High-dimensional Bayesian optimization

A survey and benchmark

Miguel González-Duque - DTU BioSustain - 04/07/2024

About me

miguelgondu.com/about

B.Sc & M.Sc in Mathematics

Universidad Nacional de Colombia

PhD

IT University of Copenhagen, Creative AI Lab

Machine Learning programmer

University of Copenhagen, MLLS centre mlls.dk

About today

High-dimensional Bayesian **optimization**

About today

High-dimensional Bayesian optimization

High-dimensional Bayesian optimization

About today

High-dimensional Bayesian **optimization**

High-dimensional **Bayesian optimization**

High-dimensional Bayesian optimization

Goals for the day

1. Give you an overview of HDBO

2. Understand the needs of practitioners

About today

Slides are available online!

I will show this link again at the end

Estimated duration: <1h

Questions welcome

miguelgondu.com/assets/hdbo_biosustain_04_07_2024.pdf

About today

A survey and benchmark of high-dimensional Bayesian optimization of discrete sequences

Miguel González-Duque*
University of Copenhagen

Richard Michael
University of Copenhagen

Simon Bartels
Paul Sabatier University Tolouse

Yevgen Zainchkovskyy
Technical University of Denmark

Søren Hauberg
Technical University of Denmark

Wouter Boomsma
University of Copenhagen

https://arxiv.org/abs/2406.04739

About today

miguelgondu.com/assets/hdbo_timeline.pdf

A survey and benchmark of high-dimensional Bayesian optimization of discrete sequences

Miguel González-Duque* University of Copenhagen

Richard Michael University of Copenhagen

Simon Bartels Paul Sabatier University Tolouse

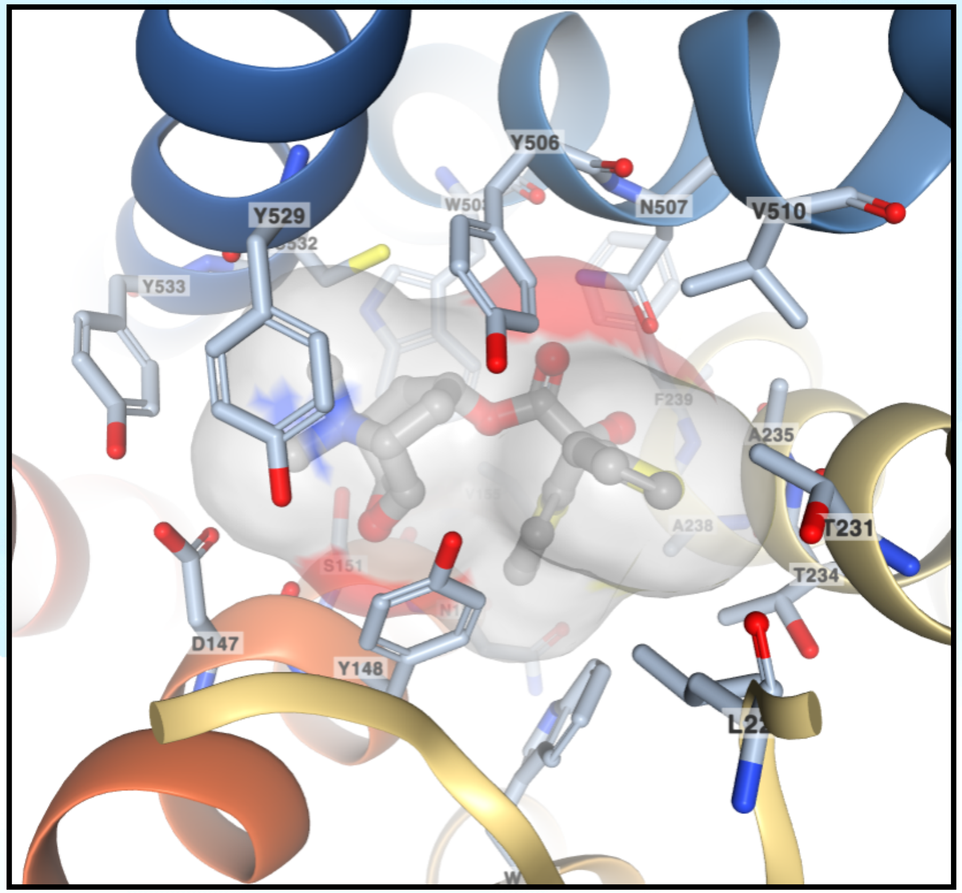
Yevgen Zainchkovskyy Technical University of Denmark

Søren Hauberg Technical University of Denmark

Wouter Boomsma University of Copenhagen

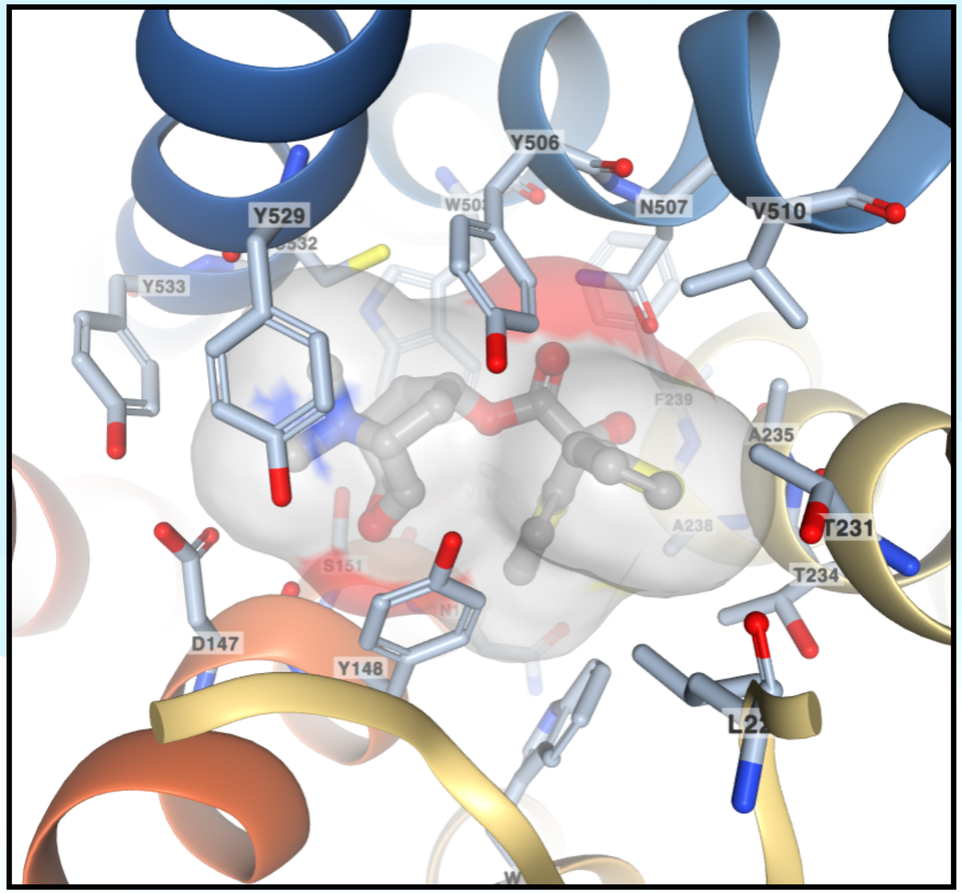
https://arxiv.org/abs/2406.04739

High-dimensional Bayesian optimization

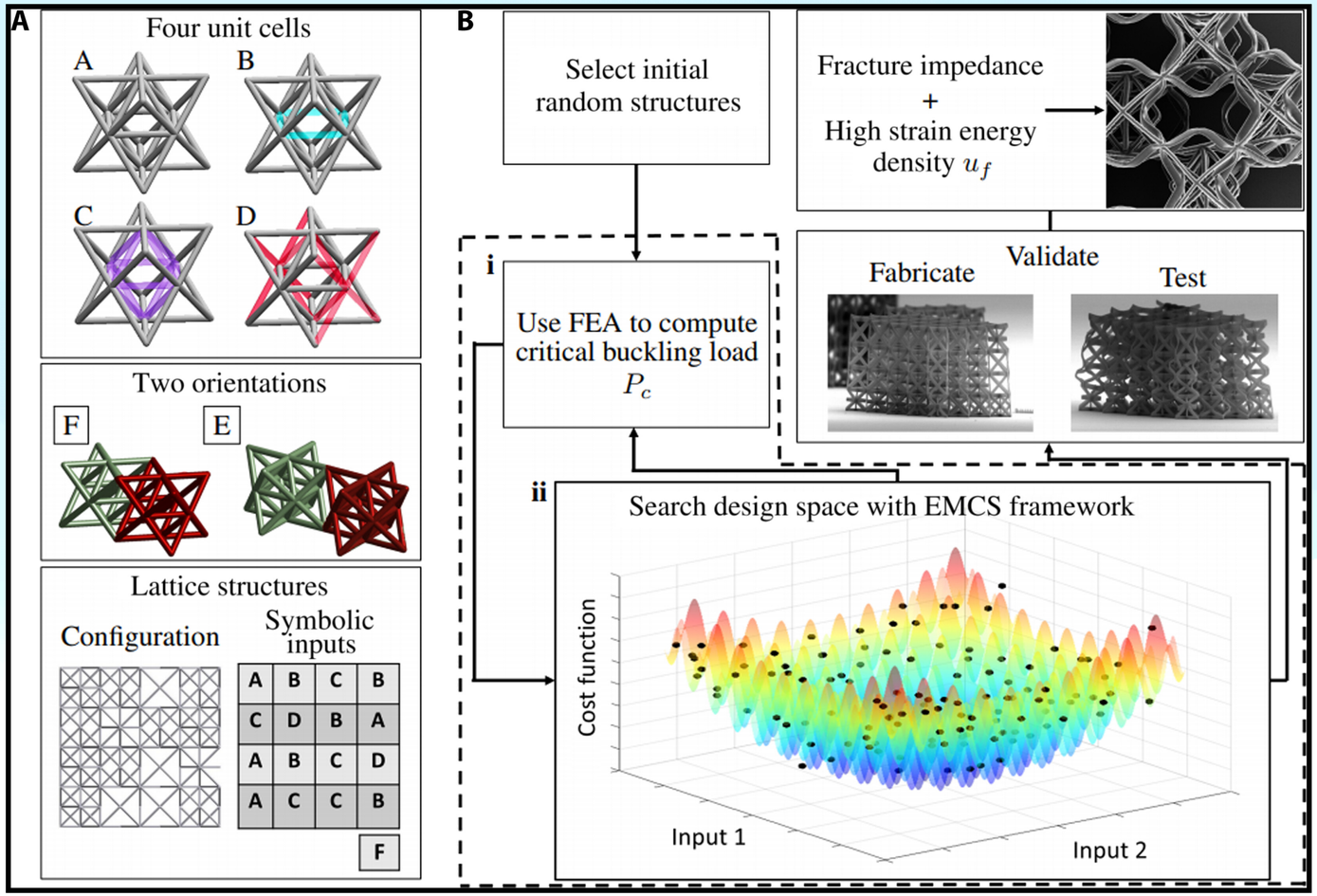


rcsb.org/news/feature/57e30fd490f5613003407f09

Optimizing molecules, proteins and materials

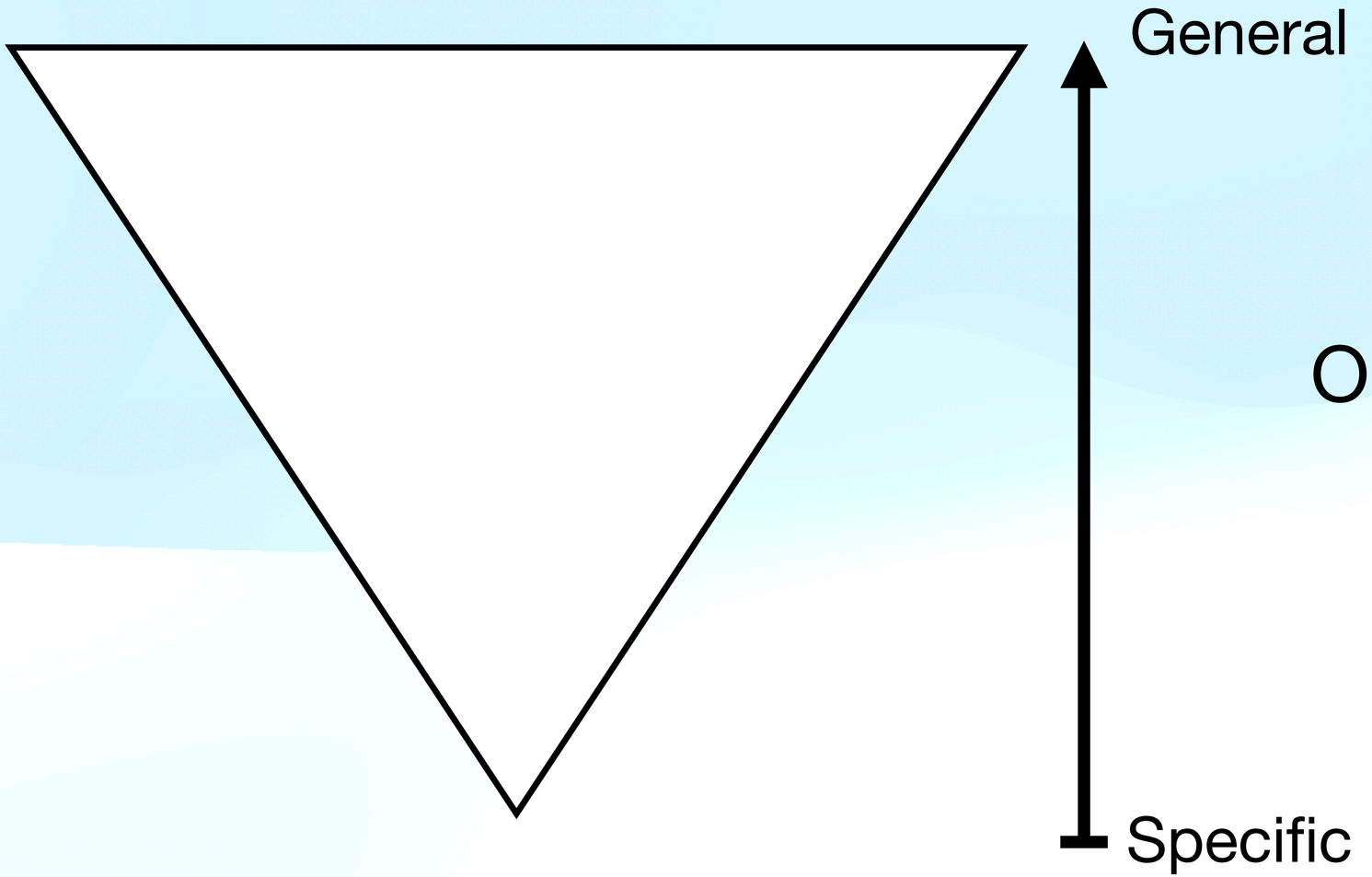


rcsb.org/news/feature/57e30fd490f5613003407f09



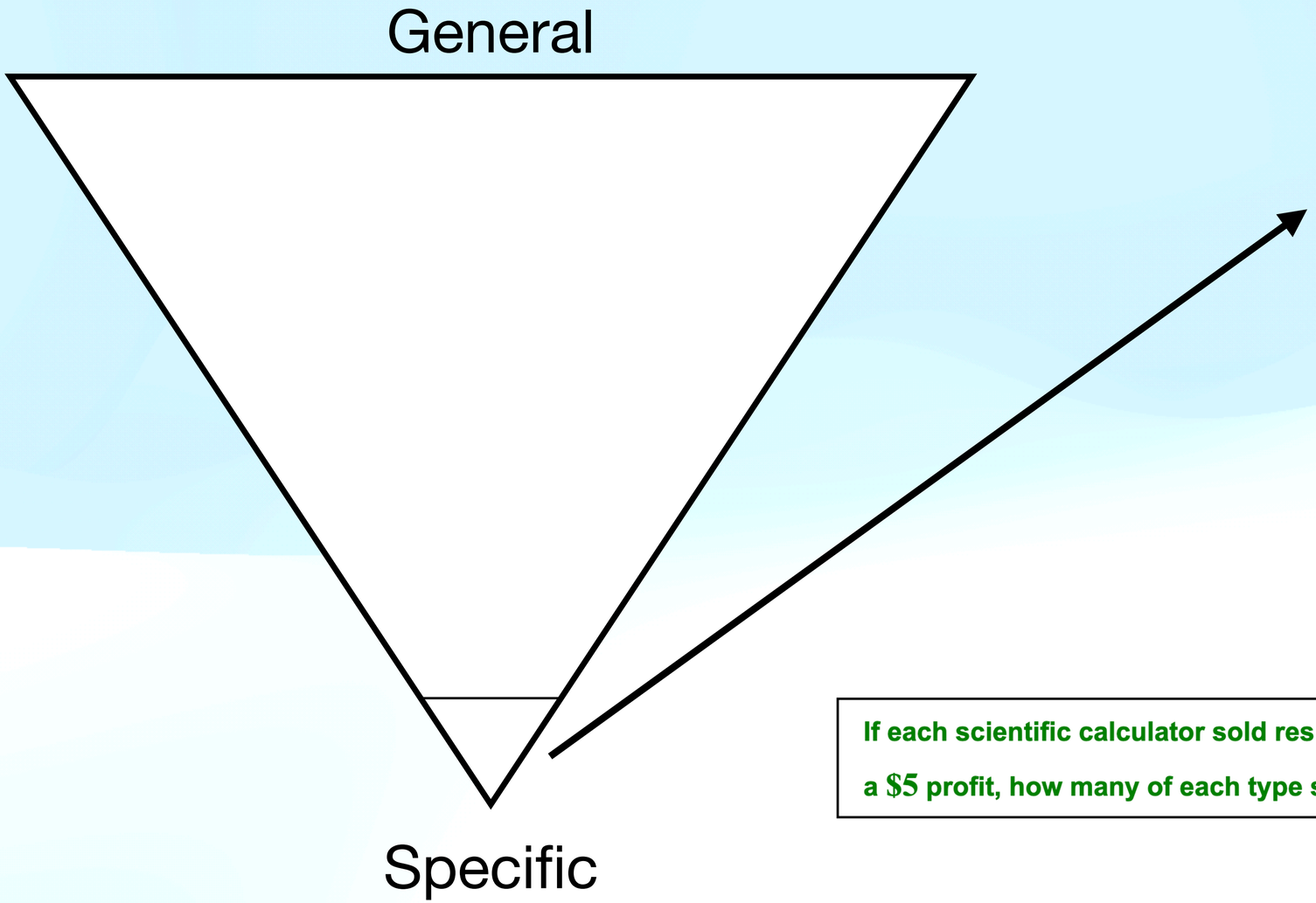
science.org/doi/10.1126/sciadv.abk2218

Optimizing molecules, proteins and materials



Optimization problems
live in a triangle*

Optimization: an introduction

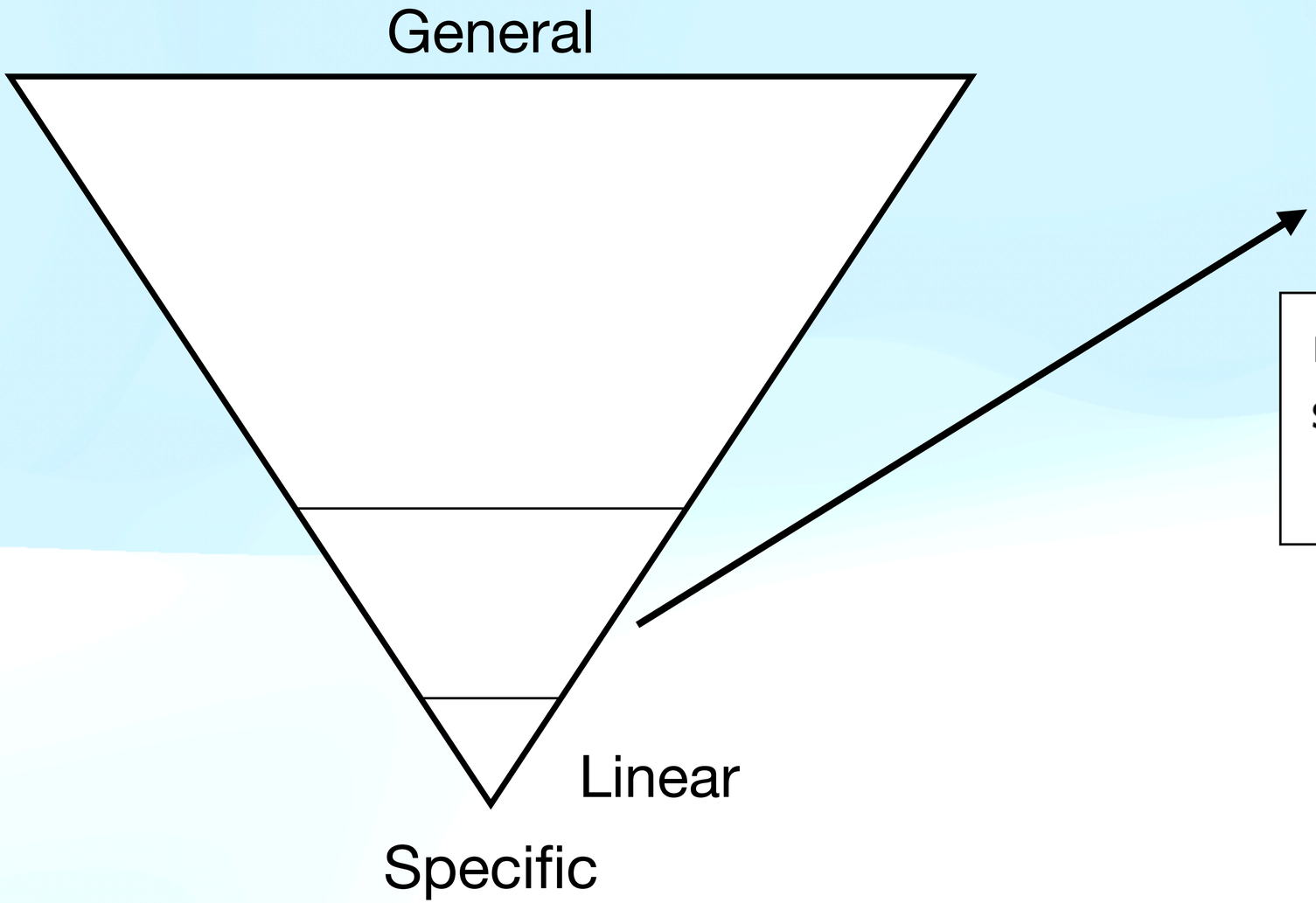


Linear problems

A calculator company produces a scientific calculator and a graphing calculator. Long-term projections indicate an expected demand of at least 100 scientific and 80 graphing calculators each day. Because of limitations on production capacity, no more than 200 scientific and 170 graphing calculators can be made daily. To satisfy a shipping contract, a total of at least 200 calculators much be shipped each day.

If each scientific calculator sold results in a $\$2$ loss, but each graphing calculator produces a $\$5$ profit, how many of each type should be made daily to maximize net profits?

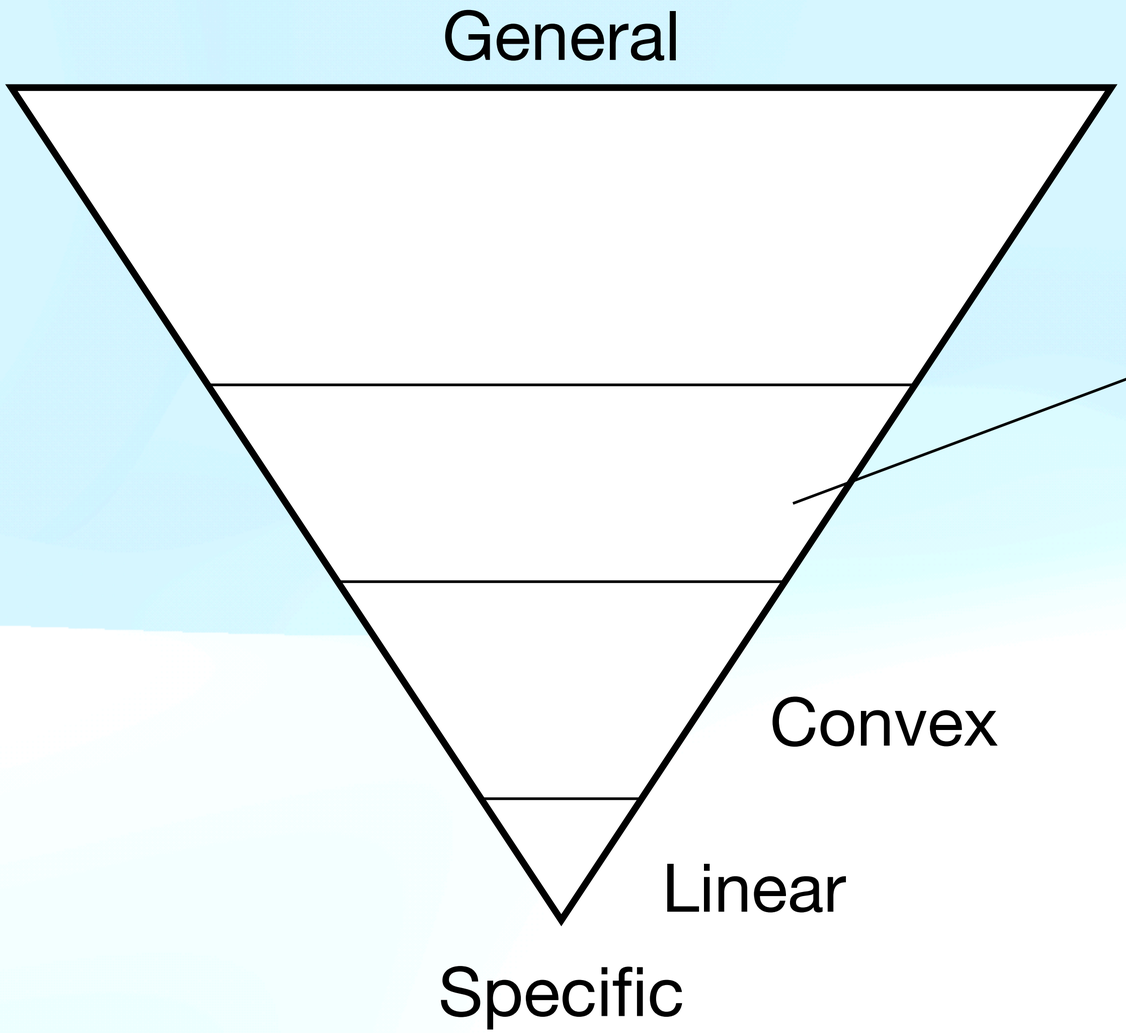
Optimization: an introduction



Convex problems

$$\begin{array}{ll} \text{minimize} & x_1^2 + x_2^2 \\ \text{subject to} & x_1 \leq 0 \\ & x_1 + x_2 = 0 \end{array}$$

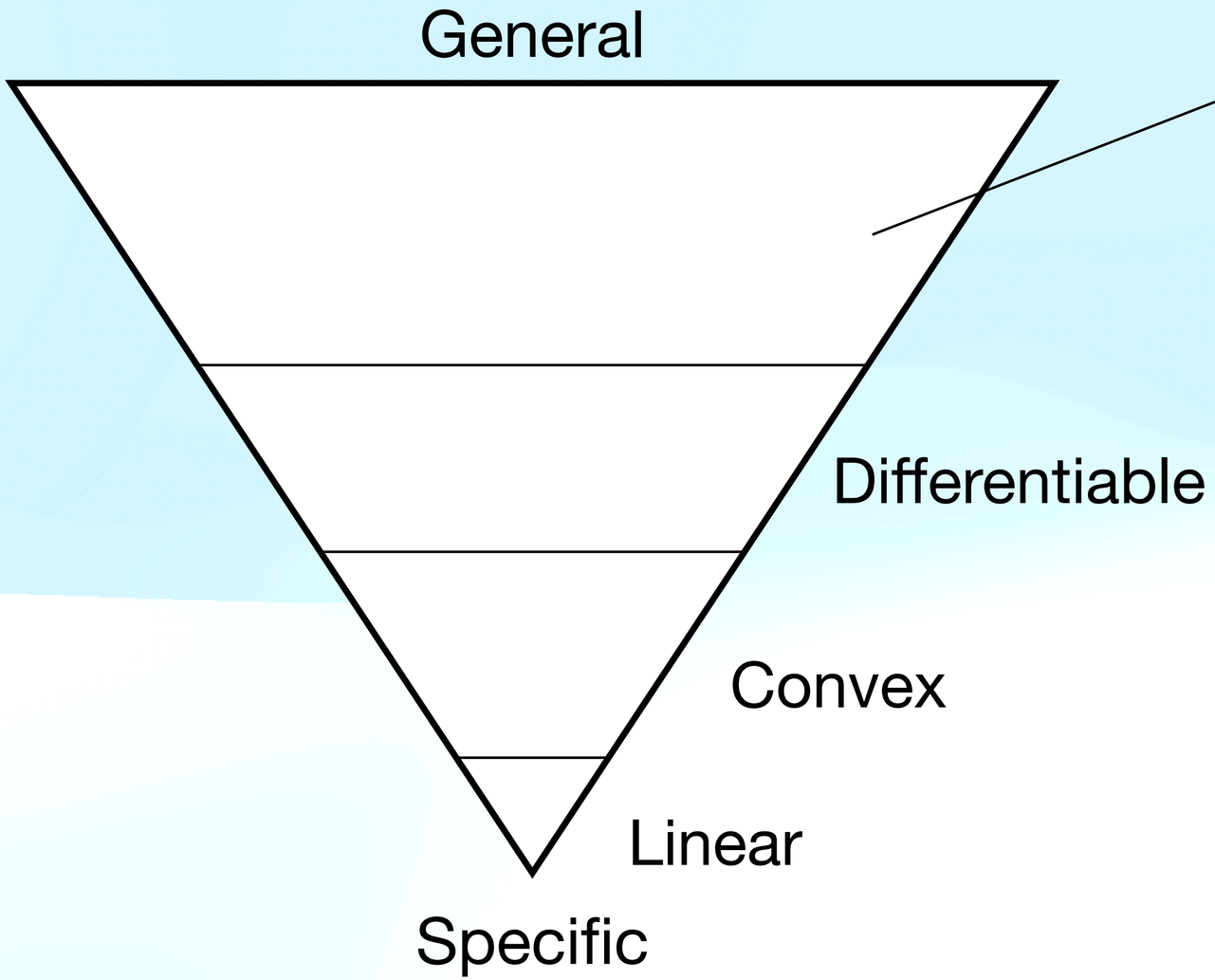
Optimization: an introduction



Differentiable problems

Train a Feed-Forward Neural Network using gradient descent.

Optimization: an introduction



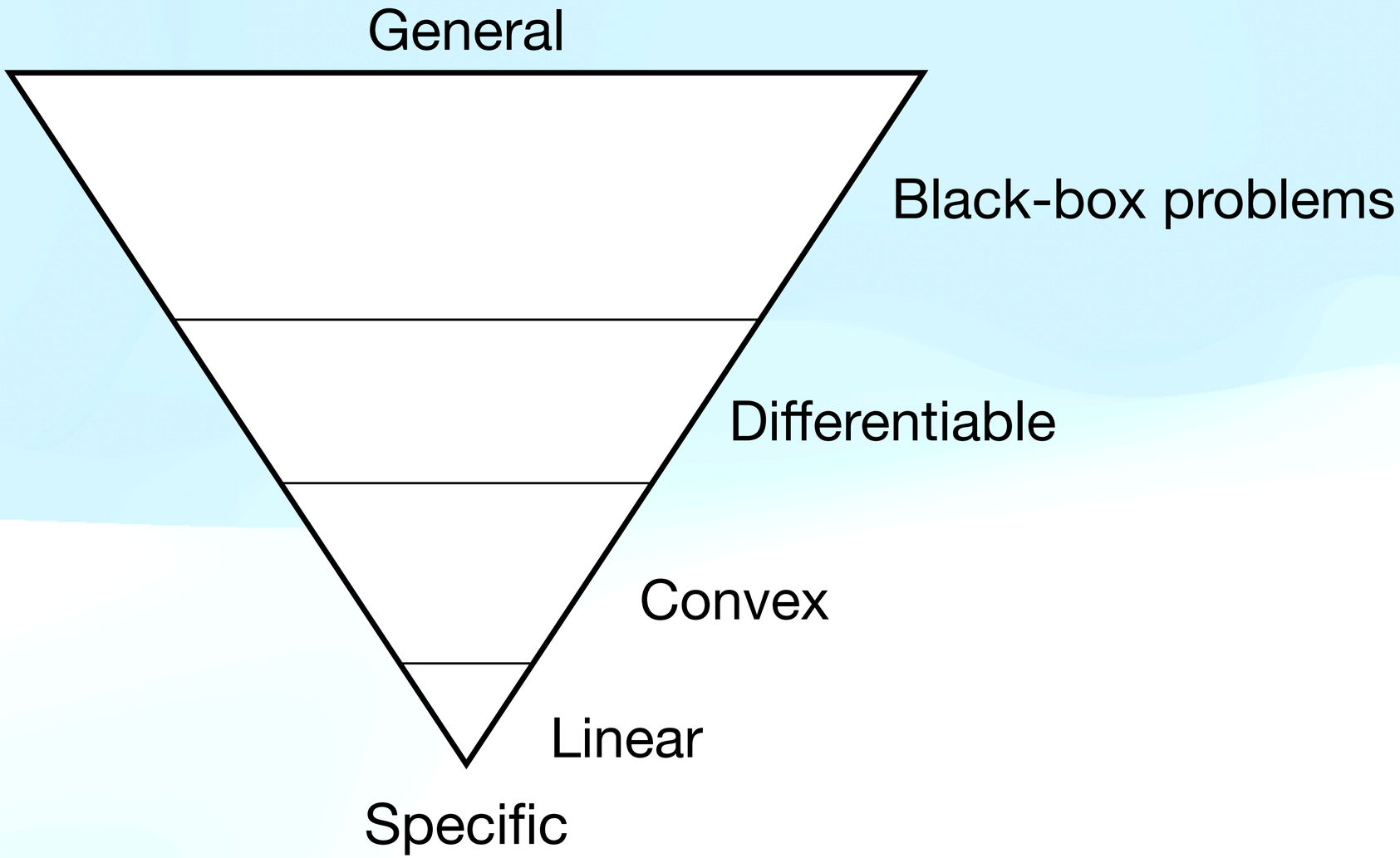
Black-box problems

Whenever the function you're optimizing doesn't have a closed form!

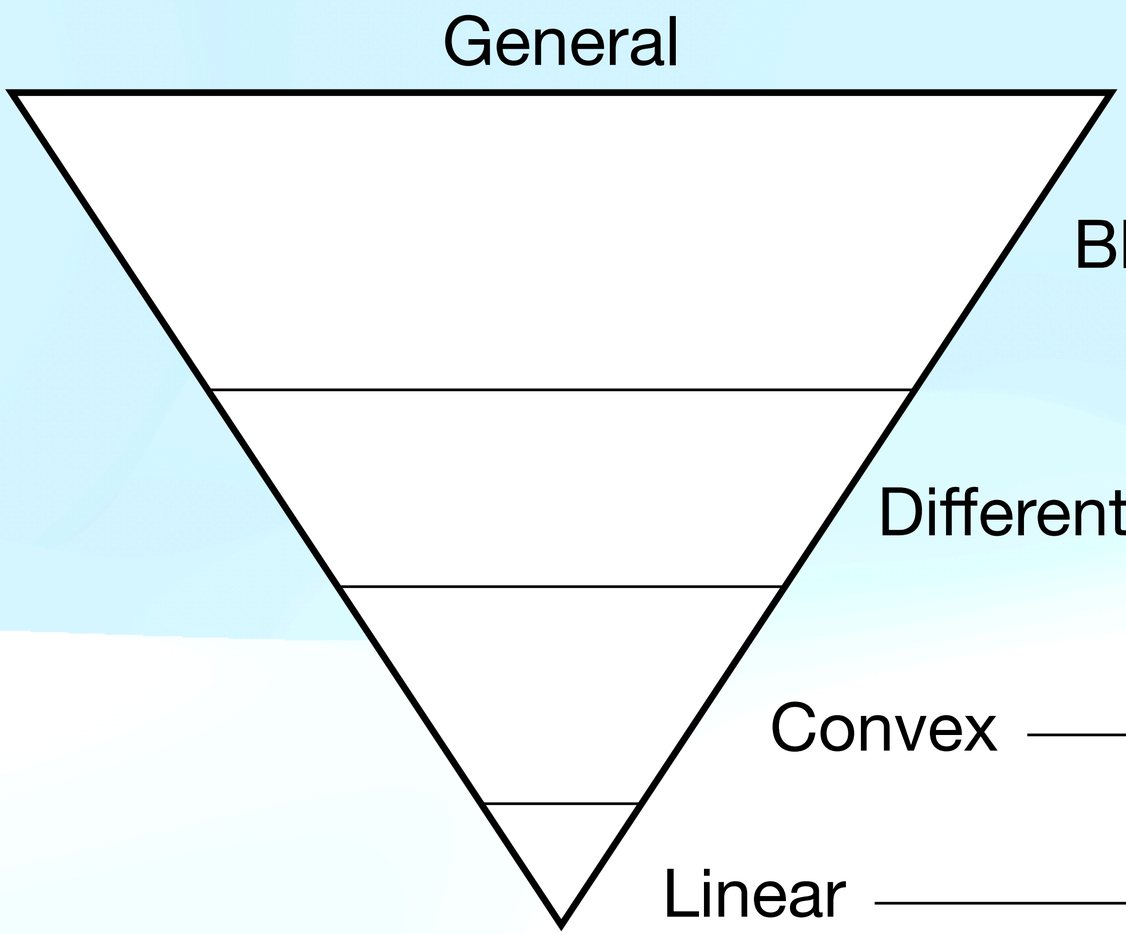
Find a molecule that binds well with a certain protein

Find the mutations of a wild type that maximize a certain function

Optimization: an introduction



Optimization: an introduction



Solved using e.g.

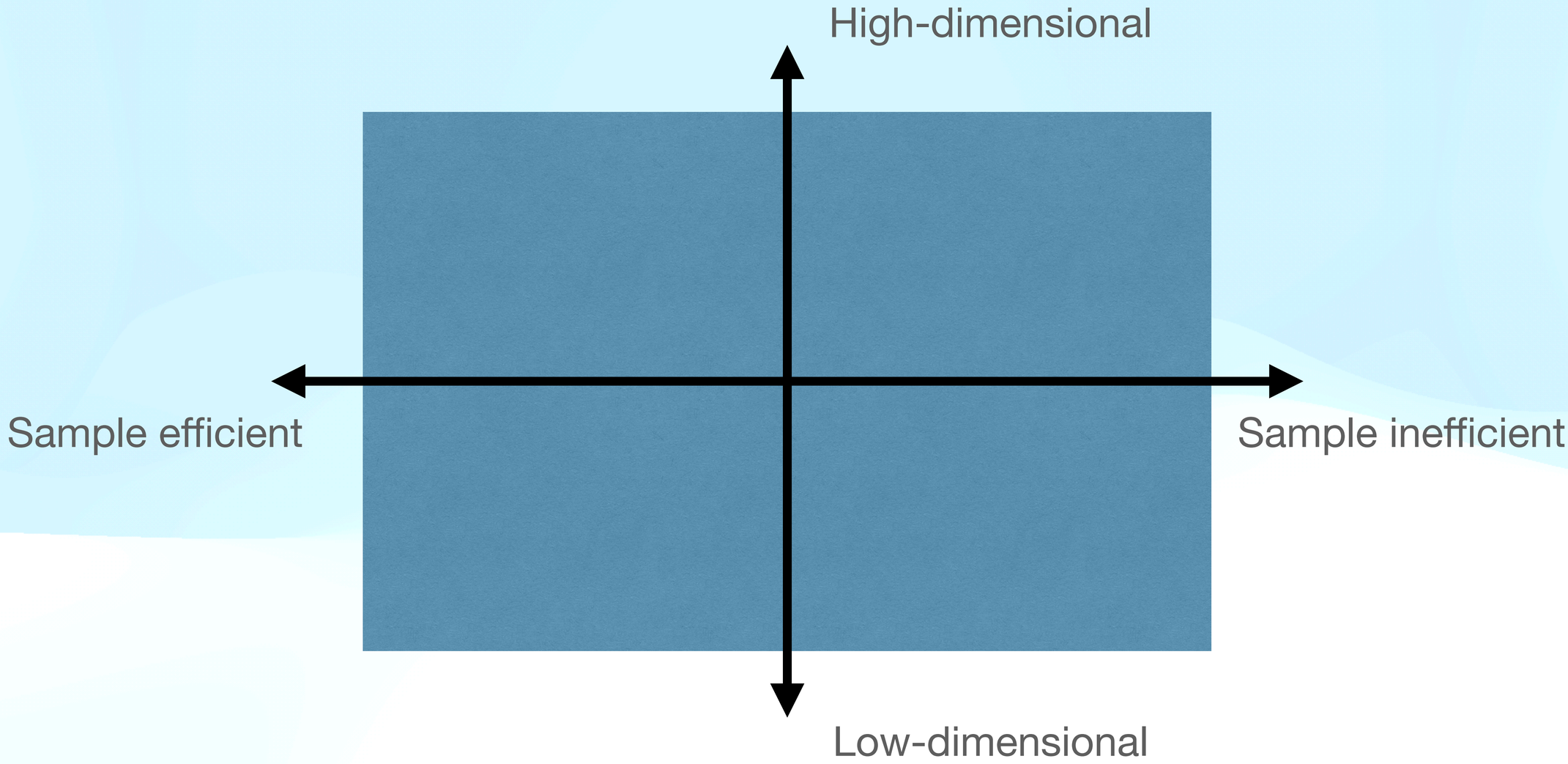
Black-box problems → Evolutionary algorithms, BO

Differentiable → Gradient descent, Newton methods

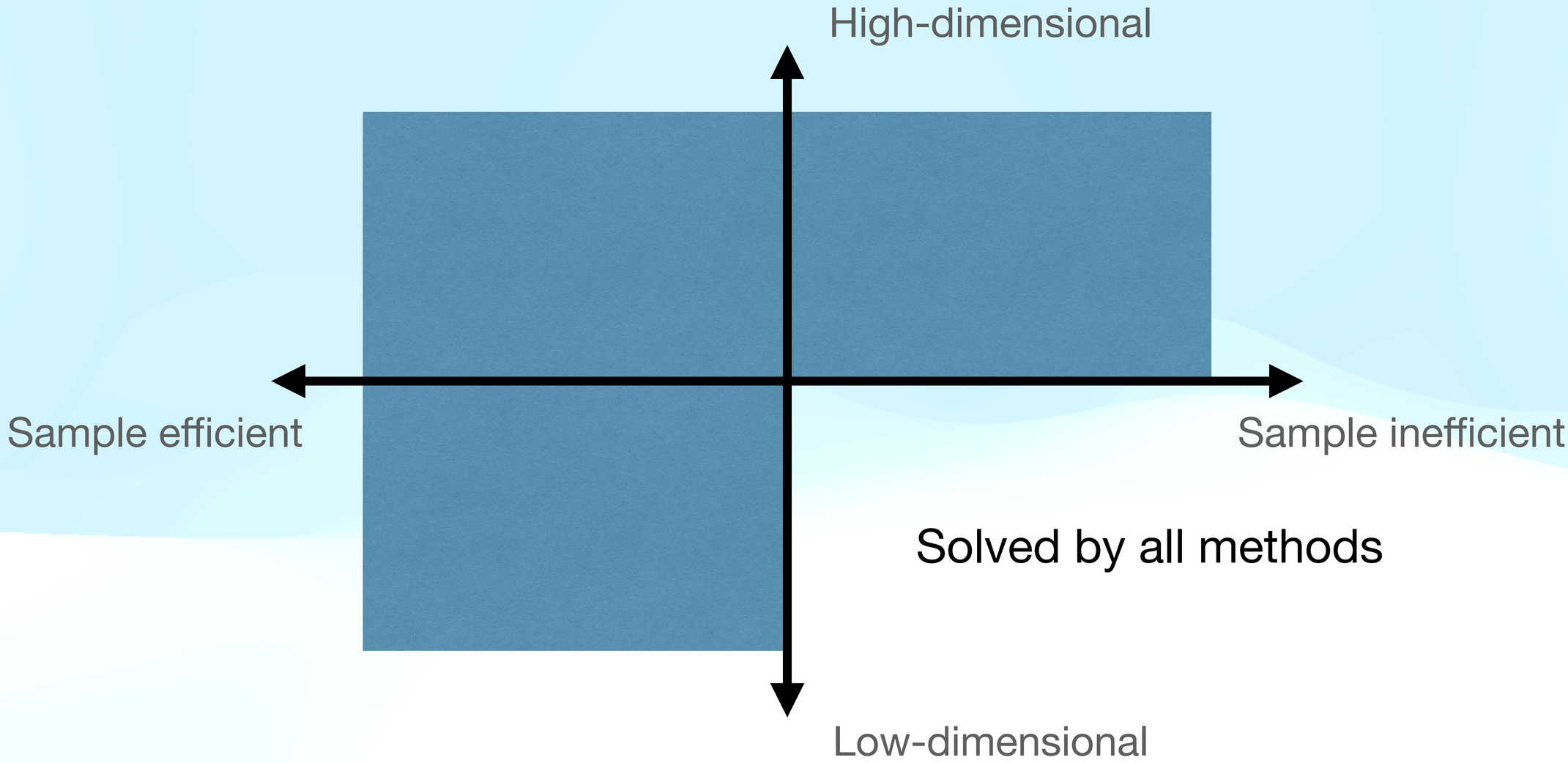
Convex → Convex opt.

Linear → Simplex algorithm

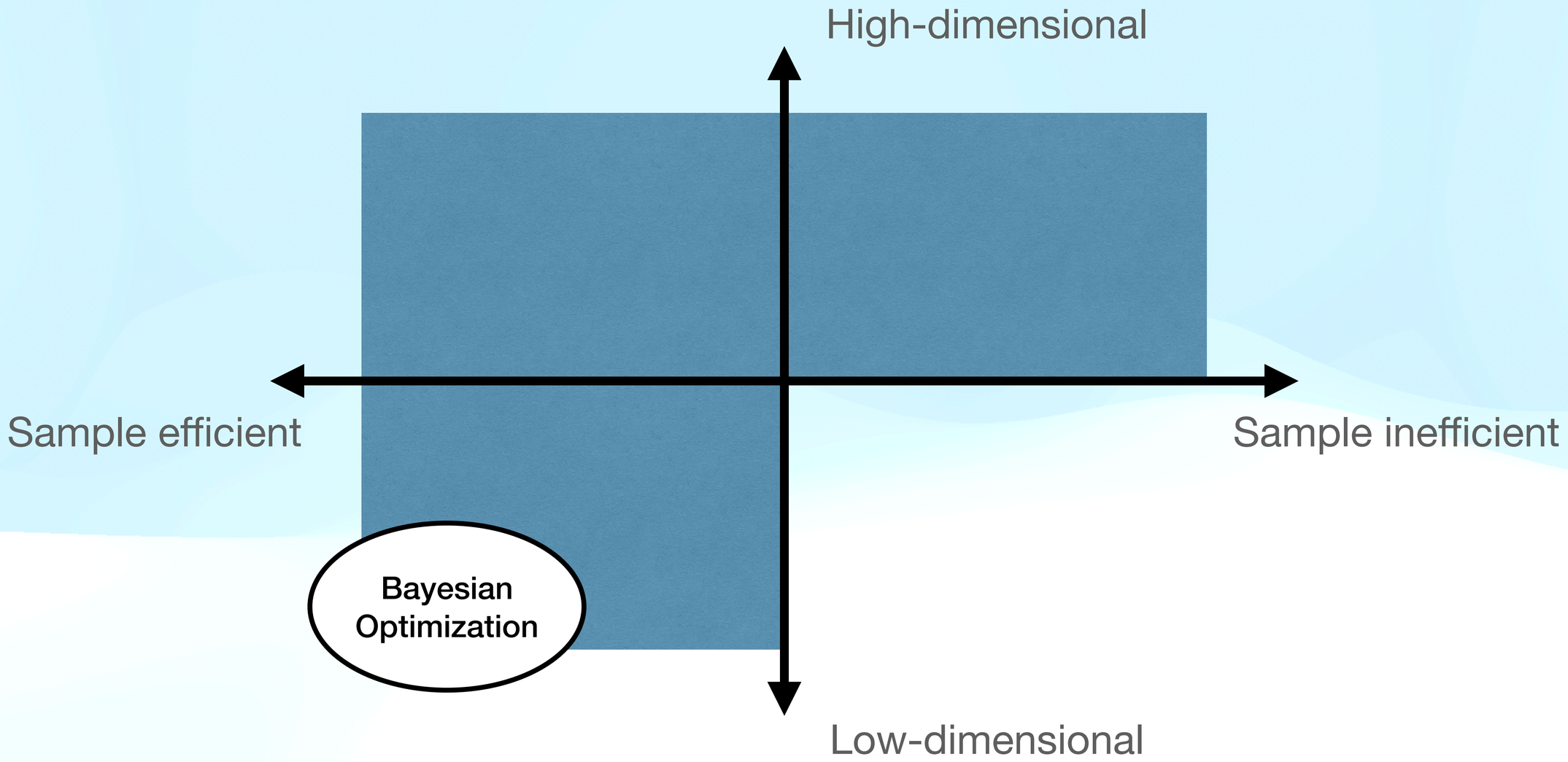
Optimization: an introduction



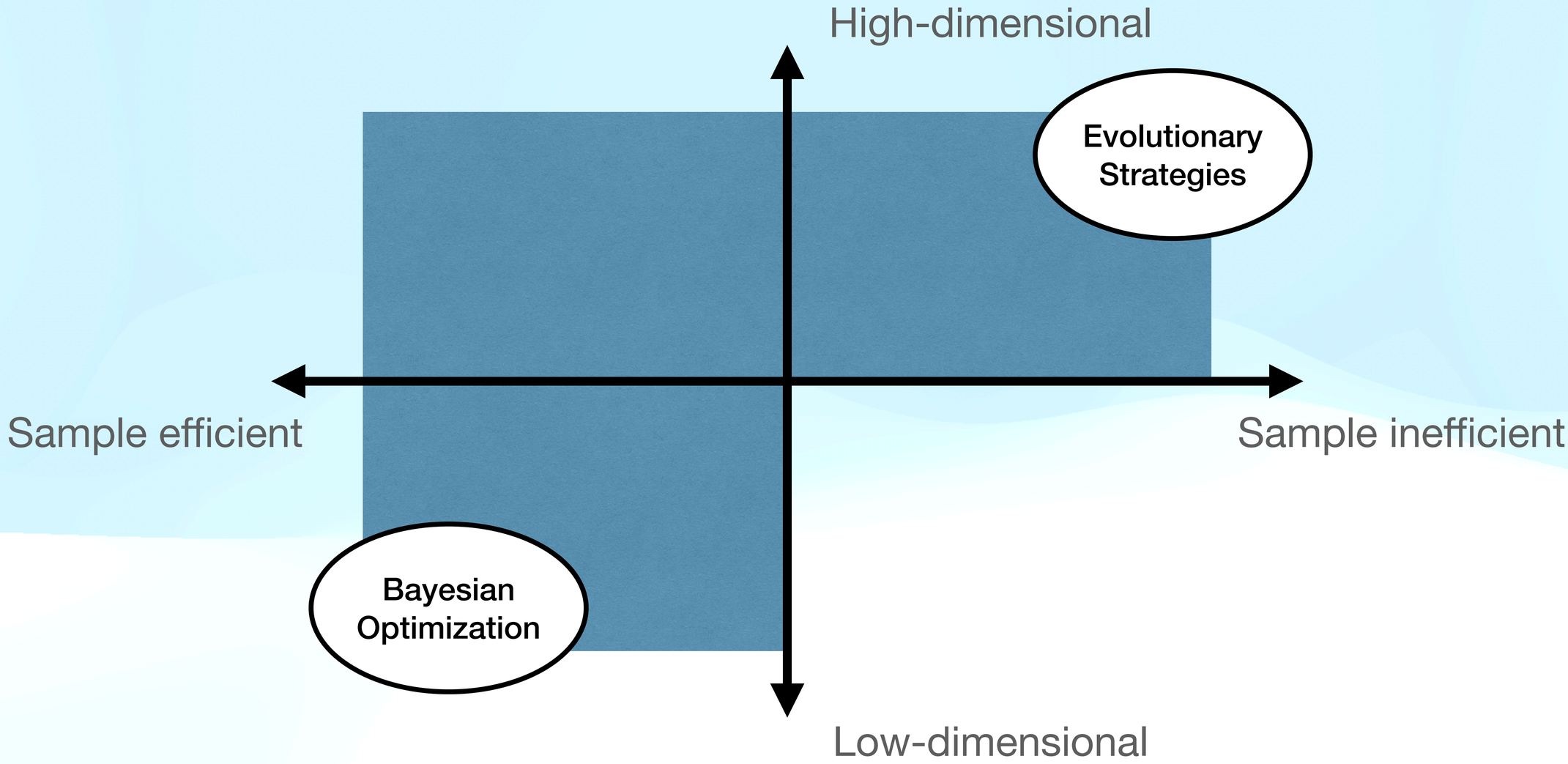
Black-box optimization algorithms



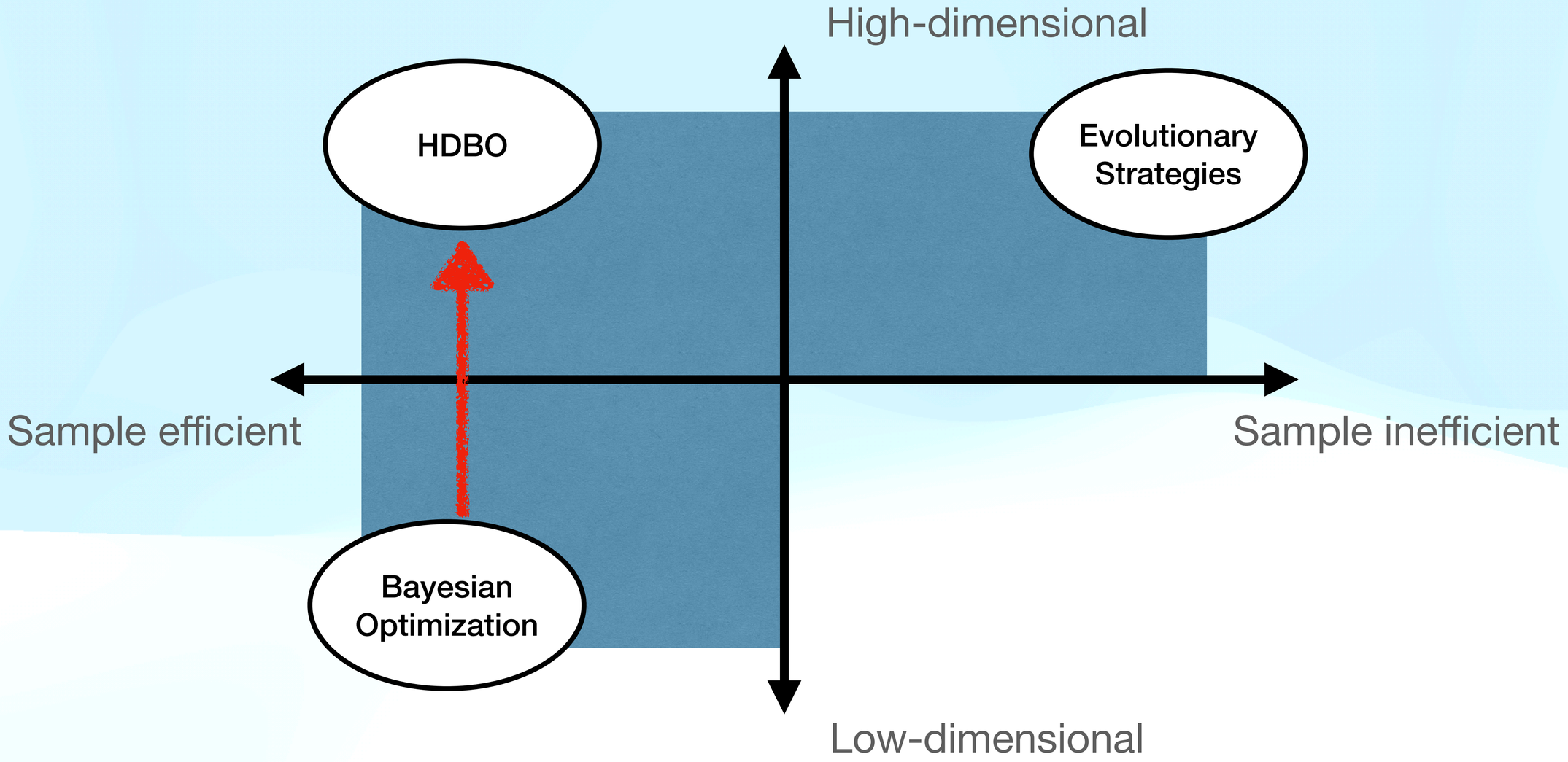
Black-box optimization algorithms



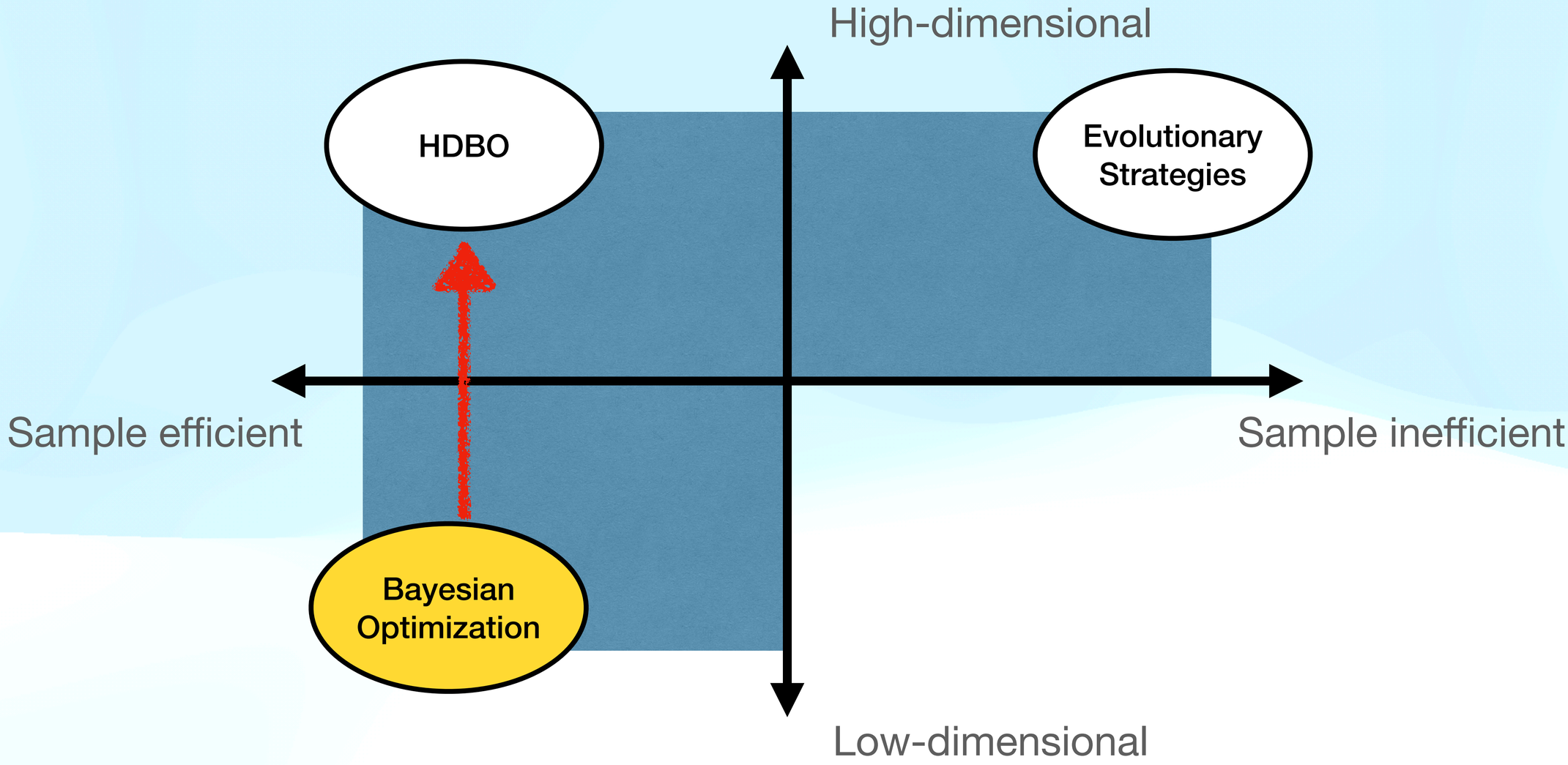
Black-box optimization algorithms



Black-box optimization algorithms



Black-box optimization algorithms



Black-box optimization algorithms

High-dimensional

HDBO

Evolutionary Strategies

Sample efficient

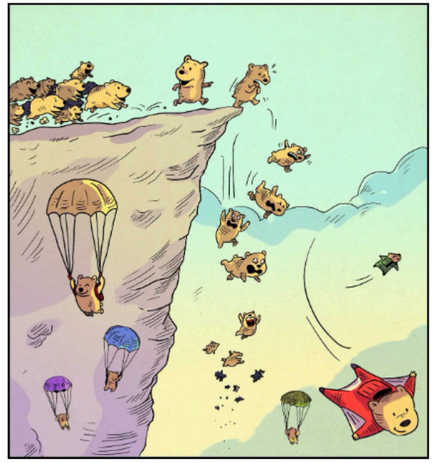
Bayesian Optimization

If you are curious…

Low-dimensiona

A Visual Guide to Evolution Strategies

OCTOBER 29, 2017



Survival of the fittest.

blog.otoro.net/2017/10/29/visual-evolution-strategies

Black-box optimization algorithms

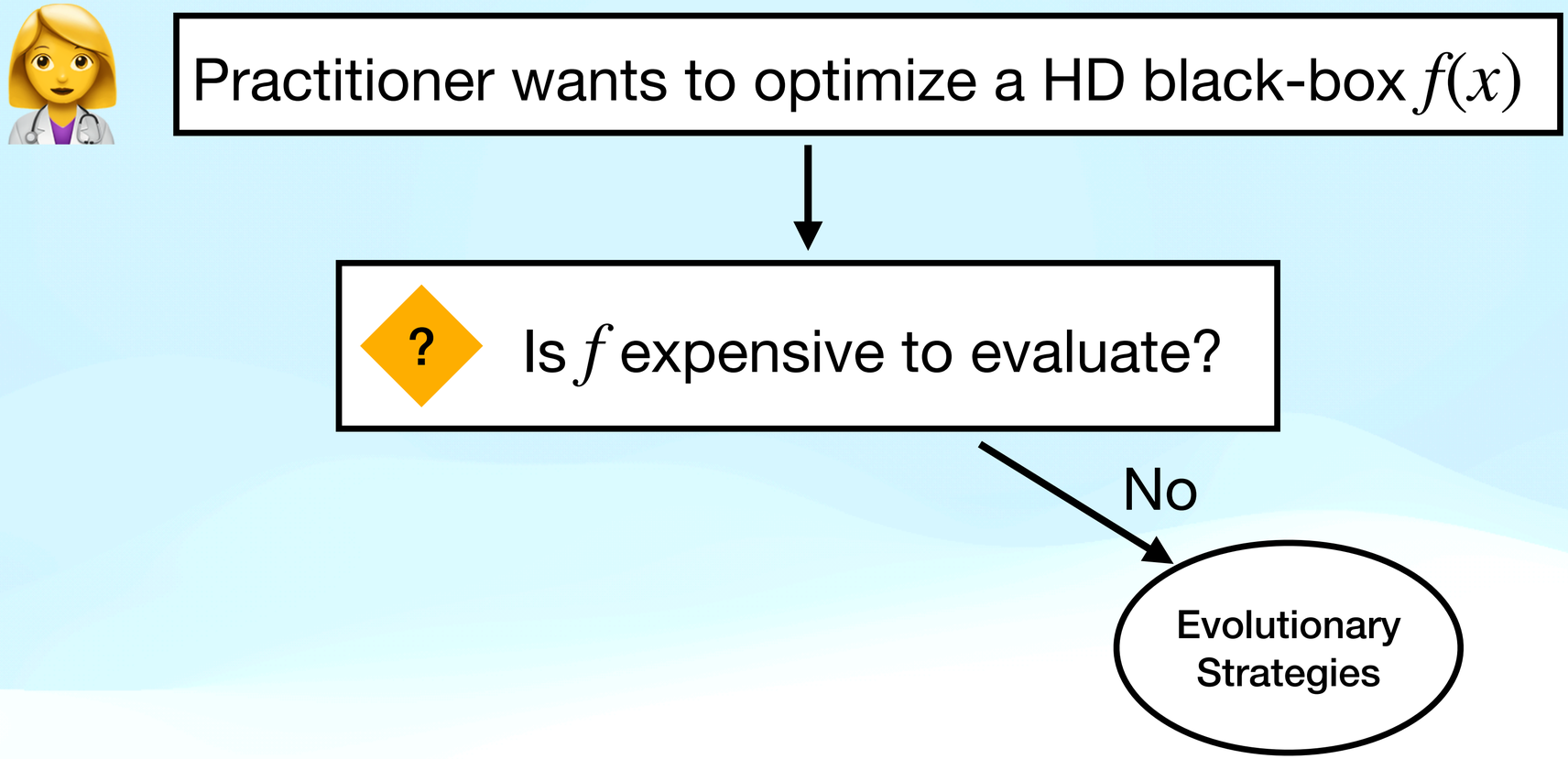
Practitioner wants to optimize a HD black-box $f(x)$

A flowchart for practitioners [work in progress]

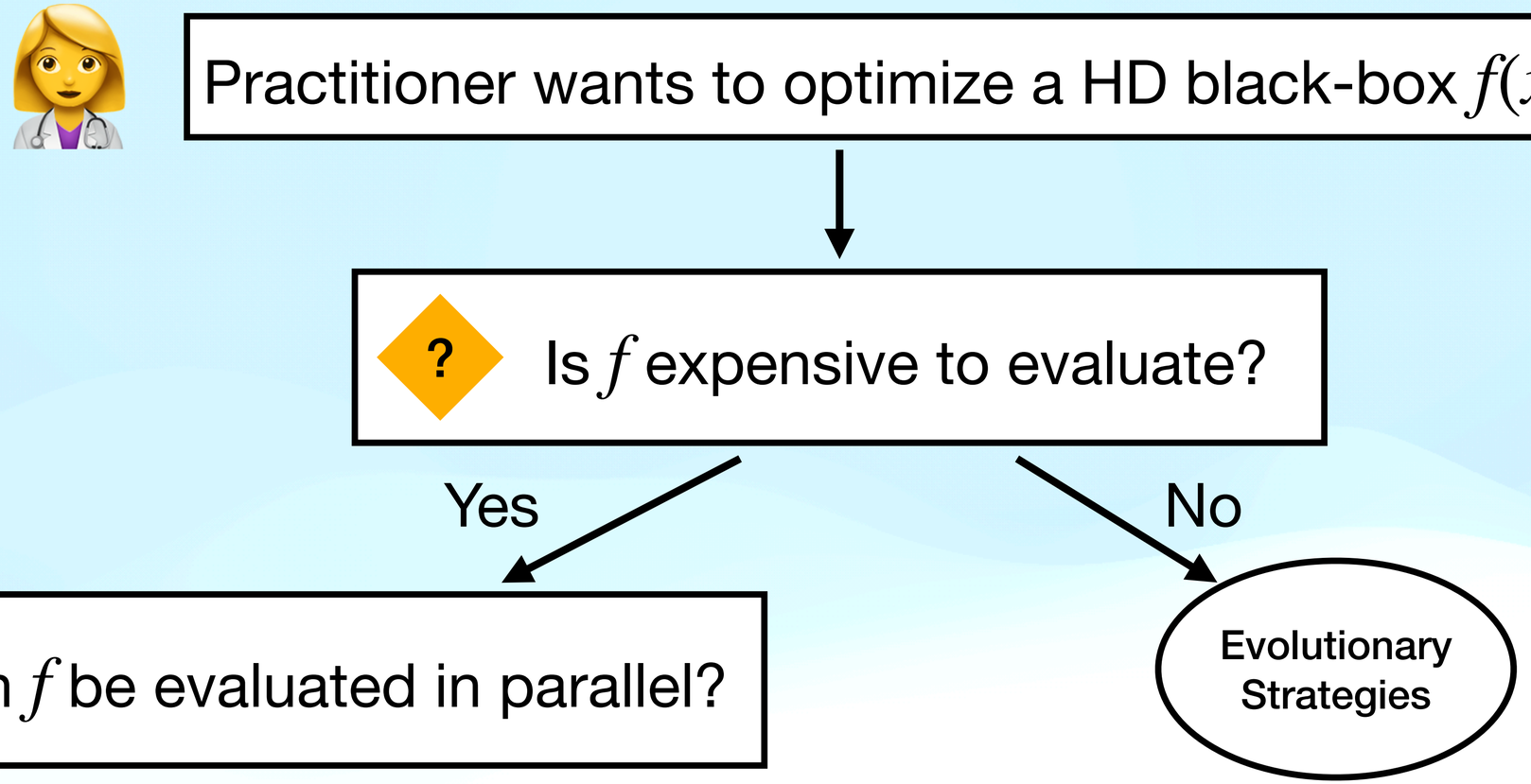
Practitioner wants to optimize a HD black-box $f(x)$

Is f expensive to evaluate?

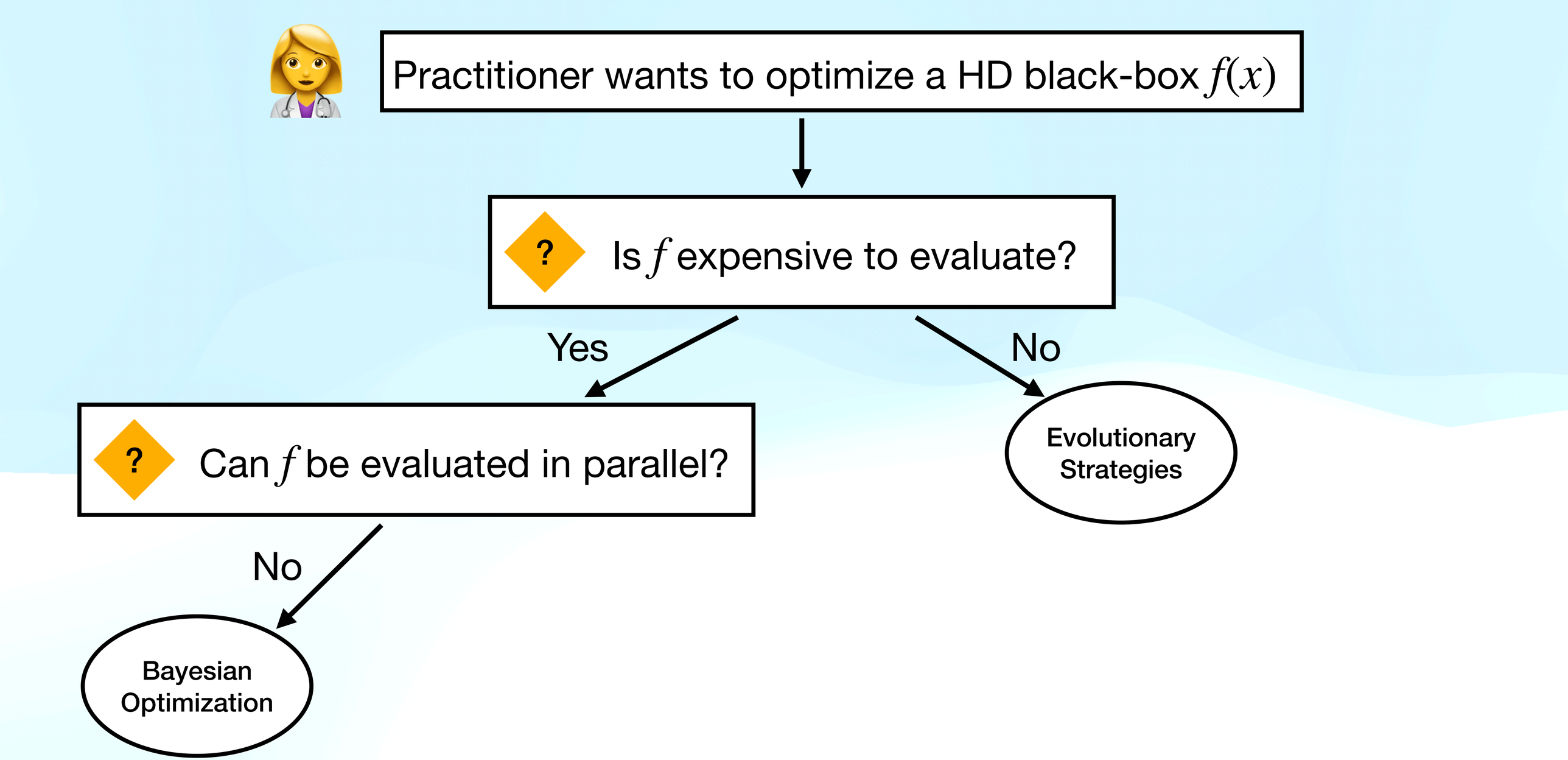
A flowchart for practitioners [work in progress]



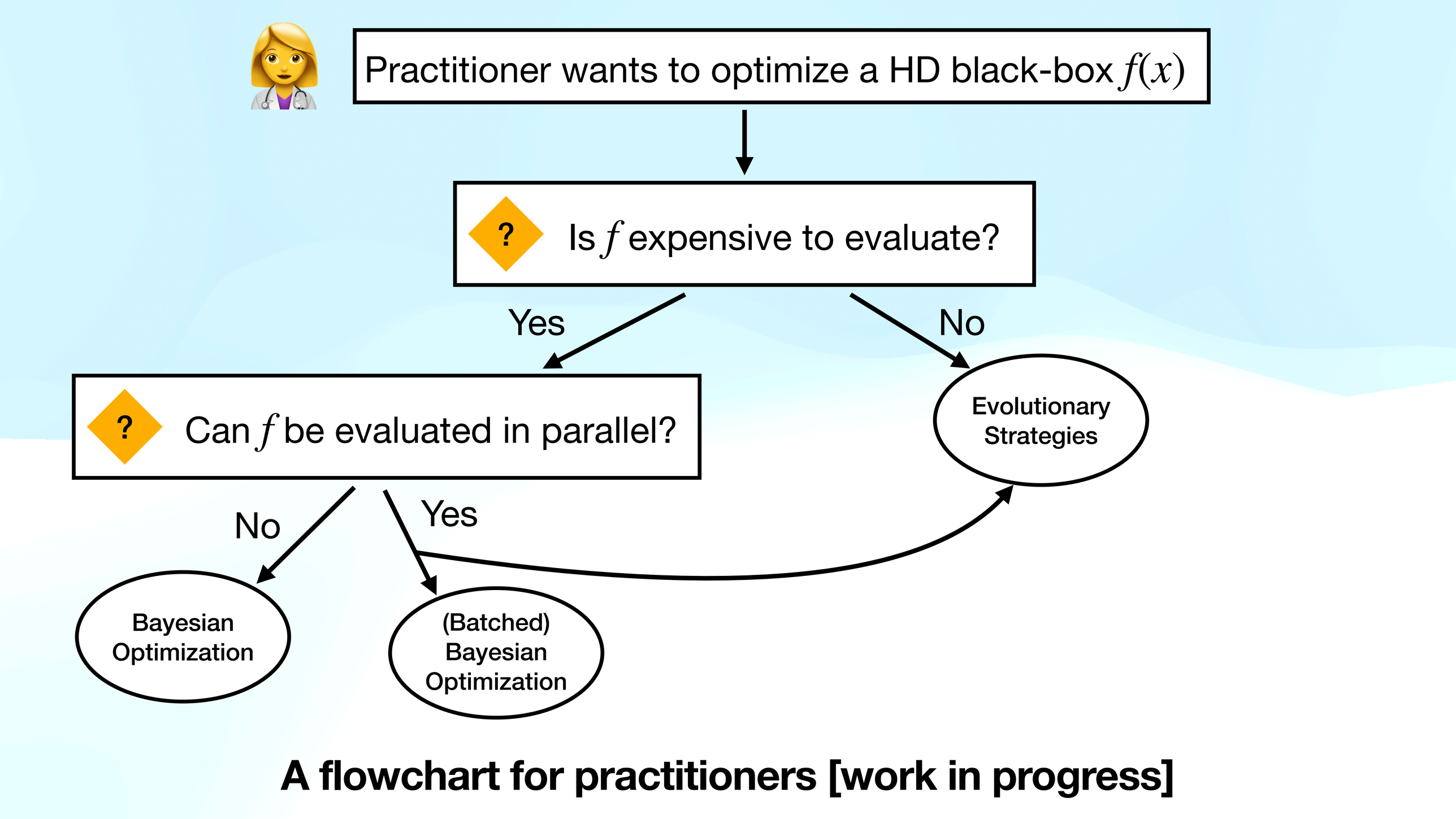
A flowchart for practitioners [work in progress]



A flowchart for practitioners [work in progress]



A flowchart for practitioners [work in progress]



A flowchart for practitioners [work in progress]

Practitioner wants to optimize a HD black-box $f(x)$

? Is f expensive to evaluate?

Yes

No

? Can f be evaluated in parallel?

Evolutionary Strategies

No

Yes

Bayesian Optimization

(Batched) Bayesian Optimization

Q. What is your set-up?

A flowchart for practitioners [work in progress]

High-dimensional Bayesian optimization

Objective function

The function you’re optimizing (e.g. docking score, function…)

BayesOpt has three core ingredients

Objective function

The function you’re optimizing (e.g. docking score, function…)

Surrogate model

A probabilistic approximation of the objective function

BayesOpt has three core ingredients

Objective function

The function you're optimizing (e.g. docking score, function…)

Surrogate model

A probabilistic approximation of the objective function

Acquisition function

An easy-to-query function that uses the surrogate model to find the next best point.

BayesOpt has three core ingredients

$\mathscr{D} = \emptyset$

Start with an empty dataset $\mathscr{D}$

Objective function $f(x)$

Surrogate model $\tilde{f}(x)$

Acquisition function $\alpha(x; \tilde{f})$

Pseudocode for a BayesOpt implementation

$\mathscr{D} = \emptyset$

While we have compute:

Fit the surrogate model $\tilde{f}$ on $\mathscr{D}$

Pseudocode for a BayesOpt implementation

Approximate the objective function

Objective function $f(x)$

Surrogate model $\tilde{f}(x)$

Acquisition function $\alpha(x; \tilde{f})$

$\mathscr{D} = \emptyset$

While we have compute:

Fit the surrogate model $\tilde{f}$ on $\mathscr{D}$

Find the optima of the acq. function $\alpha(x; \tilde{f})$ and call it x_{next}

Pseudocode for a BayesOpt implementation

Optimize the acq. function, which is easy

Objective function $f(x)$

Surrogate model $\tilde{f}(x)$

Acquisition function $\alpha(x; \tilde{f})$

$\mathscr{D} = \emptyset$

While we have compute:

Fit the surrogate model $\tilde{f}$ on $\mathscr{D}$

Find the optima of the acq. function $\alpha(x; \tilde{f})$ and call it x_{next}

Evaluate the objective function f on x_{next}

Add $f(x_{\text{next}})$ to the dataset $\mathscr{D}$

Pseudocode for a BayesOpt implementation

Evaluate the objective function (expensive)

Objective function $f(x)$

Surrogate model $\tilde{f}(x)$

Acquisition function $\alpha(x; \tilde{f})$

$\mathscr{D} = \emptyset$

While we have compute:

Fit the surrogate model $\tilde{f}$ on $\mathscr{D}$

Find the optima of the acq. function $\alpha(x; \tilde{f})$ and call it x_{next}

Evaluate the objective function f on x_{next}

Add $f(x_{\text{next}})$ to the dataset $\mathscr{D}$

Pseudocode for a BayesOpt implementation

Evaluate the objective function (expensive)

Objective function $f(x)$

Surrogate model $\tilde{f}(x)$

Acquisition function $\alpha(x; \tilde{f})$

$\mathscr{D} = \emptyset$

While we have compute:

Fit the surrogate model $\tilde{f}$ on $\mathscr{D}$

Find the optima of the acq. function $\alpha(x; \tilde{f})$ and call it x_{next}

Evaluate the objective function f on x_{next}

Add $f(x_{\text{next}})$ to the dataset $\mathscr{D}$

Pseudocode for a BayesOpt implementation

Evaluate the objective function (expensive)

Objective function $f(x)$

Surrogate model $\tilde{f}(x)$

Acquisition function $\alpha(x; \tilde{f})$

$\mathscr{D} = \emptyset$

While we have compute:

Fit the surrogate model $\tilde{f}$ on $\mathscr{D}$

Find the optima of the acq. function $\alpha(x; \tilde{f})$ and call it x_{next}

Evaluate the objective function f on x_{next}

Add $f(x_{\text{next}})$ to the dataset $\mathscr{D}$

Pseudocode for a BayesOpt implementation

Evaluate the objective function (expensive)

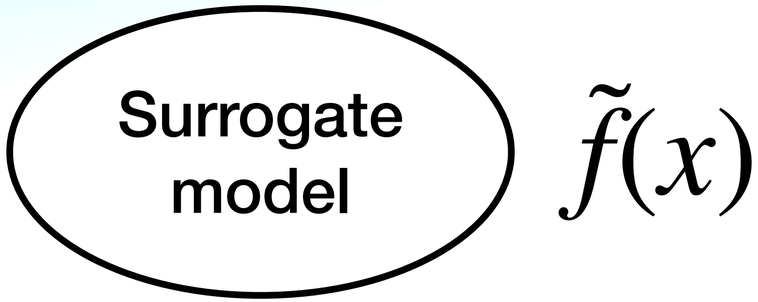
Objective function $f(x)$

Surrogate model $\tilde{f}(x)$

Acquisition function $\alpha(x; \tilde{f})$

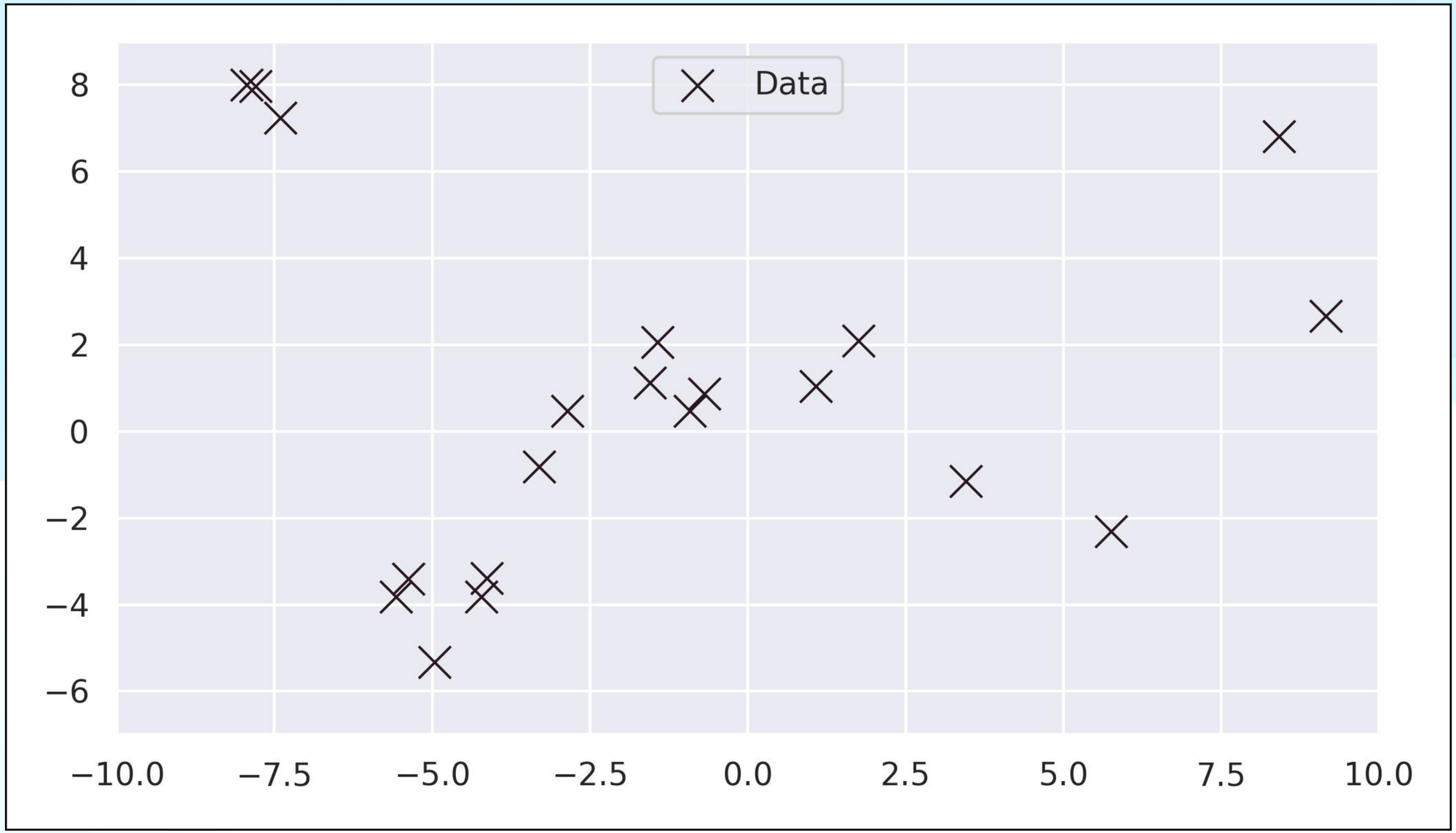
The default choice for surrogate models are

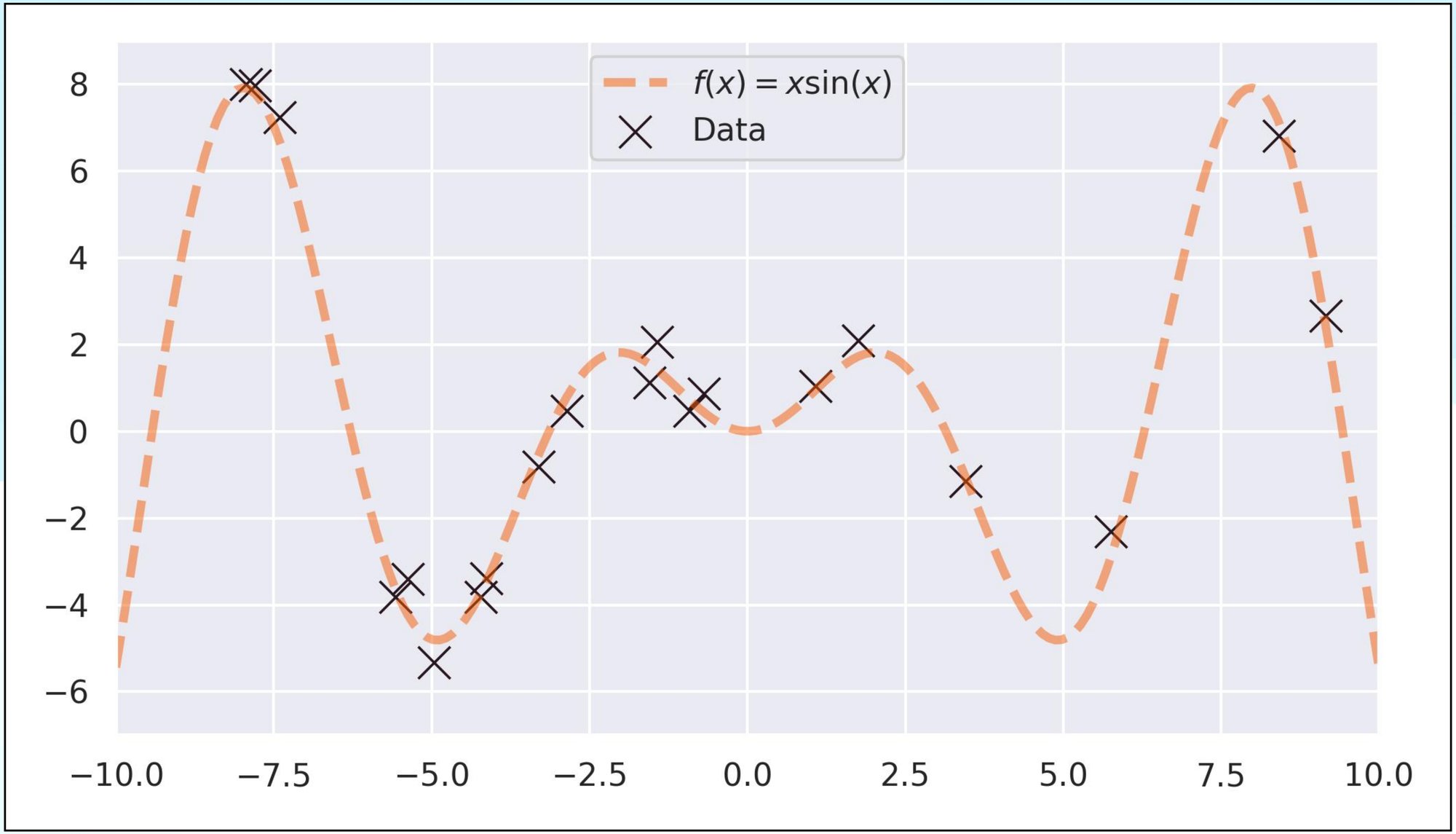
Gaussian Processes

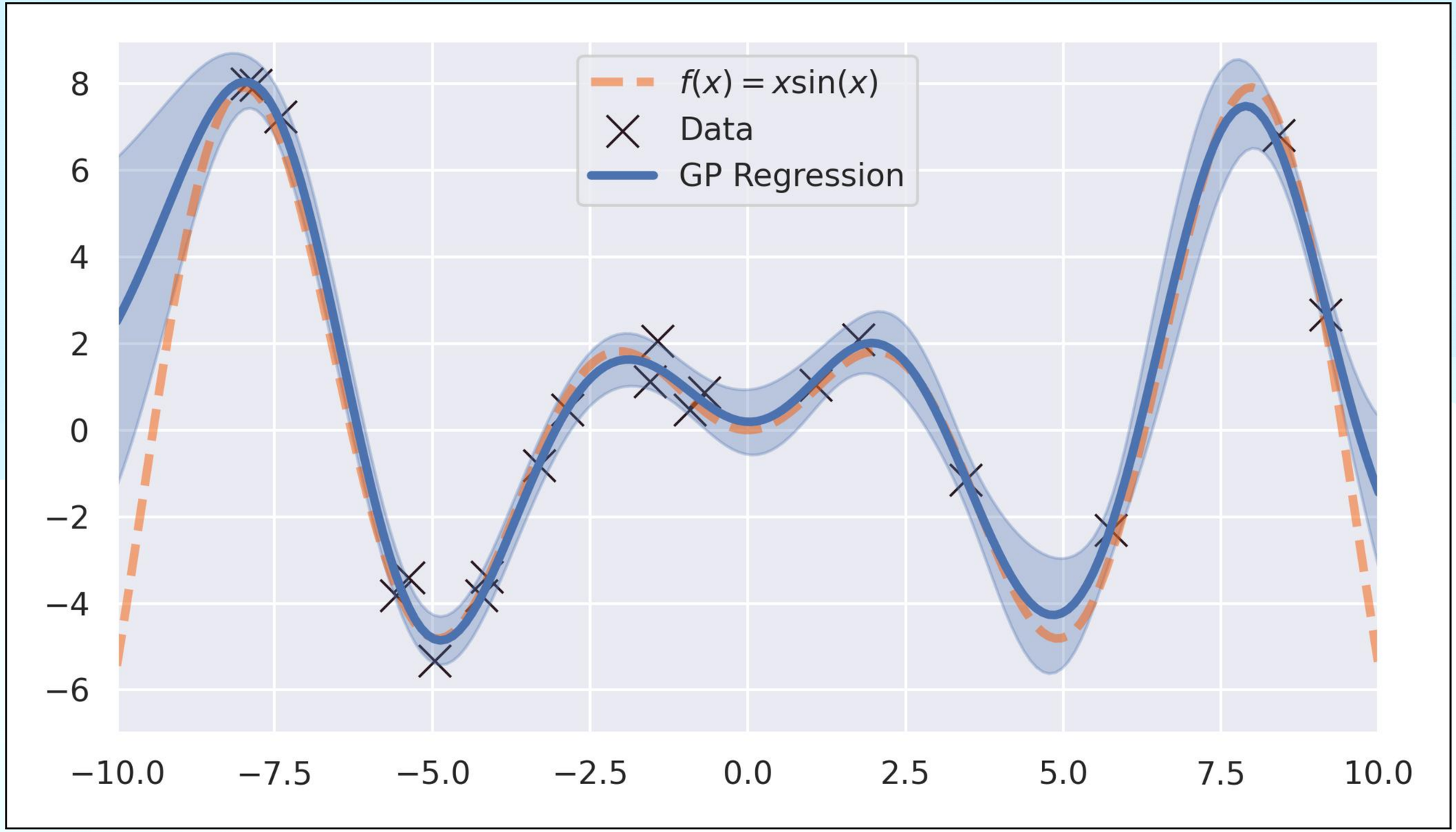


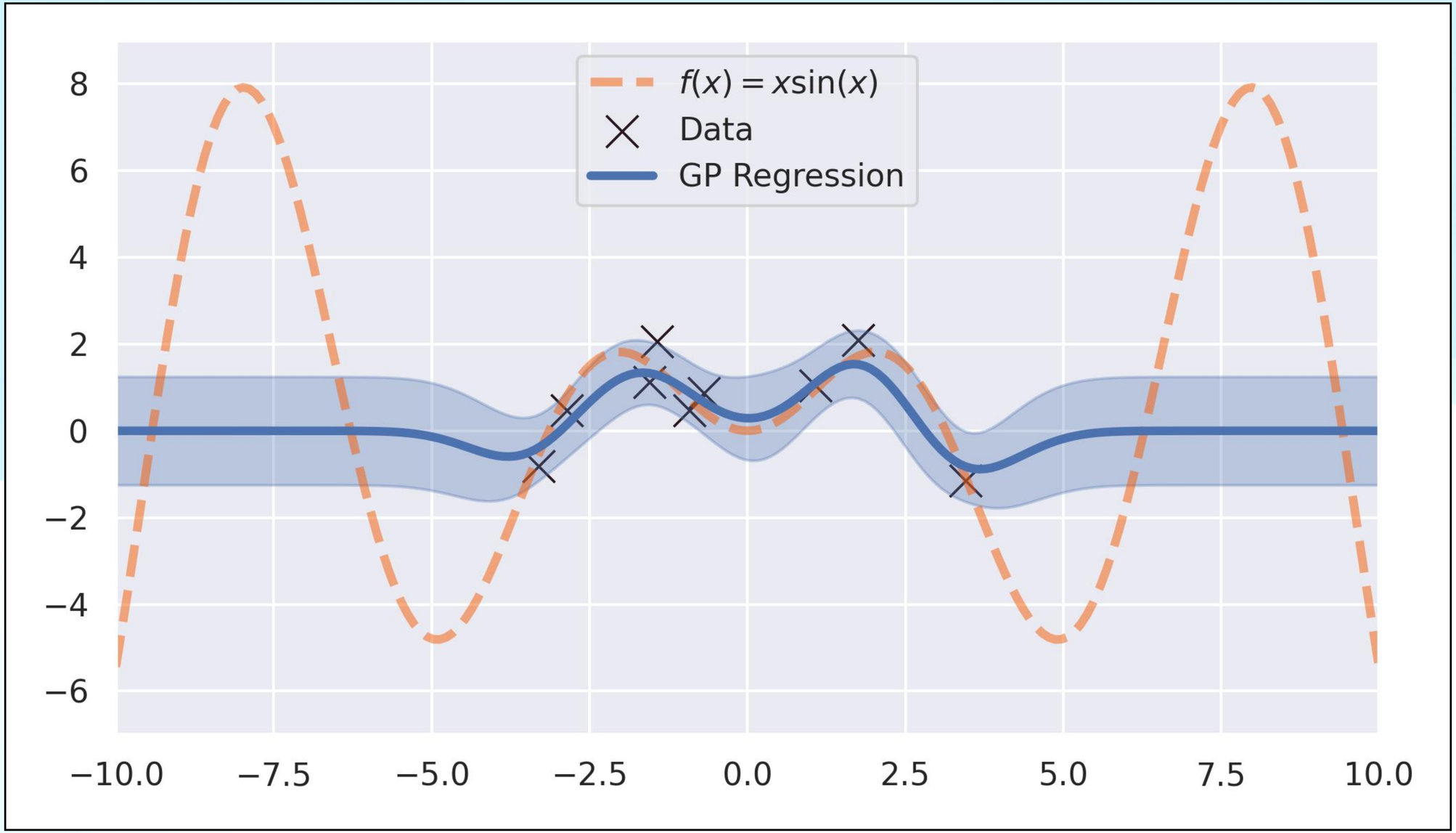
An introduction to Gaussian Processes

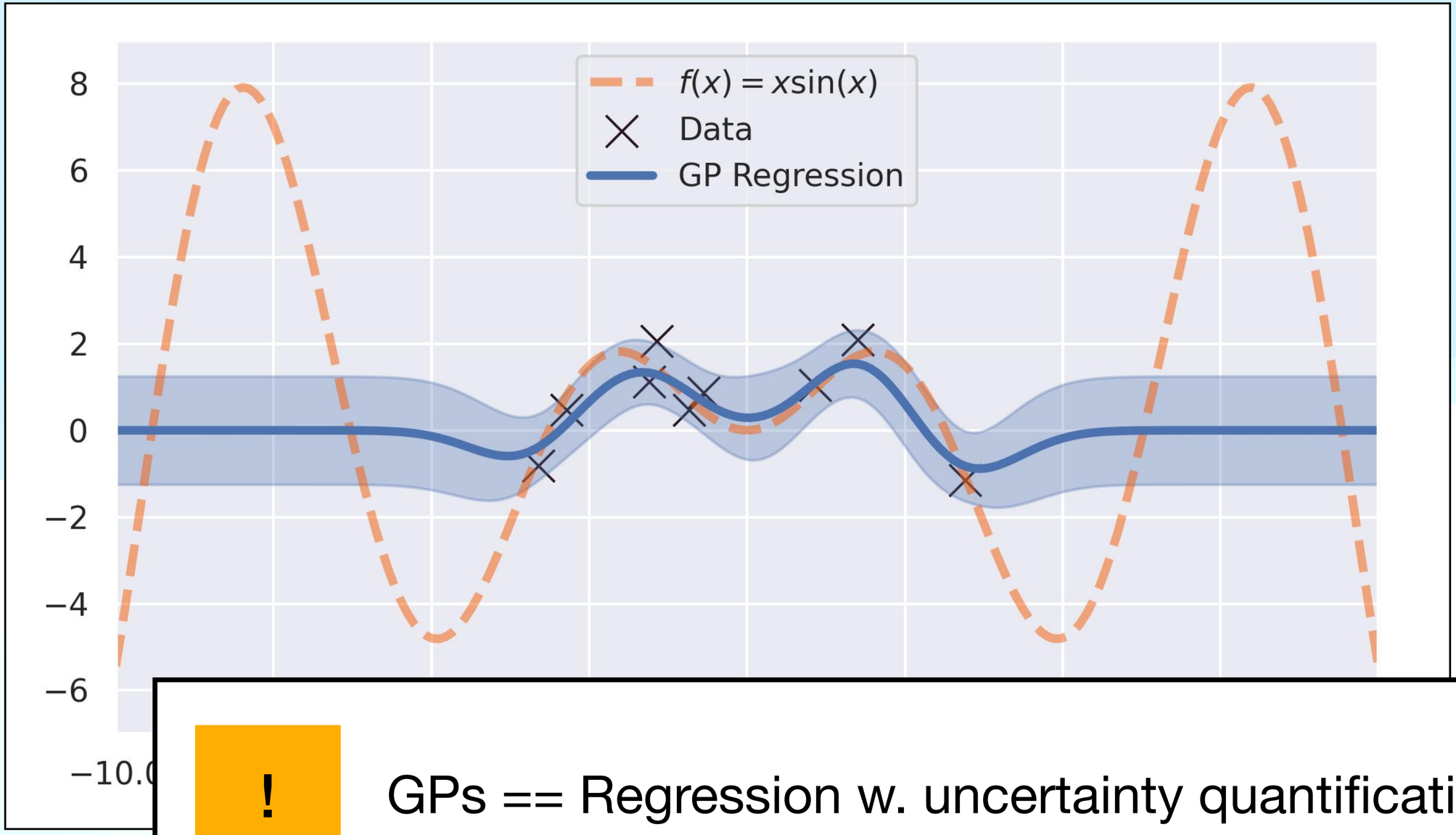
Maths ahead


Data
8
6
4
2
0
−2
−4
−6
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0


$f(x) = x\sin(x)$
Data
8
6
4
2
0
−2
−4
−6
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0


$f(x) = x\sin(x)$
Data
GP Regression
8
6
4
2
0
−2
−4
−6
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0


$f(x) = x\sin(x)$
Data
GP Regression
8
6
4
2
0
−2
−4
−6
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0



!

GPs == Regression w. uncertainty quantification

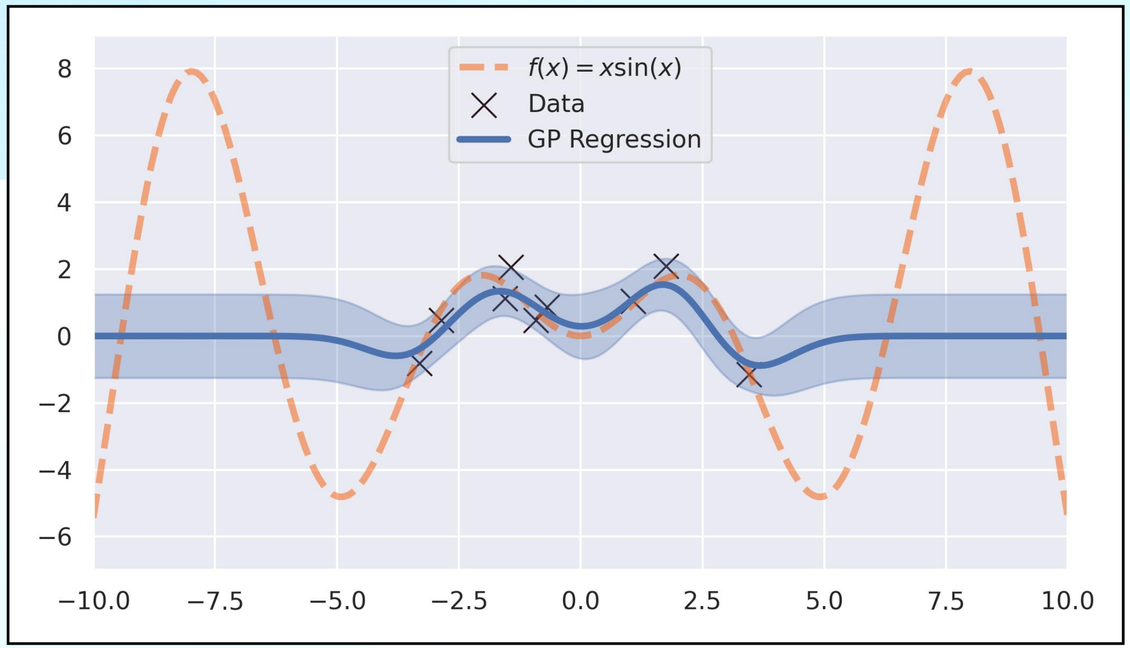
$$f \sim \mathrm{GP}(\mu, k)$$

$$f \sim \mathrm{GP}(\mu, k)$$

Mean function

$$f \sim \mathrm{GP}(\mu, k)$$

Mean function

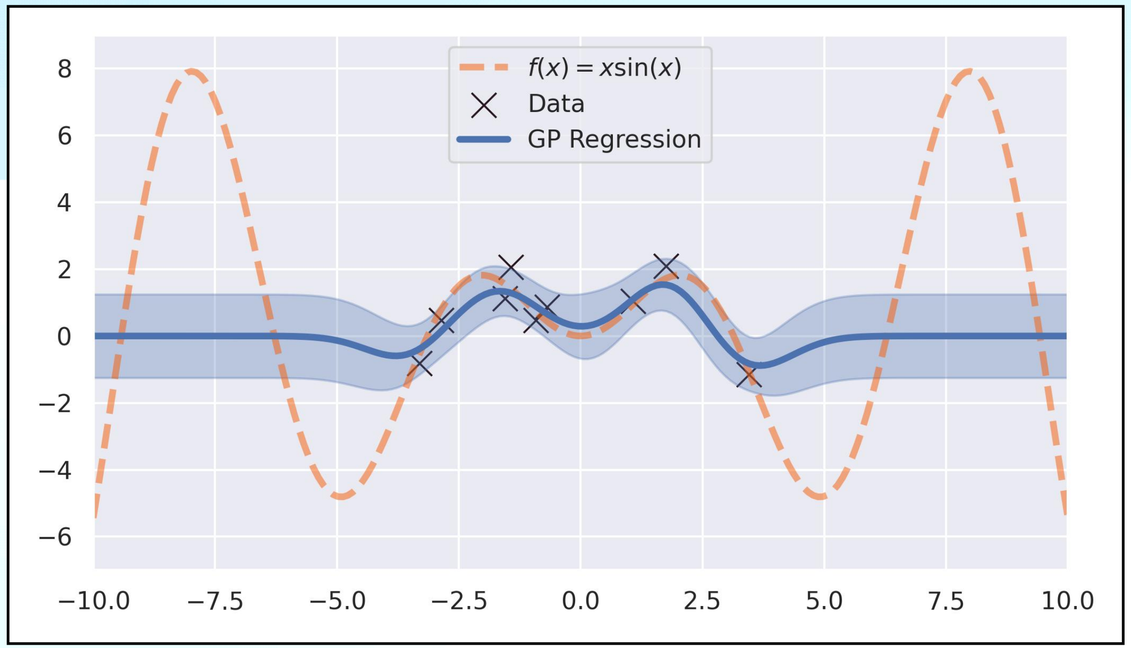


Determines the default behavior away from data

$$f \sim \mathrm{GP}(\mu, k)$$

Mean function

Kernel

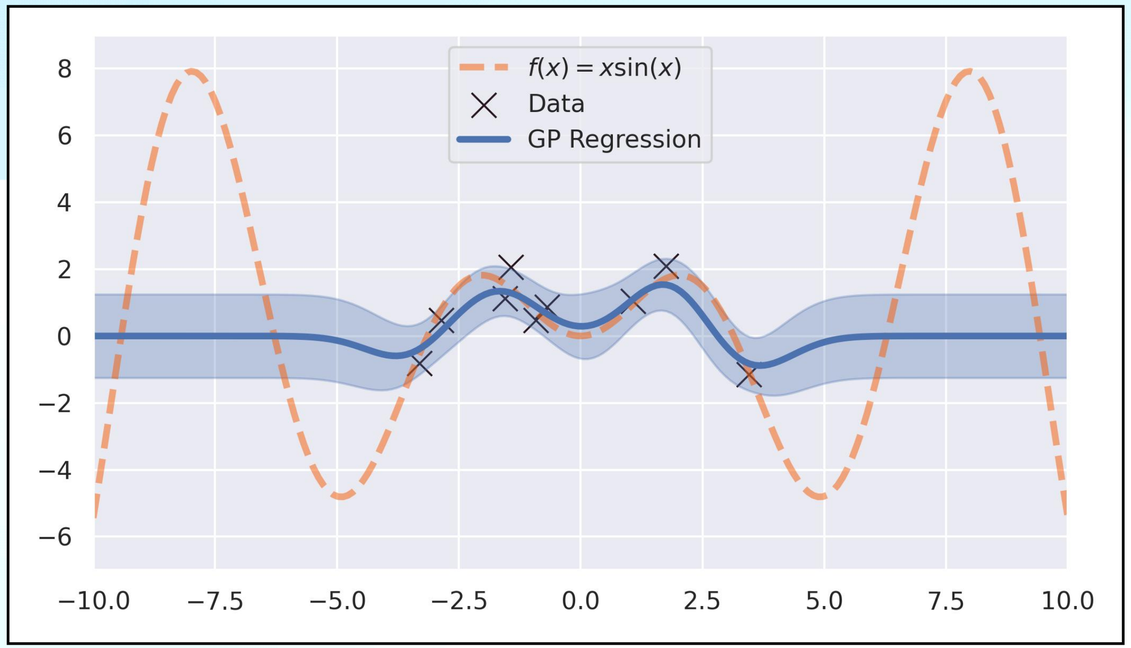


Determines the default behavior away from data

$$f \sim \mathrm{GP}(\mu, k)$$

Mean function

Kernel



Determines the family of functions to search in (e.g. periodic, smooth…)

Determines the default behavior away from data

$$f \sim \mathrm{GP}(\mu, k)$$

$$f \sim \mathrm{GP}(\mu, k)$$

$$\Longleftrightarrow$$

$$\{f(x_1), \ldots, f(x_N)\} \sim \mathcal{N}(\boldsymbol{\mu}, K)$$

$$f \sim \mathrm{GP}(\mu, k)$$

$$\Longleftrightarrow$$

$$\{f(x_1), \ldots, f(x_N)\} \sim \mathcal{N}(\boldsymbol{\mu}, K)$$

Where

$$\boldsymbol{\mu} = [\mu(x_1), \ldots, \mu(x_N)]$$

$$K = [k(x_i, x_j)]_{i,j=1}^{N}$$

$$f \sim \mathrm{GP}(\mu, k)$$

$$\Longleftrightarrow$$

$$\{f(x_1), \ldots, f(x_N)\} \sim \mathcal{N}(\boldsymbol{\mu}, K)$$

Where

$$\boldsymbol{\mu} = [\mu(x_1), \ldots, \mu(x_N)] \longrightarrow \text{Mean vector}$$

$$K = [k(x_i, x_j)]_{i,j=1}^{N} \longrightarrow \text{Gram matrix}$$

$$f \sim \mathrm{GP}(\mu, k)$$

$$\Longleftrightarrow$$

For a given pair of inputs $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^D$

$$\mathbb{E}[f(\boldsymbol{x})] = \mu(\boldsymbol{x})$$

$$\mathrm{Cov}(f(\boldsymbol{x}), f(\boldsymbol{x}')) = k(\boldsymbol{x}, \boldsymbol{x}')$$

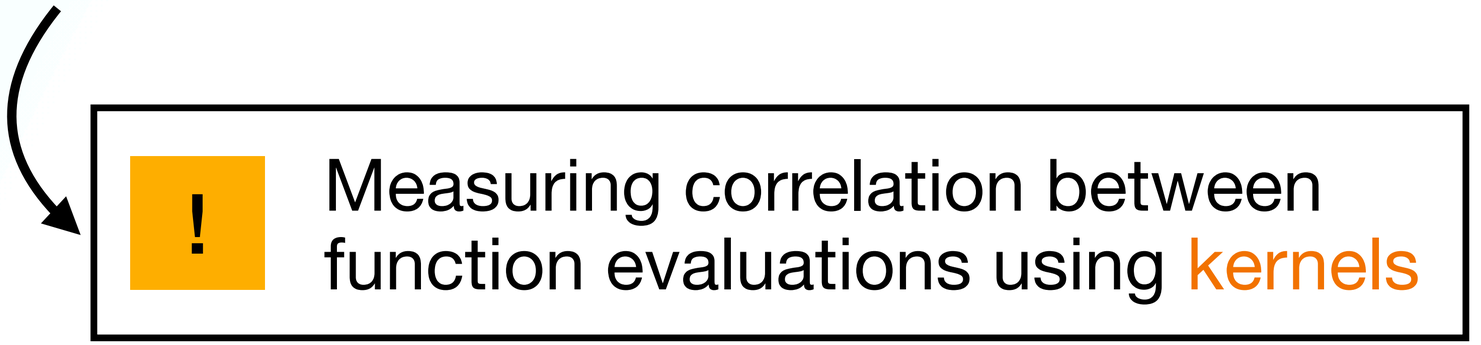
$$f \sim \mathrm{GP}(\mu, k)$$

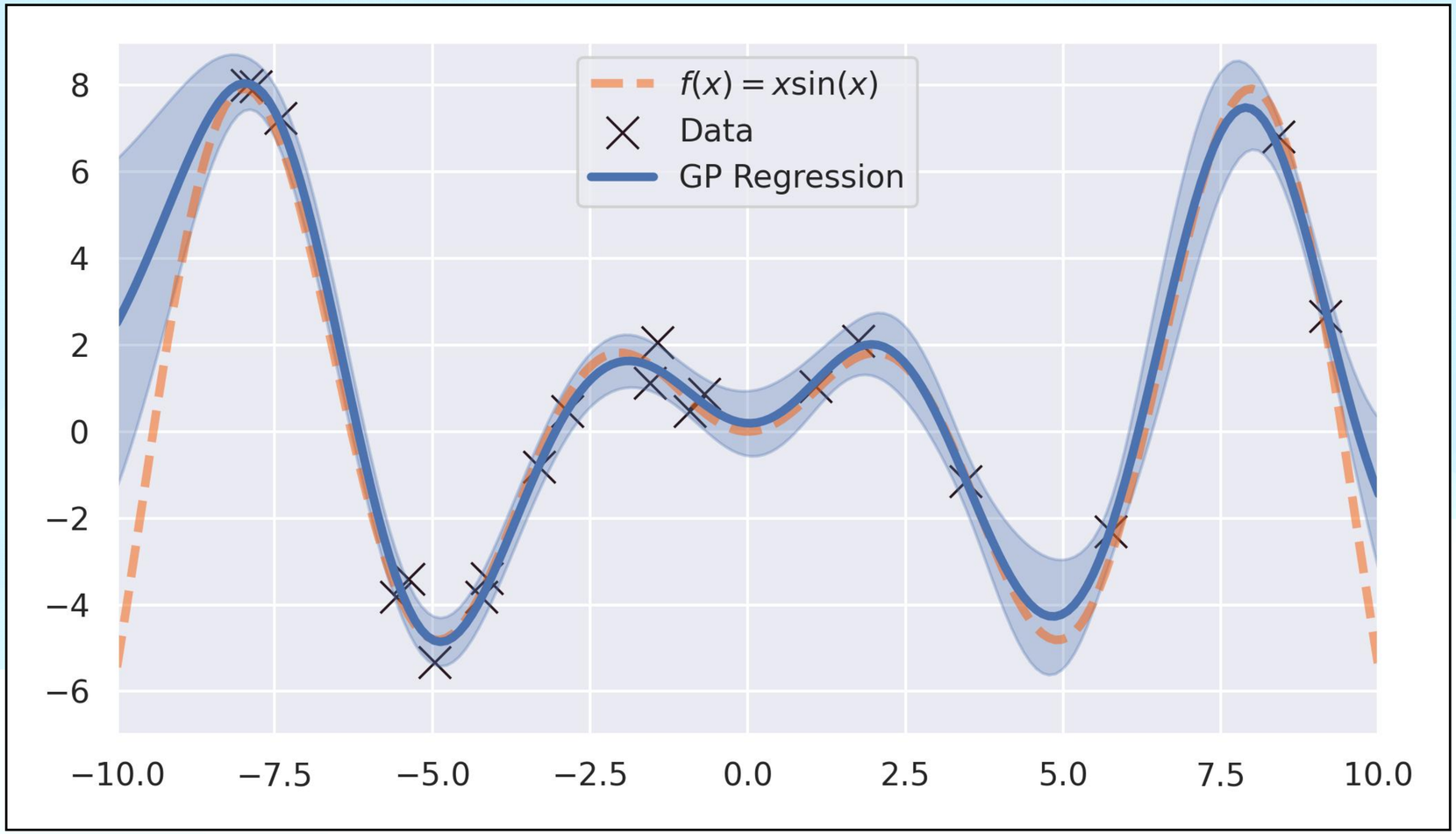
$$\Longleftrightarrow$$

For a given pair of inputs $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^D$

$$\mathbb{E}[f(\boldsymbol{x})] = \mu(\boldsymbol{x})$$

$$\mathrm{Cov}(f(\boldsymbol{x}), f(\boldsymbol{x}')) = k(\boldsymbol{x}, \boldsymbol{x}')$$

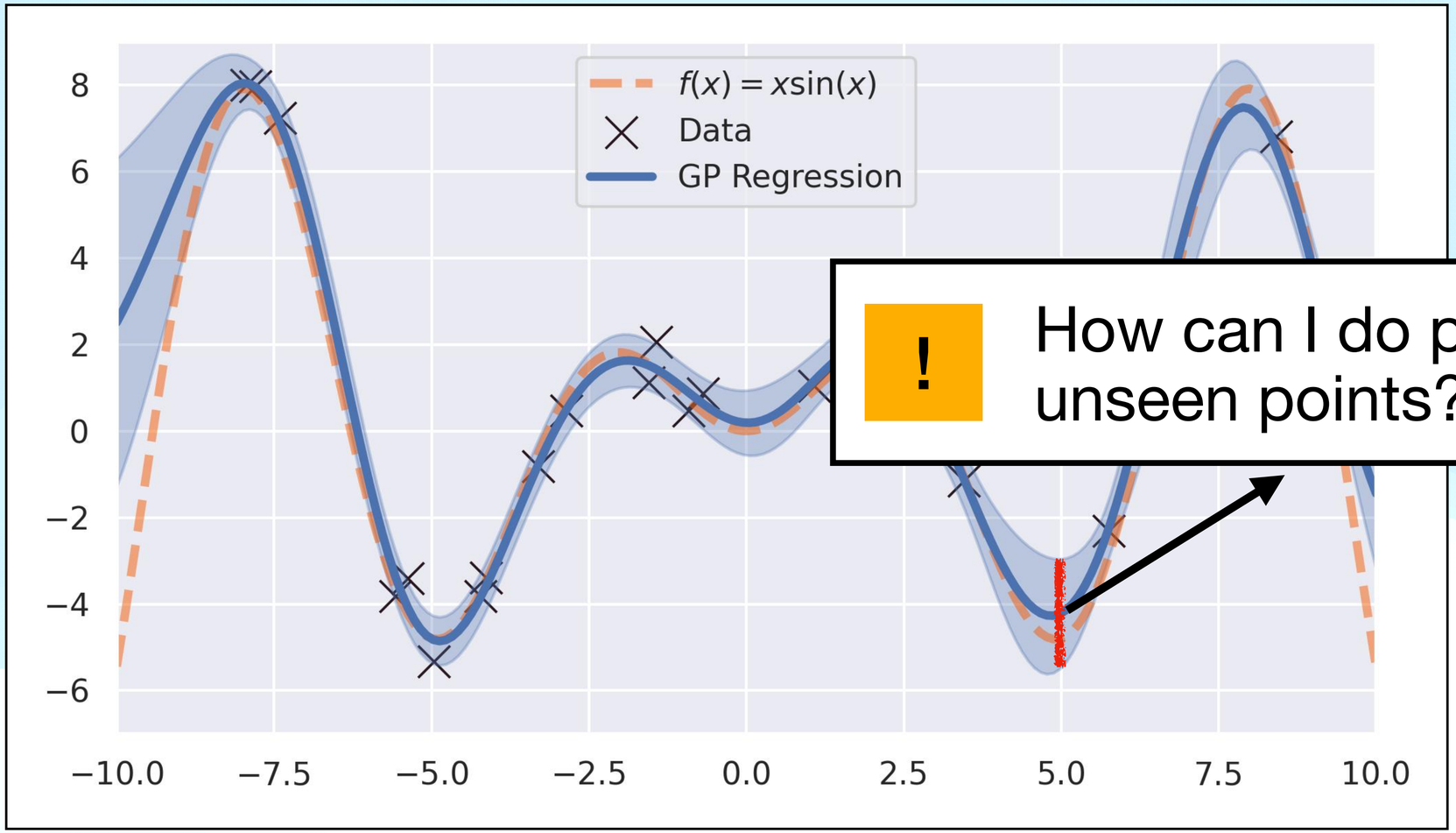




$$\{f(x_1), \ldots, f(x_N)\} \sim \mathcal{N}(\boldsymbol{\mu}, K)$$

$$\boldsymbol{\mu} = [\mu(x_1), \ldots, \mu(x_N)]$$

$$K = [k(x_i, x_j)]_{i,j=1}^{N}$$



! How can I do predictions in unseen points?

$$\{f(x_1), \ldots, f(x_N)\} \sim \mathcal{N}(\boldsymbol{\mu}, K)$$

$$\boldsymbol{\mu} = [\mu(x_1), \ldots, \mu(x_N)]$$

$$K = [k(x_i, x_j)]_{i,j=1}^{N}$$

In an unseen point $\boldsymbol{x}_* \in \mathbb{R}^D$

$$\{f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_N), f(\boldsymbol{x}_*)\} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_{1:N} \\ \mu(x_*) \end{bmatrix}, \begin{bmatrix} K_{1:N} & \boldsymbol{k}_* \\ \boldsymbol{k}_*^\top & k(\boldsymbol{x}_*, \boldsymbol{x}_*) \end{bmatrix}\right)$$

In an unseen point $\boldsymbol{x}_* \in \mathbb{R}^D$

$$\{f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_N), f(\boldsymbol{x}_*)\} \sim \mathcal{N}\left(\begin{bmatrix}\boldsymbol{\mu}_{1:N} \\ \mu(x_*)\end{bmatrix}, \begin{bmatrix}K_{1:N} & \boldsymbol{k}_* \\ \boldsymbol{k}_*^\top & k(\boldsymbol{x}_*, \boldsymbol{x}_*)\end{bmatrix}\right)$$

Using properties from the Normal distribution

$$p(f(\boldsymbol{x}_*) \,|\, f(x_1), \ldots, f(x_N)) \sim \mathcal{N}\left(\mu_{*|1:N}, k_{*|1:N}\right)$$

In an unseen point $\boldsymbol{x}_* \in \mathbb{R}^D$

$$\{f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_N), f(\boldsymbol{x}_*)\} \sim \mathcal{N}\left(\begin{bmatrix}\boldsymbol{\mu}_{1:N} \\ \mu(x_*)\end{bmatrix}, \begin{bmatrix}K_{1:N} & \boldsymbol{k}_* \\ \boldsymbol{k}_*^\top & k(\boldsymbol{x}_*, \boldsymbol{x}_*)\end{bmatrix}\right)$$

Using properties from the Normal distribution

$$p(f(\boldsymbol{x}_*) \,|\, f(x_1), \ldots, f(x_N)) \sim \mathcal{N}\left(\mu_{*|1:N}, k_{*|1:N}\right)$$

Where

$$\mu_{*|1:N} = \mu(\boldsymbol{x}_*) + \boldsymbol{k}_*^\top K_{1:N}^{-1}(\boldsymbol{f}_{1:N} - \boldsymbol{\mu}_{1:N})$$

$$k_{*|1:N} = k(\boldsymbol{x}_*, \boldsymbol{x}_*) + \boldsymbol{k}_*^\top K_{1:N}^{-1}\boldsymbol{k}_*$$

$$p(f(\boldsymbol{x}_*) \,|\, f(x_1), \ldots, f(x_N)) \sim \mathcal{N}\left(\mu_{*|1:N}, k_{*|1:N}\right)$$

$$\mu_{*|1:N} = \mu(\boldsymbol{x}_*) + \boldsymbol{k}_*^\top K_{1:N}^{-1}(\boldsymbol{f}_{1:N} - \boldsymbol{\mu}_{1:N})$$

$$k_{*|1:N} = k(\boldsymbol{x}_*, \boldsymbol{x}_*) + \boldsymbol{k}_*^\top K_{1:N}^{-1}\boldsymbol{k}_*$$

$$p(f(\boldsymbol{x}_*) \,|\, f(x_1), \ldots, f(x_N)) \sim \mathcal{N}\left(\mu_{*|1:N}, k_{*|1:N}\right)$$

$$\mu_{*|1:N} = \mu(\boldsymbol{x}_*) + \boldsymbol{k}_*^\top K_{1:N}^{-1}(\boldsymbol{f}_{1:N} - \boldsymbol{\mu}_{1:N})$$

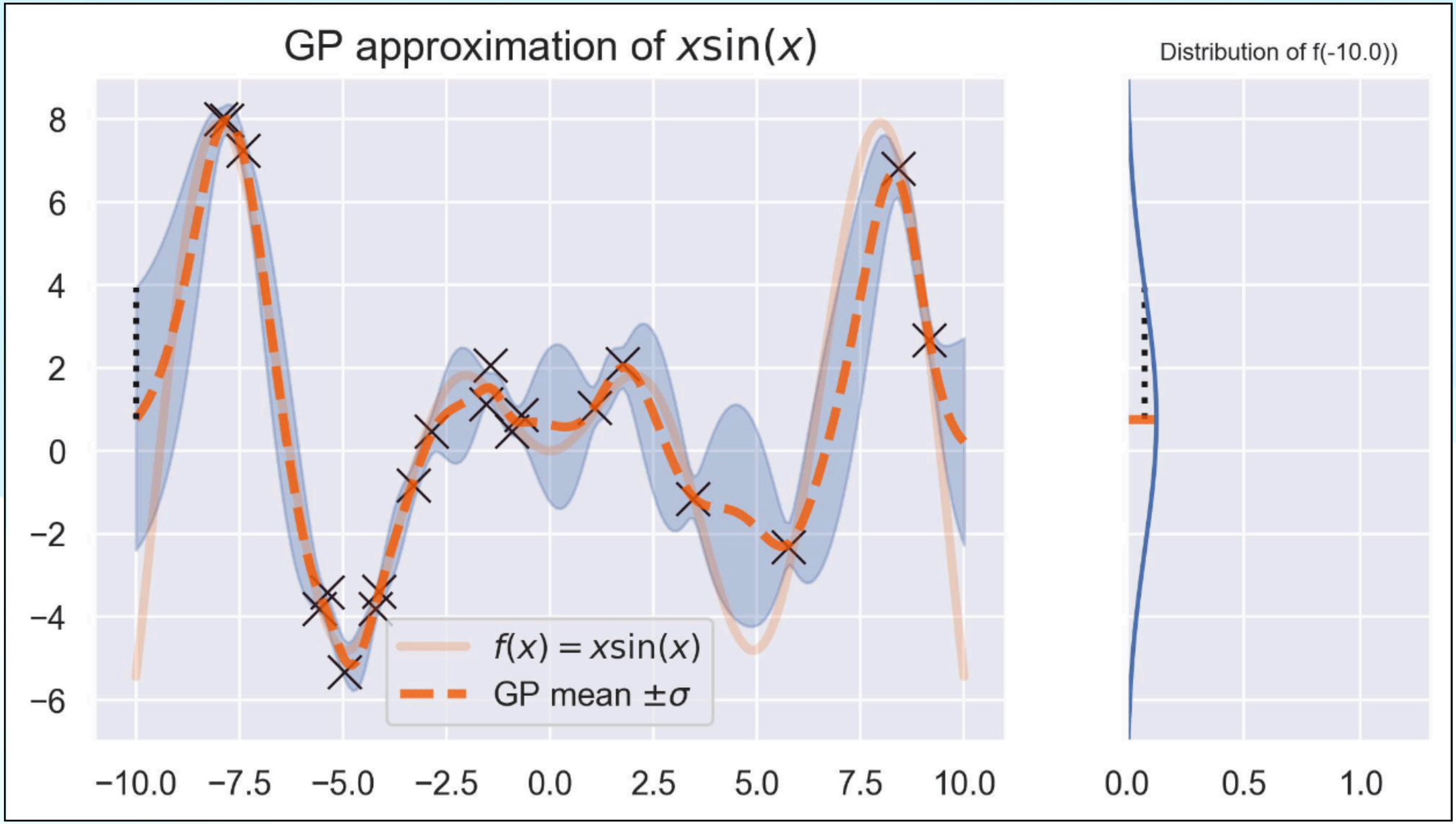
$$k_{*|1:N} = k(\boldsymbol{x}_*, \boldsymbol{x}_*) + \boldsymbol{k}_*^\top K_{1:N}^{-1}\boldsymbol{k}_*$$

$$p(f(\boldsymbol{x}_*) \,|\, f(x_1), \ldots, f(x_N)) \sim \mathcal{N}\left(\mu_{*|1:N}, k_{*|1:N}\right)$$

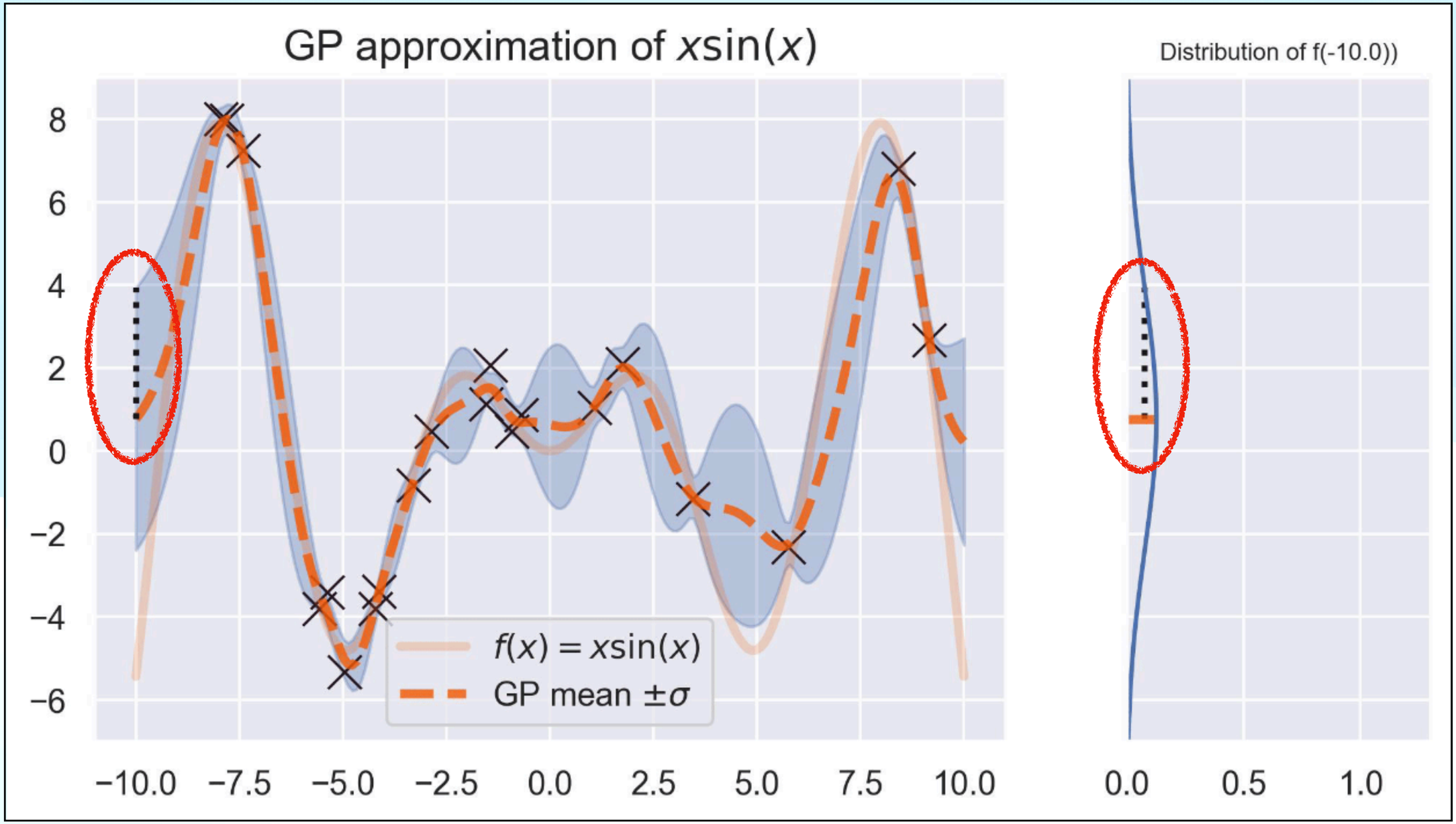
$$\mu_{*|1:N} = \mu(\boldsymbol{x}_*) + \boldsymbol{k}_*^\top K_{1:N}^{-1}(\boldsymbol{f}_{1:N} - \boldsymbol{\mu}_{1:N})$$

$$k_{*|1:N} = k(\boldsymbol{x}_*, \boldsymbol{x}_*) + \boldsymbol{k}_*^\top K_{1:N}^{-1}\boldsymbol{k}_*$$

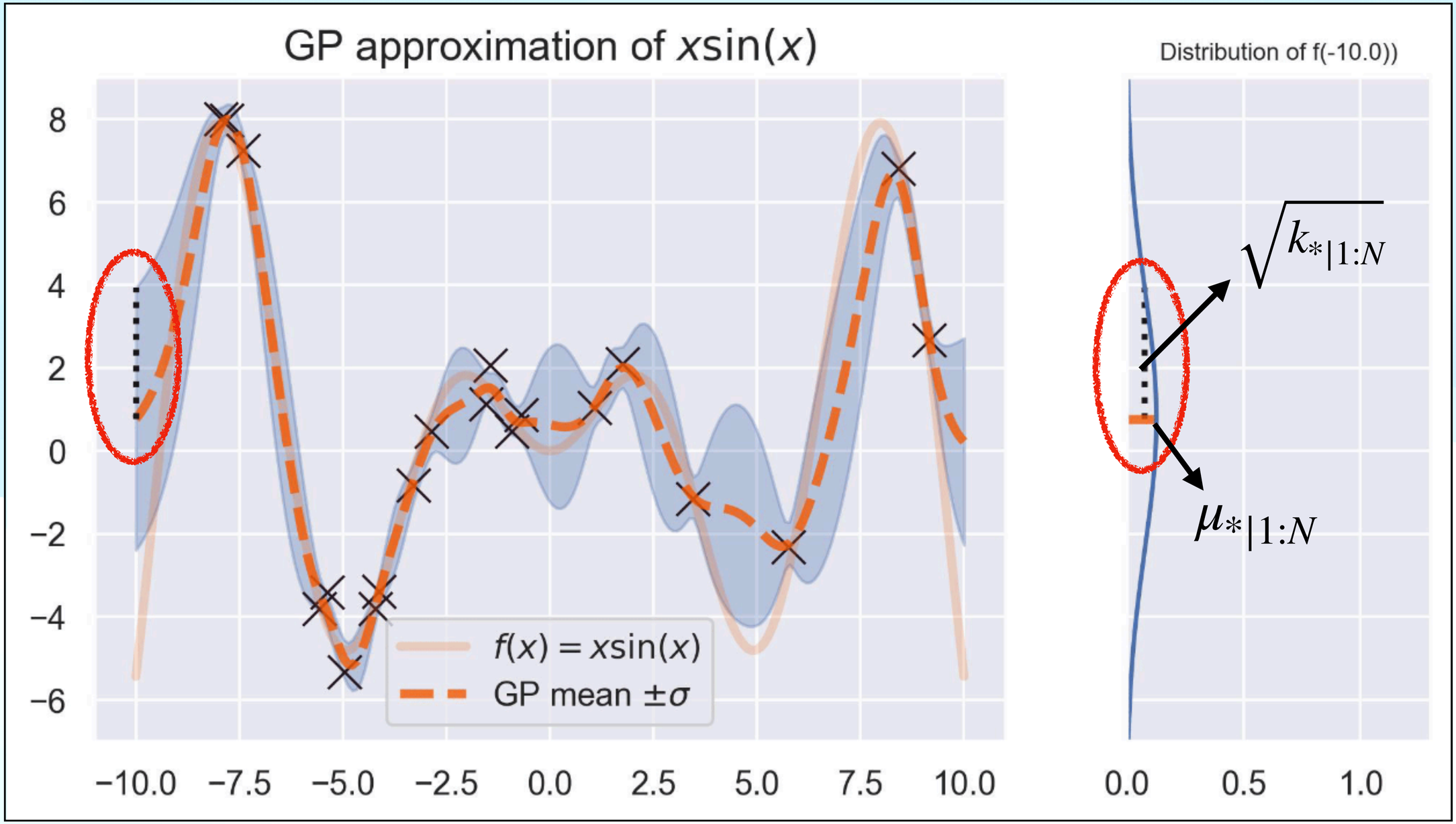
1. We need to invert a matrix of size N^*



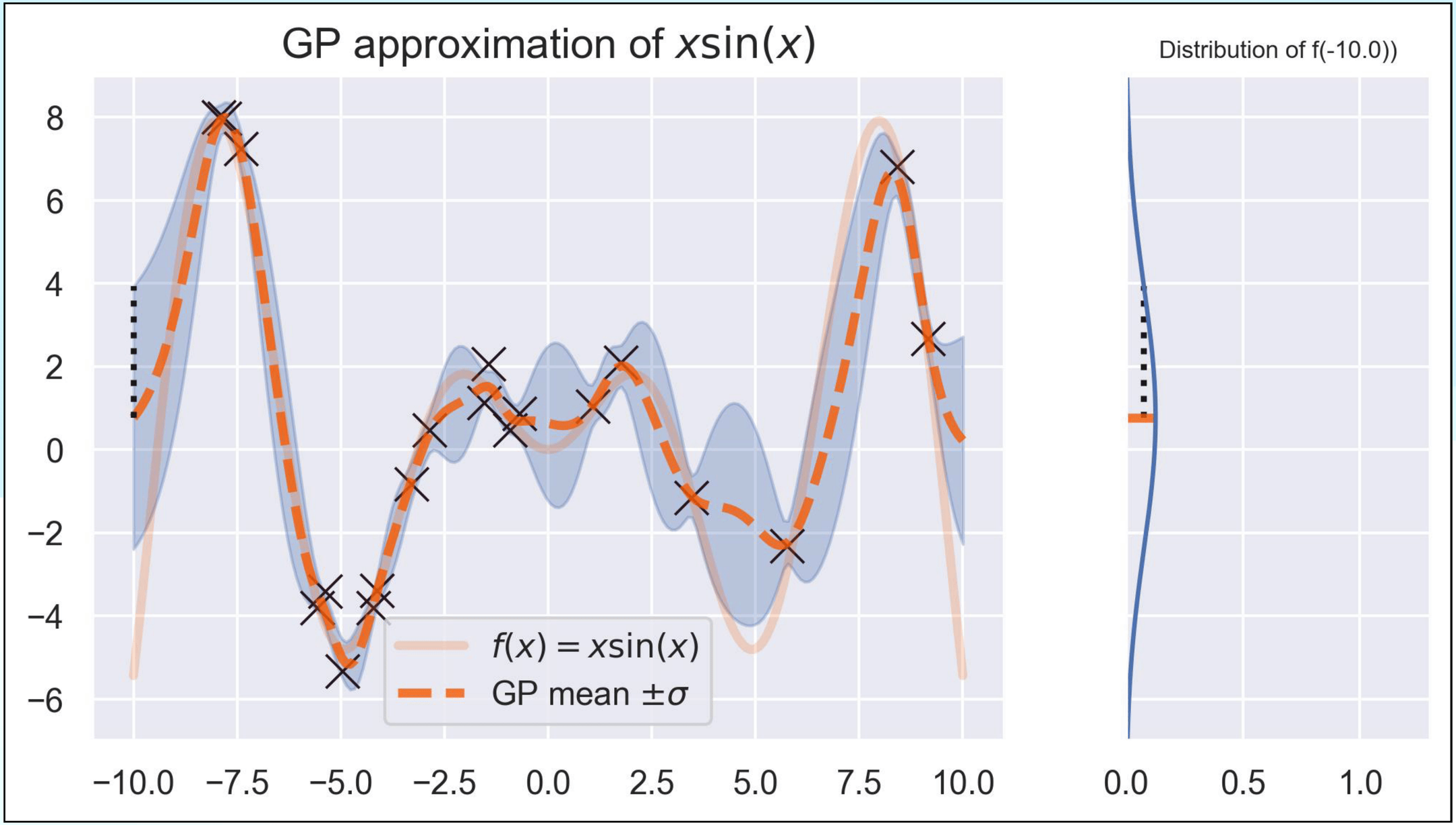
https://www.miguelgondu.com/blogposts/2023-07-31/intro-to-bo/



https://www.miguelgondu.com/blogposts/2023-07-31/intro-to-bo/



https://www.miguelgondu.com/blogposts/2023-07-31/intro-to-bo/



https://www.miguelgondu.com/blogposts/2023-07-31/intro-to-bo/

Examples of kernels

$$k(x, x') = \sigma^2(x^T x' + b)$$

Linear

$$k(x, x') = \alpha \exp\left(-\frac{l\|x - x'\|^2}{2\sigma^2}\right)$$

Squared-exponential

$$k(x, x') = \exp\left(-\frac{2\sin^2(\pi\|x - x'\|/p)}{l^2}\right)$$

Periodic

$$k(x, x') = \sigma^2(x^T x' + b)$$

Linear

$$k(x, x') = \alpha \exp\left(-\frac{l\|x - x'\|^2}{2\sigma^2}\right)$$

Squared-exponential

$$k(x, x') = \exp\left(-\frac{2\sin^2(\pi\|x - x'\|/p)}{l^2}\right)$$

Periodic

Examples of kernels

Examples of kernels

$$k(x, x') = \sigma^2(x^T x' + b)$$

$$k(x, x') = \alpha \exp\left(-\frac{l\|x - x'\|^2}{2\sigma^2}\right)$$

Squared-exponential

! Kernels have hyperparamaters

$$k(x, x') = \exp\left(-\frac{2\sin^2(\pi\|x - x'\|/p)}{l^2}\right)$$

Periodic

Examples of kernels

$$k(x, x') = \sigma^2(x^T x' + b)$$

Periodic

$$k(x, x') = \alpha \exp\left(-\frac{l\|x - x'\|^2}{2\sigma^2}\right)$$

Squared-exponential

! Kernels have hyperparamaters

https://distill.pub/2019/visual-exploration-gaussian-processes/

$$k(x, x') = \sigma^2(x^T x' + b)$$

$$k(x, x') = \alpha \exp\left(-\frac{l\|x - x'\|^2}{2\sigma^2}\right)$$

Squared-exponential

! Kernels have hyperparamaters

https://distill.pub/2019/visual-exploration-gaussian-processes/

! We train these by maximizing the marginal log-likelihood

Examples of kernels

Algebra of kernels

If k_1 and k_2 are two kernels, then the following are also kernels

$$k_1 + k_2$$

Sum

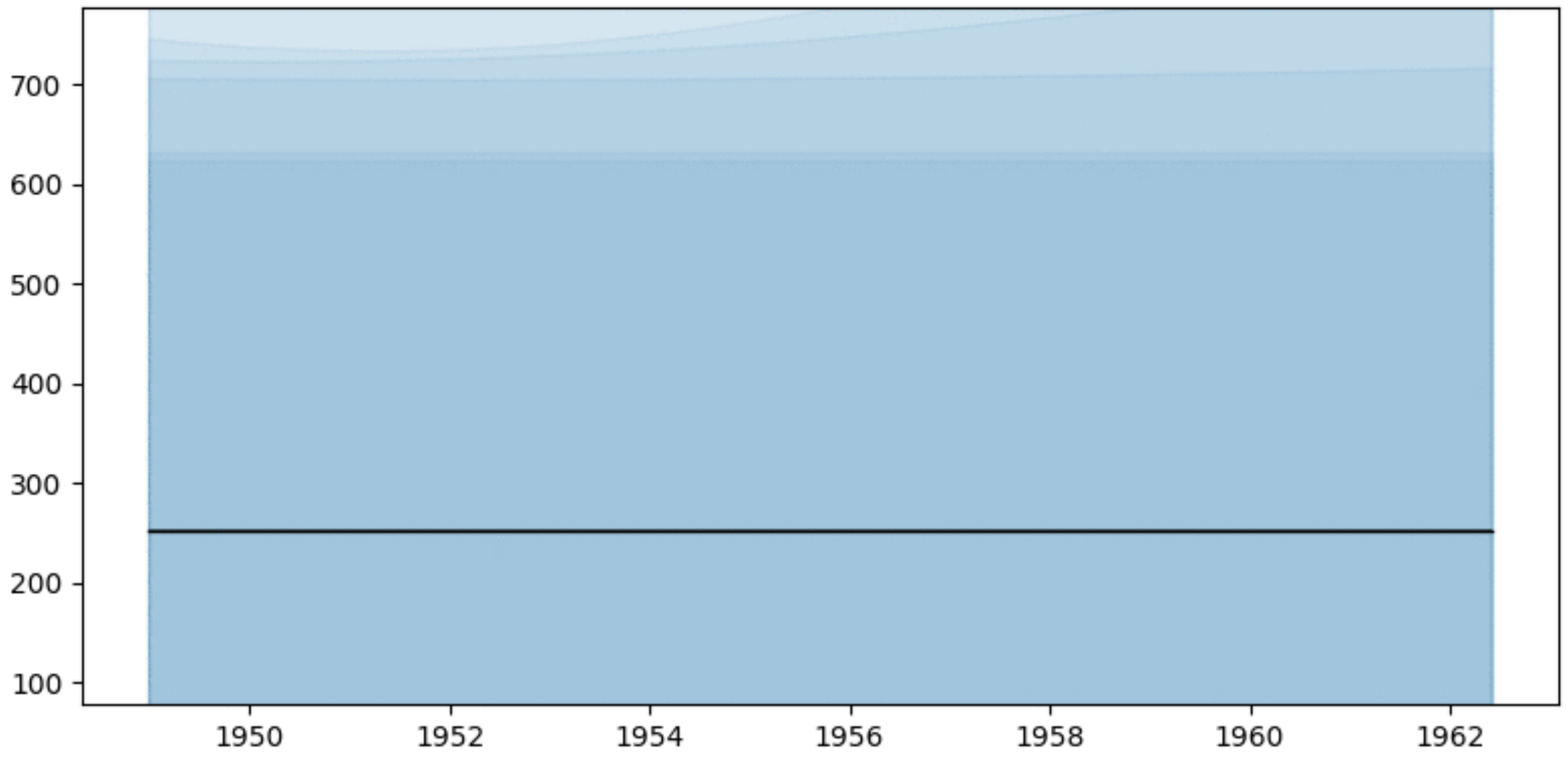
$$k_1 \cdot k_2$$

Multiplication

$$\alpha k_1$$

Positive scalar multiplication $\alpha > 0$

Kernels provide a lot of structure
(that can be automatically learned)



https://github.com/probsys/AutoGP.jl

Examples of kernels

Examples of kernels

$$k(x, x') = \sigma^2(x^T x' + b)$$

Linear

$$k(x, x') = \alpha \exp\left(-\frac{l\|x - x'\|^2}{2\sigma^2}\right)$$

Squared-exponential

$$k(x, x') = \exp\left(-\frac{2\sin^2(\pi\|x - x'\|/p)}{l^2}\right)$$

Periodic

2. We rely on distances to measure correlation

⚠ Curse of dimensionality

$$k(x, x') = \alpha \exp\left(-\frac{l\|x - x'\|^2}{2\sigma^2}\right)$$

Squared-exponential

$$k(x, x') = \exp\left(-\frac{2\sin^2(\pi\|x - x'\|/p)}{l^2}\right)$$

Periodic

Examples of kernels

Surrogate model

$$\tilde{f}(x) \sim \mathrm{GP}(\mu, k)$$

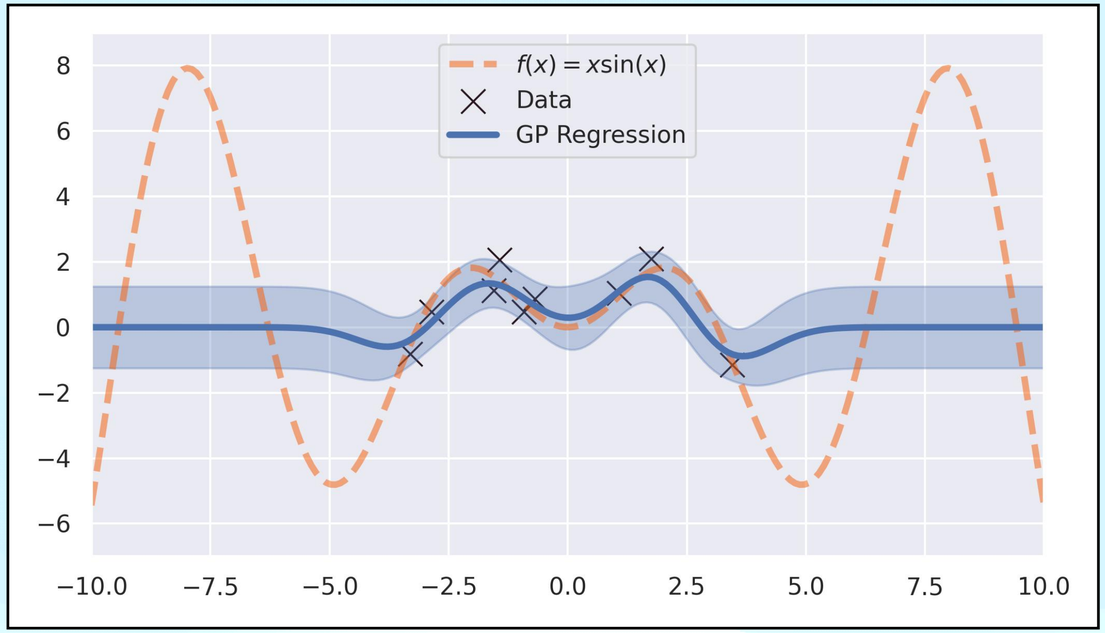
Gaussian Processes

Summary: GPs

Surrogate model

$$\tilde{f}(x) \sim \mathrm{GP}(\mu, k)$$

Gaussian Processes



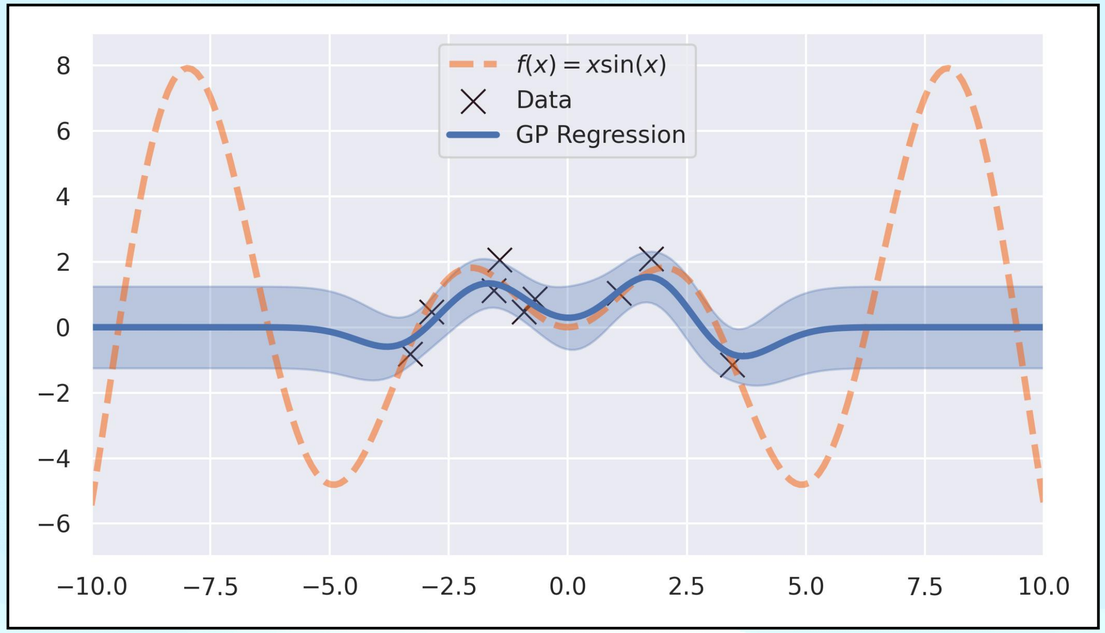
Regression with uncertainty given by k

Summary: GPs

Surrogate model

$$\tilde{f}(x) \sim \mathrm{GP}(\mu, k)$$

Gaussian Processes



Regression with uncertainty given by k

1. We need to invert a matrix of size N

2. We rely on distances to measure correlation

Summary: GP's

$\mathscr{D} = \emptyset$

While we have compute:

Fit the surrogate model $\tilde{f}$ on $\mathscr{D}$

Find the optima of the acq. function $\alpha(x; \tilde{f})$ and call it x_{next}

Evaluate the objective function f on x_{next}

Add $f(x_{\text{next}})$ to the dataset $\mathscr{D}$

Pseudocode for a BayesOpt implementation

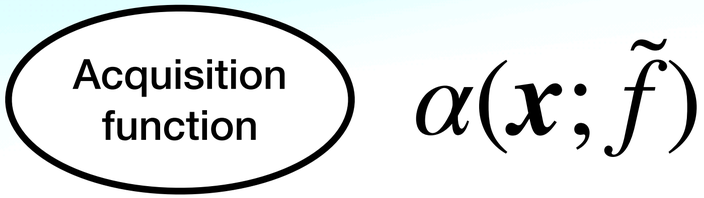
Evaluate the objective function (expensive)

Objective function $f(x)$

Surrogate model $\tilde{f}(x)$

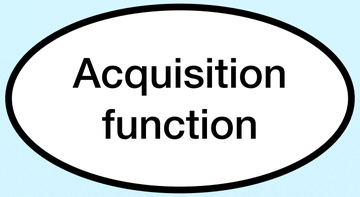
Acquisition function $\alpha(x; \tilde{f})$

Acquisition functions balance exploration and exploitation



$$\alpha(\boldsymbol{x}; \tilde{f})$$

Acquisition functions balance exploration and exploitation



$$\alpha(\boldsymbol{x}; \tilde{f})$$

Some examples:

$$\alpha(\boldsymbol{x}; \tilde{f}) = \text{Prob}[f(x) > f(x_b)]$$

(Prob. of improvement)

Acquisition functions balance exploration and exploitation

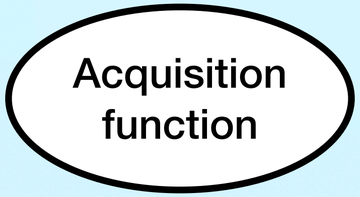
$$\alpha(\boldsymbol{x};\tilde{f})$$

Some examples:

$$\alpha(\boldsymbol{x};\tilde{f}) = \text{Prob}[f(x) > f(x_b)]$$ (Prob. of improvement)

$$\alpha(\boldsymbol{x};\tilde{f}) = \mathbb{E}[\max(0, f(x) - f(x_b))]$$ (Expected improvement)

Acquisition functions balance exploration and exploitation



$$\alpha(\boldsymbol{x}; \tilde{f})$$

Some examples:

$$\alpha(\boldsymbol{x}; \tilde{f}) = \text{Prob}[f(x) > f(x_b)]$$ (Prob. of improvement)

$$\alpha(\boldsymbol{x}; \tilde{f}) = \mathbb{E}[\max(0, f(x) - f(x_b))]$$ (Expected improvement)

$$\alpha(\boldsymbol{x}; \tilde{f}, \beta) = \mu_{*|1:N} + \beta\sigma_{*|1:N}$$ (Upper confidence bound)

Acquisition functions balance exploration and exploitation

Choice depends on your preferences and set-up

function

$\alpha(x; f)$

Some examples:

$$\alpha(\boldsymbol{x}; \tilde{f}) = \text{Prob}[f(x) > f(x_b)]$$ (Prob. of improvement)

$$\alpha(\boldsymbol{x}; \tilde{f}) = \mathbb{E}[\max(0, f(x) - f(x_b))]$$ (Expected improvement)

$$\alpha(\boldsymbol{x}; \tilde{f}, \beta) = \mu_{*|1:N} + \beta\sigma_{*|1:N}$$ (Upper confidence bound)

https://arxiv.org/abs/2310.20708

Acquisition functions balance exploration and exploitation

Choice depends on your preferences and set-up

UCB and log-Expected Improvement are good for starters

Some exa

$\alpha(\boldsymbol{x}; \tilde{f})$ mprovement)

$\alpha(\boldsymbol{x}; \tilde{f})$ improvement)

$\alpha(\boldsymbol{x}; \tilde{f}, \beta$ nfidence bound)

Unexpected Improvements to Expected Improvement for Bayesian Optimization

Sebastian Ament
Meta
ament@meta.com

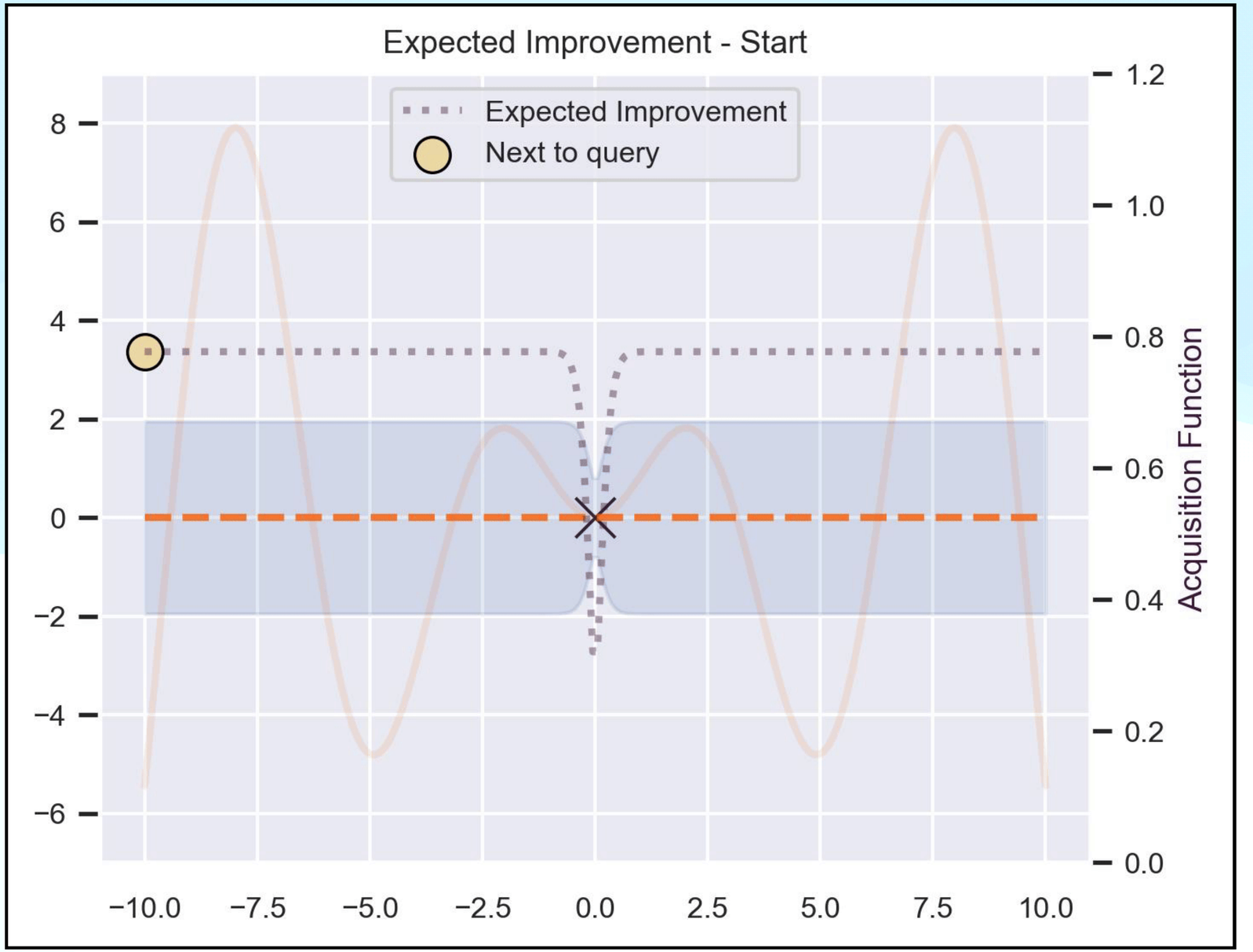
Samuel Daulton
Meta
sdaulton@meta.com

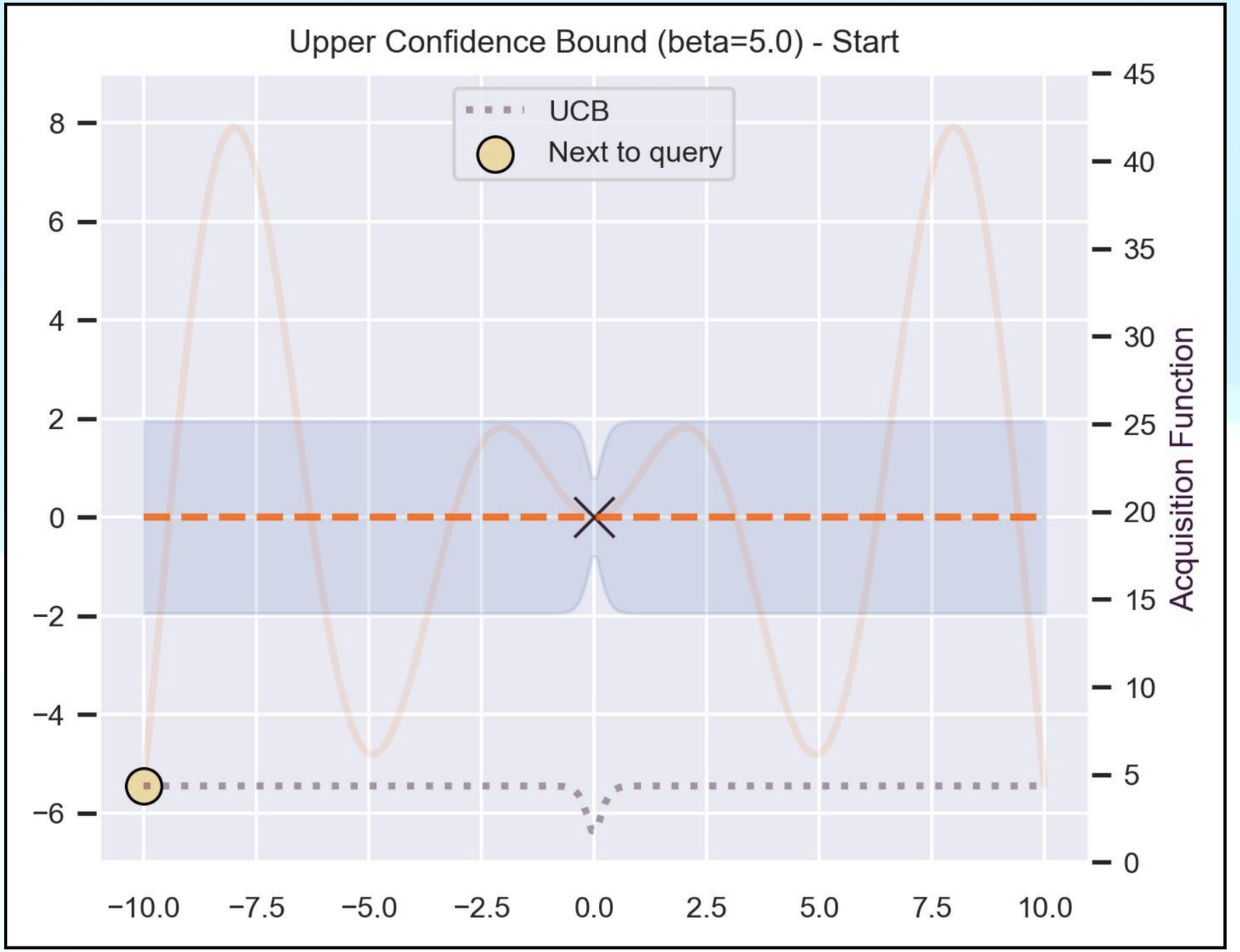
David Eriksson
Meta
deriksson@meta.com

Maximilian Balandat
Meta
balandat@meta.com

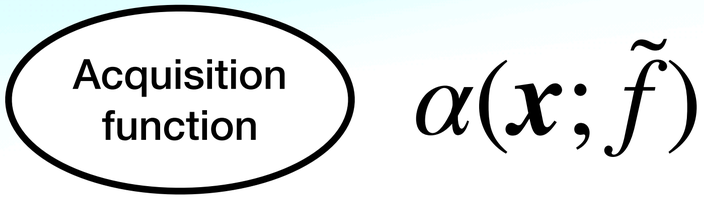
Eytan Bakshy
Meta
ebakshy@meta.com

https://arxiv.org/abs/2310.20708


Expected Improvement - Start
Expected Improvement
Next to query
8
6
4
2
0
−2
−4
−6
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Acquisition Function
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

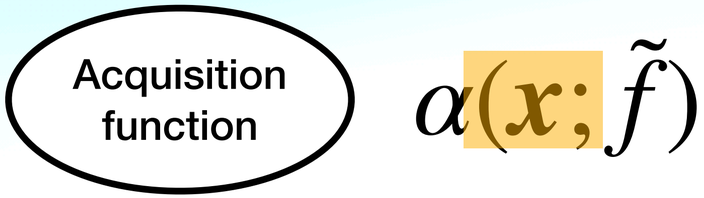

Upper Confidence Bound (beta=5.0) - Start
UCB
Next to query
8
6
4
2
0
−2
−4
−6
45
40
35
30
25
20
15
10
5
0
Acquisition Function
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

Acquisition functions balance exploration and exploitation



$$\alpha(\boldsymbol{x}; \tilde{f})$$

Acquisition functions balance exploration and exploitation



Acquisition functions balance exploration and exploitation

Acquisition function

$$\alpha(\boldsymbol{x}; \tilde{f})$$

3. Even if α is easy to query, we'll have to optimize a high-dimensional function.

1. We need to invert a matrix of size N

2. We rely on distances to measure correlation

3. Even if α is easy to query, we'll have to optimize a high-dimensional function.

Three reasons why BO doesn't scale with dimension

1. We need to invert a matrix of size N

Folk knowledge:

"GPs don't scale beyond 20 input dimensions"

3. Even if α is easy to query, we'll have to optimize a high-dimensional function.

Three reasons why BO doesn't scale with dimension

1. We need to invert a matrix of size N

Folk knowledge:

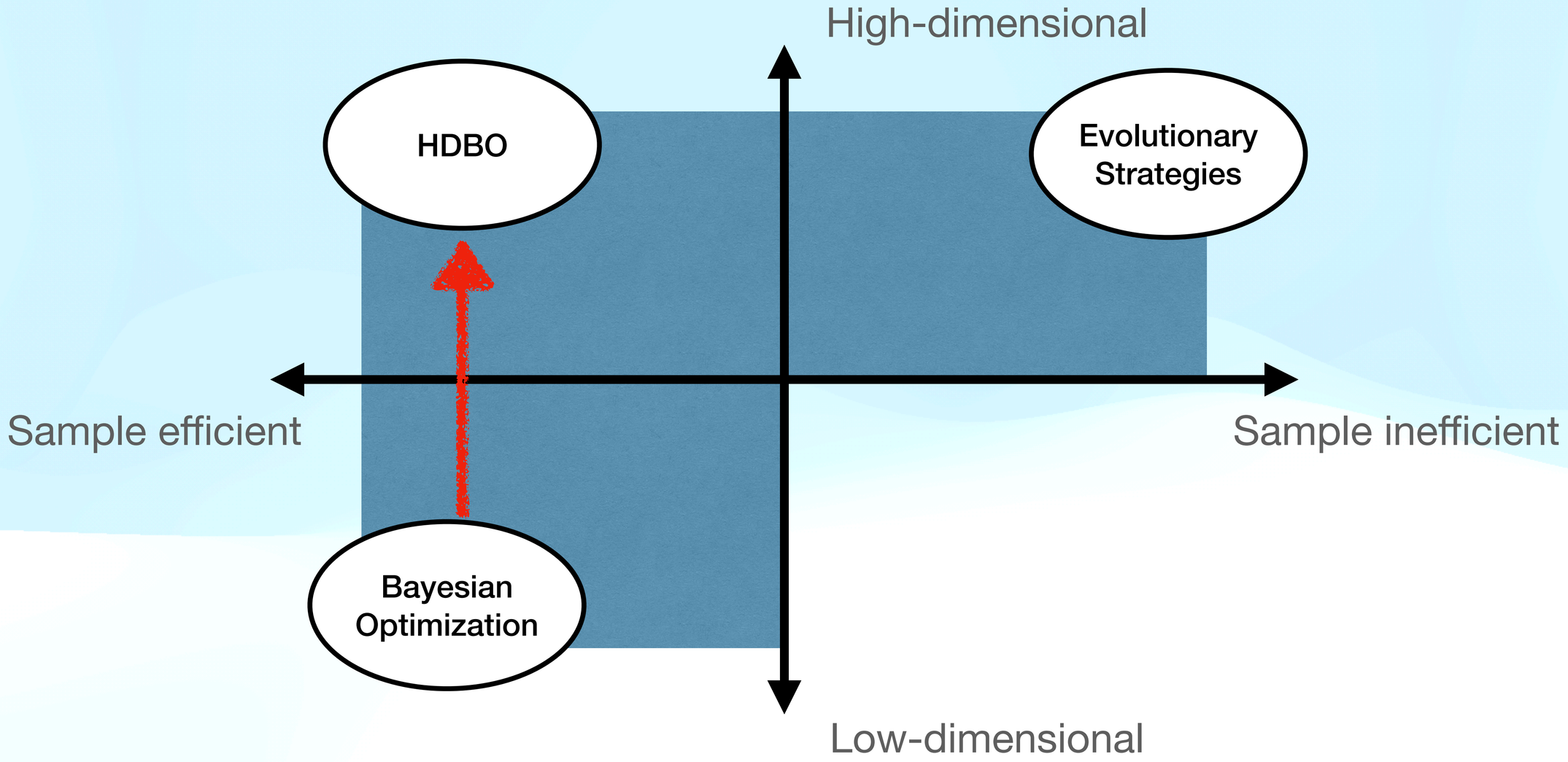
"GPs don't scale beyond 20 input dimensions"

This has been disputed recently.

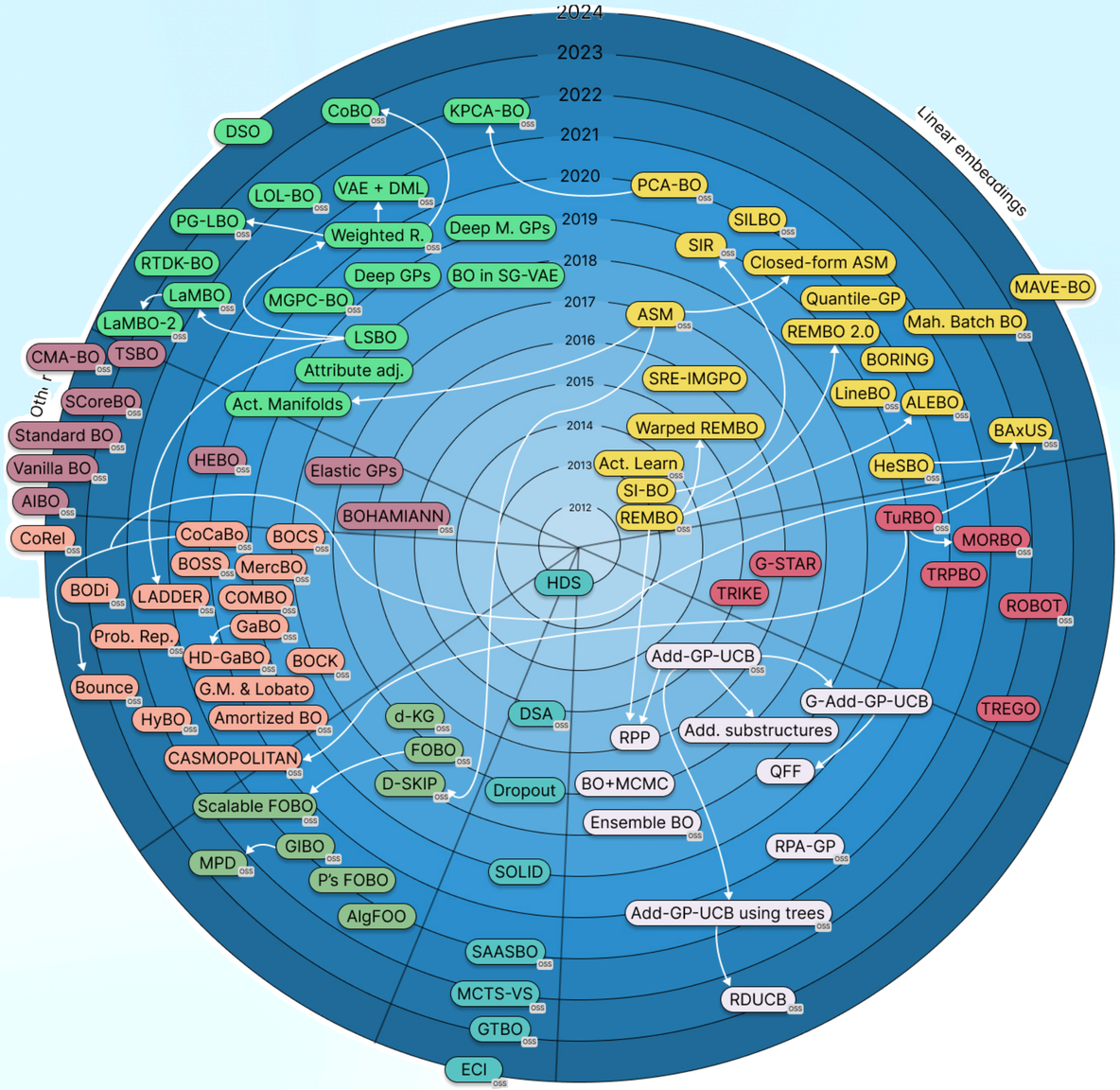
High-dimensional BO methods are being proposed

Three reasons why BO doesn't scale with dimension

High-dimensional Bayesian optimization



Black-box optimization algorithms



miguelgondu.com/assets/hdbo_timeline.pdf

A survey and benchmark of high-dimensional Bayesian optimization of discrete sequences

Miguel González-Duque*
University of Copenhagen

Richard Michael
University of Copenhagen

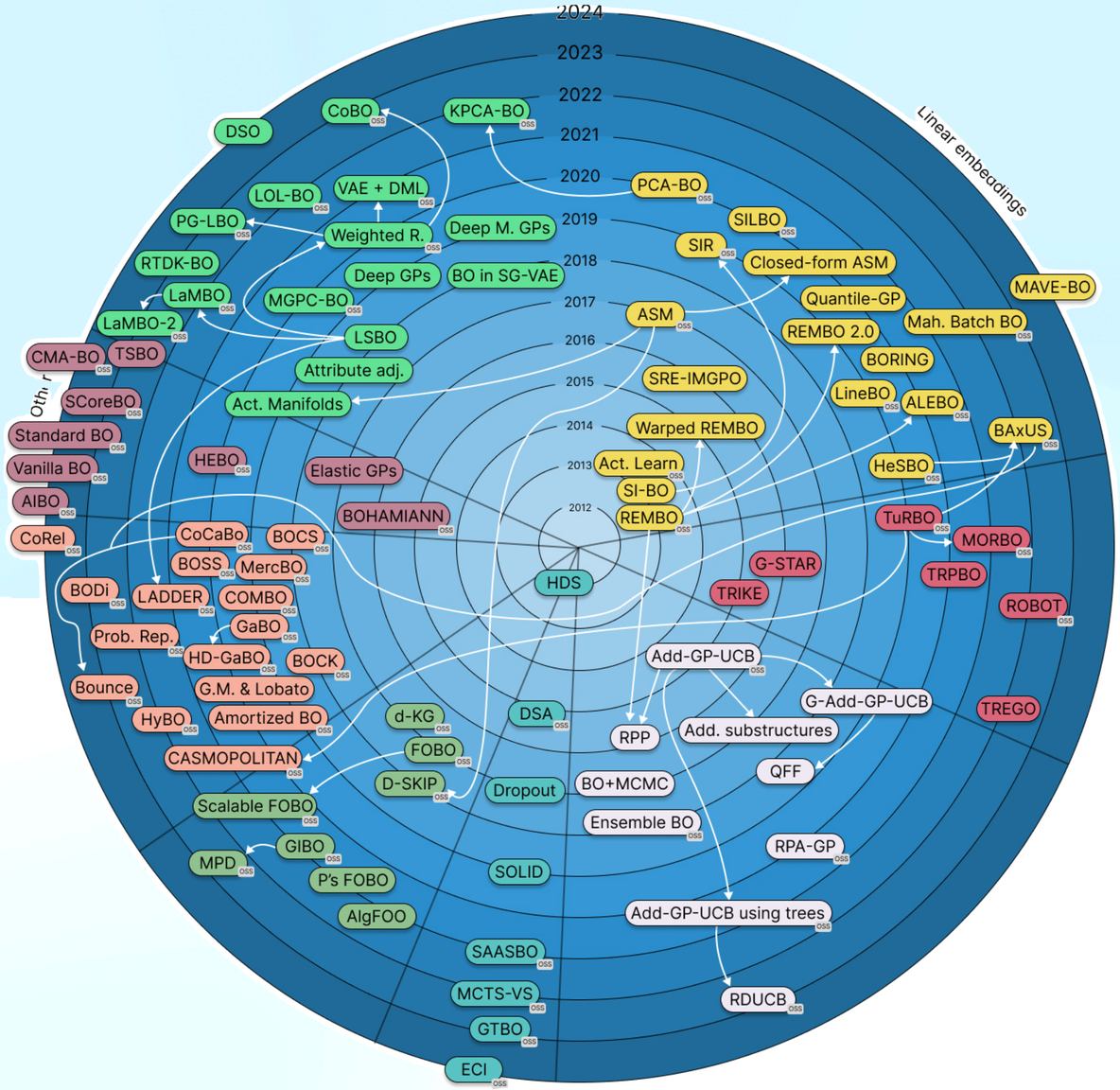
Simon Bartels
Paul Sabatier University Tolouse

Yevgen Zainchkovskyy
Technical University of Denmark

Søren Hauberg
Technical University of Denmark

Wouter Boomsma
University of Copenhagen

https://arxiv.org/abs/2406.04739



A survey and benchmark of high-dimensional Bayesian optimization of discrete sequences

Miguel González-Duque*
University of Copenhagen

Richard Michael
University of Copenhagen

Simon Bartels
Paul Sabatier University Tolouse

Yevgen Zainchkovskyy
Technical University of Denmark

Søren Hauberg
Technical University of Denmark

Wouter Boomsma
University of Copenhagen

We have been exploring HDBO methods, and we would like to propose a new **taxonomy** of 7 different families, expanding previous work.

Comparison of High-Dimensional Bayesian Optimization Algorithms on BBOB
https://arxiv.org/abs/2303.00890

A survey on high-dimensional Gaussian process modeling with applications to Bayesian optimization
https://hal.science/hal-03419959/

Variable selection

Additive models

Linear embeddings

Gradient information

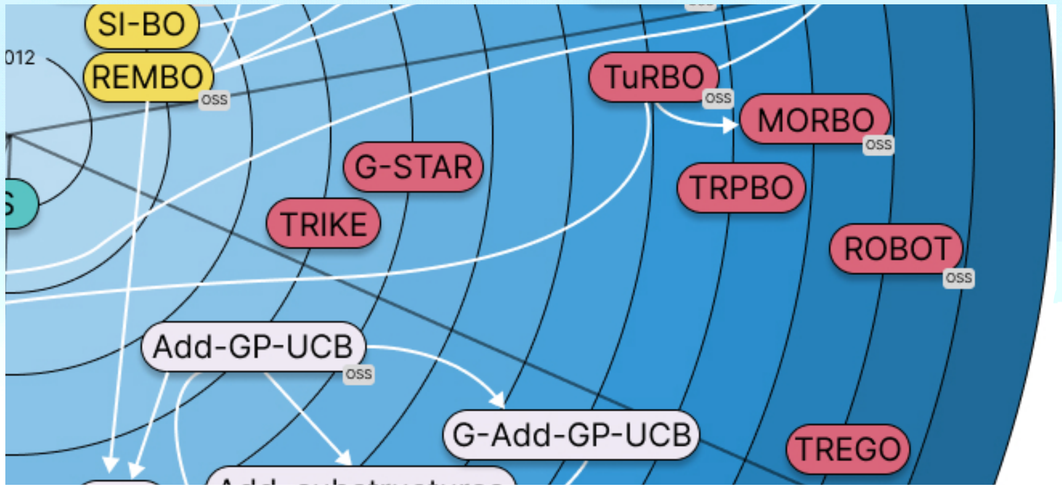
Non-linear embeddings

Structured spaces

Trust regions

A taxonomy of 7

Trust regions

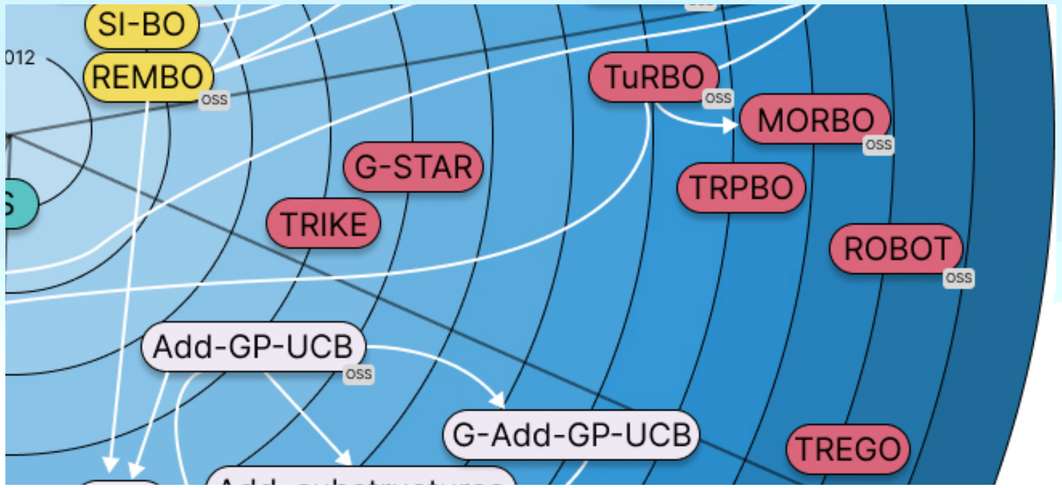


A taxonomy of 7

Trust regions

Core idea:

Optimize the acquisition function in a tiny hypercube in $\mathbb{R}^D$.



A taxonomy of 7

Trust regions

Scalable Global Optimization via Local Bayesian Optimization

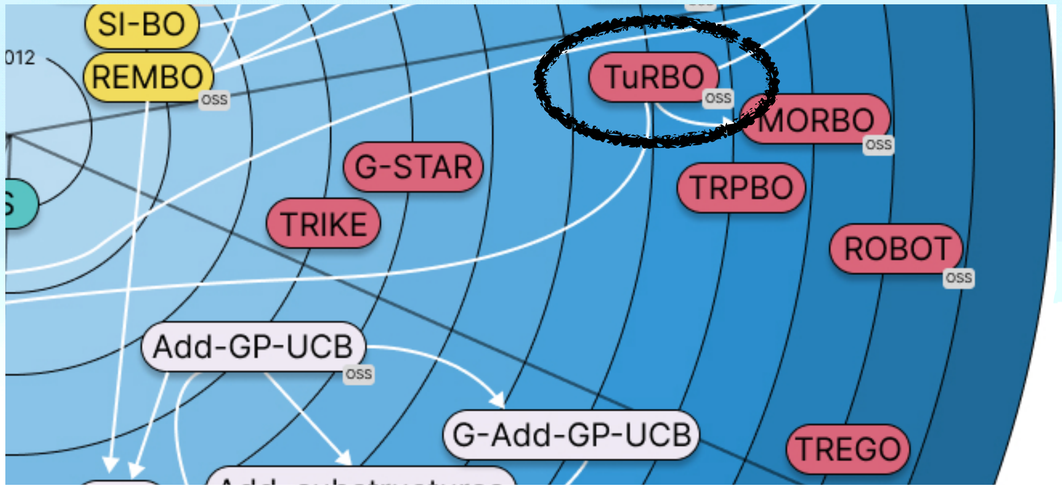
David Eriksson
Uber AI
eriksson@uber.com

Michael Pearce
University of Warwick
m.a.l.pearce@warwick.ac.uk

Jacob R Gardner
Uber AI
jake.gardner@uber.com

Ryan Turner
Uber AI
ryan.turner@uber.com

Matthias Poloczek
Uber AI
poloczek@uber.com



A taxonomy of 7

Trust regions

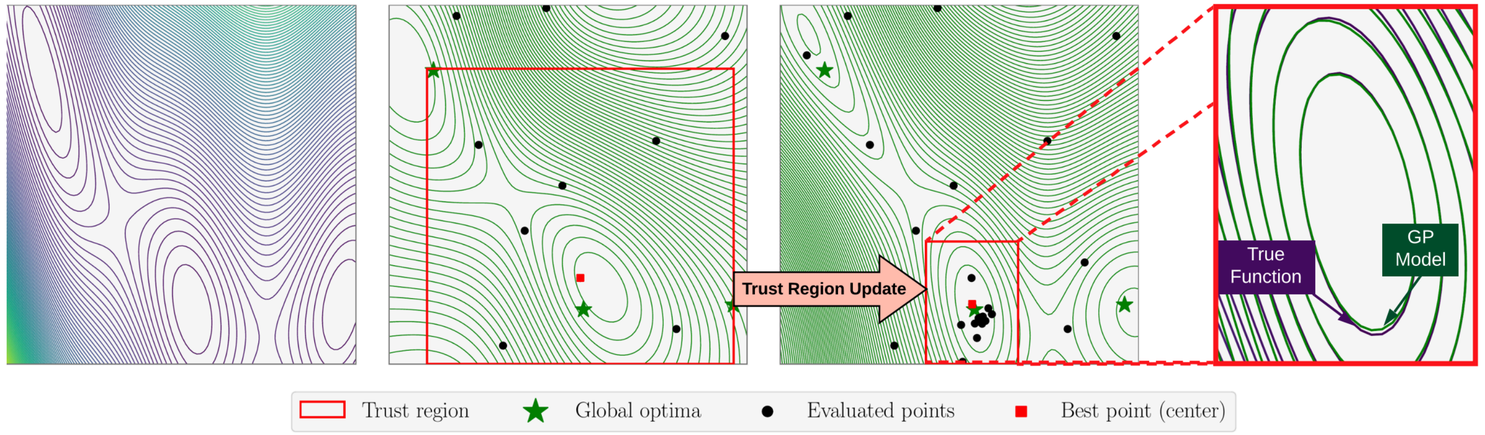
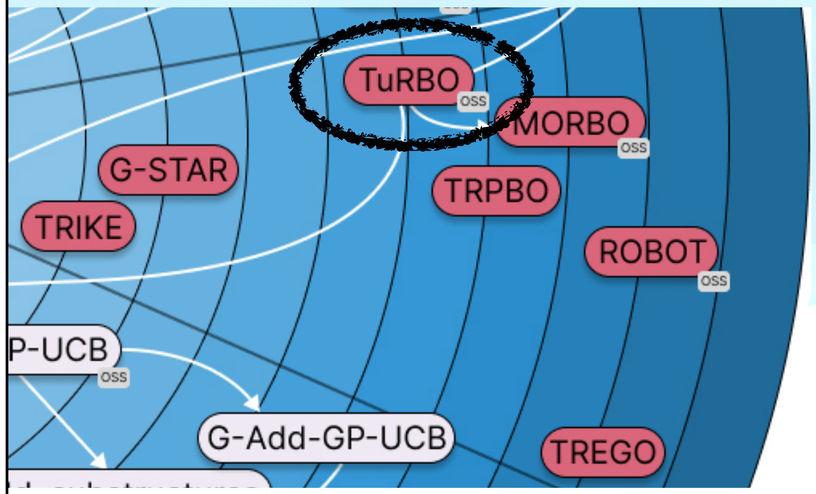
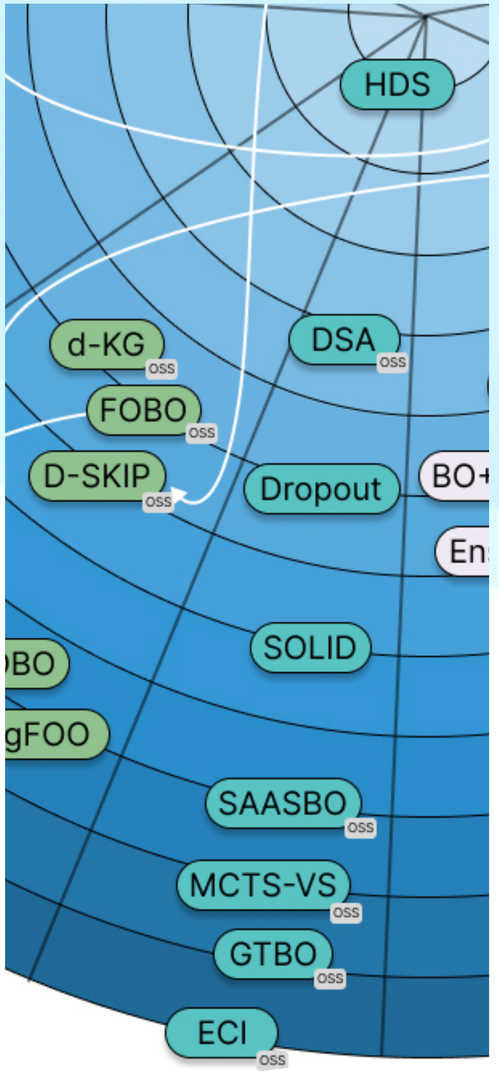


Figure 1: Illustration of the TuRBO algorithm. **(Left)** The true contours of the Branin function. **(Middle left)** The contours of the GP model fitted to the observations depicted by black dots. The current TR is shown as a red square. The global optima are indicated by the green stars. **(Middle right)** During the execution of the algorithm, the TR has moved towards the global optimum and has reduced in size. The area around the optimum has been sampled more densely in effect. **(Right)** The local GP model almost exactly fits the underlying function in the TR, despite having a poor global fit.



A taxonomy of 7

Variable selection

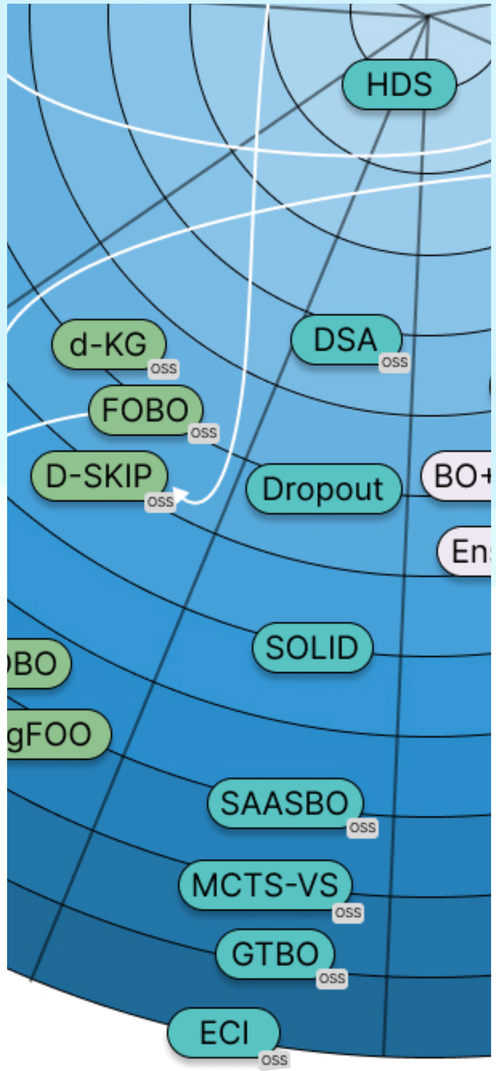


A taxonomy of 7

Variable selection

Core assumption:

Most of the input variables have almost no effect on the objective



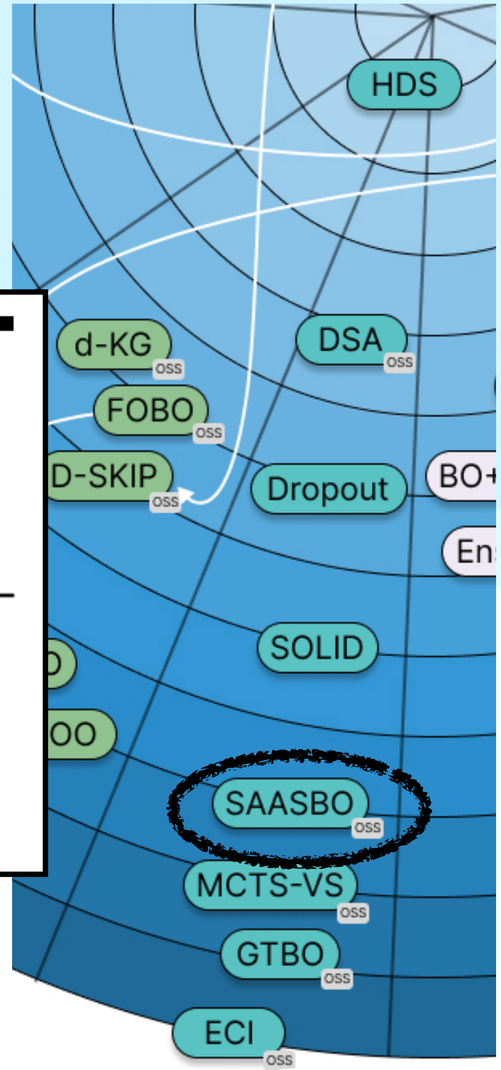
A taxonomy of 7

Variable selection

High-Dimensional Bayesian Optimization with Sparse Axis-Aligned Subspaces

David Eriksson[*,1] **Martin Jankowiak**[*,2]

[1]Facebook, Menlo Park, California, USA
[2]Broad Institute of Harvard and MIT, Cambridge, Massachusetts, USA

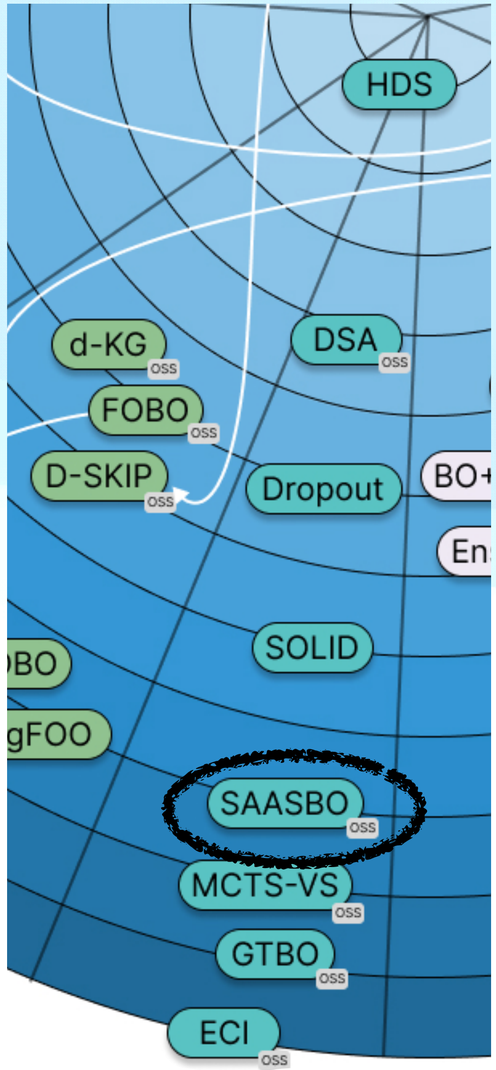


A taxonomy of 7

Variable selection

4.1 SAAS FUNCTION PRIOR

$$
\begin{aligned}
&\text{[kernel variance]} && \sigma_k^2 \sim \mathcal{LN}(0, 10^2) && (8)\\
&\text{[global shrinkage]} && \tau \sim \mathcal{HC}(\alpha) &&\\
&\text{[length scales]} && \rho_i \sim \mathcal{HC}(\tau) && \text{for } \; i = 1, ..., D.\\
&\text{[function values]} && \mathbf{f} \sim \mathcal{N}(\mathbf{0}, K_{\mathbf{XX}}^{\psi}) && \text{with } \; \psi = \{\rho_{1:d}, \sigma_k^2\}\\
&\text{[observations]} && \mathbf{y} \sim \mathcal{N}(\mathbf{f}, \sigma^2 \mathbb{1}_N) &&
\end{aligned}
$$



A taxonomy of 7

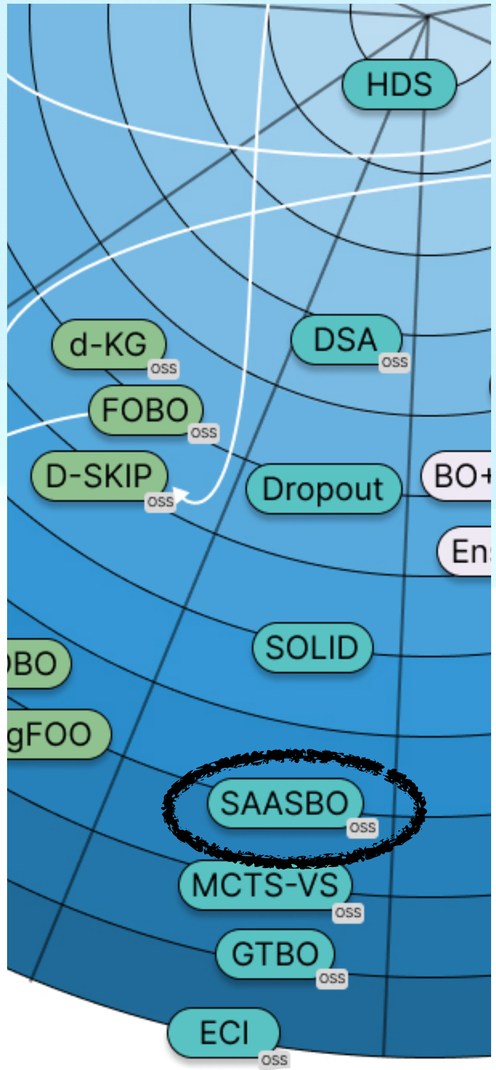
Variable selection

4.1 SAAS FUNCTION PRIOR

$$
\begin{aligned}
&\text{[kernel variance]} && \sigma_k^2 \sim \mathcal{LN}(0, 10^2) && (8)\\
&\text{[global shrinkage]} && \tau \sim \mathcal{HC}(\alpha) &&\\
&\text{[length scales]} && \rho_i \sim \mathcal{HC}(\tau) && \text{for } i = 1, ..., D.\\
&\text{[function values]} && \mathbf{f} \sim \mathcal{N}(\mathbf{0}, K_{\mathbf{XX}}^{\psi}) && \text{with } \psi = \{\rho_{1:d}, \sigma_k^2\}\\
&\text{[observations]} && \mathbf{y} \sim \mathcal{N}(\mathbf{f}, \sigma^2 \mathbb{1}_N) &&
\end{aligned}
$$

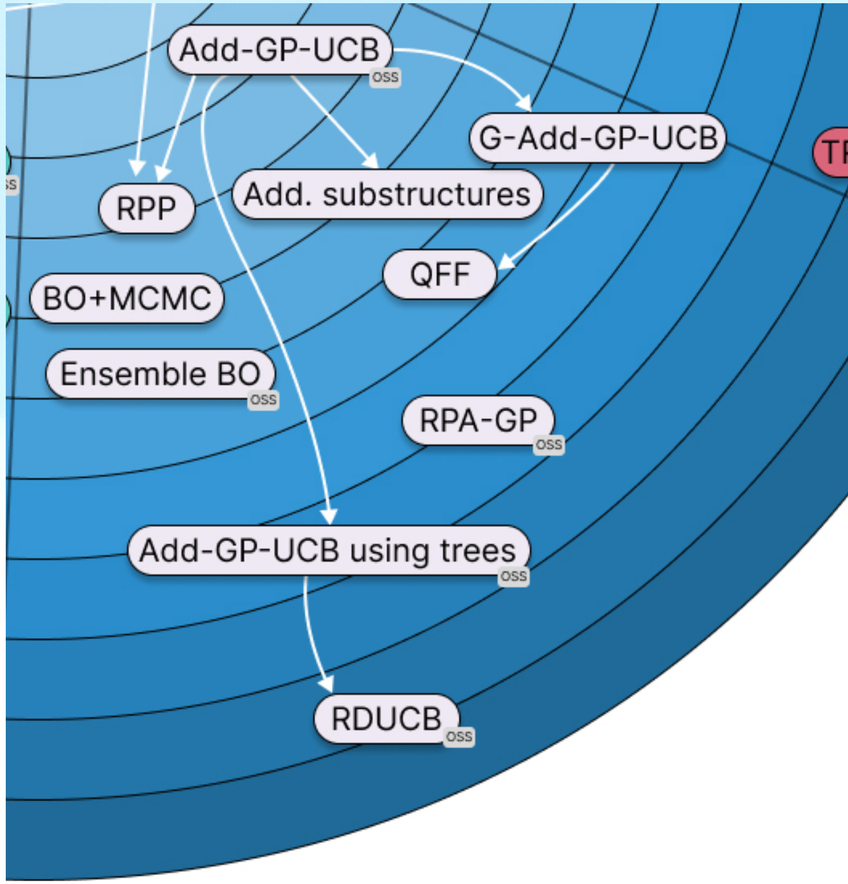
- Regularizing the training loss of GPs to encourage axis-aligned sparcity.

- Training GPs in a fully Bayesian way.



A taxonomy of 7

Additive models



A taxonomy of 7

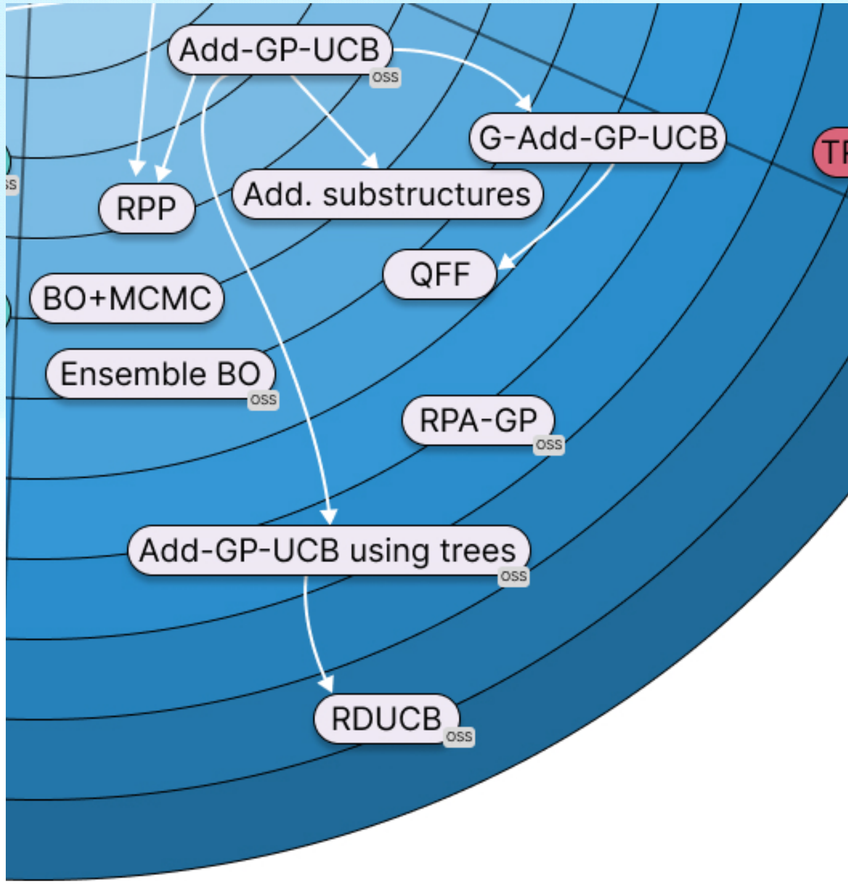
Additive models

Core assumption:

f can be decomposed as

$$f(\boldsymbol{x}) = f(x_1, \ldots, x_D)$$
$$= f^{(1)}(\{x_i\}_1) + \cdots + f^{(G)}(\{x_j\}_G)$$

Where the subsets are potentially disjoint



A taxonomy of 7

Additive models

Key structural assumption: In order to make progress in high dimensions, we will assume that f decomposes into the following additive form,

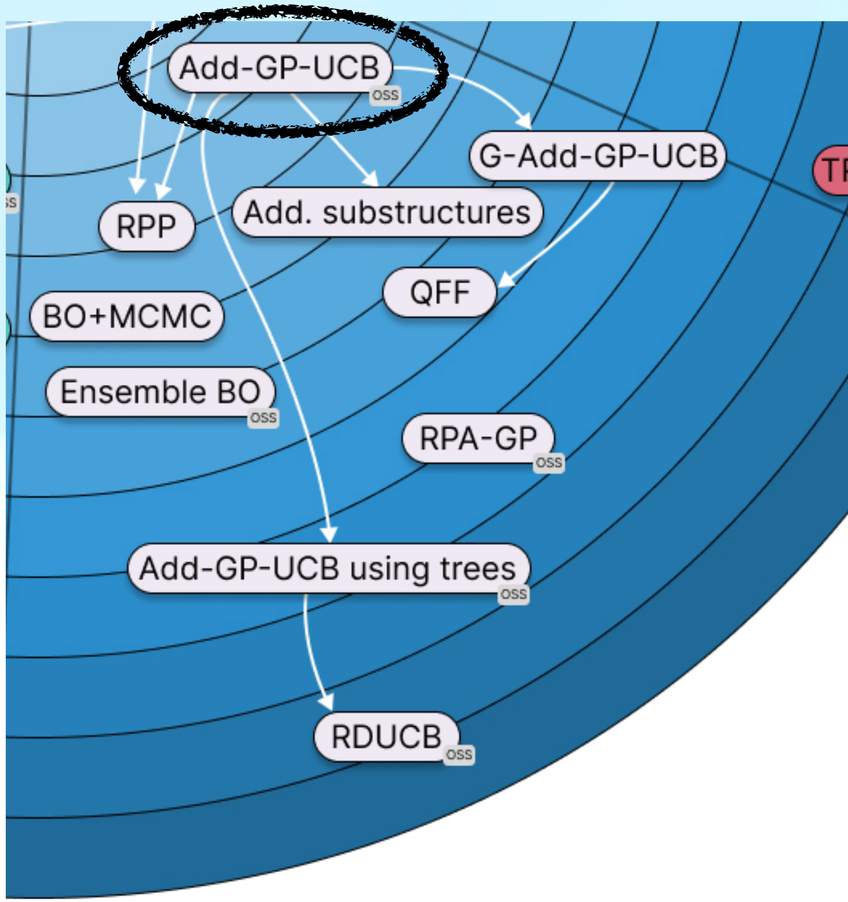
$$f(x) = f^{(1)}(x^{(1)}) + f^{(2)}(x^{(2)}) + \cdots + f^{(M)}(x^{(M)}). \quad (1)$$

Here each $x^{(j)} \in \mathcal{X}^{(j)} = [0,1]^{d_j}$ are lower dimensional components. We will refer to the $\mathcal{X}^{(j)}$'s as "groups" and

$$\mu(x) = \mu^{(1)}(x^{(1)}) + \cdots + \mu^{(M)}(x^{(M)}) \quad (4)$$
$$\kappa(x, x') = \kappa^{(1)}(x^{(1)}, x^{(1)\prime}) + \cdots + \kappa^{(M)}(x^{(M)}, x^{(M)\prime}).$$

an additive kernel. We define the *Additive Gaussian Process Upper Confidence Bound* (**Add-GP-UCB**) to be

$$\widetilde{\varphi}_t(x) = \mu_{t-1}(x) + \beta_t^{1/2} \sum_{j=1}^{M} \sigma_{t-1}^{(j)}(x^{(j)}). \quad (6)$$

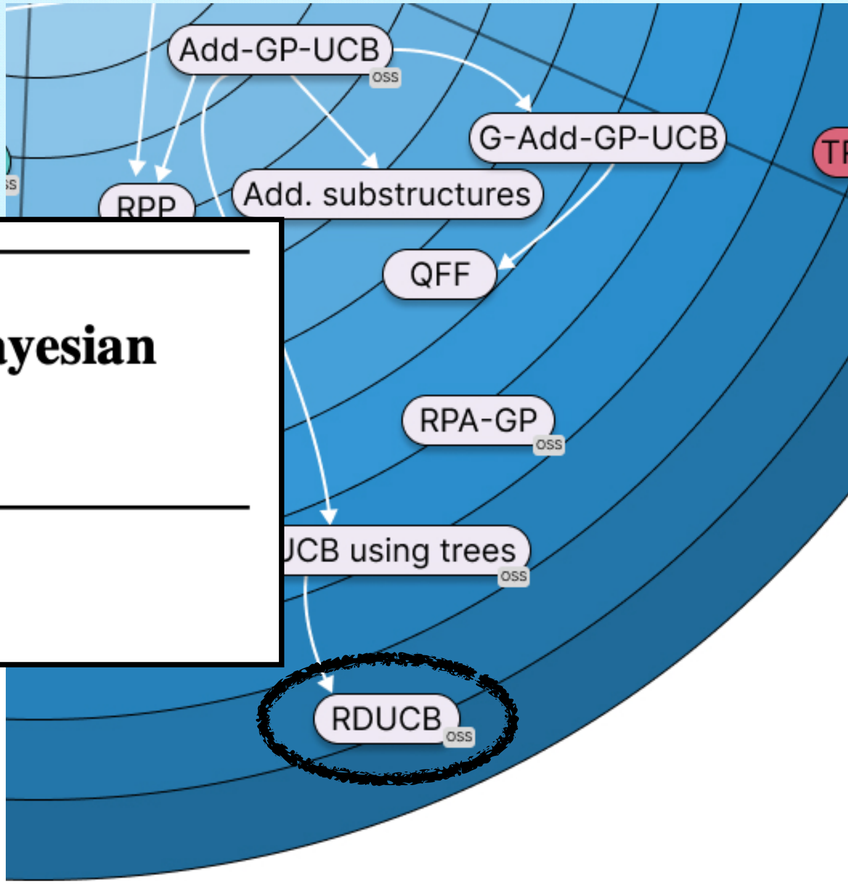


A taxonomy of 7

Additive models

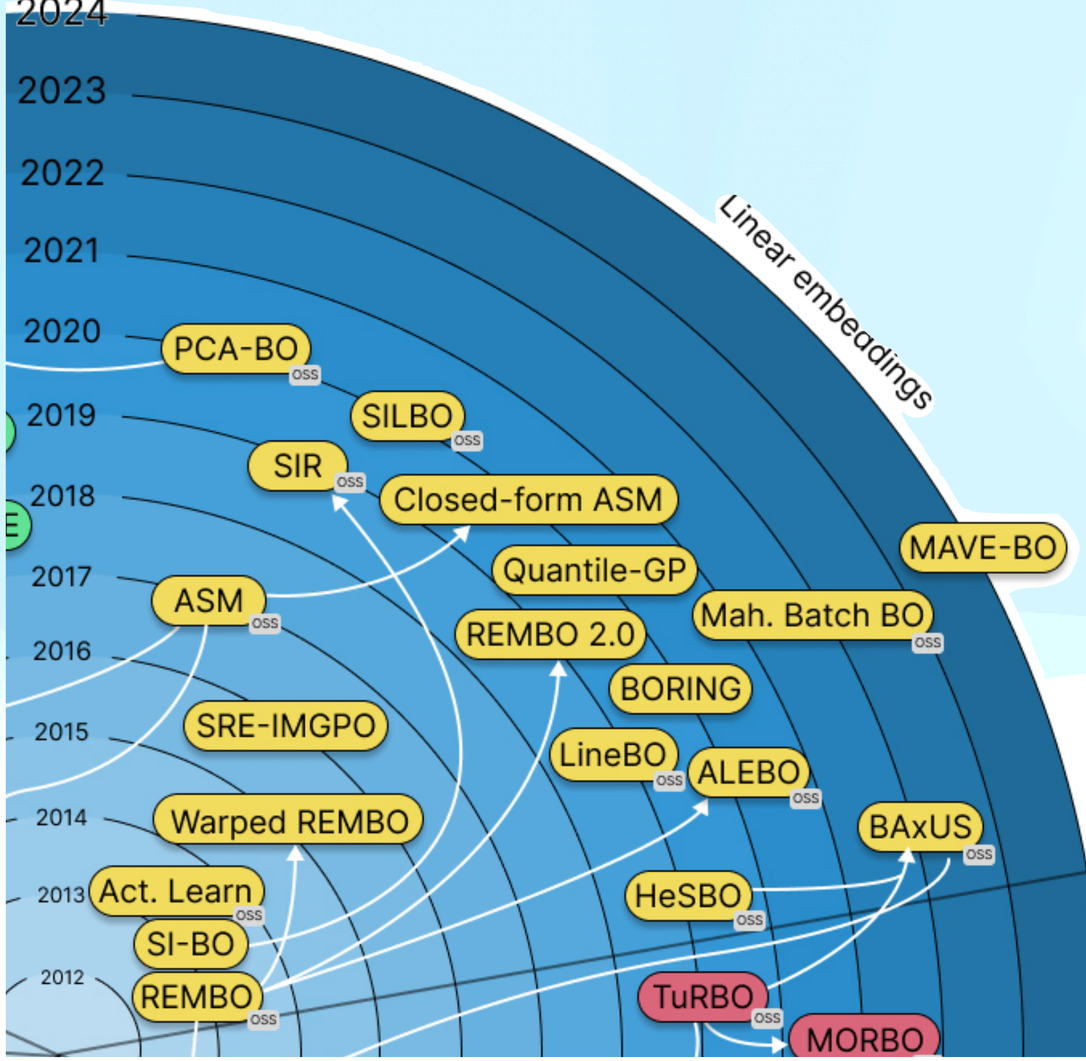
Are Random Decompositions all we need in High Dimensional Bayesian Optimisation?

Juliusz Ziomek [1] **Haitham Bou-Ammar** [1]



A taxonomy of 7

Linear embeddings

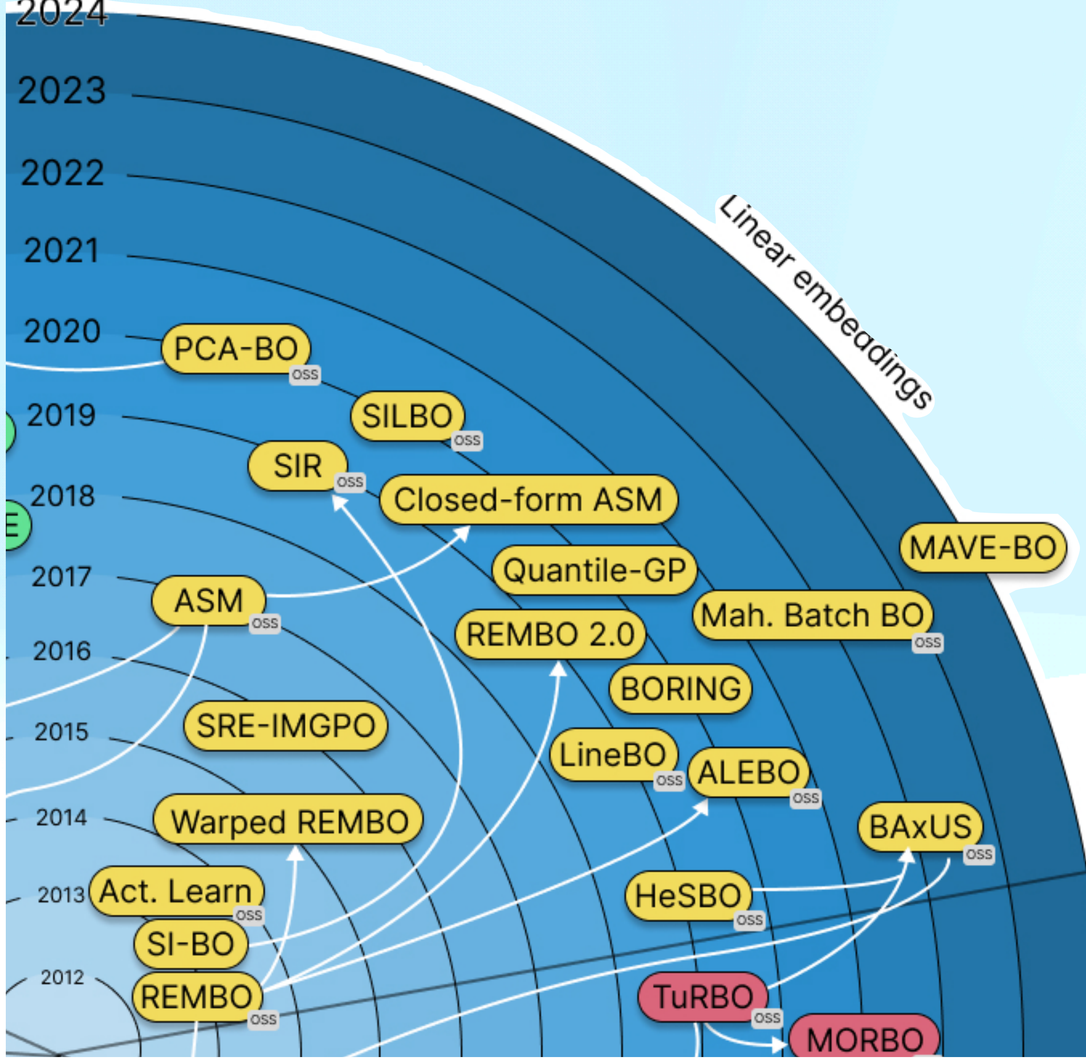


A taxonomy of 7

Linear embeddings

Core assumption:

The variables that have an impact are a linear subspace of $\mathbb{R}^D$.



A taxonomy of 7

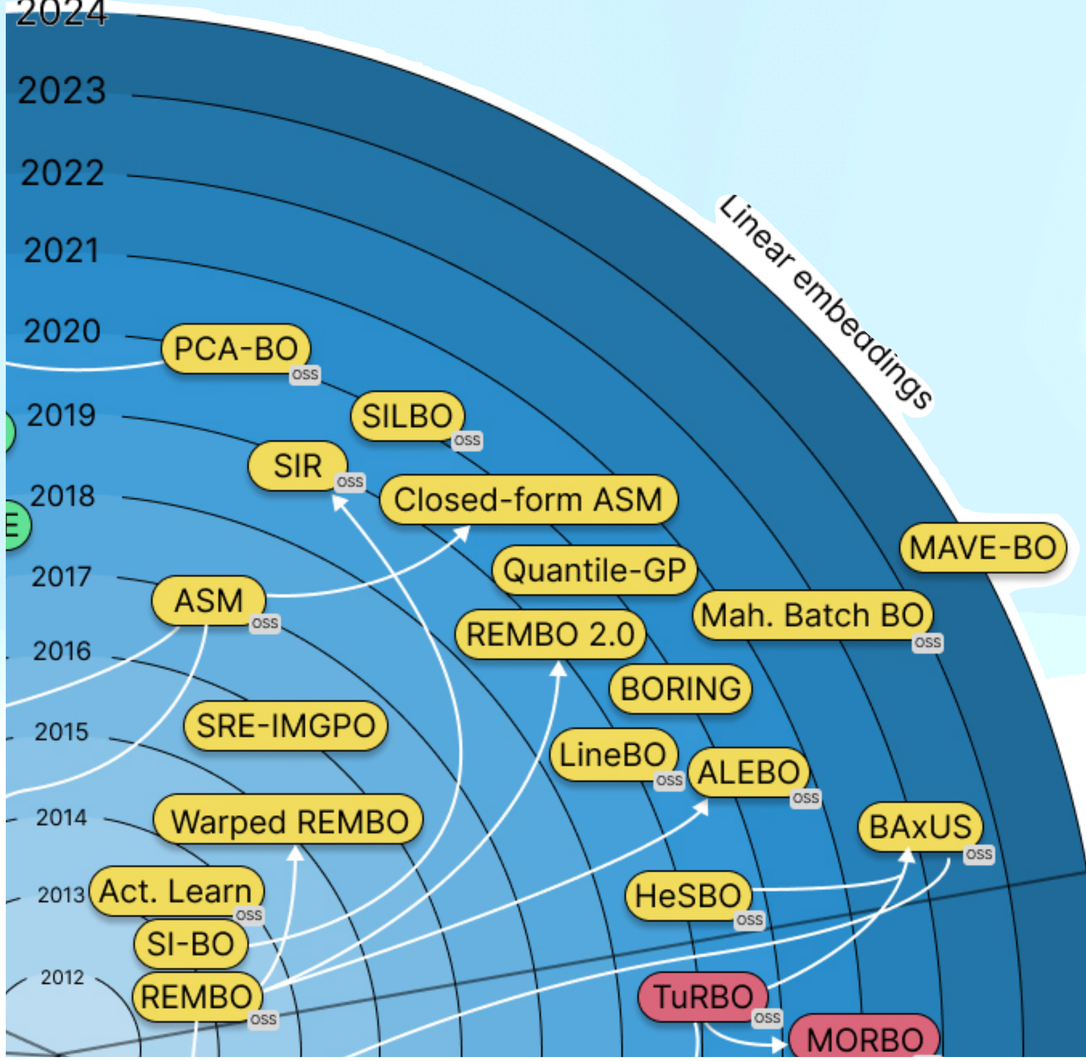
Linear embeddings

Core assumption:

The variables that have an impact are a linear subspace of $\mathbb{R}^D$.

In practice:

Optimize $z \in \mathbb{R}^d$ by choosing $A \in \mathbb{R}^{D \times d}$ and evaluating $f(Az)$



A taxonomy of 7

Linear embeddings

Bayesian Optimization in a Billion Dimensions via Random Embeddings

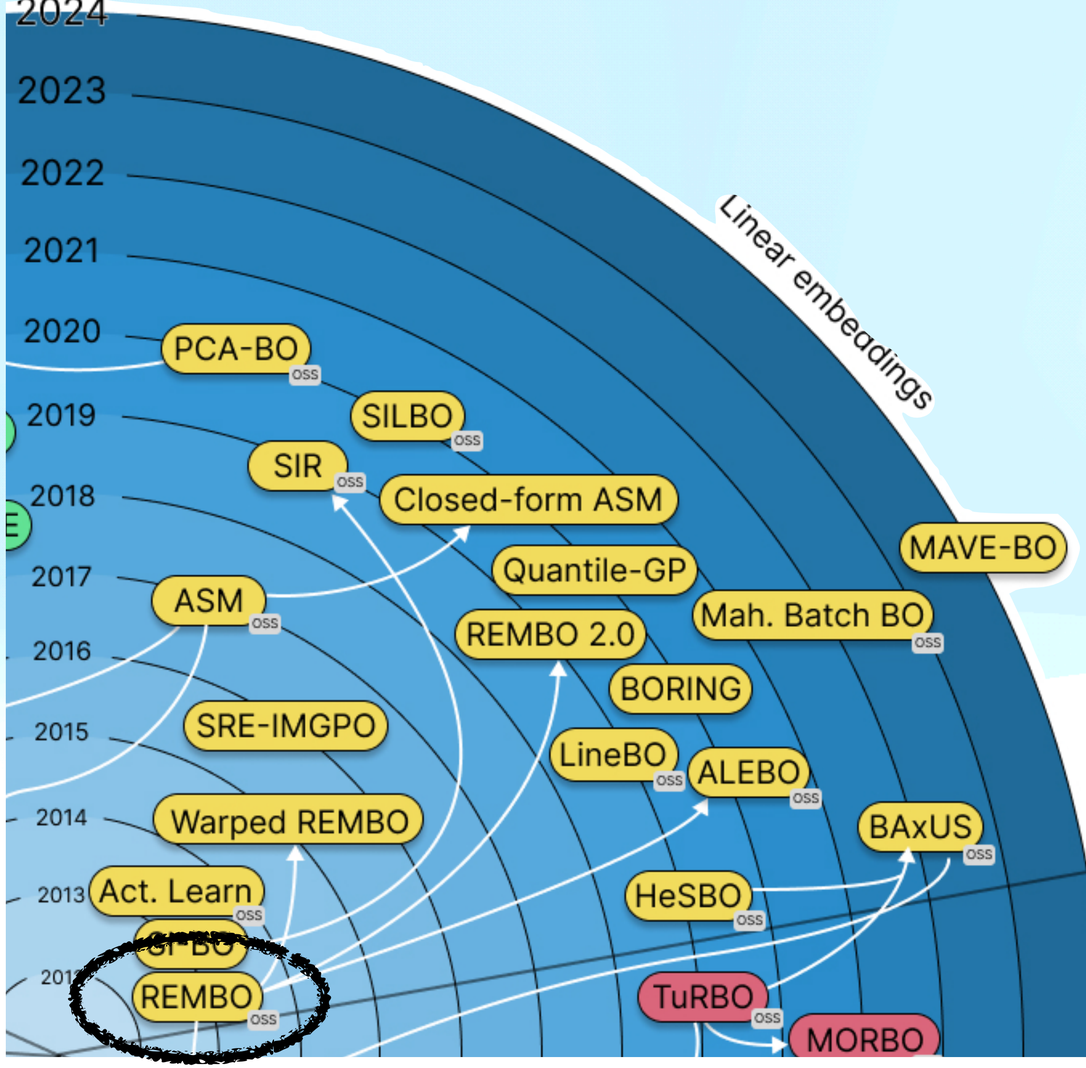
Ziyu Wang ZIYU.WANG@CS.OX.AC.UK
Department of Computer Science, University of Oxford

Frank Hutter FH@CS.UNI-FREIBURG.DE
Department of Computer Science, University of Freiburg

Masrour Zoghi M.ZOGHI@UVA.NL
Department of Computer Science, University of Amsterdam

David Matheson DAVIDM@CS.UBC.CA
Department of Computer Science, University of British Columbia

Nando de Freitas NANDO@CS.OX.AC.UK
Department of Computer Science, University of Oxford
Canadian Institute for Advanced Research

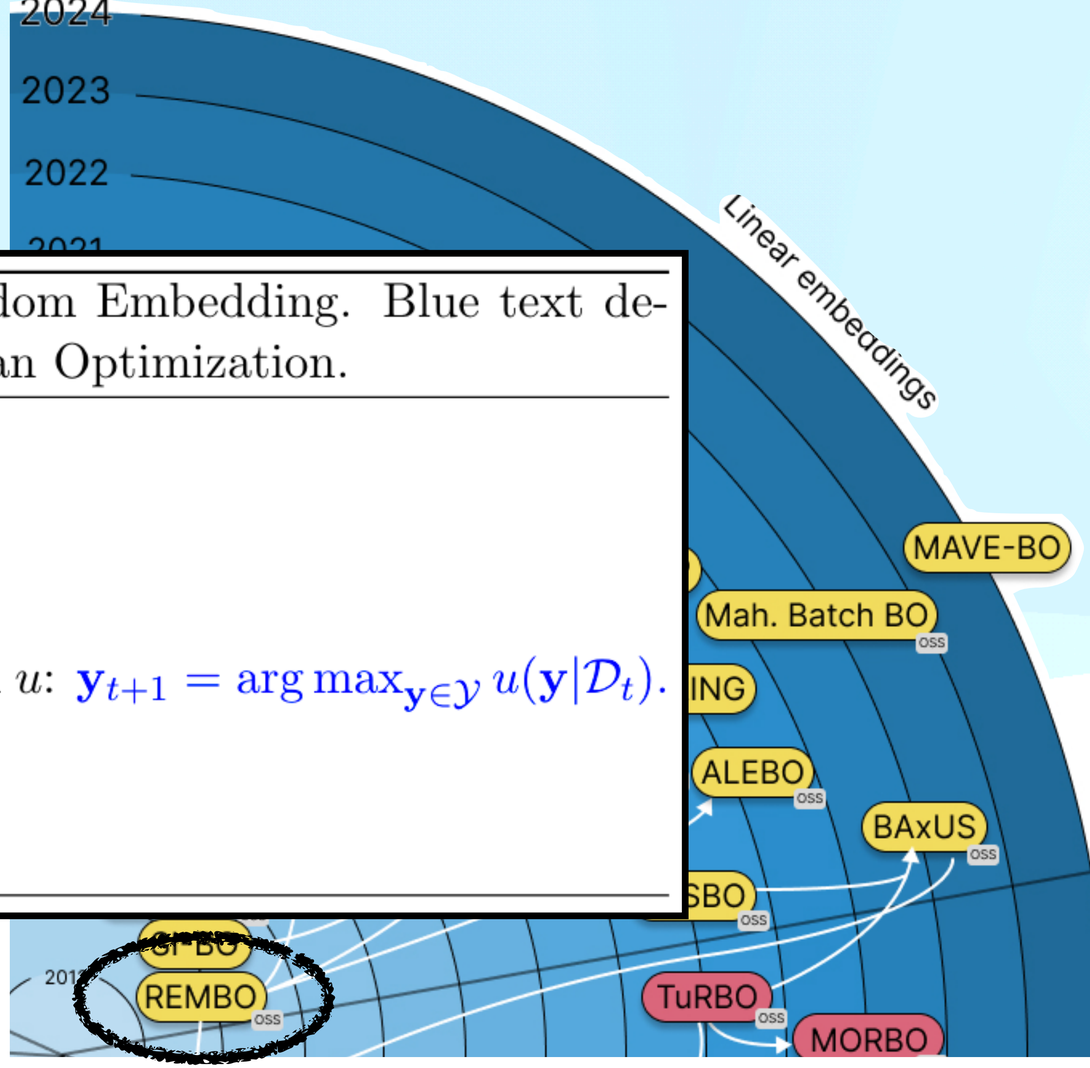


A taxonomy of 7

Linear embeddings

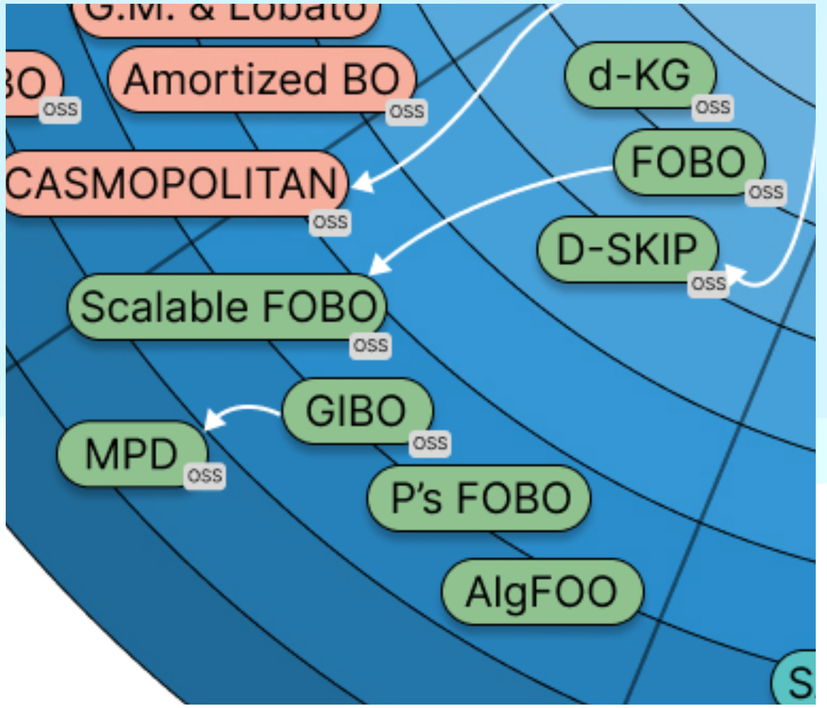
Algorithm 2 REMBO: Bayesian Optimization with Random Embedding. Blue text denotes parts that are changed compared to standard Bayesian Optimization.

1: Generate a random matrix $\mathbf{A} \in \mathbb{R}^{D \times d}$
2: Choose the bounded region set $\mathcal{Y} \subset \mathbb{R}^d$
3: Initialize $\mathcal{D}_0$ as $\emptyset$.
4: **for** $t = 1, 2, \ldots$ **do**
5: Find $\mathbf{y}_{t+1} \in \mathbb{R}^d$ by optimizing the acquisition function u: $\mathbf{y}_{t+1} = \arg\max_{\mathbf{y} \in \mathcal{Y}} u(\mathbf{y} | \mathcal{D}_t)$.
6: Augment the data $\mathcal{D}_{t+1} = \mathcal{D}_t \cup \{(\mathbf{y}_{t+1}, f(\mathbf{A}\mathbf{y}_{t+1}))\}$.
7: Update the kernel hyper-parameters.
8: **end for**



A taxonomy of 7

Gradient information

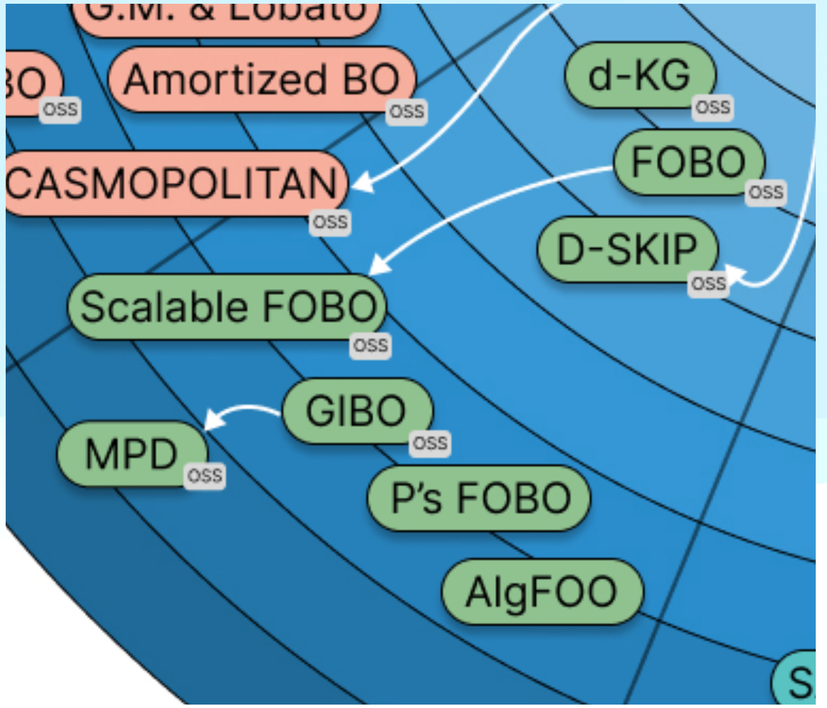


A taxonomy of 7

Gradient information

Core assumption:

Suppose your function is differentiable, then predict the gradient using GPs



A taxonomy of 7

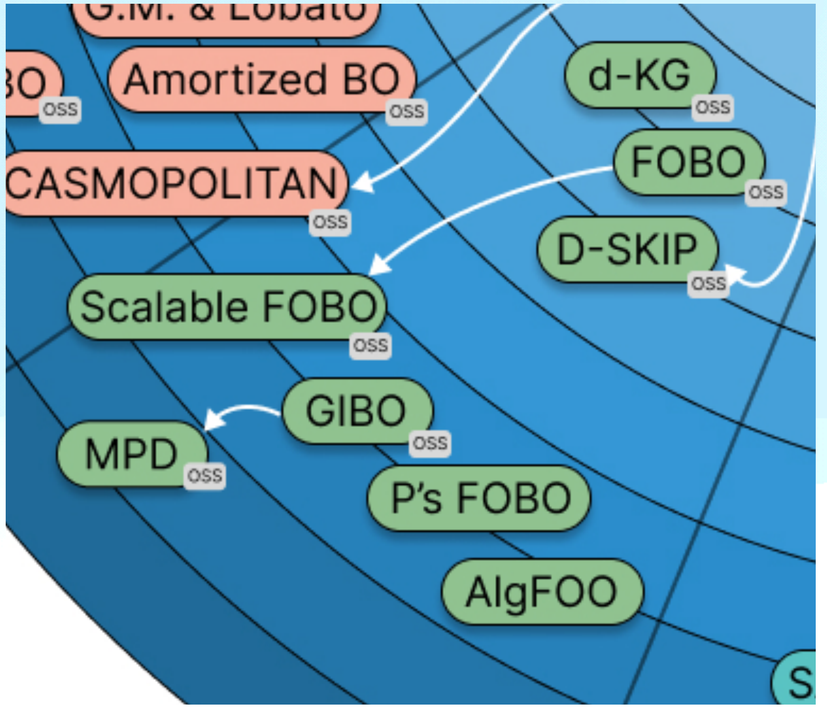
Gradient information

Core assumption:

Suppose your function is differentiable, then predict the gradient using GPs

$$f \sim \mathrm{GP}(\mu, k)$$

$$\Rightarrow \partial f \sim \mathrm{GP}(\partial\mu, \partial^2 k)$$

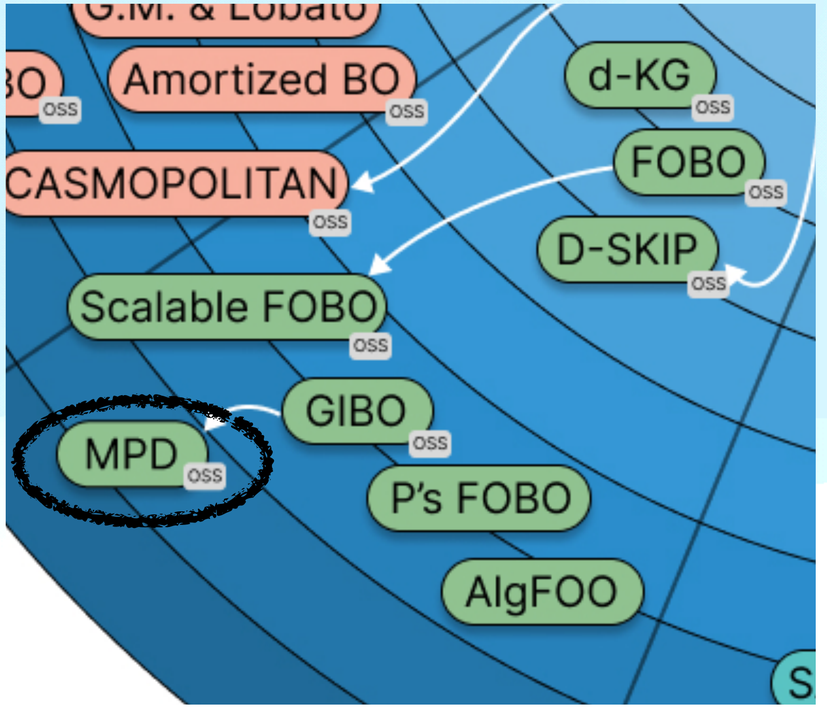


A taxonomy of 7

Gradient information

Local Bayesian optimization via maximizing probability of descent

Quan Nguyen[*1] **Kaiwen Wu**[*2] **Jacob R. Gardner**[2] **Roman Garnett**[1]
[1]Washington University in St. Louis [2]University of Pennsylvania
{quan,garnett}@wustl.edu
{kaiwenwu,jacobrg}@seas.upenn.edu



A taxonomy of 7

Gradient information

$$p\left(\begin{bmatrix}\mathbf{y}\\ \nabla f(\mathbf{x})\end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix}\mu(\mathbf{X})\\ \nabla\mu(\mathbf{x})\end{bmatrix}, \begin{bmatrix}K(\mathbf{X},\mathbf{X})+\sigma^2\mathbf{I} & K(\mathbf{X},\mathbf{x})\nabla^\top\\ \nabla K(\mathbf{x},\mathbf{X}) & \nabla K(\mathbf{x},\mathbf{x})\nabla^\top\end{bmatrix}\right).$$

Here, when placed in front of K, the differential operator ∇ indicates that we are taking the derivative of K with respect to its first input; when placed behind K, it indicates the derivative is with respect to its second input. Conditioned on the observations $(\mathbf{X}, \mathbf{y})$, the posterior distribution of the derivative $\nabla f(\mathbf{x})$ may be obtained as:

$$p\big(\nabla f(\mathbf{x}) \mid \mathbf{x}, \mathbf{X}, \mathbf{y}\big) = \mathcal{N}(\boldsymbol{\mu}_\mathbf{x}, \Sigma_\mathbf{x}),$$

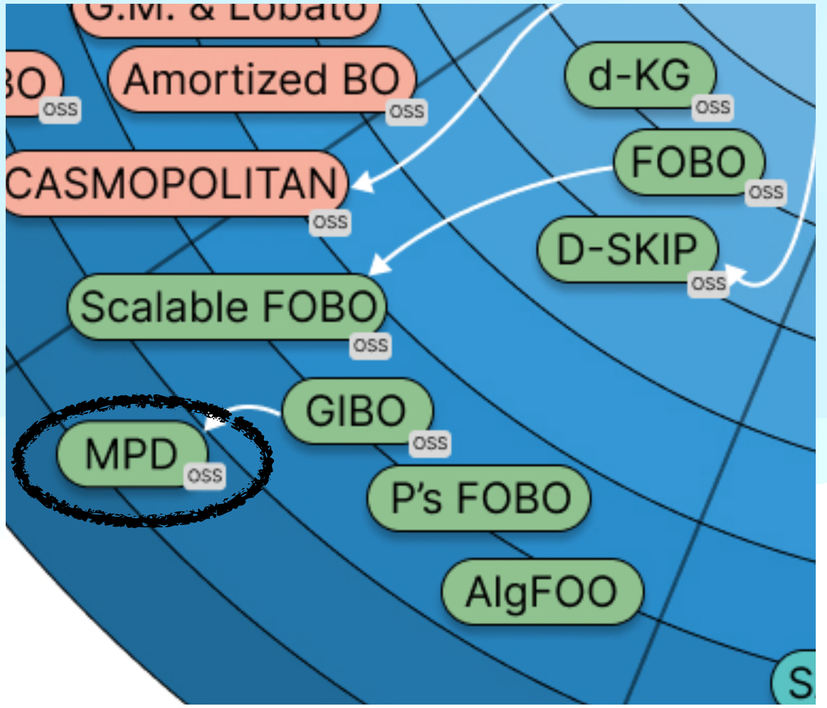
$$\text{where } \boldsymbol{\mu}_\mathbf{x} = \nabla\mu(\mathbf{x}) + \nabla K(\mathbf{x},\mathbf{X})\big(K(\mathbf{X},\mathbf{X})+\sigma^2\mathbf{I}\big)^{-1}\big(\mathbf{y}-\mu(\mathbf{X})\big), \tag{1}$$

$$\Sigma_\mathbf{x} = \nabla K(\mathbf{x},\mathbf{x})\nabla^\top - \nabla K(\mathbf{x},\mathbf{X})\big(K(\mathbf{X},\mathbf{X})+\sigma^2\mathbf{I}\big)^{-1}K(\mathbf{X},\mathbf{x})\nabla^\top.$$

Definition 3.1 (Descent probability and most probable descent direction)**.** *Given a unit vector* $\mathbf{v}$*, the descent probability of the direction* $\mathbf{v}$ *at the location* $\mathbf{x}$ *is given by*

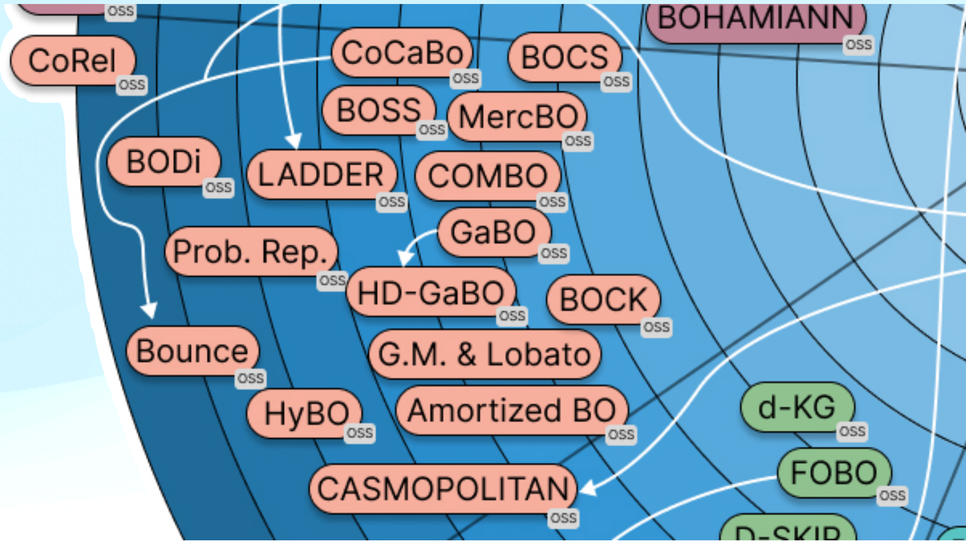
$$\Pr\left(\nabla_\mathbf{v} f(\mathbf{x}) < 0 \mid \mathbf{x}, \mathbf{v}\right) = \Phi\left(-\frac{\mathbf{v}^\top\boldsymbol{\mu}_\mathbf{x}}{\sqrt{\mathbf{v}^\top\Sigma_\mathbf{x}\mathbf{v}}}\right), \tag{2}$$

where Φ *is the* CDF *of the standard normal distribution. If* $\mathbf{v}^*$ *achieves the maximum descent probability* $\mathbf{v}^* \in \arg\max_\mathbf{v} \Pr\left(\nabla_\mathbf{v} f(\mathbf{x}) < 0 \mid \mathbf{x}, \mathbf{v}\right)$*, then we call* $\mathbf{v}^*$ *a most probable descent direction.*



A taxonomy of 7

Structured spaces

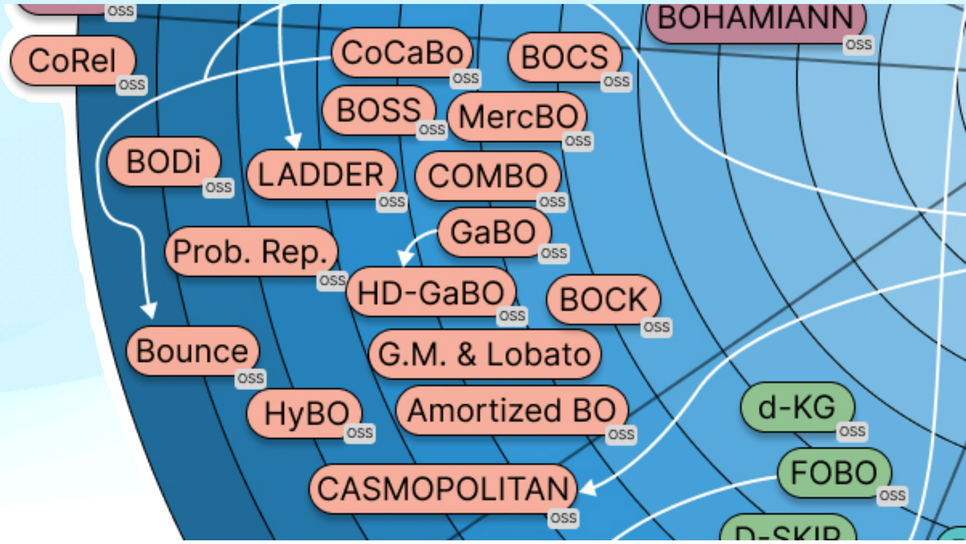


A taxonomy of 7

Structured spaces

Core assumption:

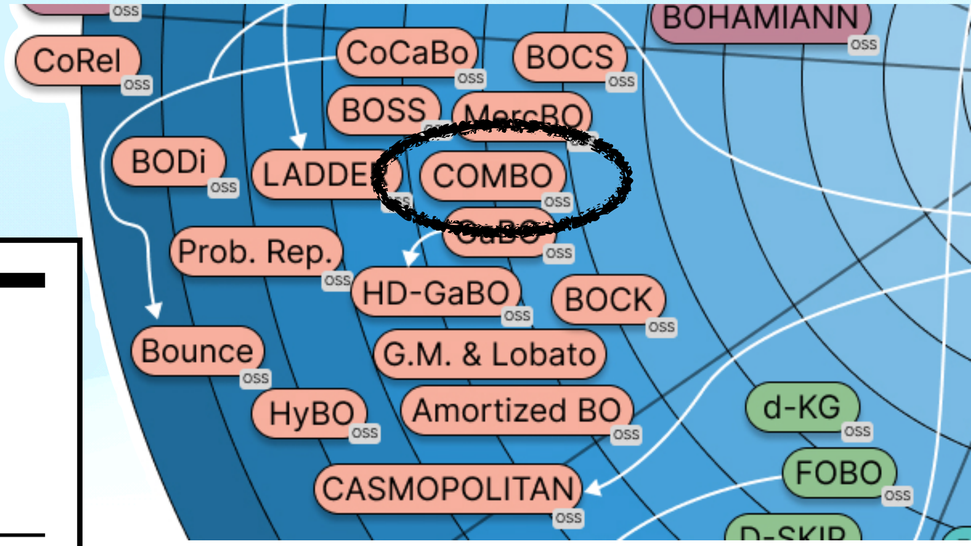
Your data is highly structured (e.g. lives in a Riemannian manifold, or is discrete/mixed input)



A taxonomy of 7

Structured spaces

Core assumption:



Combinatorial Bayesian Optimization using the Graph Cartesian Product

Changyong Oh[1] Jakub M. Tomczak[2] Efstratios Gavves[1] Max Welling[1,2,3]
[1] University of Amsterdam [2] Qualcomm AI Research [3] CIFAR
C.Oh@uva.nl, jtomczak@qti.qualcomm.com, egavves@uva.nl, m.welling@uva.nl

A taxonomy of 7

Structured spaces

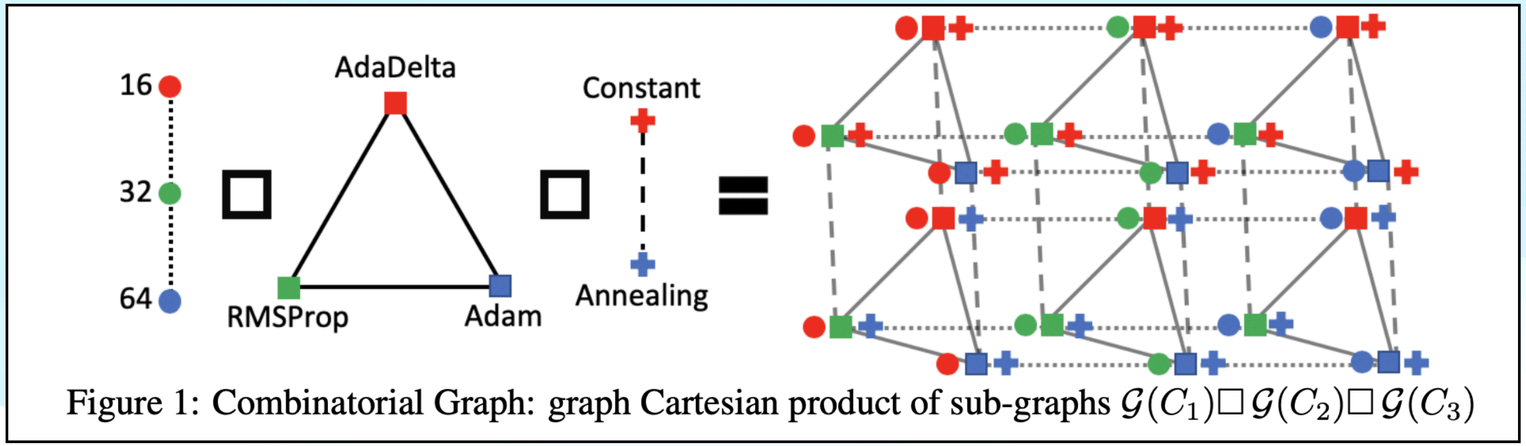
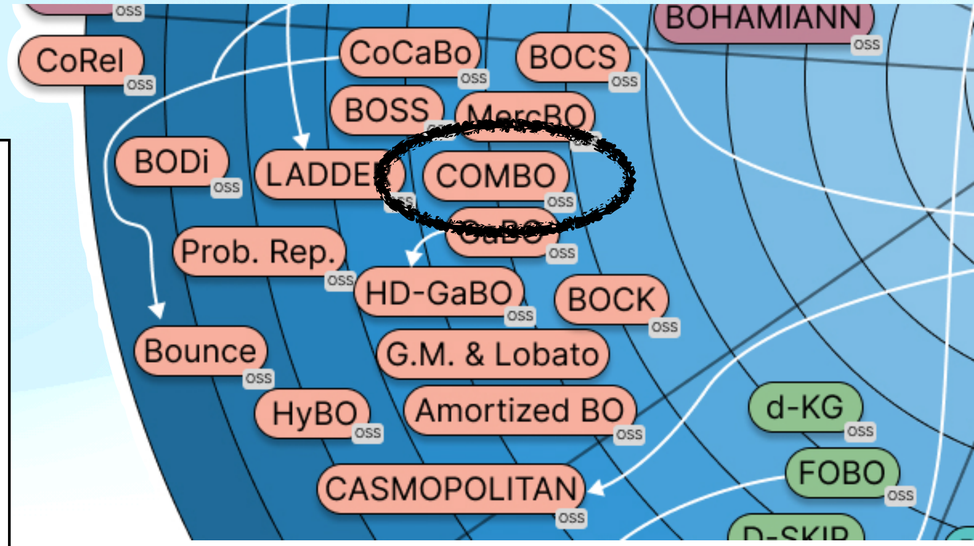


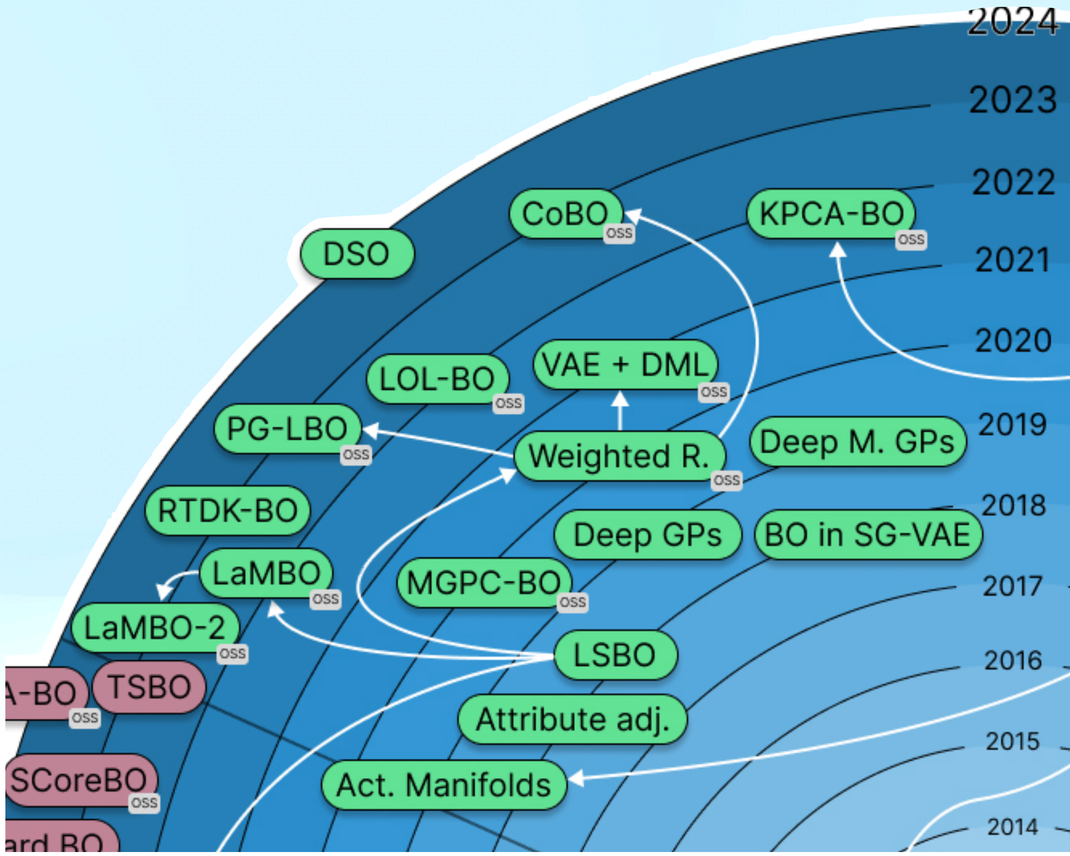
Figure 1: Combinatorial Graph: graph Cartesian product of sub-graphs $\mathcal{G}(C_1)\square\mathcal{G}(C_2)\square\mathcal{G}(C_3)$



(i) build a combinatorial graph, and (ii) use graph kernels

A taxonomy of 7

Non-linear embeddings



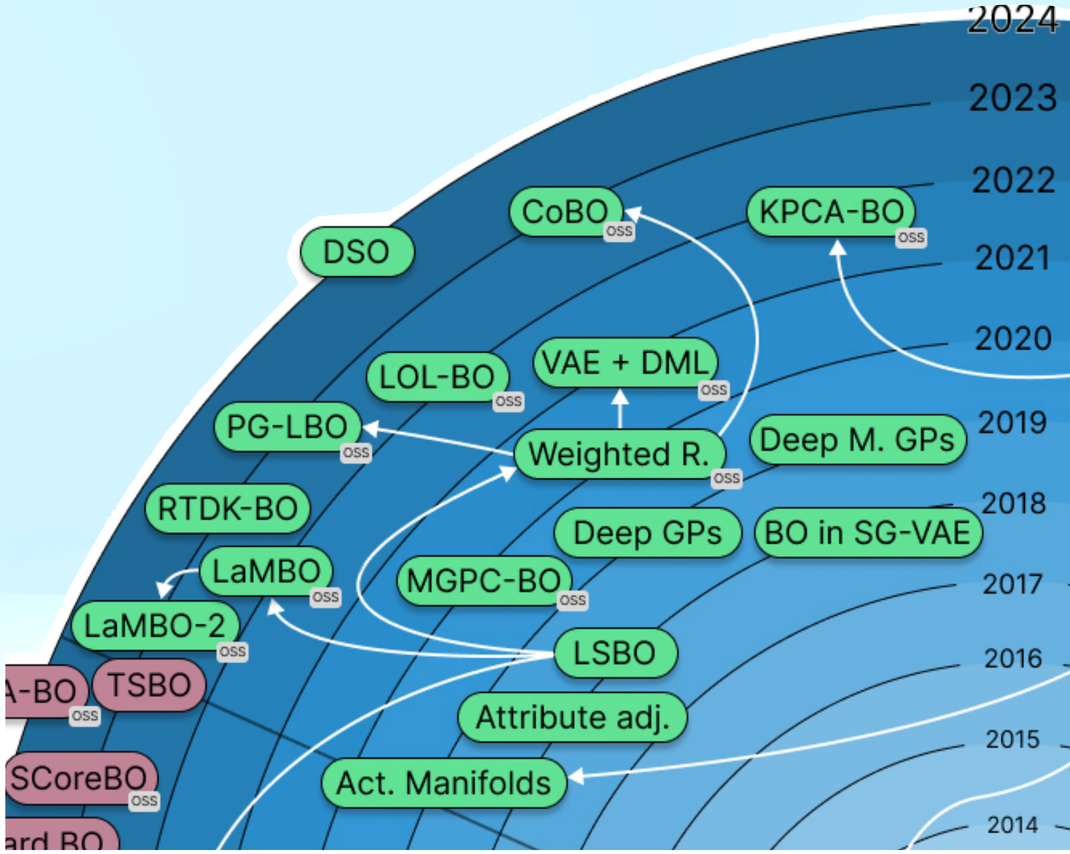
A taxonomy of 7

Non-linear embeddings

Core idea:

If you have plenty of unsupervised data $\{\boldsymbol{x}_n\}$, you can learn a latent representation and optimize therein.

(e.g. using autoencoders)



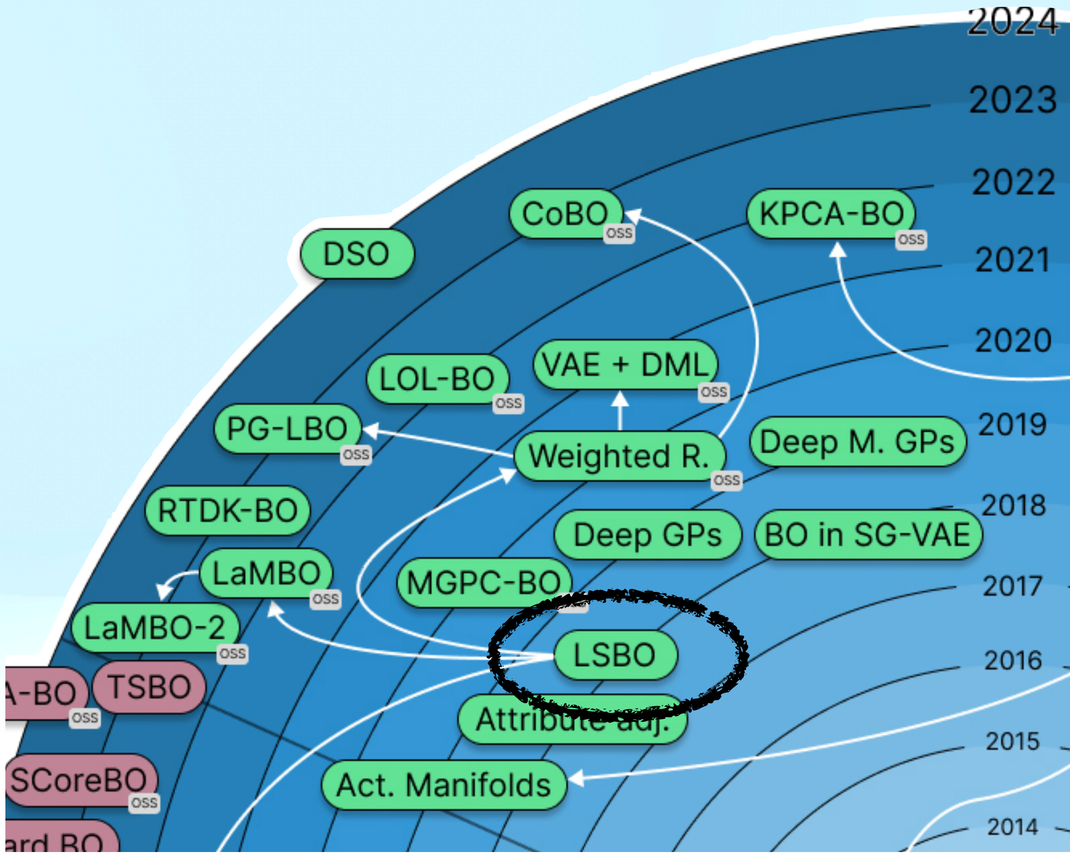
A taxonomy of 7

Non-linear embeddings

ACS central science

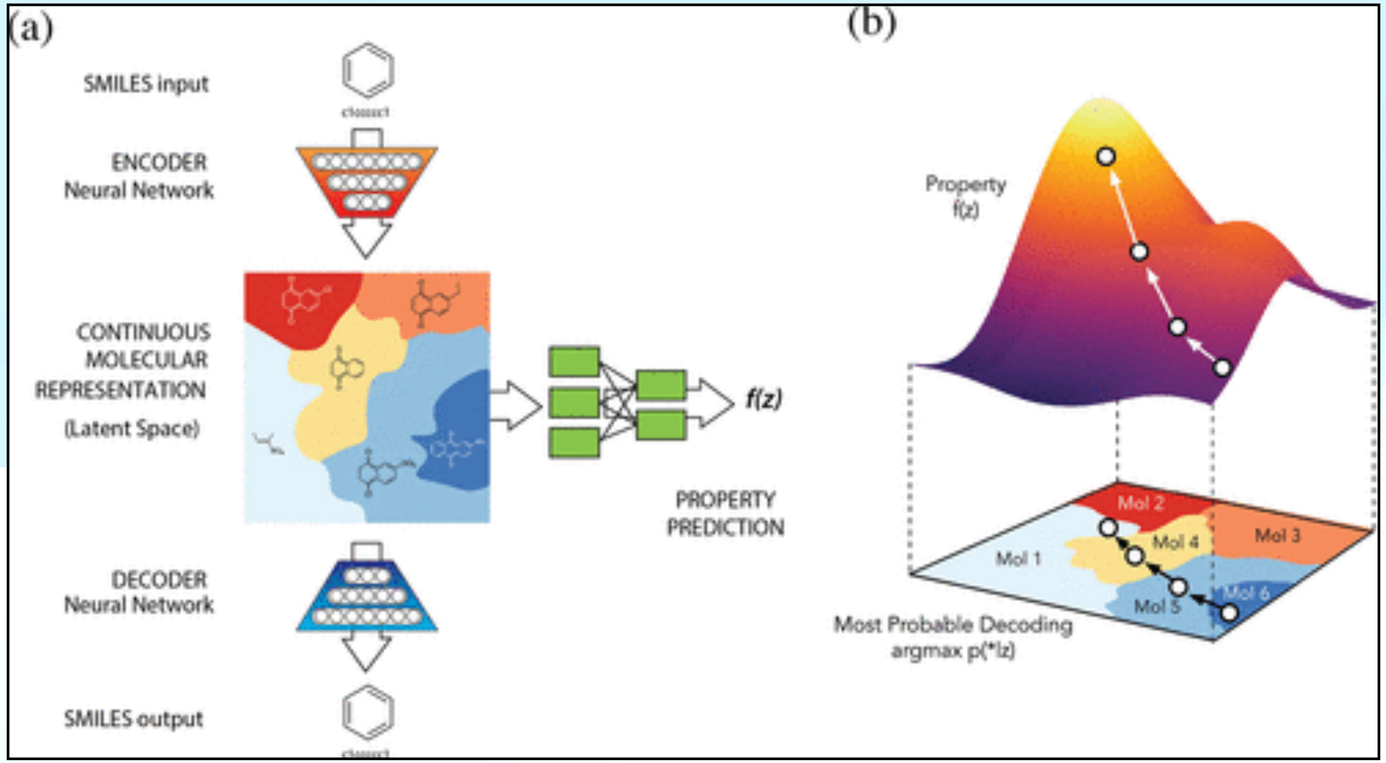
Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules

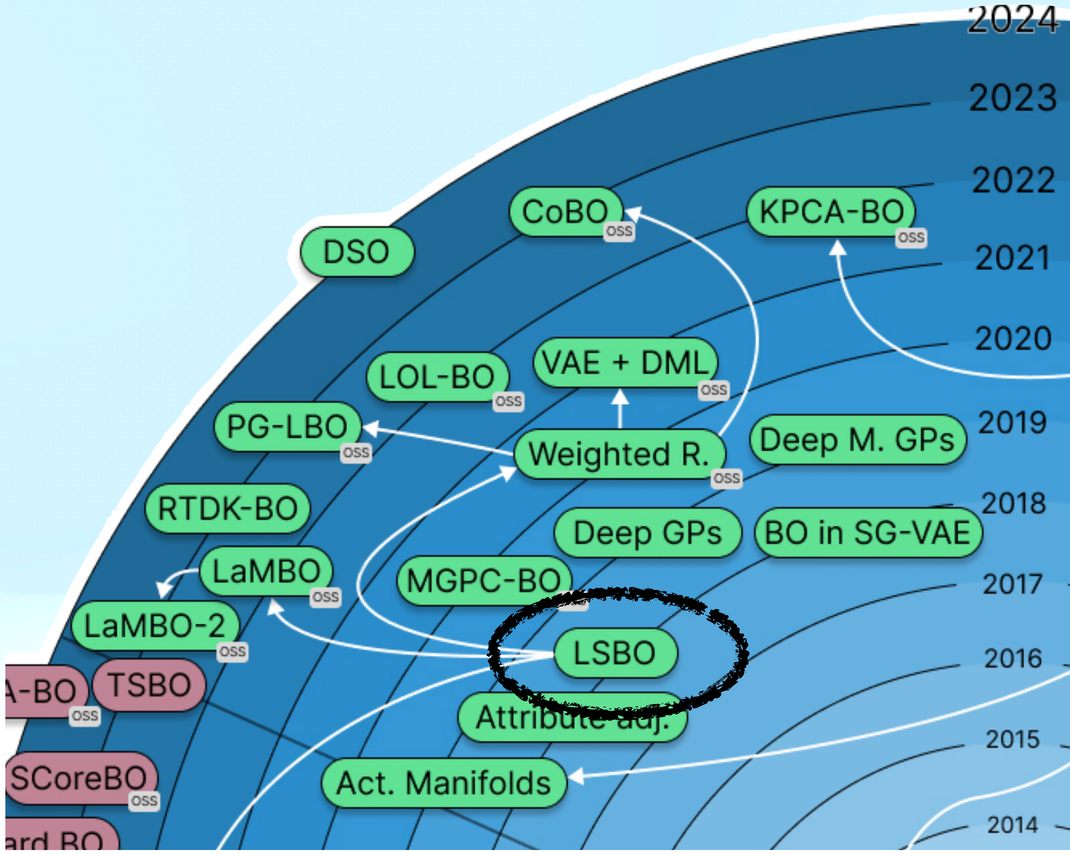
Rafael Gómez-Bombarelli[†#], Jennifer N. Wei[‡#], David Duvenaud[¶#], José Miguel Hernández-Lobato[§#], Benjamín Sánchez-Lengeling[‡], Dennis Sheberla[‡], Jorge Aguilera-Iparraguirre[†], Timothy D. Hirzel[†], Ryan P. Adams[∇], and Alán Aspuru-Guzik[*†‡⊥]



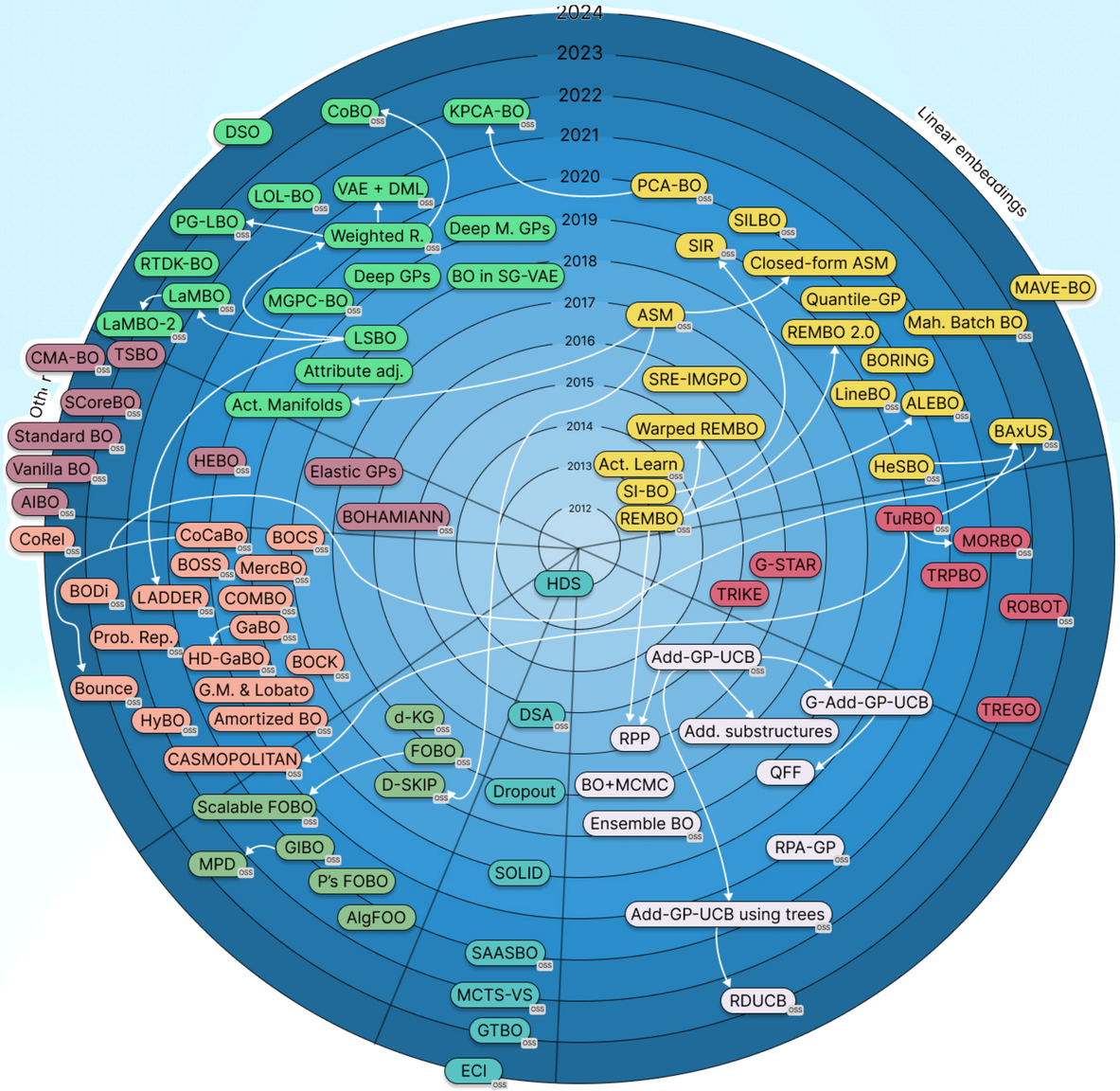
A taxonomy of 7

Non-linear embeddings





A taxonomy of 7



A taxonomy of 7

?

How is the field doing?

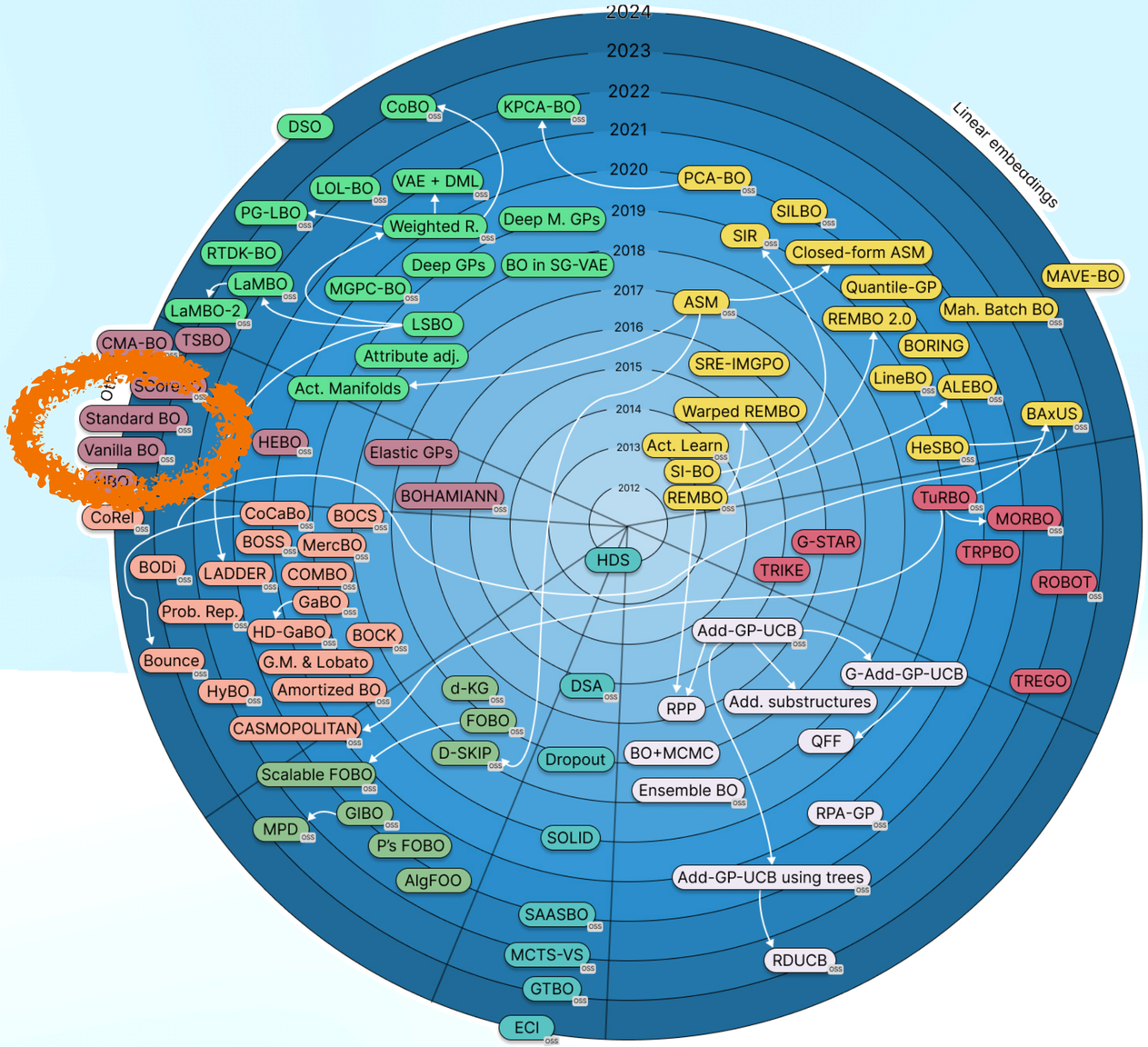
A taxonomy of 7

?

How is the field doing?

There is some controversy on the narratives and benchmarks.

A taxonomy of 7

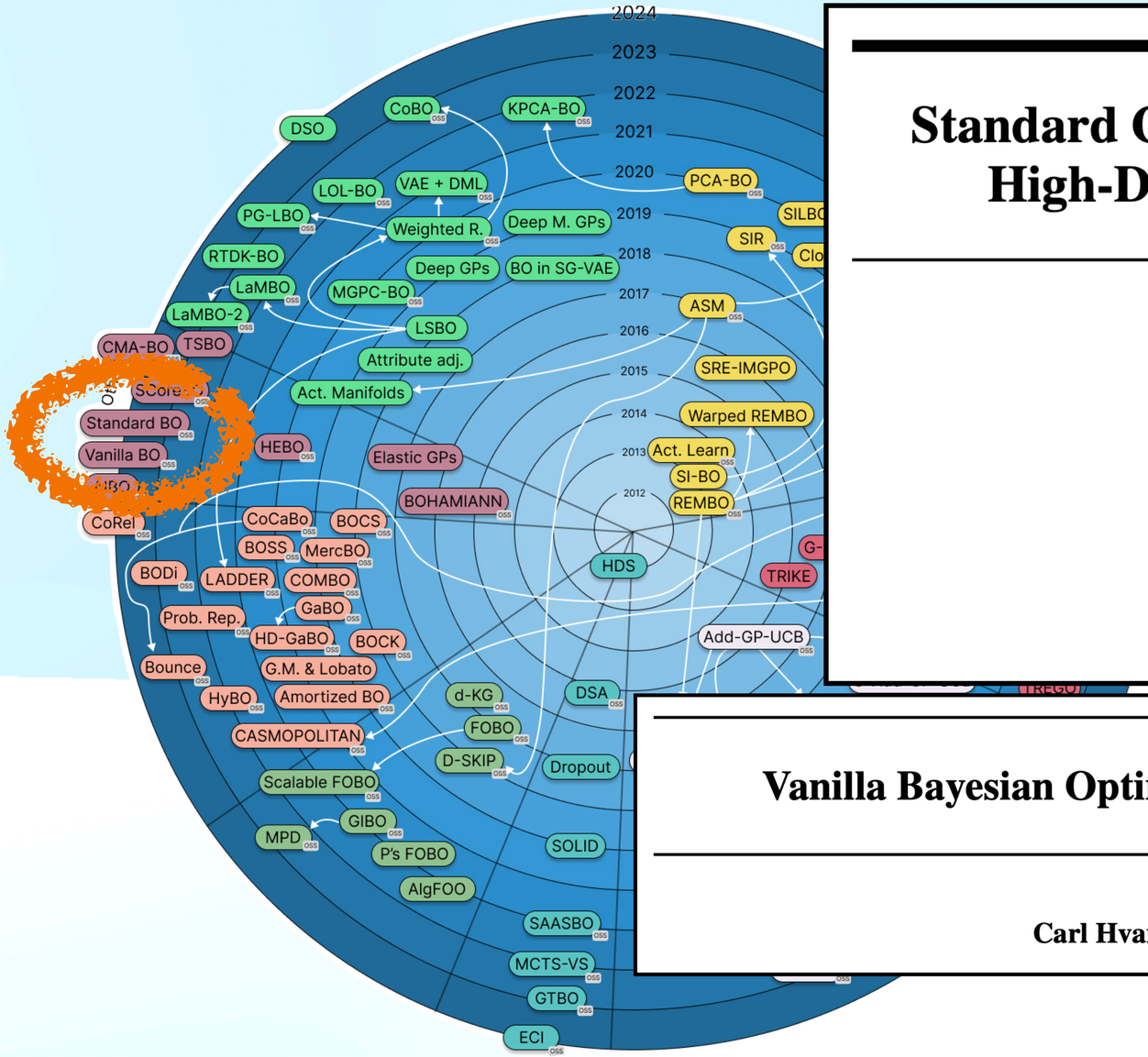


?

How is the field doing?

There is some controversy on the narratives and benchmarks.

A taxonomy of 7



Standard Gaussian Process Can Be Excellent for High-Dimensional Bayesian Optimization

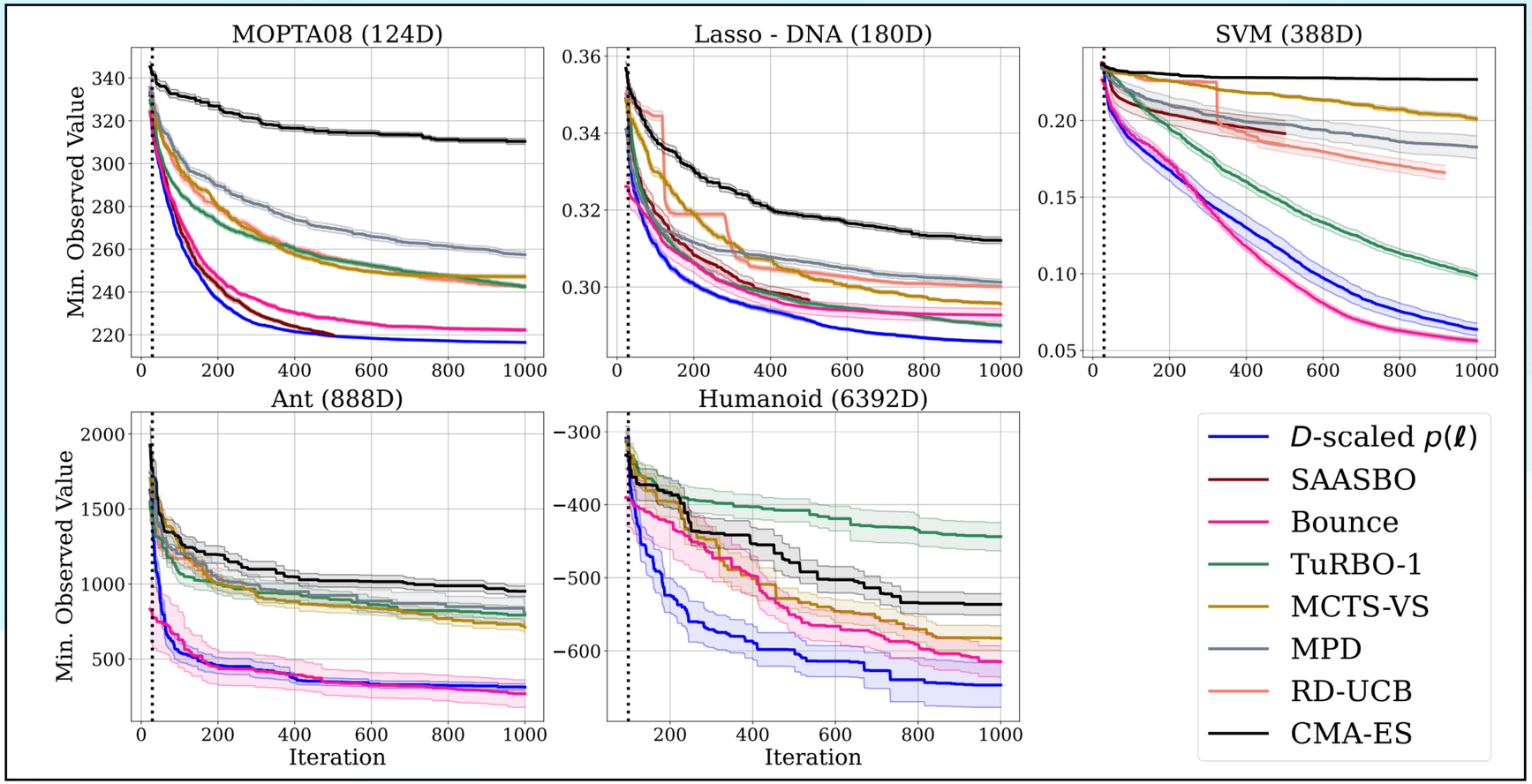
Zhitong Xu
Kahlert School of Computing
The University of Utah
zhitongx008@gmail.com

Shandian Zhe*
Kahlert School of Computing
The University of Utah
zhe@cs.utah.edu

Vanilla Bayesian Optimization Performs Great in High Dimensions

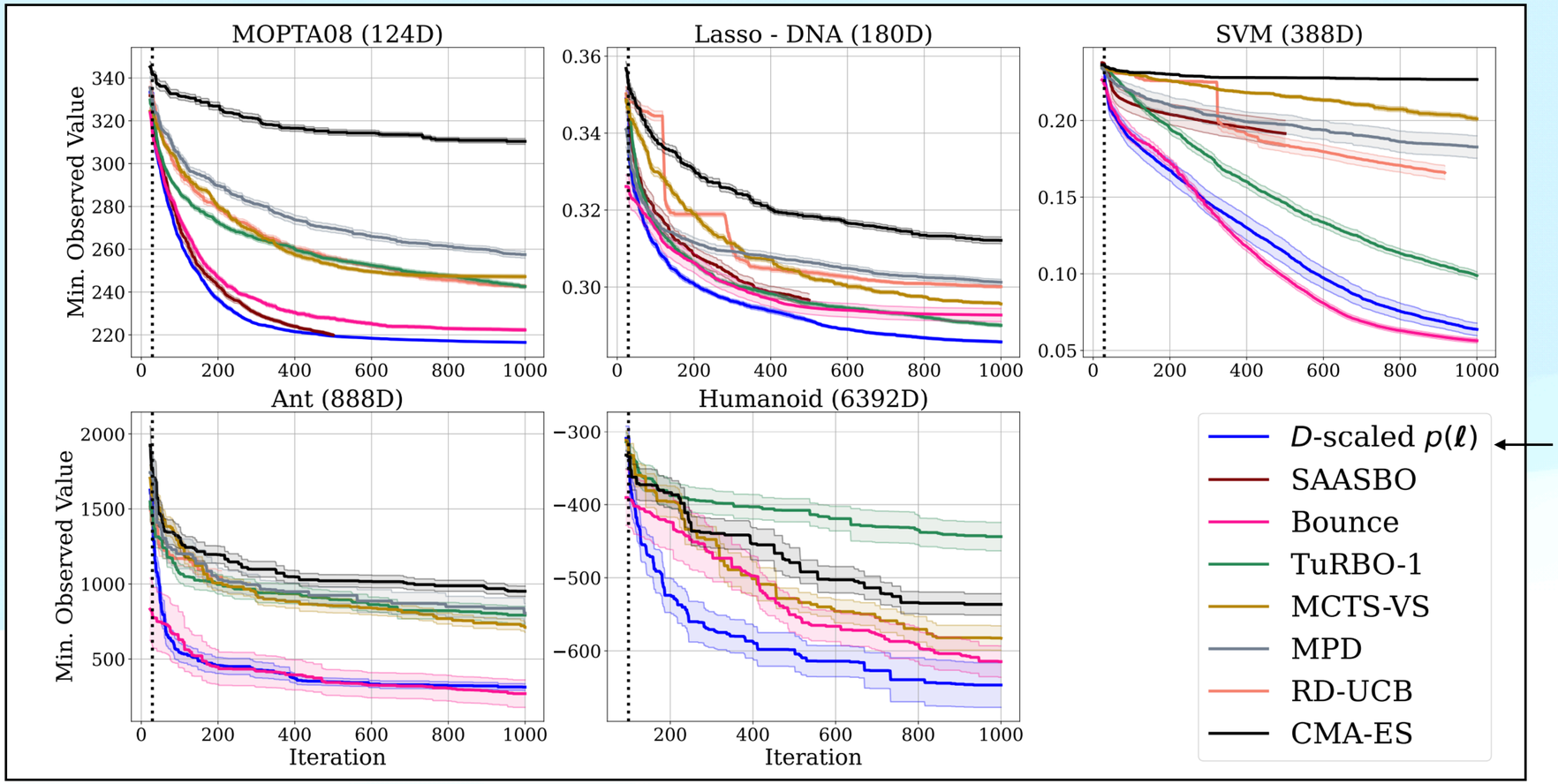
Carl Hvarfner [1] **Erik O. Hellsten** [1 2] **Luigi Nardi** [1 2]

A taxonomy of 7



Vanilla Bayesian Optimization Performs Great in High Dimensions

Carl Hvarfner [1] **Erik O. Hellsten** [1 2] **Luigi Nardi** [1 2]



A simple regularization on the training loss of GPs

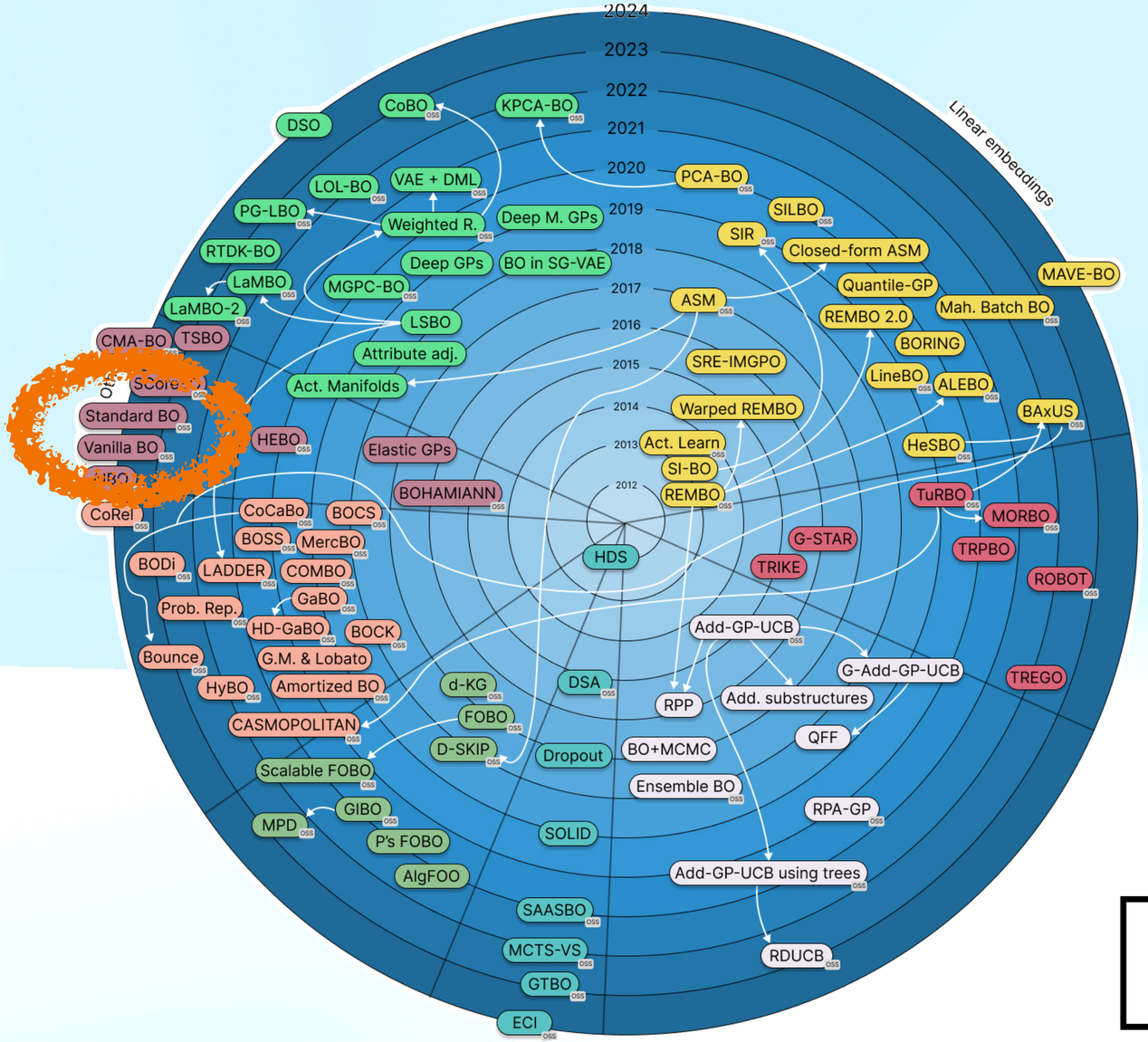
Vanilla Bayesian Optimization Performs Great in High Dimensions

Carl Hvarfner [1] **Erik O. Hellsten** [1 2] **Luigi Nardi** [1 2]

? How is the field doing?

There is some controversy on the narratives and benchmarks.

A taxonomy of 7



?

How is the field doing?

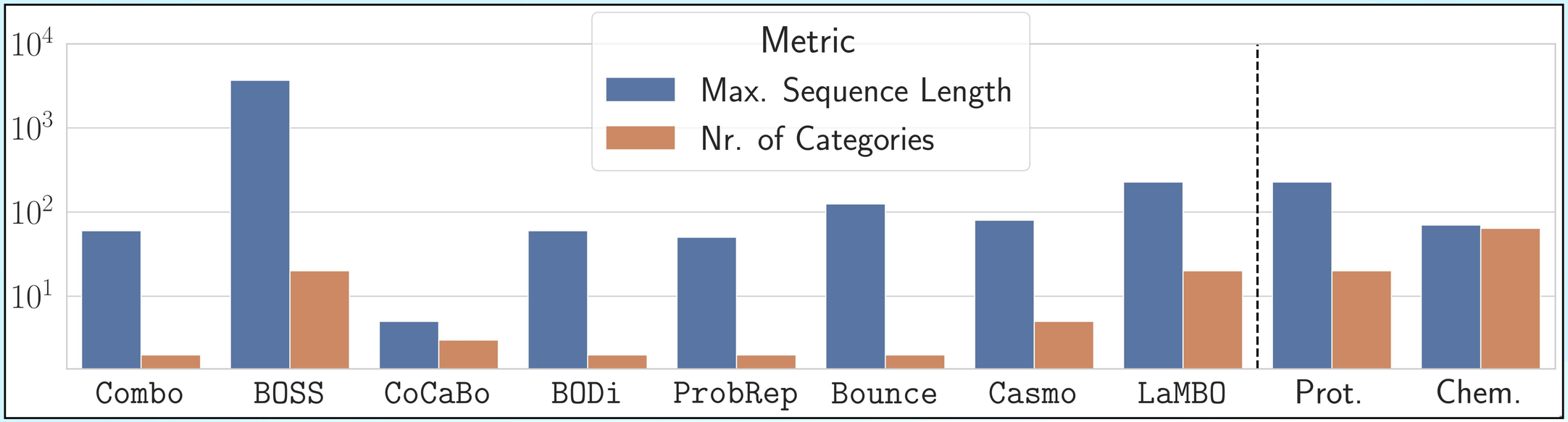
There is some controversy on the narratives and benchmarks.

The jury is still out.

Benchmarking is difficult.

Different problems, different solvers

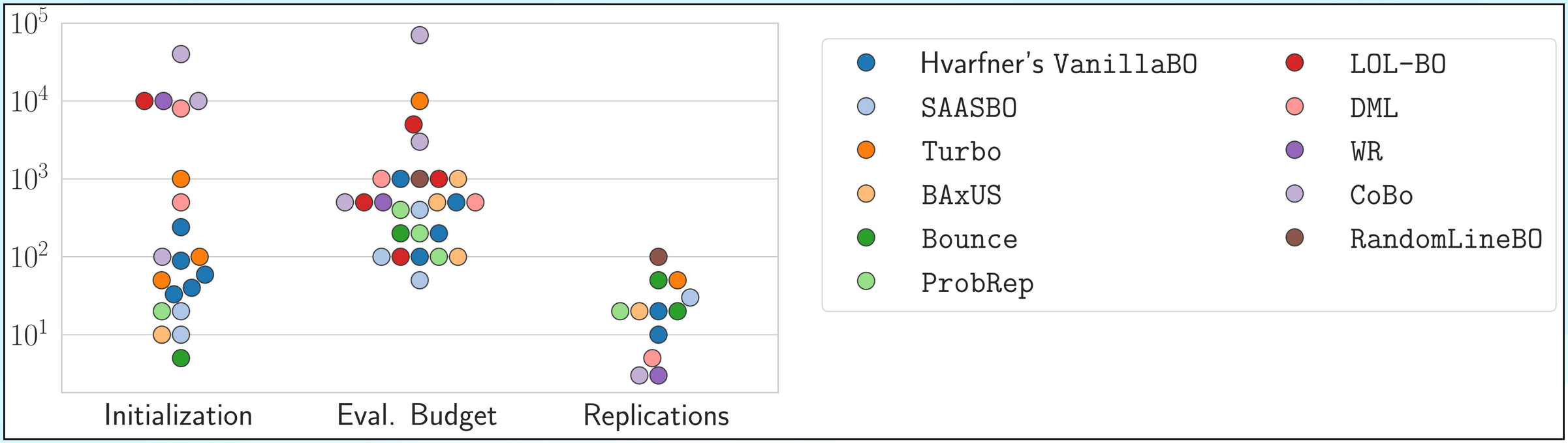
A taxonomy of 7



Different definitions of *high dimensions*

Benchmarking is difficult.

Discrete sequence optimization



Different (and sometimes underreported) problem setups

Benchmarking is difficult.

Discrete sequence optimization

Conclusion

Practitioner wants to optimize an expensive HD black-box $f(x)$

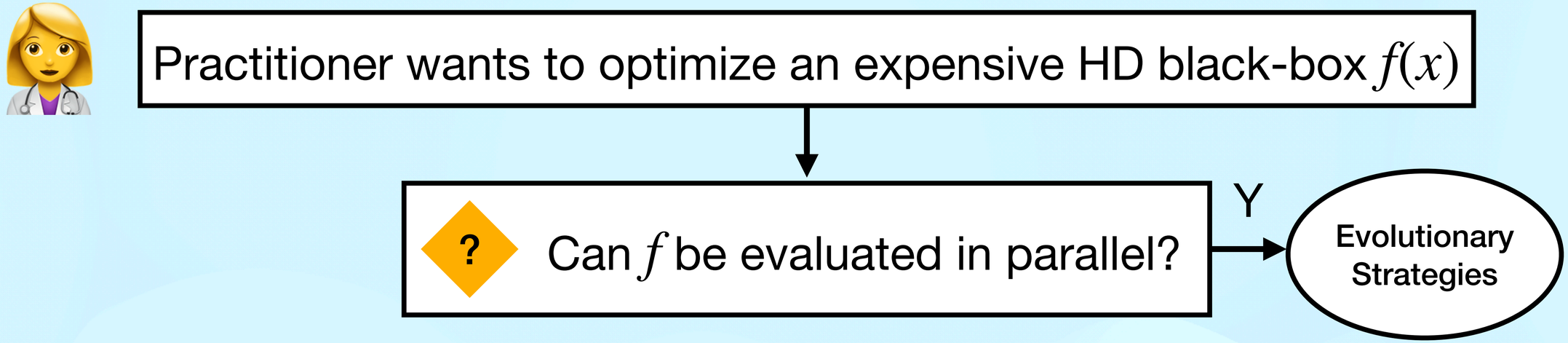
A flowchart for practitioners [work in progress]

Practitioner wants to optimize an expensive HD black-box $f(x)$

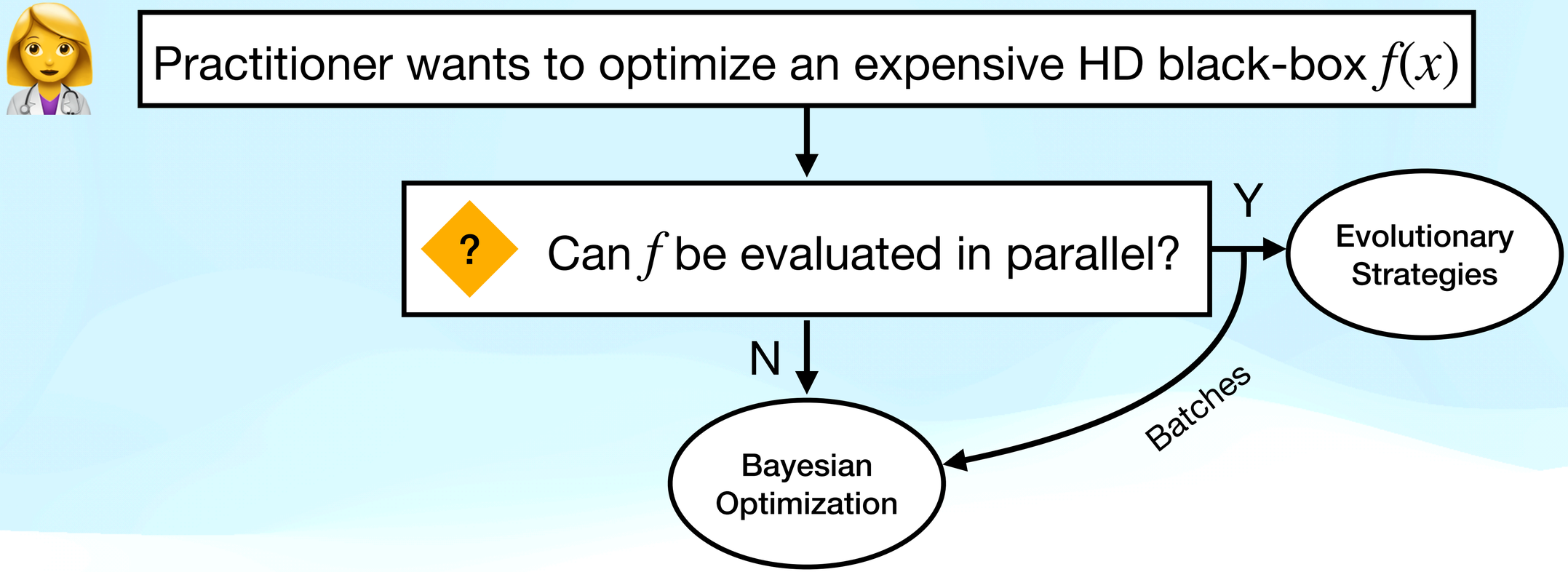
?

Can f be evaluated in parallel?

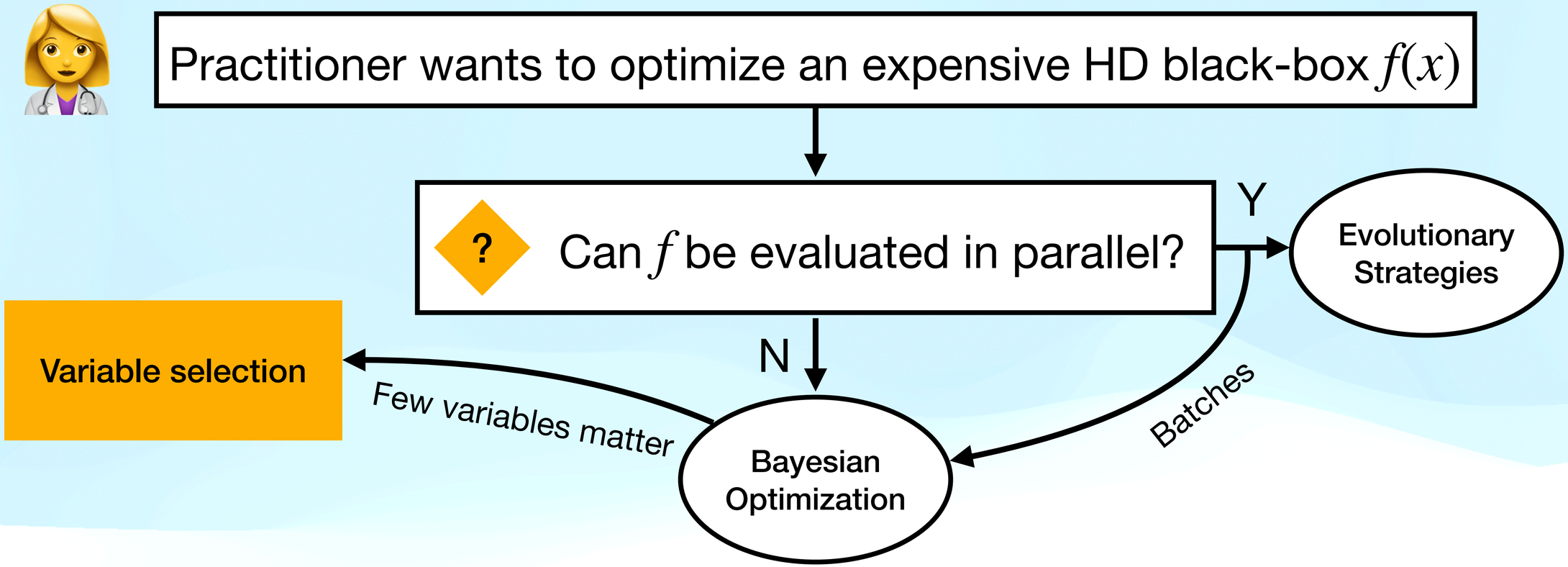
A flowchart for practitioners [work in progress]



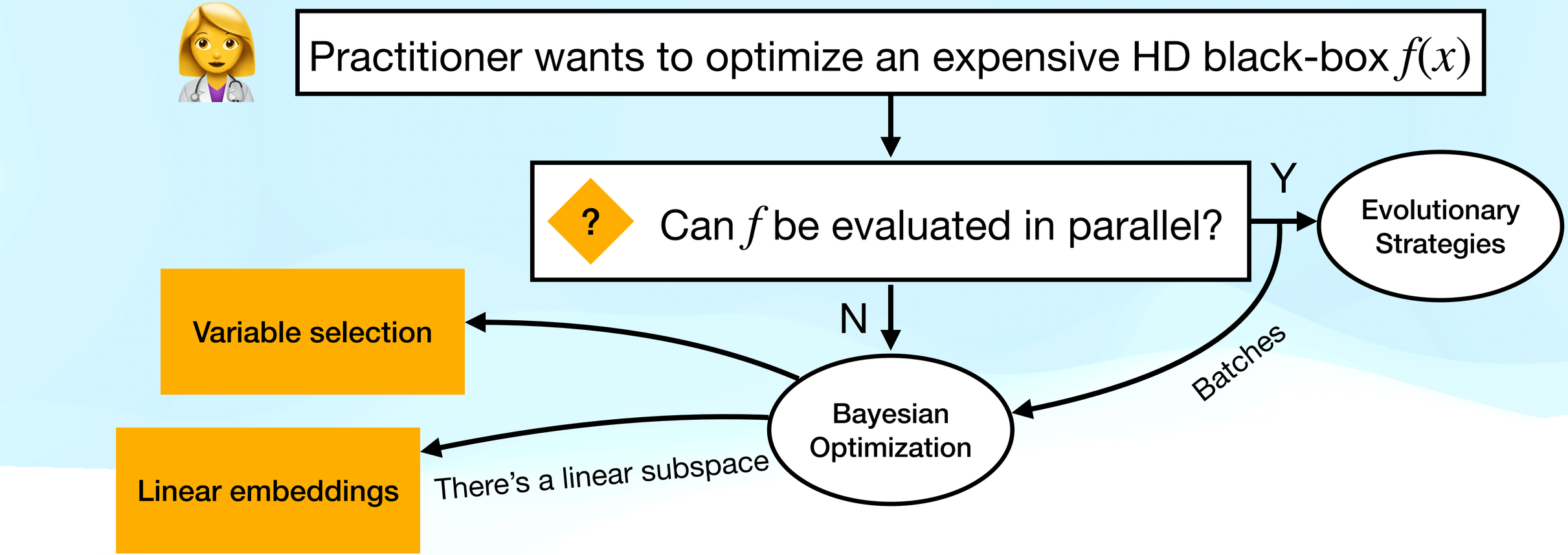
A flowchart for practitioners [work in progress]



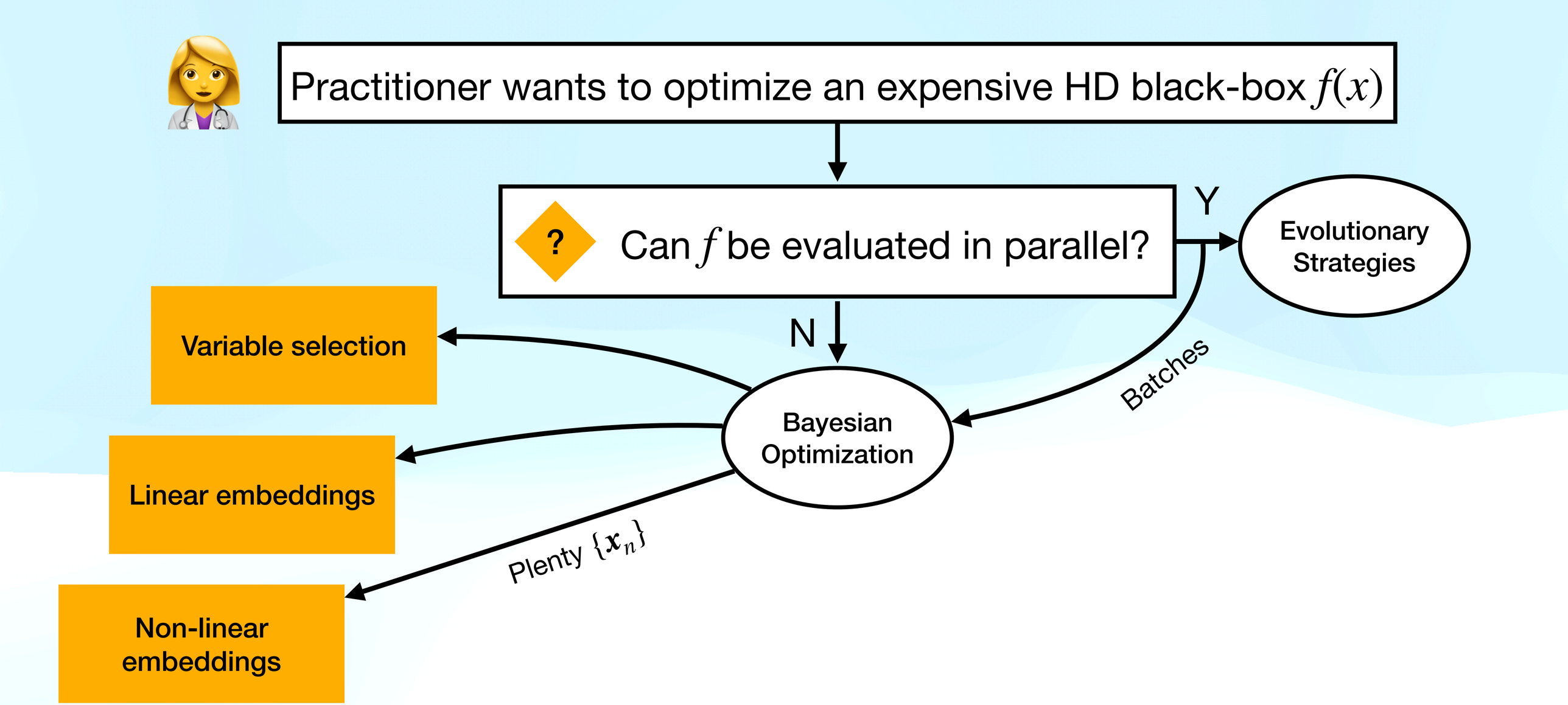
A flowchart for practitioners [work in progress]



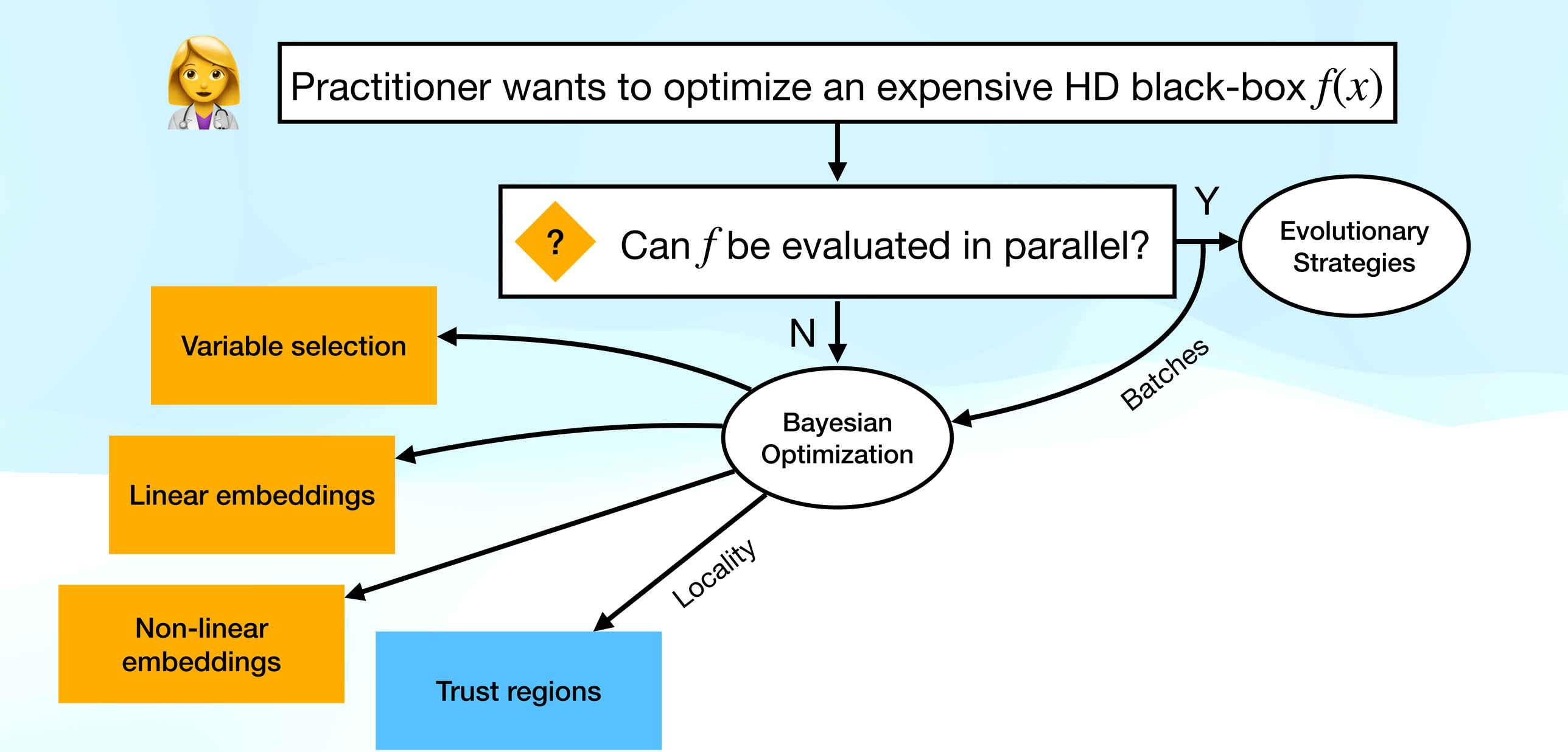
A flowchart for practitioners [work in progress]



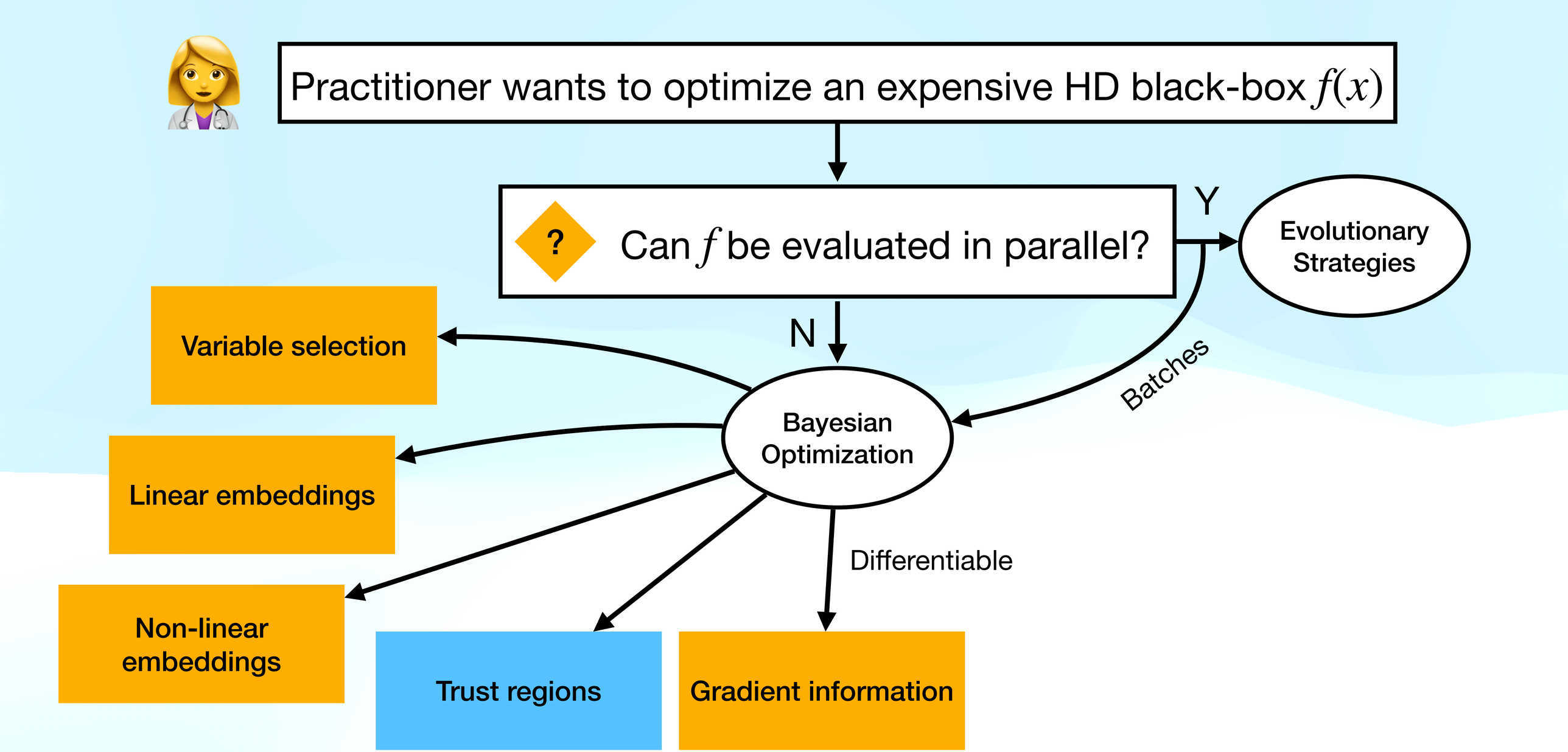
A flowchart for practitioners [work in progress]



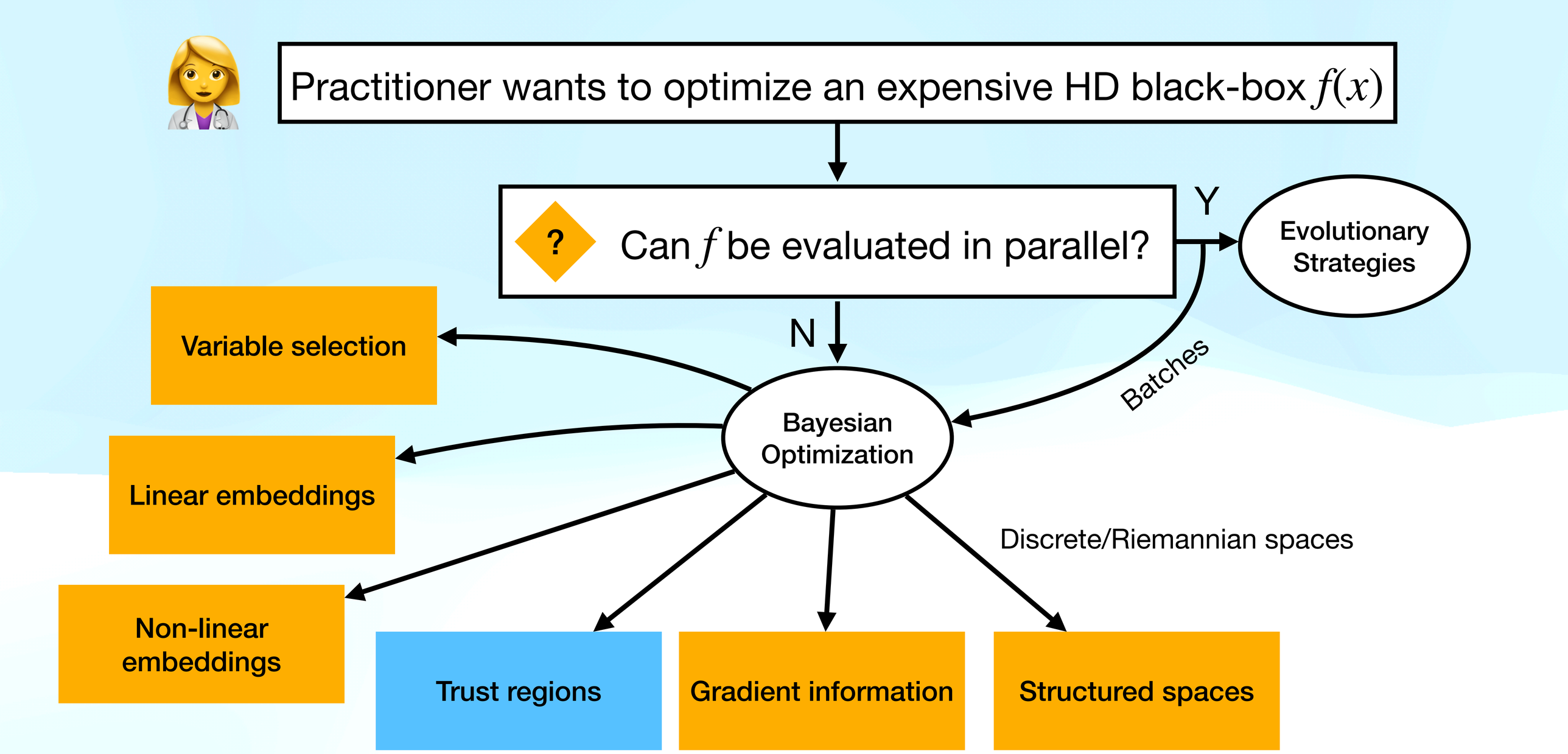
A flowchart for practitioners [work in progress]



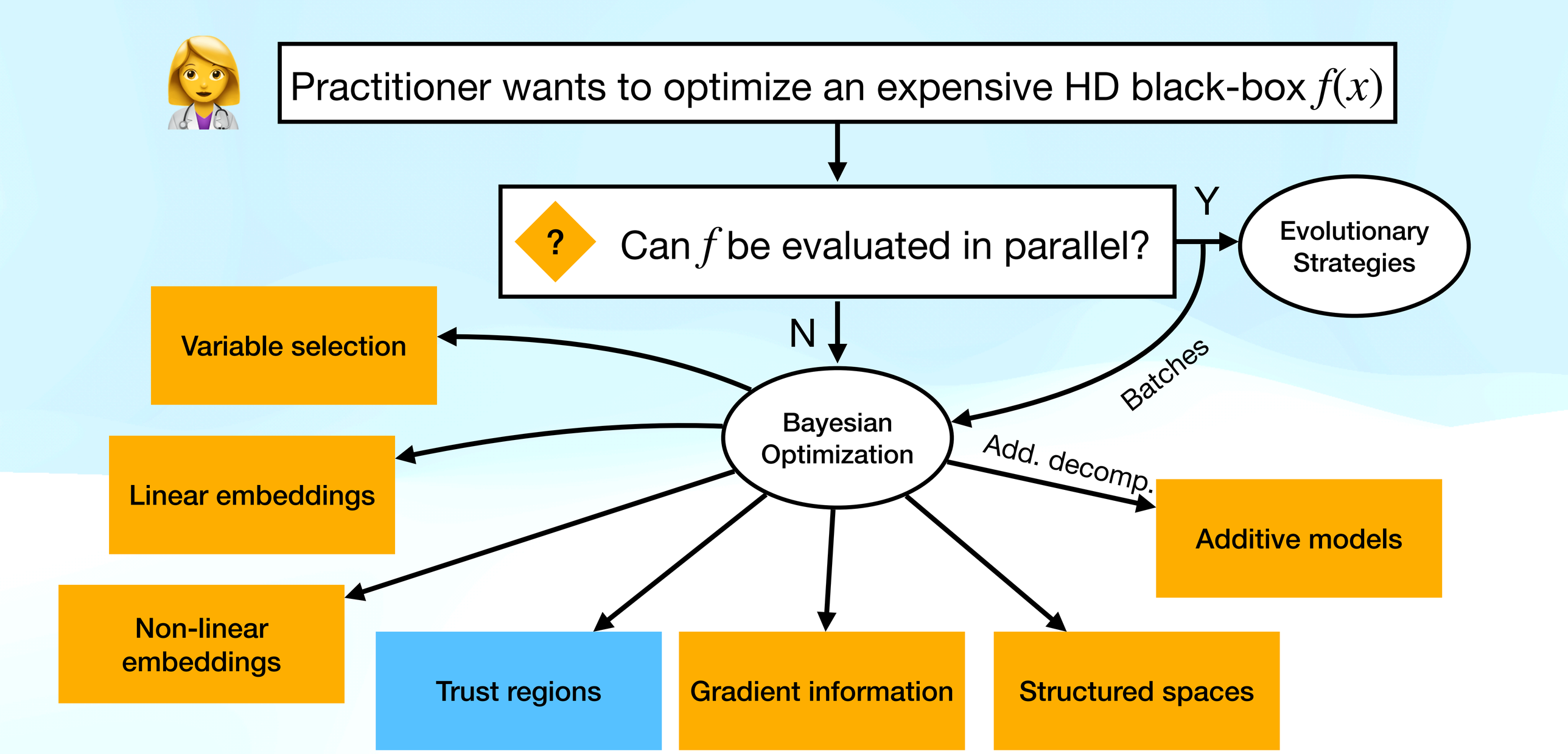
A flowchart for practitioners [work in progress]



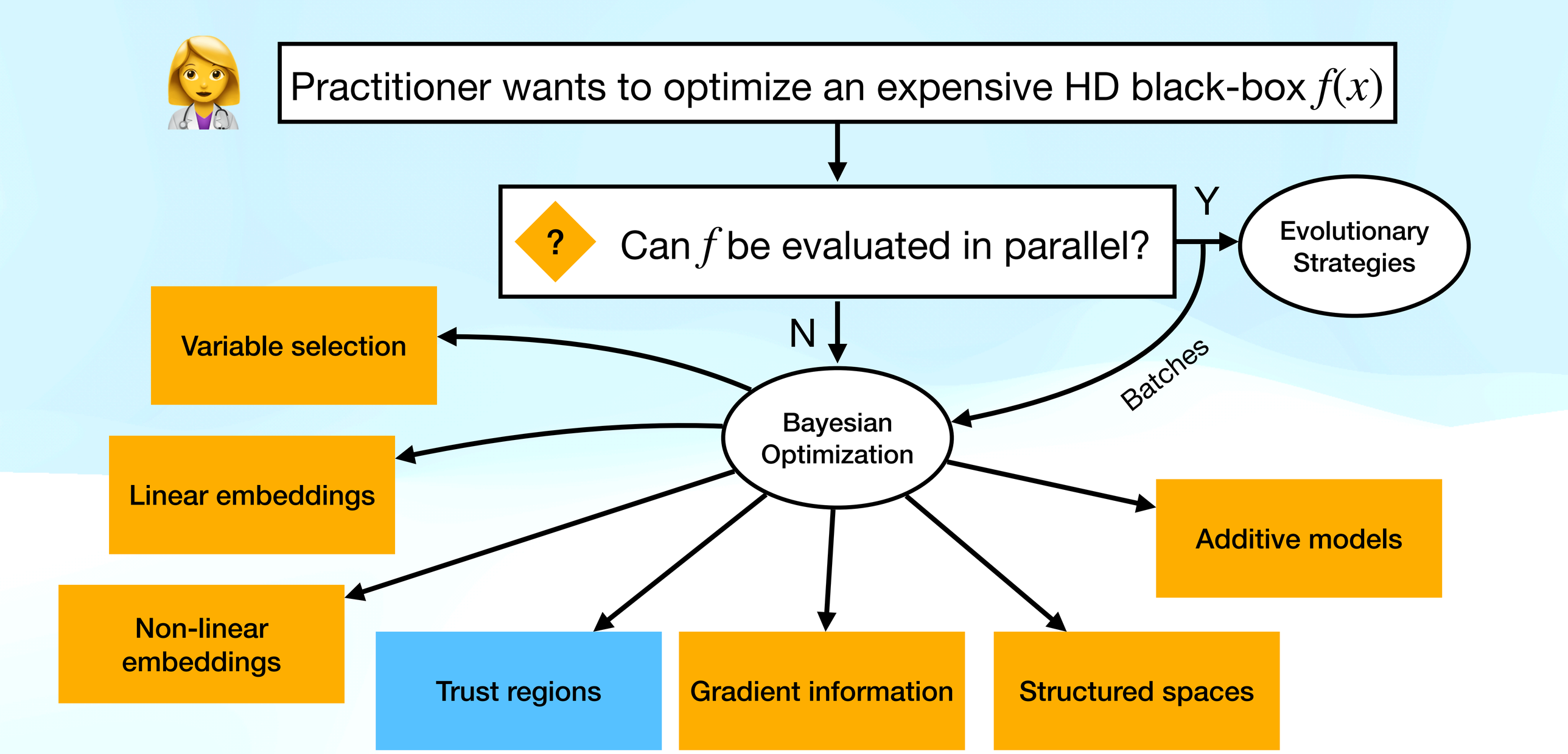
A flowchart for practitioners [work in progress]



A flowchart for practitioners [work in progress]



A flowchart for practitioners [work in progress]



A flowchart for practitioners [work in progress]

Thank you!

miguelgondu.com/assets/hdbo_biosustain_04_07_2024.pdf

Slides

machinelearninglifescience.github.io/hdbo_benchmark/

Project website

linkedin.com/in/miguelgondu/

LinkedIn

Thank you!

We’re running a benchmark!

miguelgondu.com/assets/hdbo_biosustain_04_07_2024.pdf

Slides

machinelearninglifescience.github.io/hdbo_benchmark/

Project website

linkedin.com/in/miguelgondu/

LinkedIn

Thank you!

MAP vs. MLE estimates

In the previous blogpost, I vaguely stated that kernel hyperparameters can be trained by maximizing the log-marginal likelihood w.r.t. the training points. Being a little bit more explicit, the function we are trying to maximize is this:

$$\log p(\mathbf{y}|\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N, \theta) = -\frac{1}{2}\mathbf{y}^T(K + \sigma_n^2 I)^{-1}\mathbf{y} - \frac{1}{2}\log\det(K + \sigma_n^2 I) - \frac{n}{2}\log(2\pi),$$

where $\boldsymbol{y}$ are the noisy observations, $K = [k(\boldsymbol{x}_i, \boldsymbol{x}_j)]$ is the Gram matrix, $\sigma_n > 0$ is a noise scale, and θ represents all kernel hyperparameters, including of course the lengthscales and σ_n .

Maximizing this quantity w.r.t. θ results in what is called the *Maximum Likelihood Estimate*, or MLE.

To encourage larger lengthscales, we can add a prior distribution for them, and try to maximize the *a posteriori* distribution $p(\theta|\boldsymbol{y}, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$, which is proportional to the product of the likelihood $p(\boldsymbol{y}|\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N, \theta)$ and the prior $p(\theta)$. The new function we would try to maximize would then be:

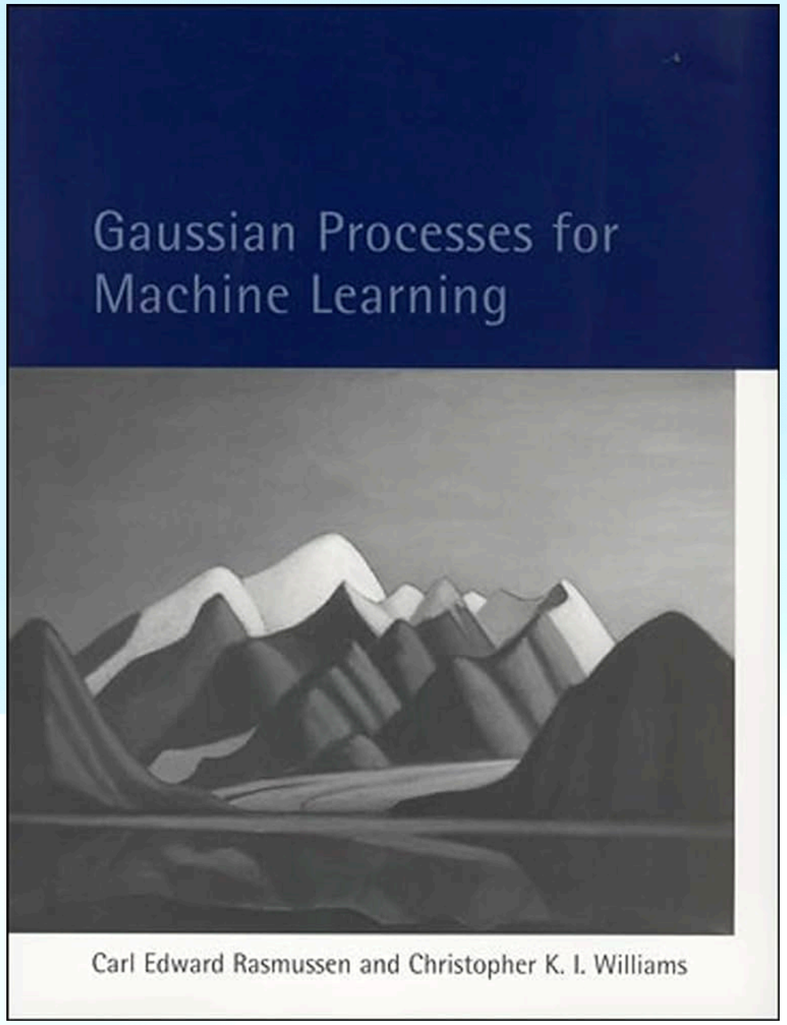
$$\underbrace{-\frac{1}{2}\mathbf{y}^T(K + \sigma_n^2 I)^{-1}\mathbf{y} - \frac{1}{2}\log\det(K + \sigma_n^2 I) - \frac{n}{2}\log(2\pi)}_{\text{The usual marginal log-likelihood}} + \underbrace{\log p(\theta)}_{\text{Regularizer}}.$$

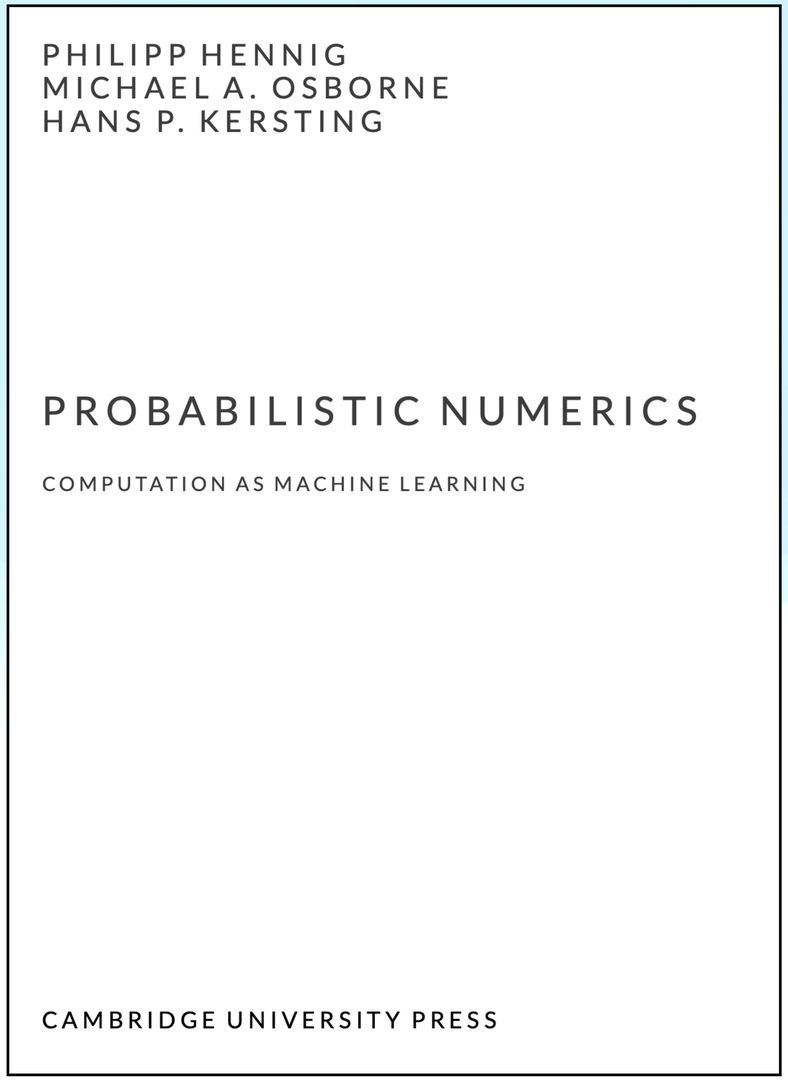
Maximizing this renders the *Maximum a posteriori* (MAP) estimate. Tools like `GPyTorch` allow us to add these regularizers easily using keyword arguments of kernels.

!

I didn’t mention GP training in detail

miguelgondu.com/blogposts/2024-03-16/when-does-vanilla-gpr-fail/





!

I didn’t mention GP training in detail