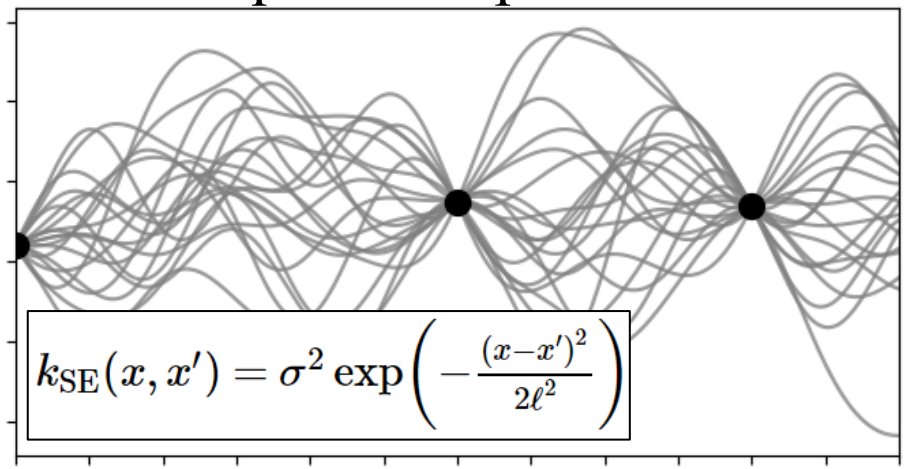
- Upper Confidence Bound (UCB)

$$\alpha_{\mathrm{UCB}} = \mu(x) + \beta\sigma(x)$$

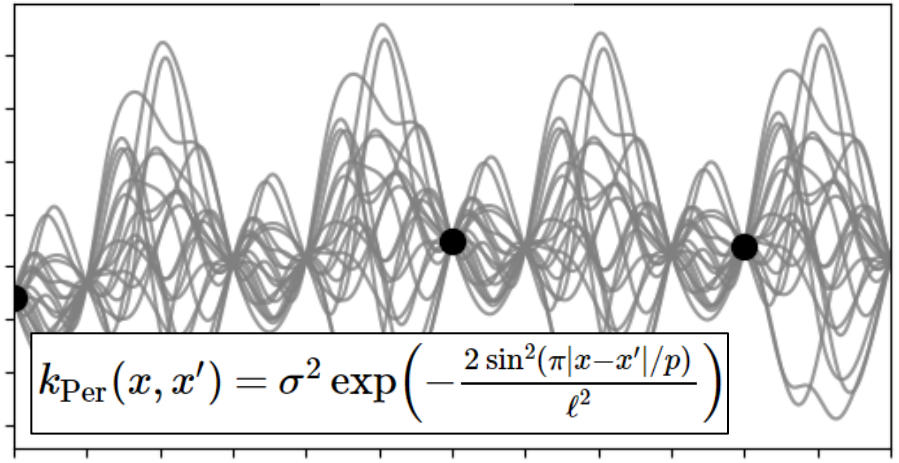
Kernel functions

- Important part of the Gaussian process used **to model the objective function**.
- Capture the **smoothness** and **correlation** between points.

Squared Exponential



Periodic



$$\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^{\boldsymbol{d}}$$

Problem statement

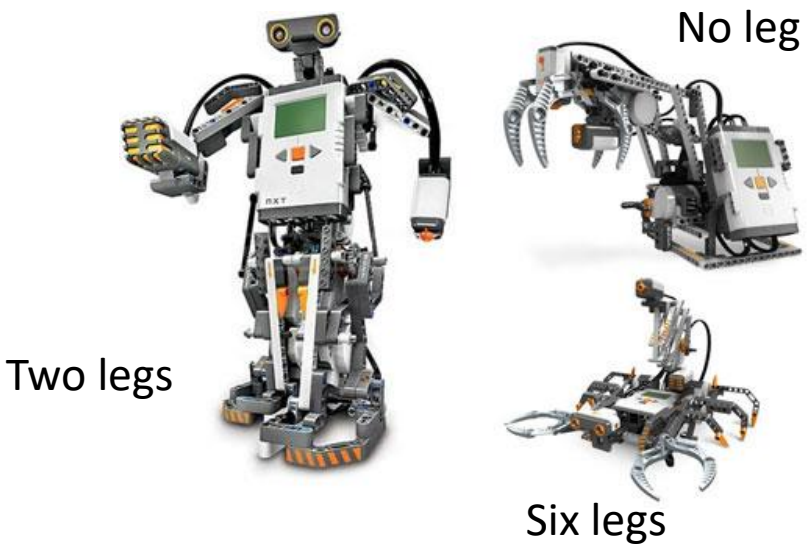
- Consider tuning hyper-parameters of a machine learning model as an expensive black-box function.
- Use BO to <u>find the best configuration of hyper-parameters</u>.
- Some important hyper-parameters:
  - **Continuous**: Learning rate, momentum, dropout rate,..
  - **Discrete**: Number of layer/hidden units, batch size, epochs,..
- BO incorporates a Gaussian Process to sample a next configuration by maximizing an acquisition function in **a continuous domain**.

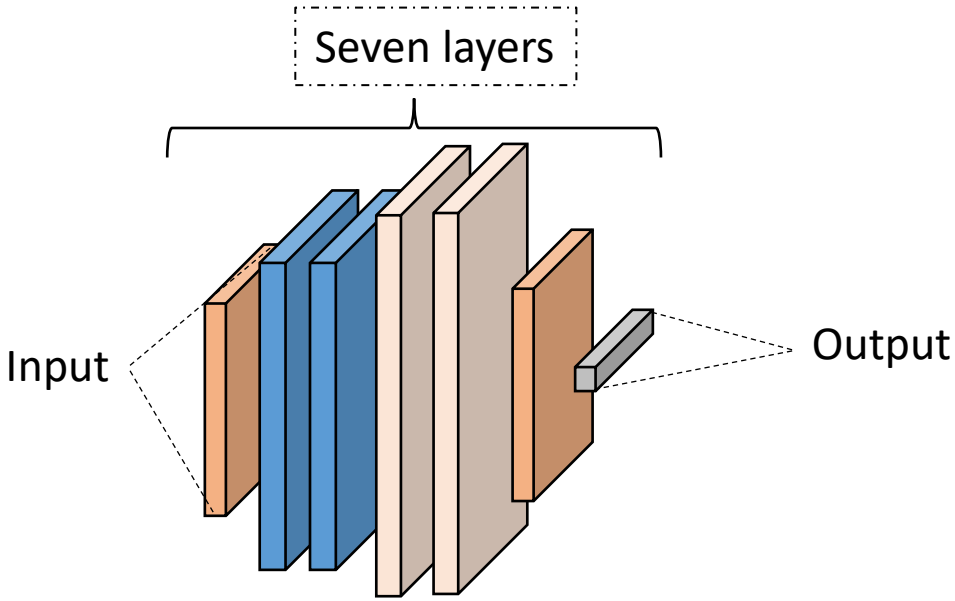
How to deal with discrete variables?

Discrete variables

- **Discrete variable**: numeric variables. Have a <u>countable</u> number of values between any two values.

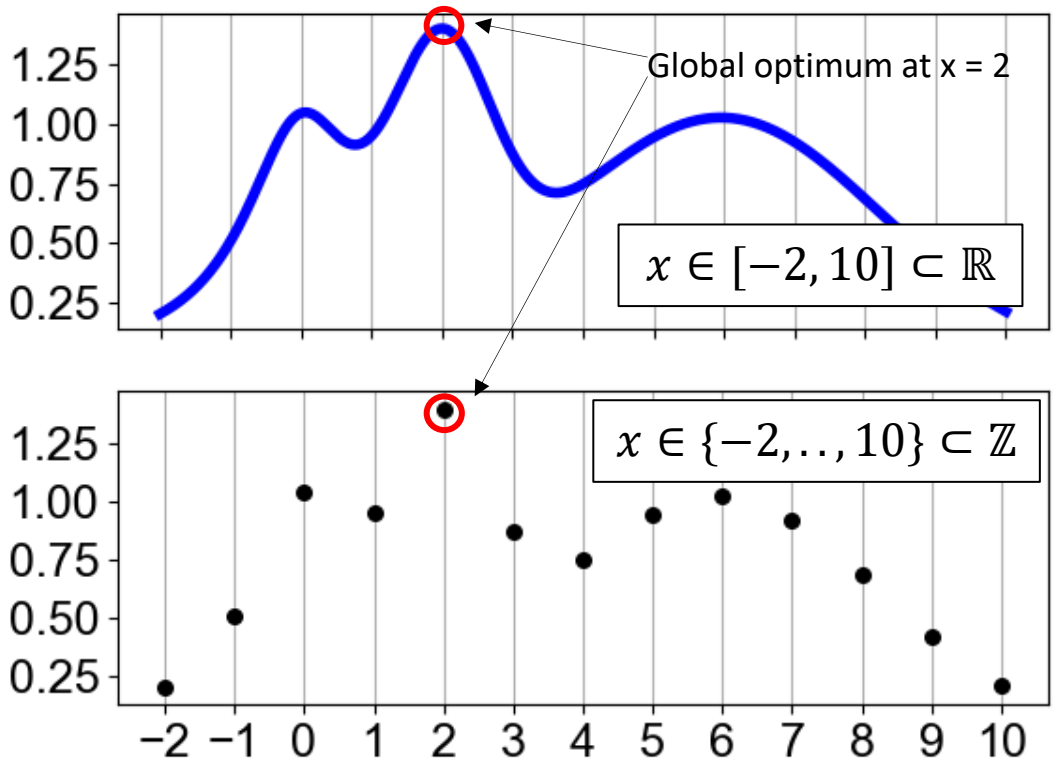
  Example: number of wheels/legs/arms of a robot;
  number of hidden units/layers of a neural network model.





Optimization with discrete variables

Assume a discrete variable $\boldsymbol{x}$ has a relation to the function evaluation $\boldsymbol{y}$.



We want to find the optimal point among **numerous** discrete points.

<u>For example</u>:

- Tune a neural network's hyper-parameters:

  Hidden layers: 1 – 10

  Hidden units: 1 – 1000

- The number of discrete points grows **exponentially.**
- **Expensive** to try all settings**.**

Example function: $y = f(x) = e^{-(x-2)^2} + e^{\left(\frac{-(x-6)^2}{10}\right)} + \frac{1}{e^{x^2}+1}$

Optimization with discrete variables

Existing methods for dealing with discrete inputs in optimization:

- Bayesian Optimization with Naïve Rounding
- Transformation of inputs [1]
- Sequential model-based optimization (SMAC) [2]
- Tree-Parzen Estimators (TPE) [3]

[1] Garrido-Merchán, E. C., & Hernández-Lobato, D. (2018). **Dealing with categorical and integer-valued variables in Bayesian optimization with Gaussian processes**. *arXiv preprint arXiv:1805.03463*.
[2] Hutter, F., Hoos, H.H., Leyton-Brown, K.: Sequential model-based optimization for general algorithm configuration. In: LION11. pp. 507{523. Springer (2011)
[3] Bergstra, J. S., Bardenet, R., Bengio, Y., & Kégl, B. (2011). **Algorithms for hyper-parameter optimization**. In *Advances in neural information processing systems* (pp. 2546-2554).

Existing method 1: BO with Naïve Rounding

- Treat **discretes variables** as **continuous**.
- Need to **round** the continuous value of a discrete variable to the nearest discrete value.

BO with Naïve Rounding Algorithm

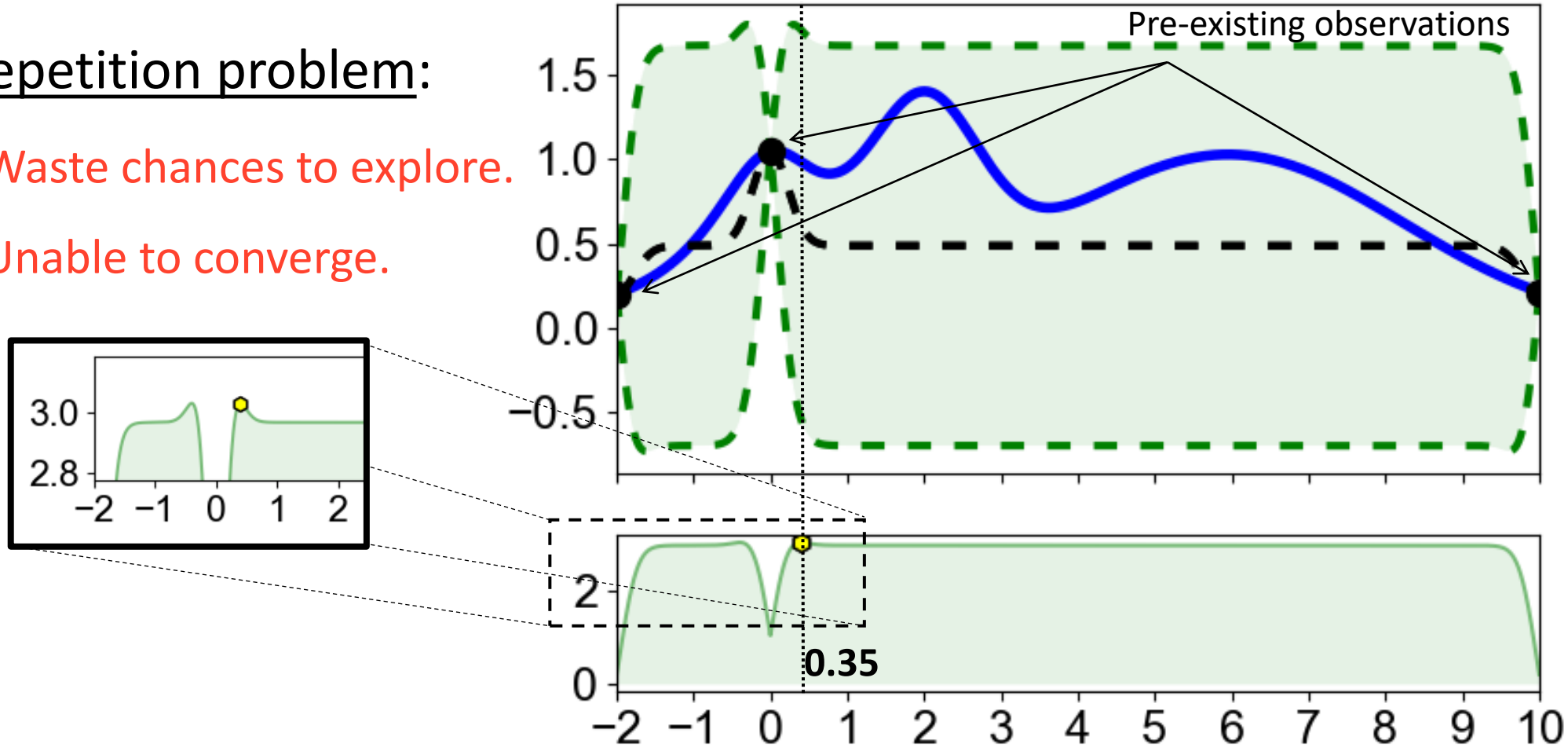
1. **for** $t = 1, 2, \ldots, T$ **do**
2. Build a model GP with D_t
3. Select new x_{t+1} by optimizing acquisition α
   $x_{t+1} = \text{argmax}_{x \in X \subseteq \mathbb{R}^d} \alpha_{UCB}(\beta_t)$
4. $\boldsymbol{x_{t+1} = \text{round}(x_{t+1})}$
5. Query the objective function to obtain y_{t+1}
6. Augment data $D_{t+1} = \{D_t, (x_{t+1}, y_{t+1})\}$
7. **end for**

Round the continuous value of discrete variables

Existing method 1: BO with Naïve Rounding

<u>Repetition problem</u>:

- Waste chances to explore.
- Unable to converge.



Existing method 2: Transformation of inputs [1]

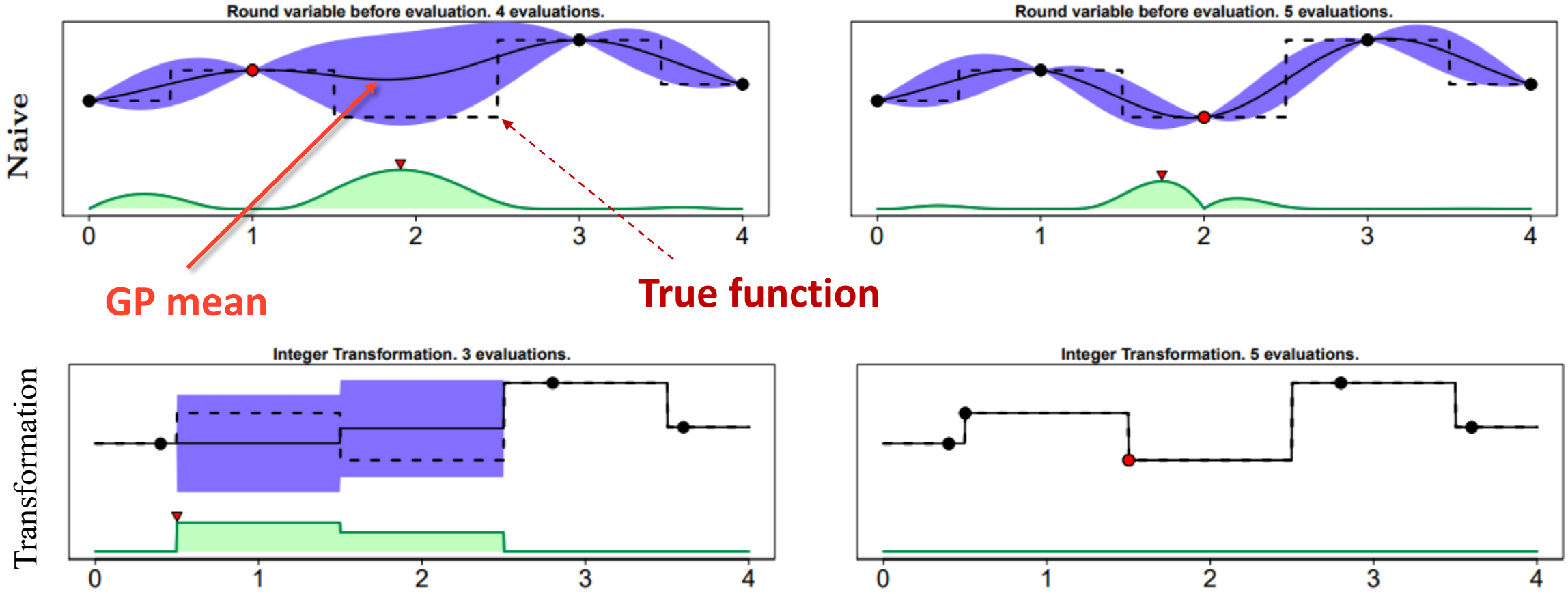
Transform the input points to ***k*** to obtain an alternative covariance.

$$\boxed{k'(x, x') = k\big(T(x), T(x')\big),}$$

where $\boldsymbol{T}(\boldsymbol{x})$ is a transformation of all discrete inputs of $f(\cdot)$.

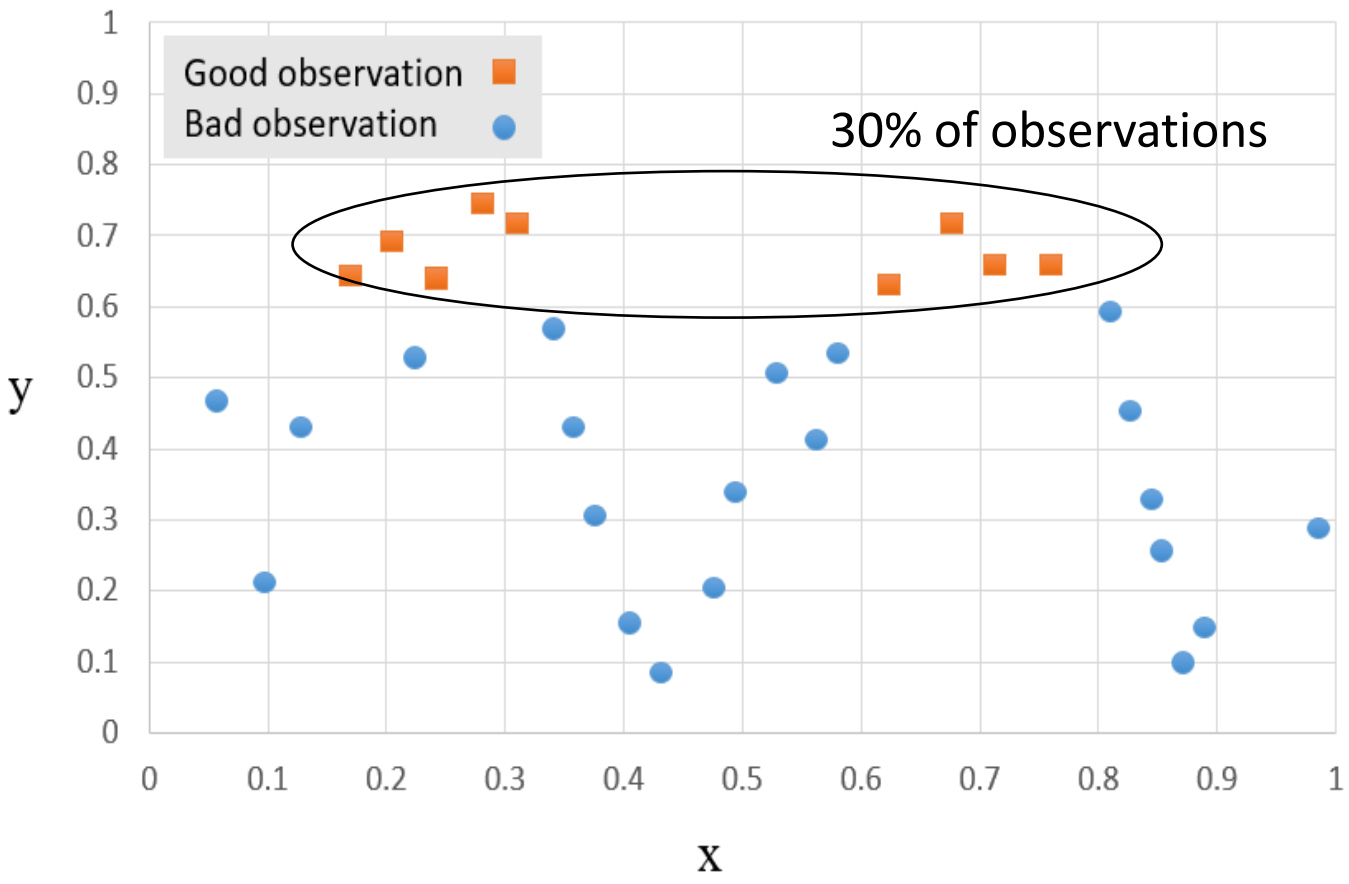
The values of discrete variables **are rounded to the closest discrete** value.

Comparison between Naïve and Transformation methods



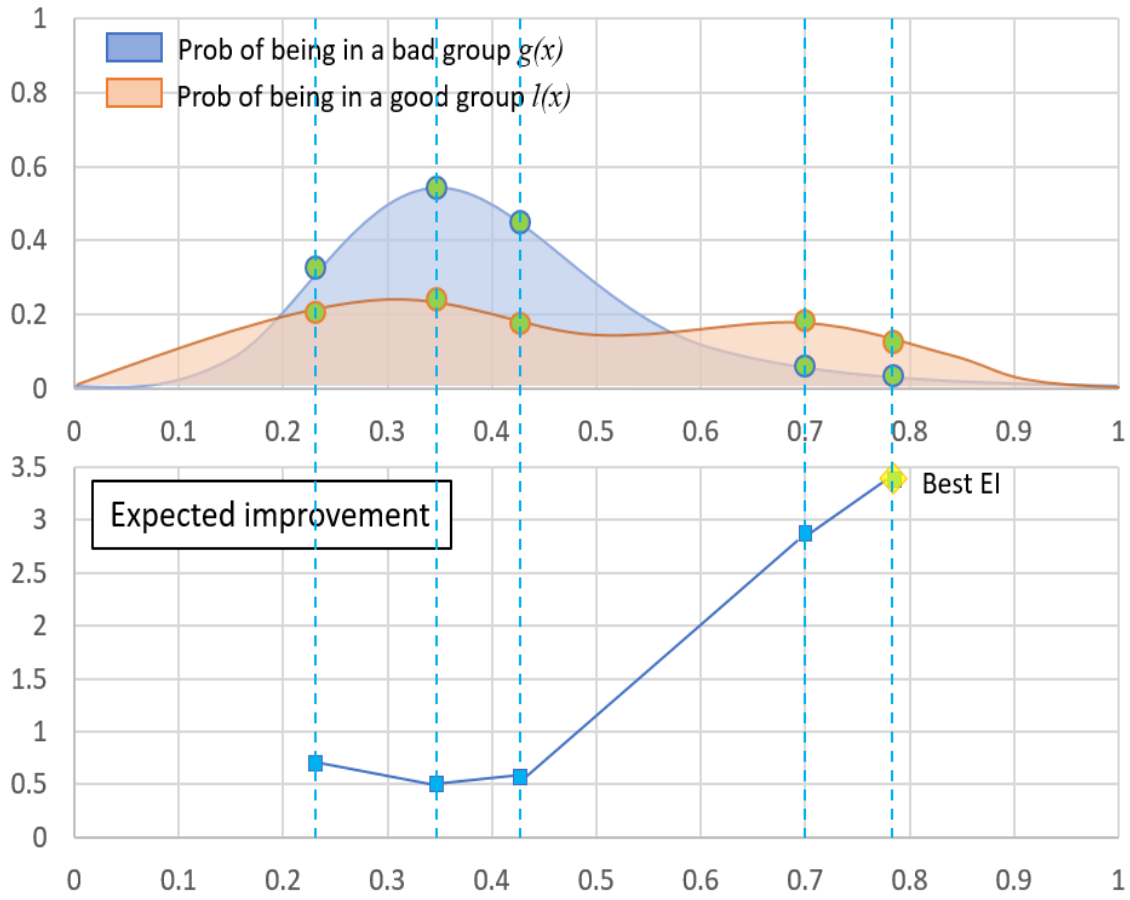
Images reference [1]

Existing method 3: Tree-Parzen Estimators [2]



1. Sort observations by evaluation.
2. Take a percentage of **best observations** for good group.
3. Remaining observations belong to bad group.

Existing method 3: Tree-Parzen Estimators [2]



$$EI_{y*} = \left(\gamma + \frac{g(x)}{l(x)}(1-\gamma)\right)^{-1}$$

Advantages:

- Sample from the discrete distribution.

Disadvantages:

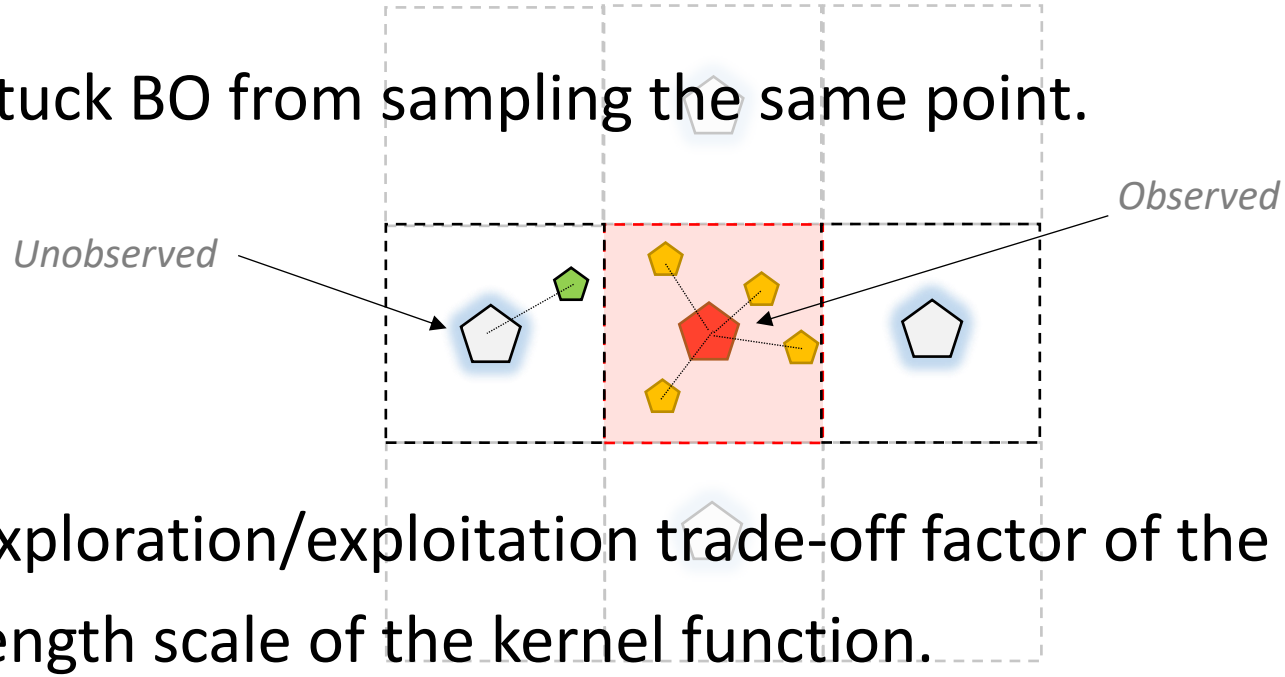
- Need sufficient amount of observations.
- Randomly draw from distributions.

Proposed method

- Question:

  *"How to **avoid repetition** in optimizing functions with **discrete inputs**?*

- *Idea:* Unstuck BO from sampling the same point.

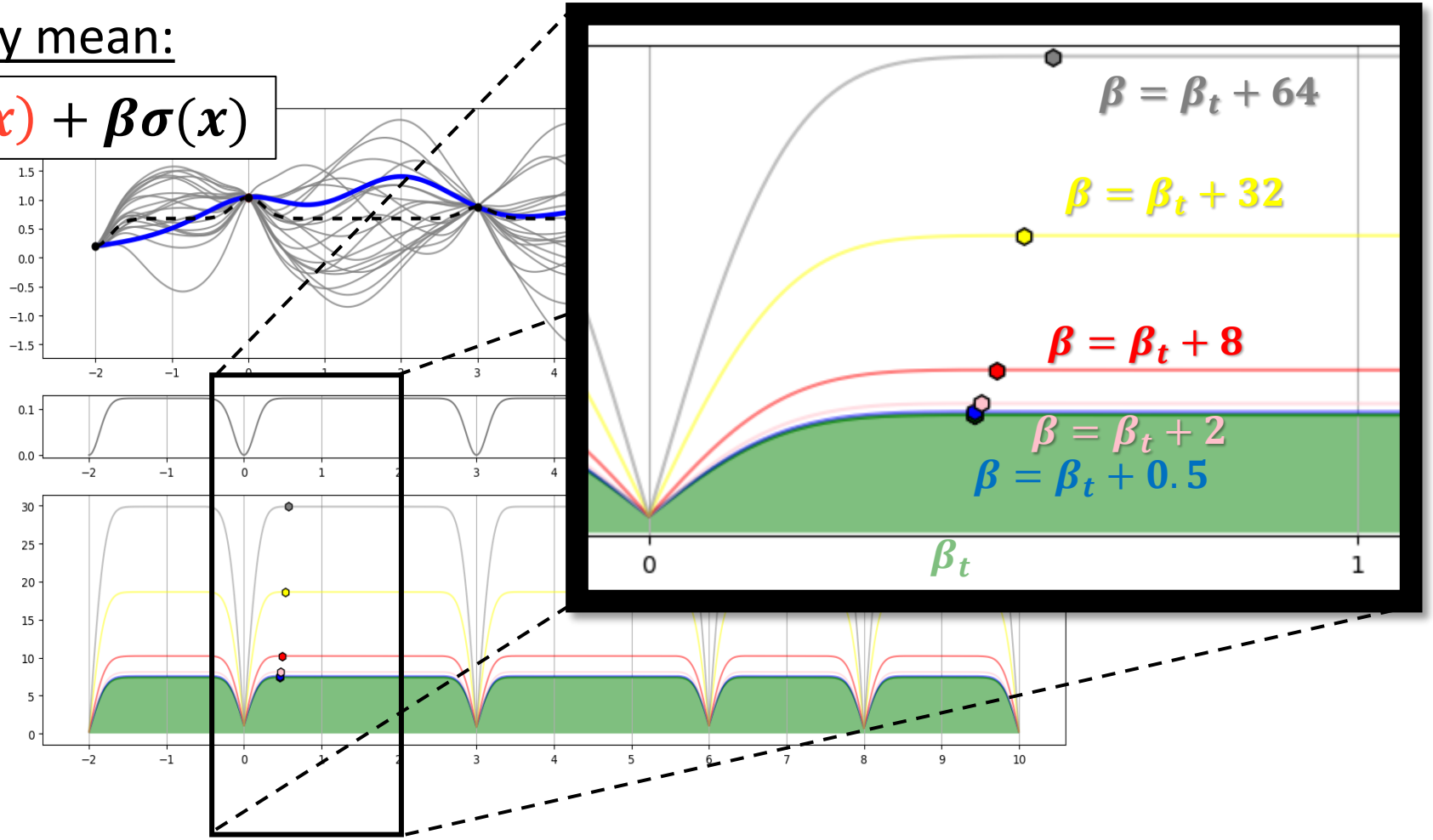


- *Solution:*
  - Adjust exploration/exploitation trade-off factor of the acquisition function.
  - Adjust length scale of the kernel function.

Proposed method - Adjust β

<u>Dominated by mean:</u>

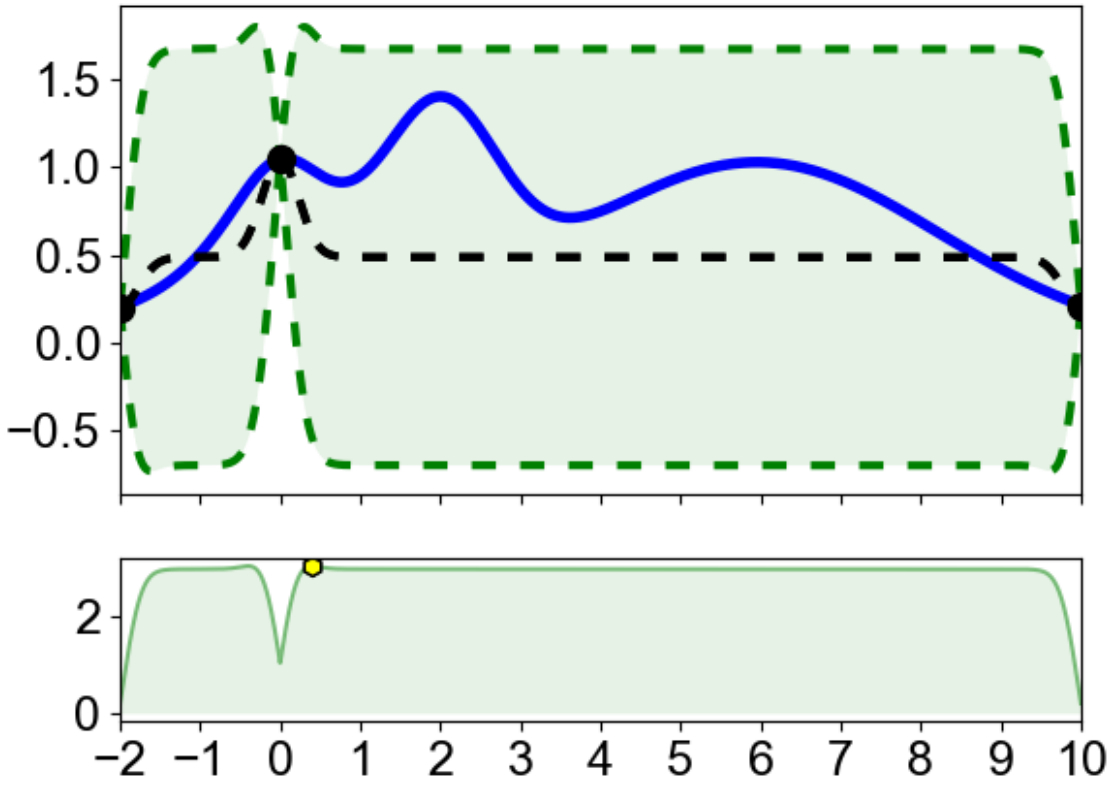
$$\alpha_{UCB} = \mu(x) + \beta\sigma(x)$$



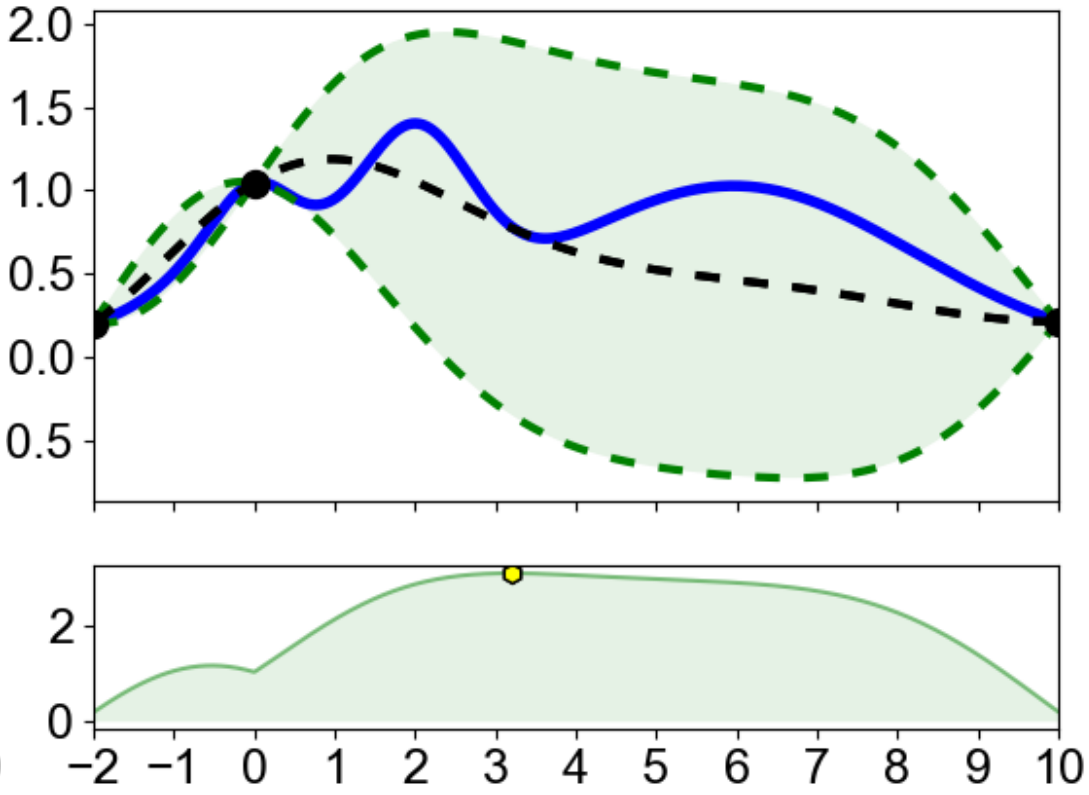
Proposed method - Adjust l

<u>Dominated by variance:</u> $\boldsymbol{\alpha_{UCB} = \mu(x) + \beta\sigma(x)}$

Small length scale



Large length scale



Proposed method

Discrete BO Algorithm

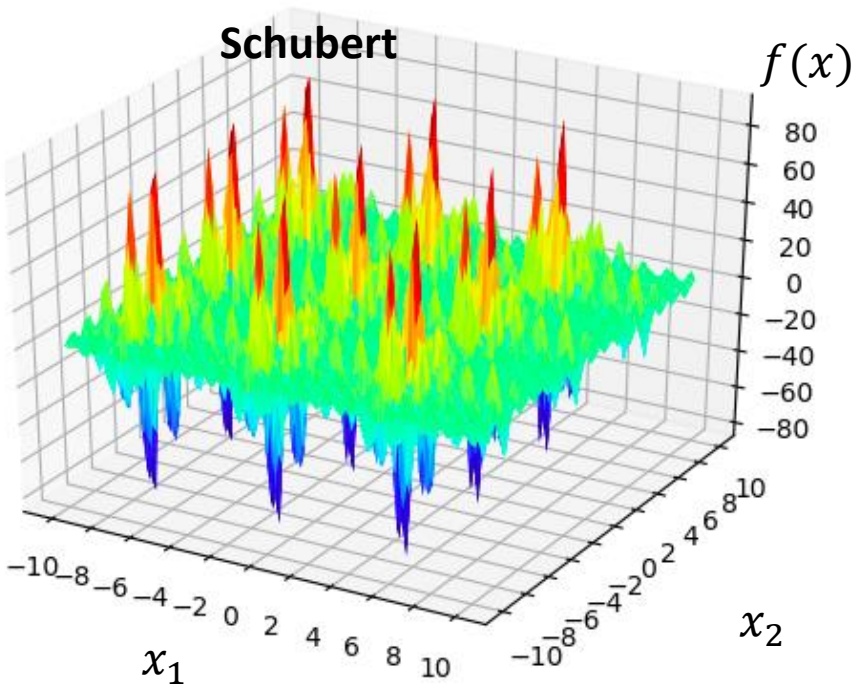
1. **for** t = 1, 2, …, T do
2. Build a model GP with D_t
3. Sample the next point x_{t+1} = $\mathrm{argmax}_{x \in X \subseteq \mathbb{R}^d}\ \alpha_{UCB}\ (\beta_t)$
4. $x_{t+1} = \mathrm{round}(x_{t+1})$ #
5. **if** $x_{t+1} \in D_t$:
   # Find $\boldsymbol{\beta}, \boldsymbol{l}$ just
6. Search for opti
7. x_{t+1} = $\mathrm{argmax}_{x \in X \subseteq \mathbb{R}^d}\ \alpha_{UCB}\ (\beta_t)$ with $\boldsymbol{\beta}^*, \boldsymbol{l}$
8. $x_{t+1} = \mathrm{round}(x_{t+1})$
9. Query the objective function to obtain y_{t+1}
10. Augment data $D_{t+1} = \{D_t, (x_{t+1}, y_{t+1})\}$
11. **end for**

$$\beta^*, l^* = \mathrm{argmin}\boldsymbol{g}(\beta_t + \Delta\beta, l)$$

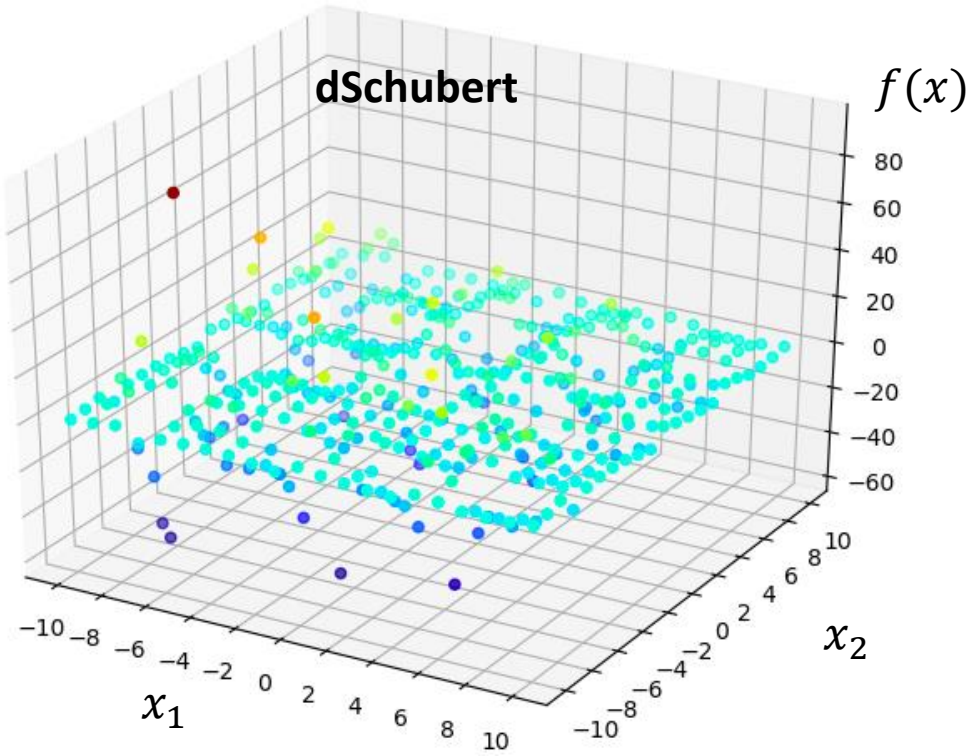
$$\boldsymbol{g}(\beta_t + \Delta\beta, l) = \Delta\beta + \|x_{t+1} - x'_{t+1}\|_2 + P(x'_{t+1})$$

Result – Synthetic functions

Synthetic function	Dimension	Range
dSchubert 2D	2	$x_1, x_2 \in \{-10, \dots 10\} \subset \mathbb{Z}$

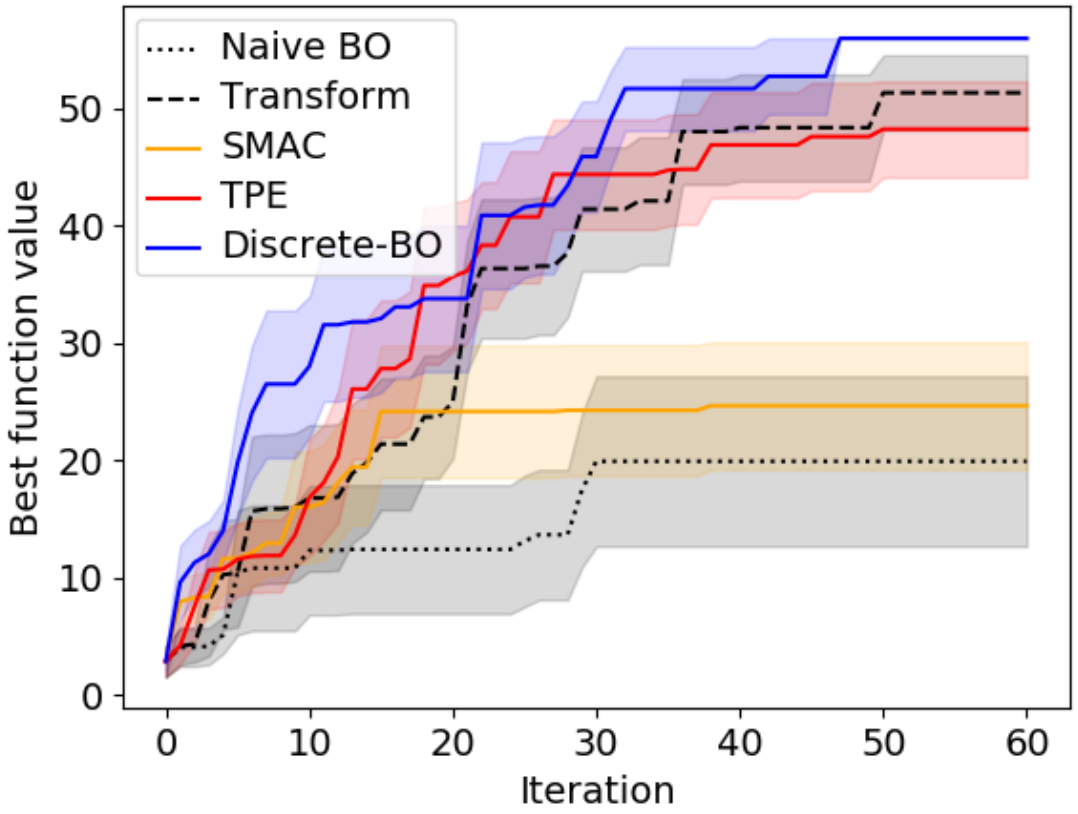


Discretize

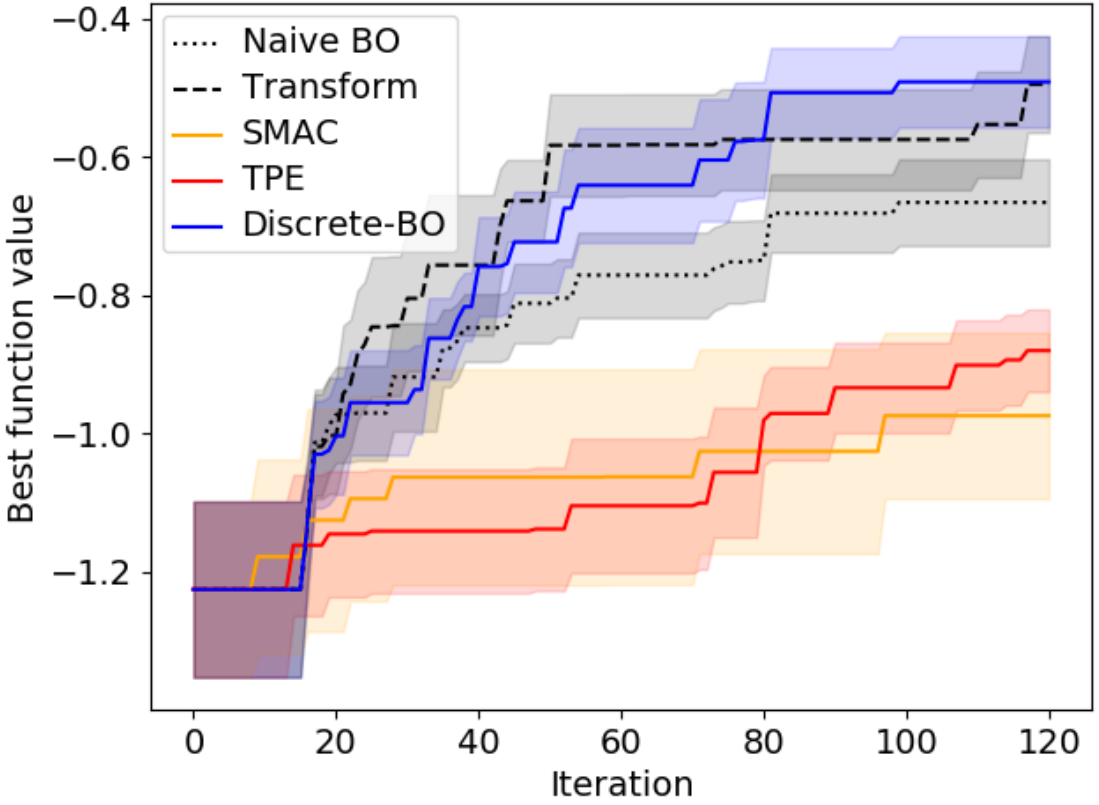


Result – Synthetic functions

Schubert2d

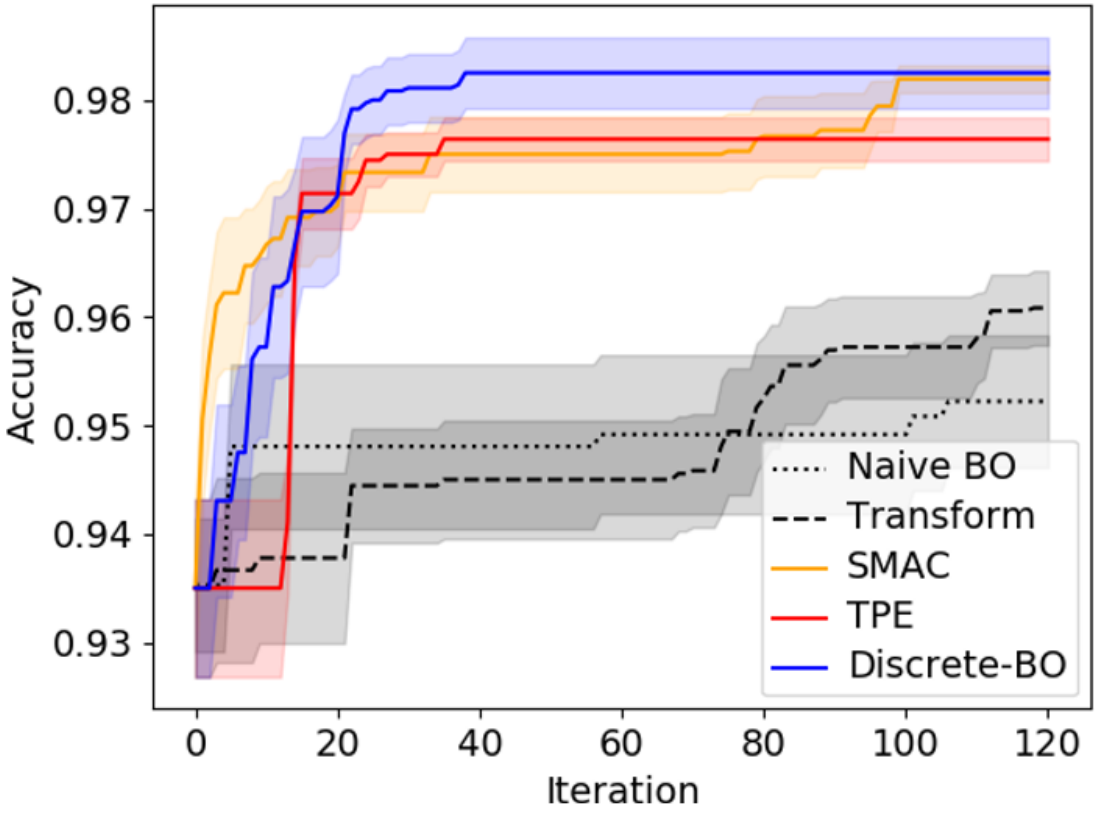


Griewank3d

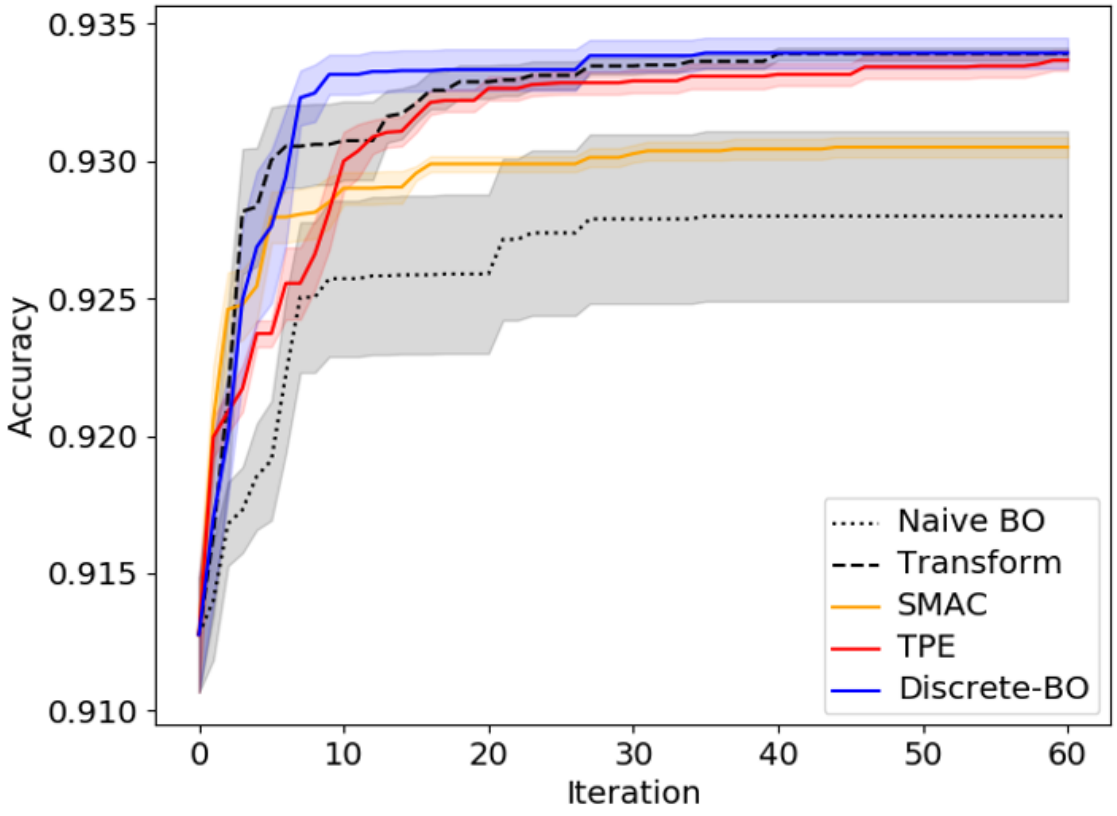


Result – Classifier models

ANN - Digits

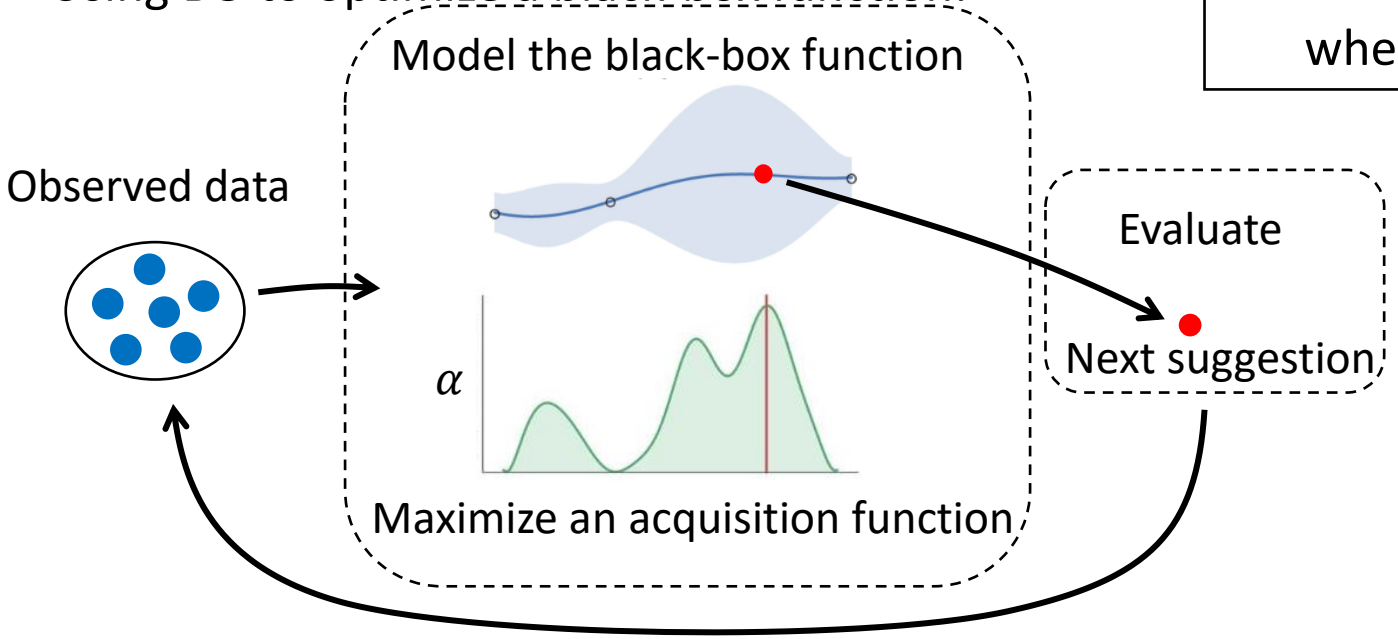


Random forest

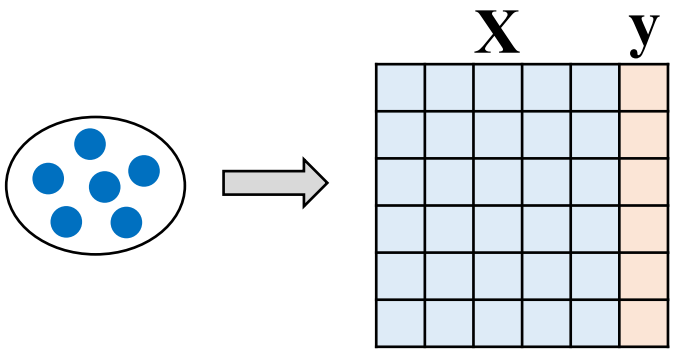


Missing Inputs

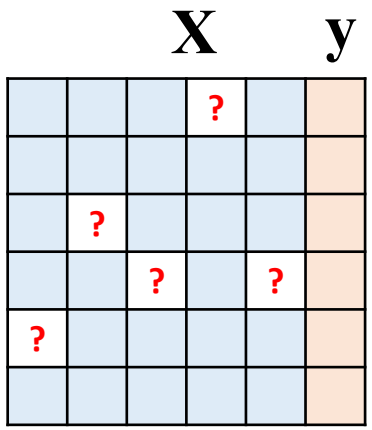
Using BO to optimize a black-box function:



- A data point $(\boldsymbol{x}, y) = \{x_1, x_2, \dots, x_d, y\}$ where $\boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^d, y \in \mathbb{R}$



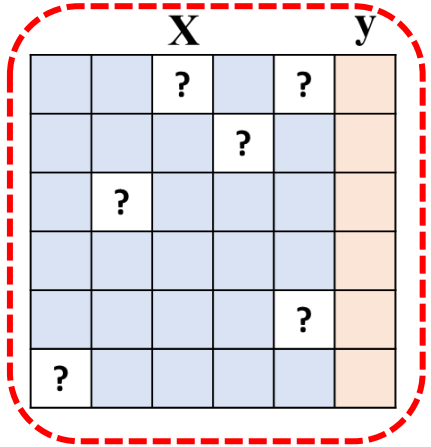
$$\alpha^{UCB} = \mu(\boldsymbol{x}) + \beta\sigma(\boldsymbol{x})$$



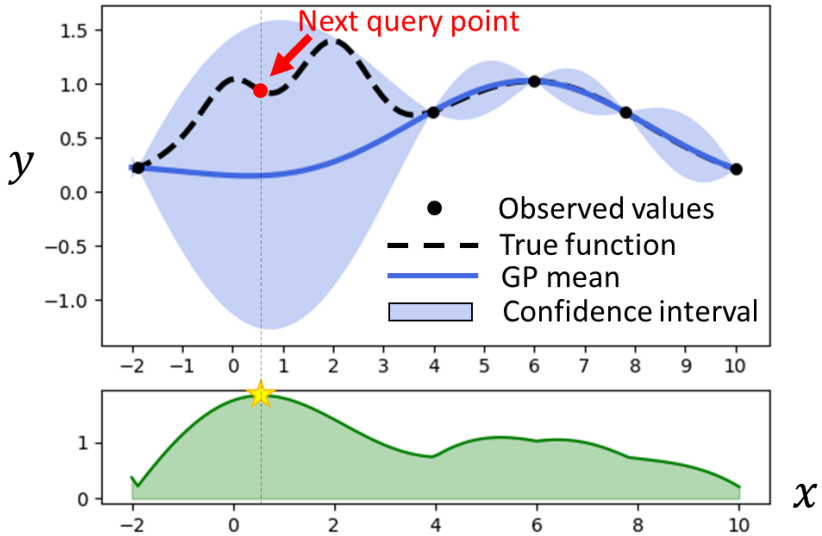
What if there are missing values in observed data?

Missing Inputs

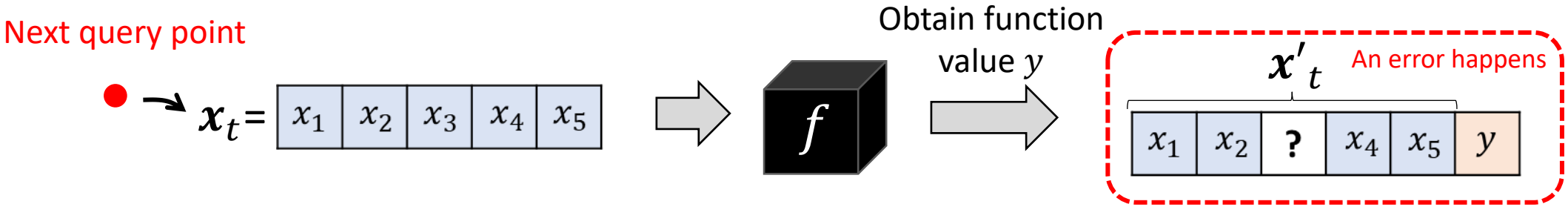
❖ **Case M1**: Missing values already in historical data



How to build the GP?



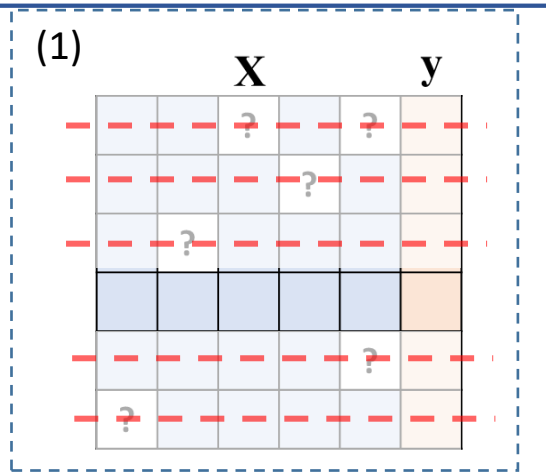
❖ **Case M2**: Missing values in new observed point $\boldsymbol{x}'$ at iteration t

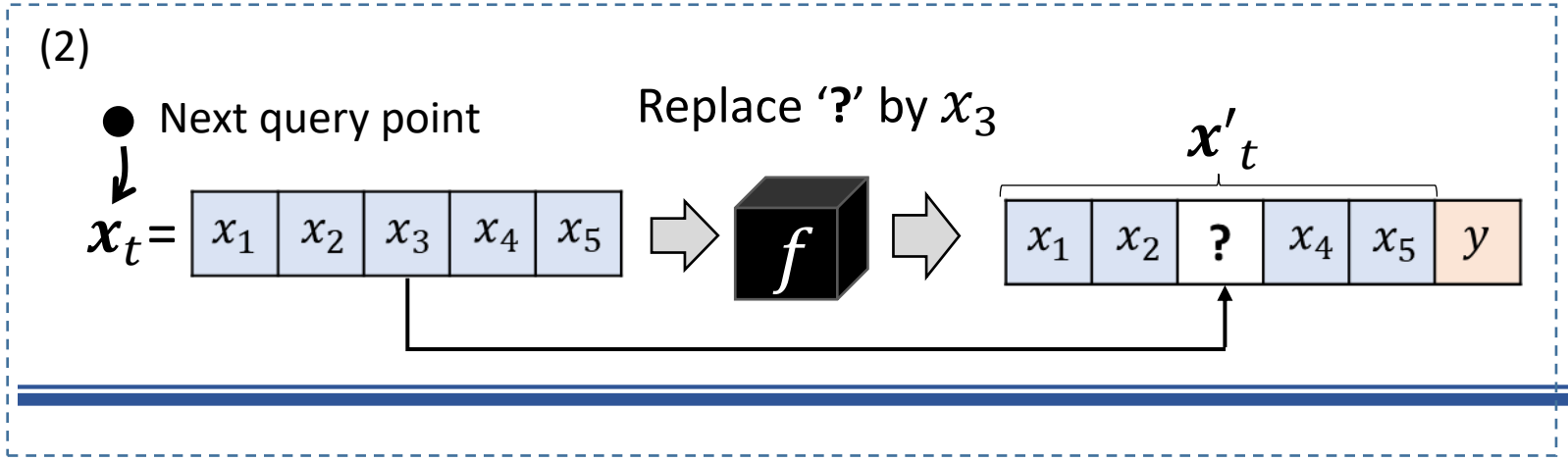


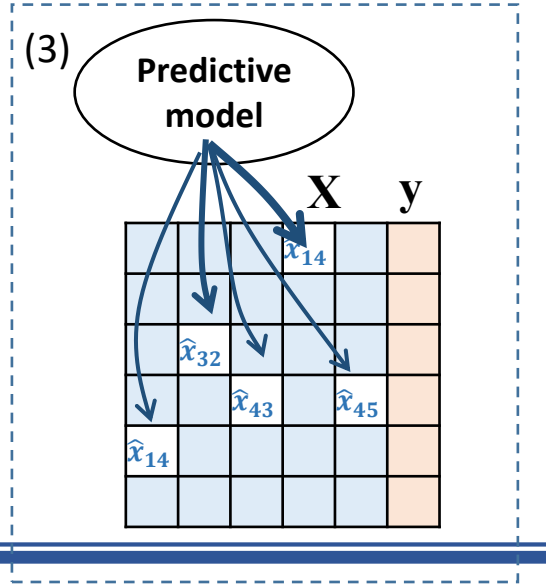
Optimization with Missing Inputs

What if we have missing values?

1. **Remove** observations which have missing values.
2. In case *M2*, we might **use the suggested point** obtained from maximizing the acquisition α (e.g., EI, UCB).
3. **Predict** missing values.

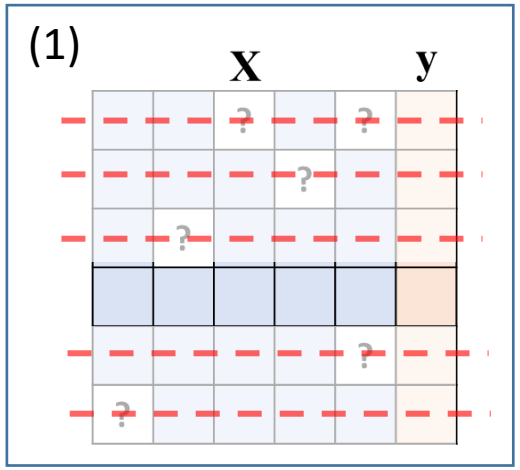






Optimization with Missing Inputs

(1) <u>Remove observations</u> (*DropBO*)

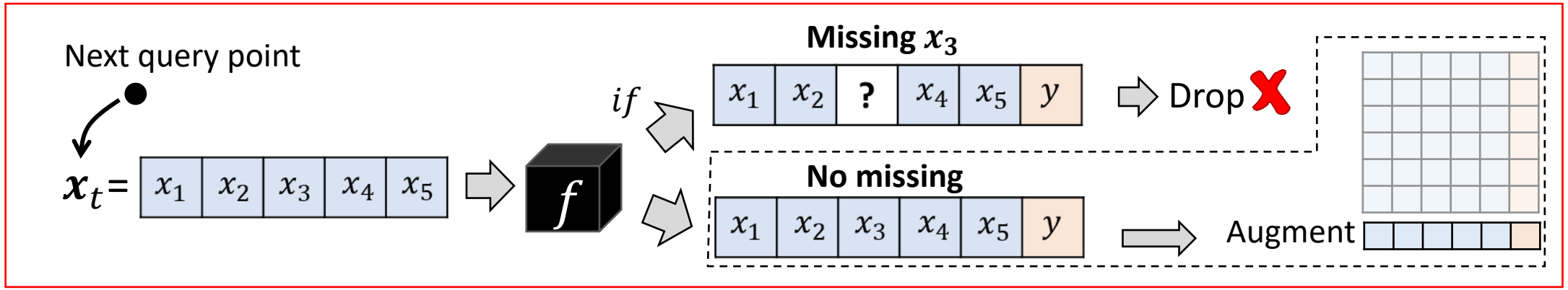


Pros:

- Very accurate data in observations set

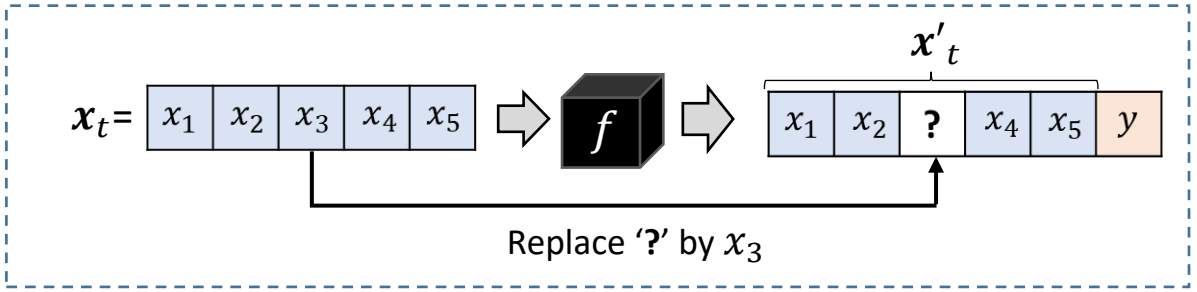
Cons:

- Have less number of observations
- Inefficient if missing rate is high



Optimization with Missing Inputs

(2) <u>Use suggested point</u> (*SuggestBO*)



Pros:

- Perform well when missing noise is *small*
- No need to train a predictive model

Cons:

- Inefficient if missing noise is *high*
- Unable to solve case *M1* (missing values in historical data)

Given $\boldsymbol{x}_t \in \mathcal{X}^d$, when evaluate $\boldsymbol{x}_t$, we obtain $y_t = f(\boldsymbol{x}'_t)$ instead of $y_t = f(\boldsymbol{x}_t)$.
SuggestBO performs well only when $\|\boldsymbol{x}_t - \boldsymbol{x}'_t\|_2$ is small.

Optimization with Missing Inputs

(3) Predict missing values

How to correctly predict missing values?

Imputations

- Mean/mode: does not consider the correlation between values.
- K-nearest neighbors: depends on available data and distance metric.
- Regression (e.g., random forest, neural network): requires many training data.

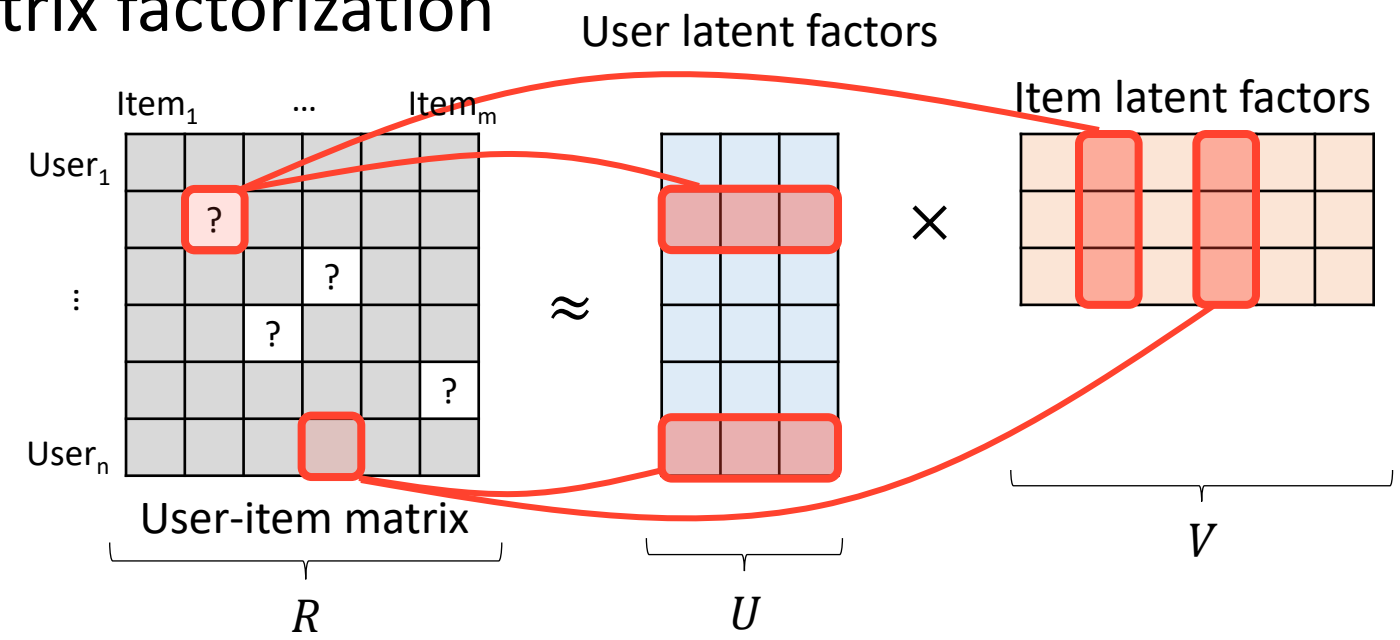
BO under uncertain inputs (BO-uGP[1]) : unable to work if actual value is unknown.

Find a way which takes into account the **correlation** between values to build the **distribution of missing data**

[1] Oliveira, R., Ott, L., Ramos, F.: Bayesian optimisation under uncertain inputs. Proceedings of the 22nd International Conference AISTAT (2019)

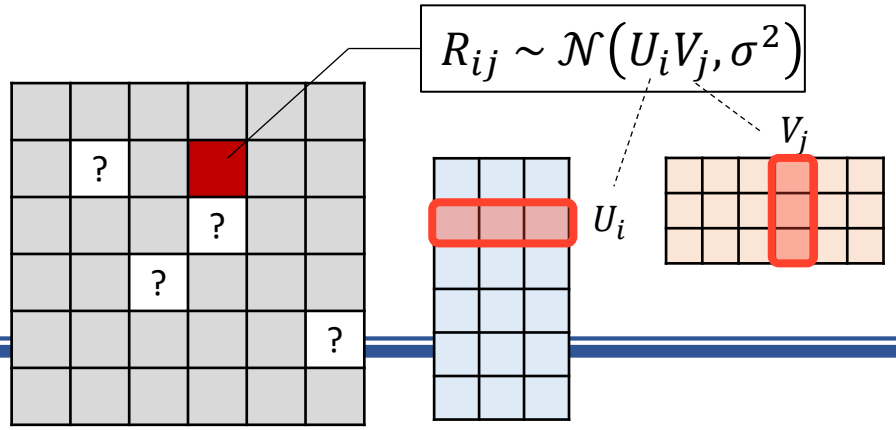
Bayesian Probabilistic Matrix Factorization (BPMF)

Matrix factorization



$R \approx UV$,
where $U \in \mathbb{R}^{N\times K}$, $V \in \mathbb{R}^{K\times M}$

BPMF



$U_i \sim \mathcal{N}(\mu_U, \Lambda_U)$
$V_j \sim \mathcal{N}(\mu_V, \Lambda_V)$

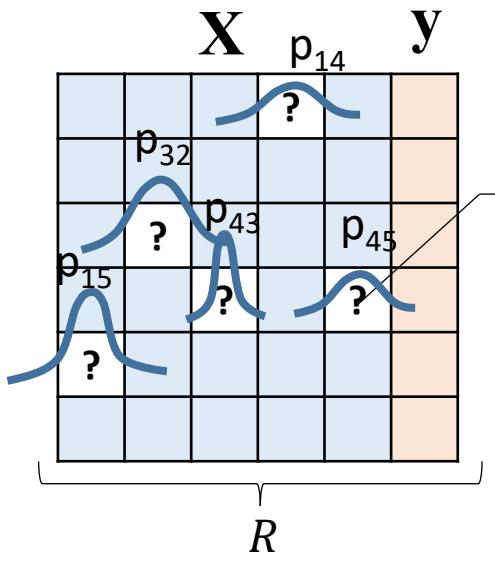
Inference using MCMC

Gaussian-Wishart priors

$$\mu_U \sim \mathcal{N}(\mu_0, \Lambda_0)$$
$$\mu_V \sim \mathcal{N}(\mu_0, \Lambda_0)$$
$$\Lambda_U \sim \mathcal{W}(\nu_0, W_0)$$
$$\Lambda_V \sim \mathcal{W}(\nu_0, W_0)$$

Our method

Consider observations set $[\boldsymbol{X}, y]$ as the matrix R, we can compute the probability distribution of each value R_{ij} in the matrix in the same way as BPMF:



$$R_{ij} \sim \hat{p}(R_{ij}) = p(R_{ij}|U_i V_j, \sigma^2) \approx p(R_{ij})$$

New values R_{ij} can be **directly sampled** from the distributions. However, it is **uncertain** to randomly sample one value as a substitution to the missing value.

Our method

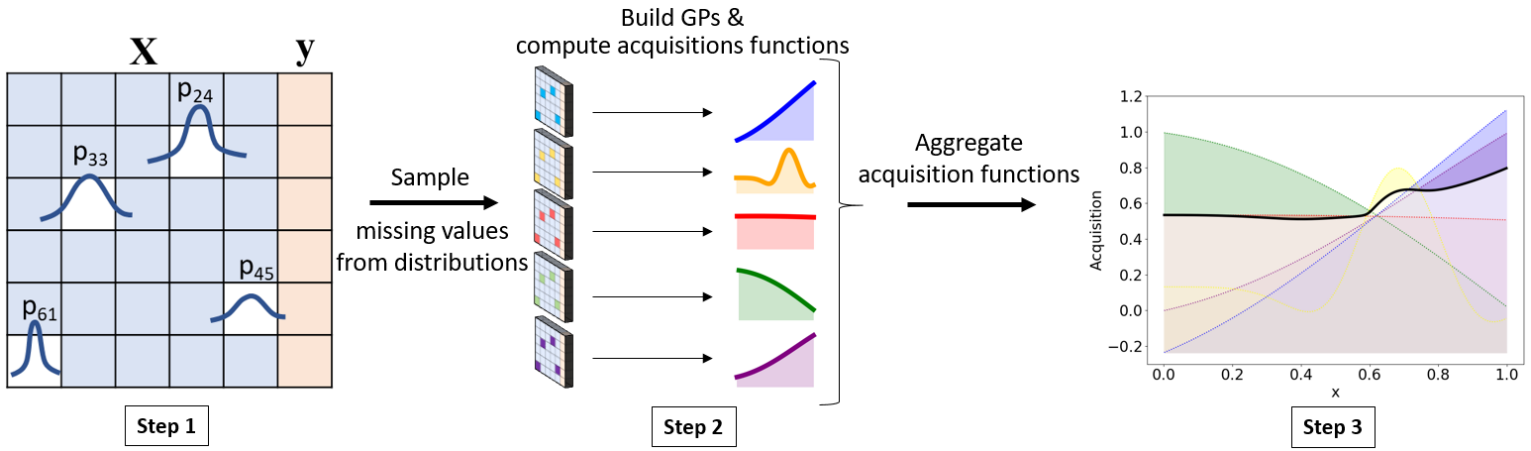
Instead of sampling **one** value for each missing location, we can sample $\boldsymbol{n}$ values.

We use the following acquisition function:

Agreement tradeoff term

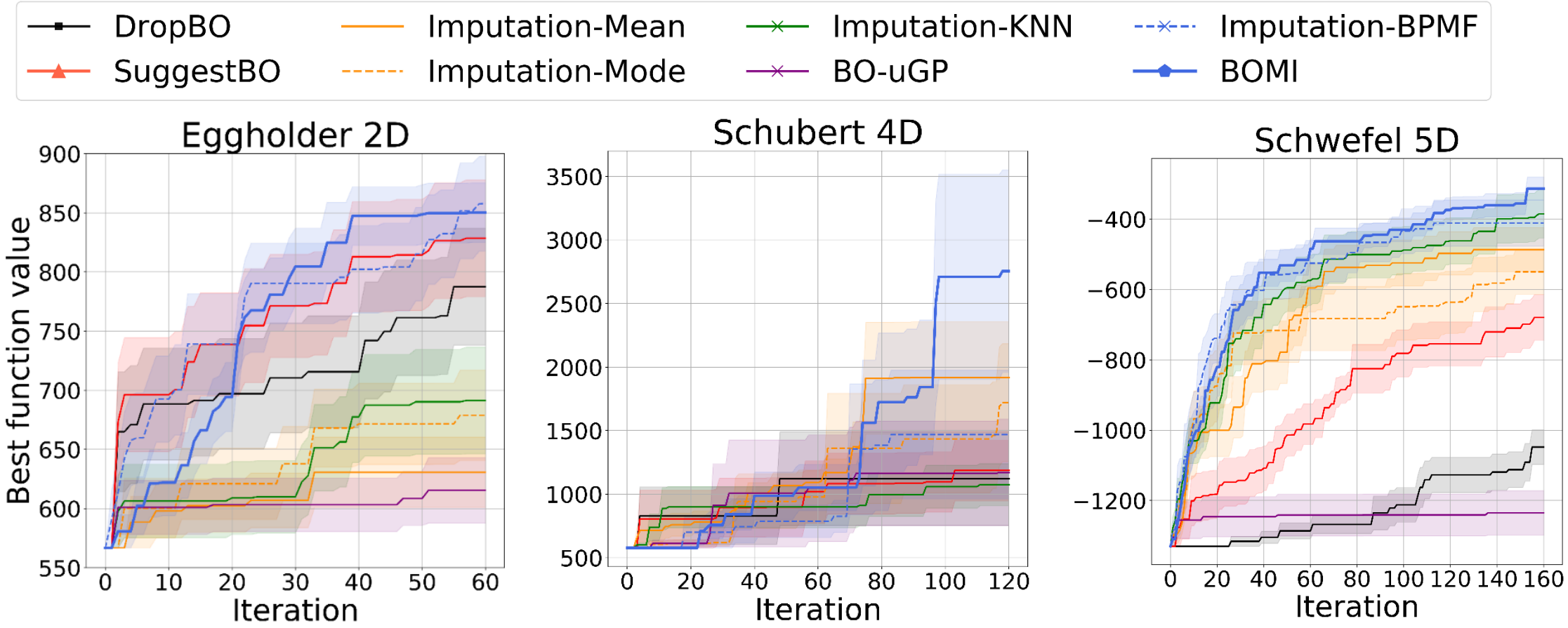
$$\alpha^{UCB-MI} = \mu_\alpha\left(\boldsymbol{\alpha}^{UCB}(x)\right) + \beta_\alpha \sigma_\alpha\left(\boldsymbol{\alpha}^{UCB}(x)\right)$$

$$= \frac{1}{Q}\sum_{q=1}^{Q}\left(\boldsymbol{\alpha}_q^{UCB}(x)\right) + \beta_\alpha \sqrt{\frac{\sum_{q=1}^{Q}\left(\boldsymbol{\alpha}_q^{UCB}(x) - \frac{1}{Q}\sum_{q=1}^{Q}\boldsymbol{\alpha}_q^{UCB}(x)\right)^2}{Q-1}}$$



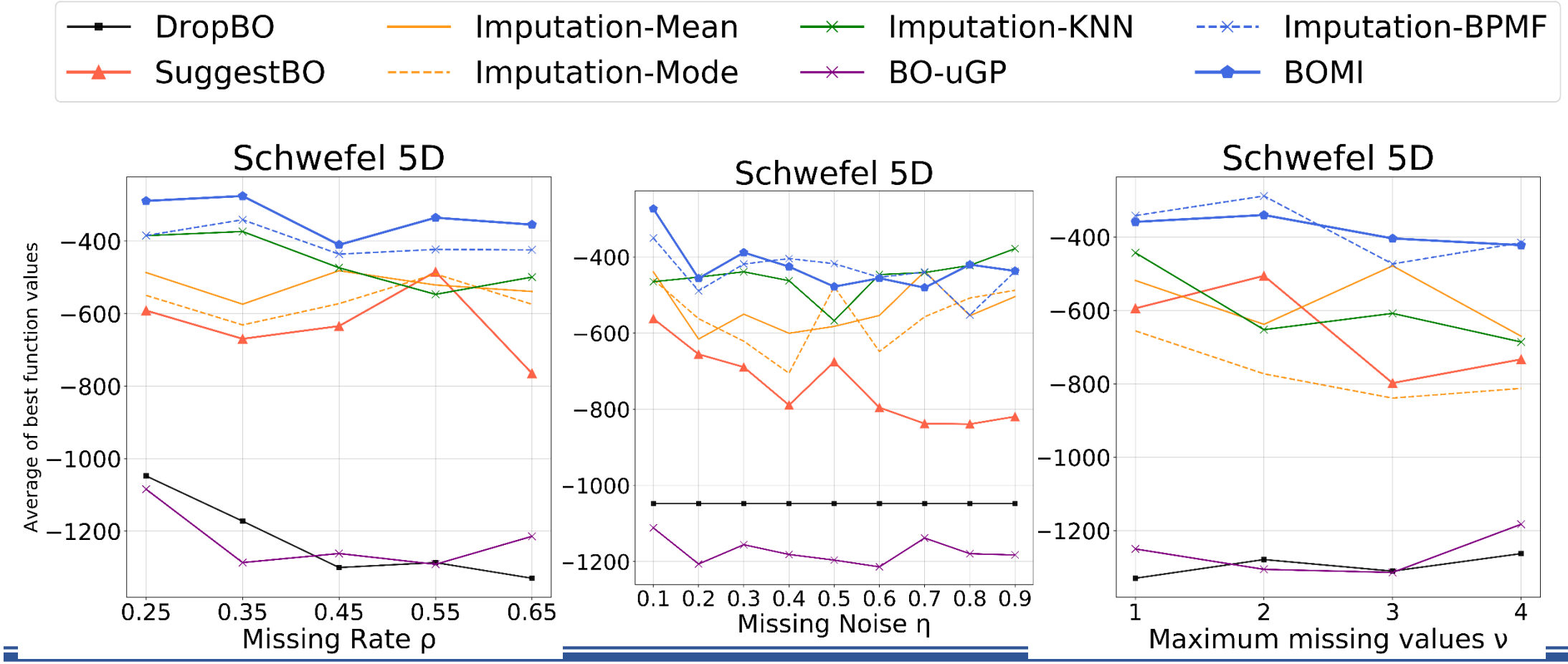
Experiments

Comparison on synthetic functions

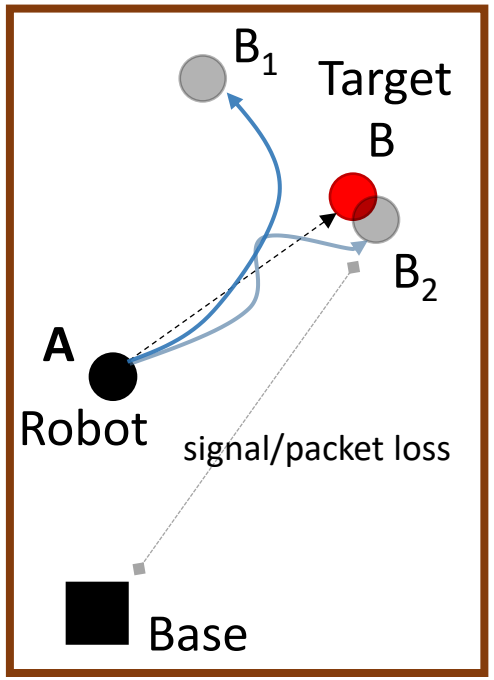


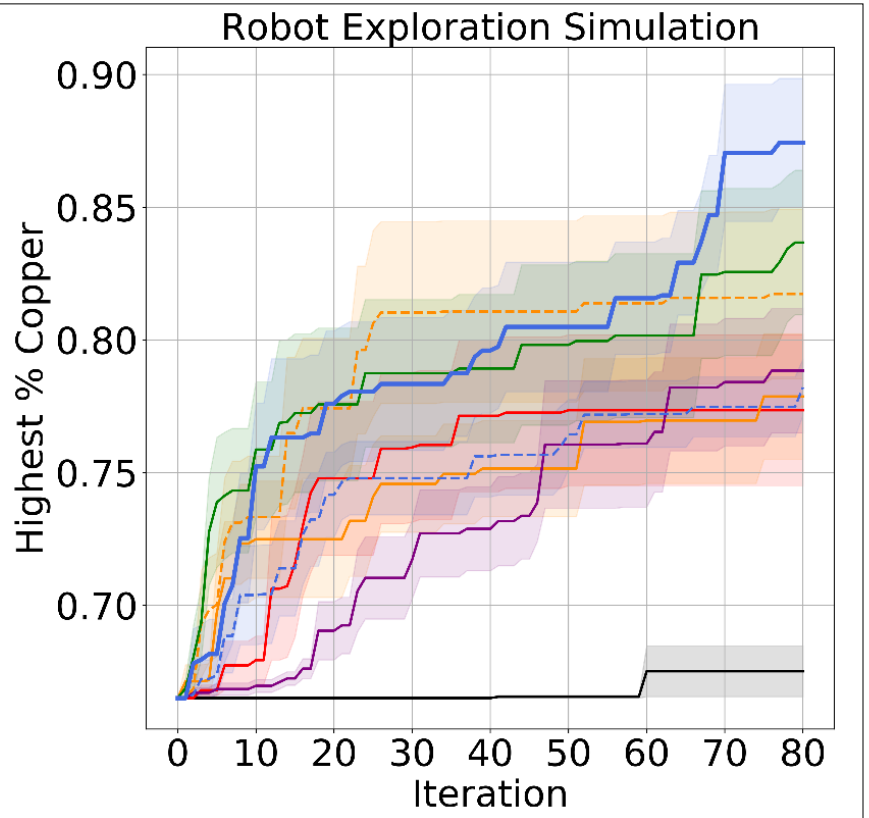
Experiments

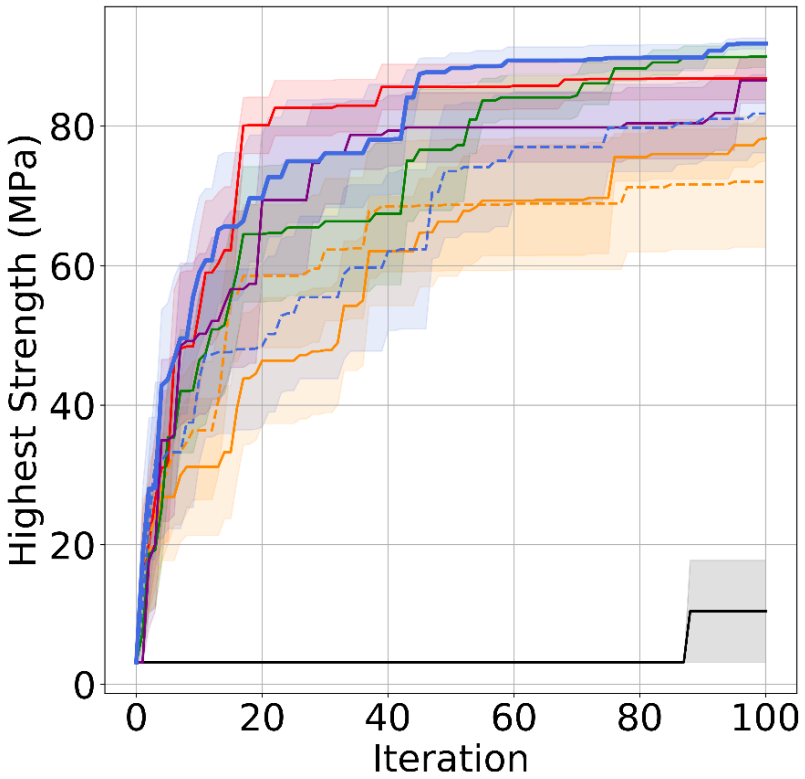
Stability



Experiments – Simulation of real problems







THANK YOU!!!