Going Beyond Global Optima with **Bayesian Algorithm Execution**

Willie Neiswanger

Going Beyond Global Optima with **Bayesian Algorithm Execution**

Willie Neiswanger

Willie

Ke Alex Wang

Stefano Ermon

Going Beyond Global Optima with **Bayesian Algorithm Execution**

Willie Neiswanger

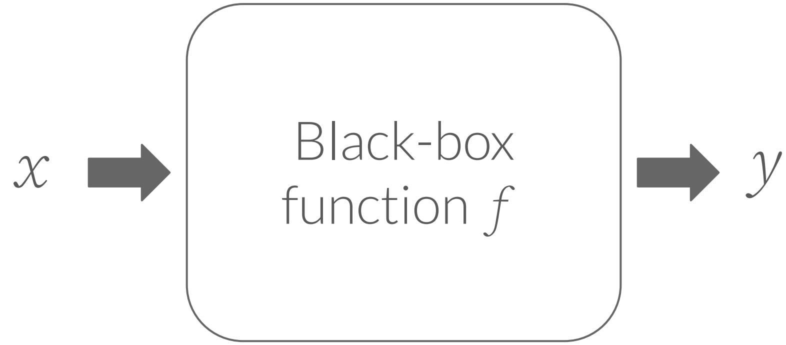
Extending Bayesian optimization from estimating *global optima* to estimating *other function properties defined by algorithms*

BACKGROUND

Background on *Black-box Global Optimization*

Suppose we have a noisy *"black-box"* function f.

Can ***query*** f:



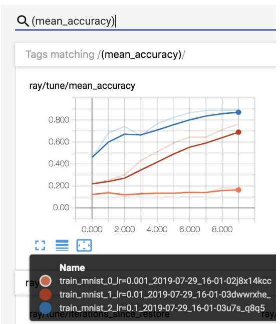
Assume:

- Observations are **noisy**: $y \sim f(x) + \varepsilon$
- Each function query is **costly**
  - E.g. in money, time, labor, etc.
- *Goal of this task*: estimate the location of **global optima** of f
- **Budget** of T queries

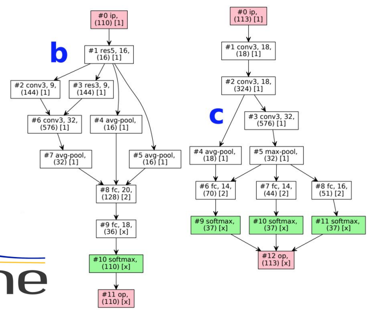
BACKGROUND

Black-box Global Optimization — many applications

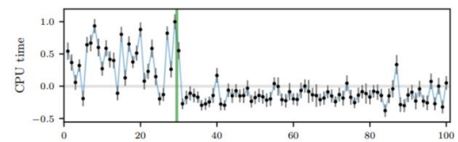
Hyperparameter Opt & Neural Architecture Search



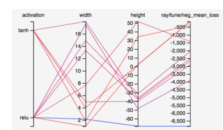
tune

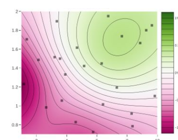


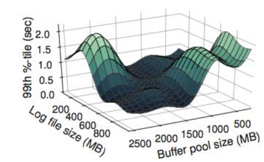
Systems Auto-tuning



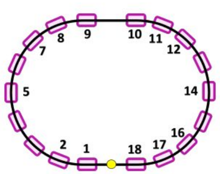
OTTER TUNE



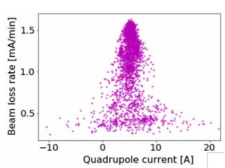




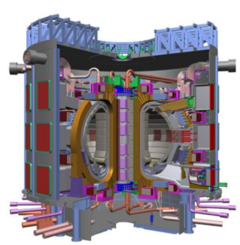
Optimizing Laboratory Equipment & Machines



(a) Layout of SPEAR3

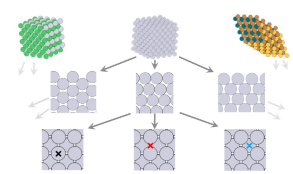


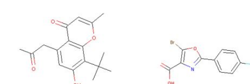
(b) Beam loss rate



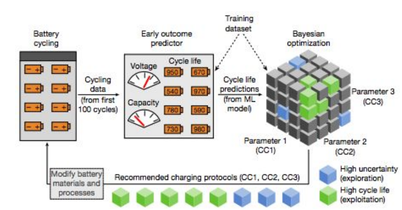
SLAC PPPL

Materials Discovery & Protocols





(a) QED 0.92083 (b) QED 0.92145



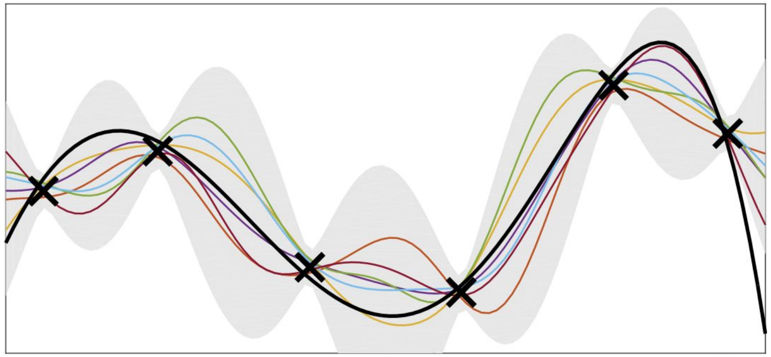
[1] Korovina, Ksenia, Sailun Xu, Kirthevasan Kandasamy, Willie Neiswanger, Barnabas Poczos, Jeff Schneider, and Eric Xing. "Chembo: Bayesian optimization of small organic molecules with synthesizable recommendations." AISTATS, 2020..
[2] Duris, Joseph, Dylan Kennedy, Adi Hanuka, Jane Shtalenkova, Auralee Edelen, P. Baxevanis, Adam Egger et al. "Bayesian optimization of a free-electron laser." Physical review letters 124, no. 12 (2020): 124801.
[3] Tran, Kevin, Willie Neiswanger, Junwoong Yoon, Qingyang Zhang, Eric Xing, and Zachary W. Ulissi. "Methods for comparing uncertainty quantifications for material property predictions." Machine Learning: Science and Technology 1, no. 2 (2020): 025006.
[4] Attia et al., "Closed-loop optimization of fast-charging protocols for batteries with machine learning", Nature, 2020
[5] Char, Ian, Youngseog Chung, Willie Neiswanger, Kirthevasan Kandasamy, Andrew O. Nelson, Mark Boyer, Egemen Kolemen, and Jeff Schneider. "Offline contextual Bayesian optimization." Advances in Neural Information Processing Systems 32 (2019).
[6] Facebook blog, "Efficient tuning of online systems using Bayesian optimization", Ben Letham, Brian Karrer, Guilherme Ottoni, Eytan Bakshy, September 27, 2018
[7] Kandasamy, K., Neiswanger, W., Schneider, J., Póczos, B., & Xing, E. P. "Neural architecture search with Bayesian optimisation and optimal transport". NeurIPS 2018.
[8] Van Aken, Dana, et al. "Automatic database management system tuning through large-scale machine learning." Proceedings of the 2017 ACM International Conference on Management of Data. 2017.

BACKGROUND

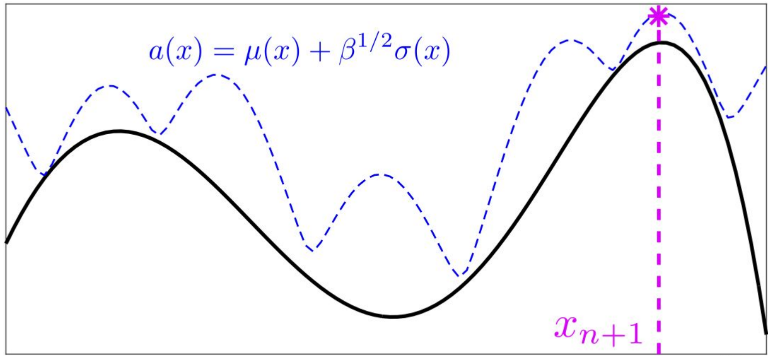
A popular method is **Bayesian optimization (BO)**

- Leverages a probabilistic model of f to sequentially choose queries.
- The model can:
  - incorporate prior beliefs about f (e.g. smoothness)
  - tell us where we are certain vs uncertain about f
- ⇒ Sample efficient optimization.

Probabilistic model of f



Use model to choose queries



Estimating other properties

In a variety of real-world tasks, there are **many other properties** of black-box functions that we also want to estimate:

Estimating other properties

In a variety of real-world tasks, there are **many other properties** of black-box functions that we also want to estimate:

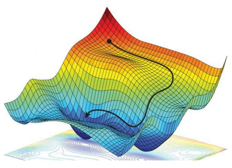
- Optimization variations (global/local/top-k optima)
- Multi-objective optimization (Pareto frontiers)
- Level set estimation (sublevel sets, superlevel sets)
- Search (subset w/ value matching some criteria)
- Phase identification (boundaries / partitions)
- Root finding / noisy bisection (roots)
- Quadrature (integrals, expectations, averages)
- Graph-structured estimation (shortest paths)
- Sensor placement (function value at set of locations)

BEYOND GLOBAL OPTIMA

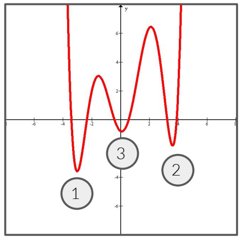
Estimating other properties

In a variety of real-world tasks, there are **many other properties** of black-box functions that we also want to estimate:

- Optimization variations (global/local/top-k optima)
- Multi-objective optimization (Pareto frontiers)
- Level set estimation (sublevel sets, superlevel sets)
- Search (subset w/ value matching some criteria)
- Phase identification (boundaries / partitions)
- Root finding / noisy bisection (roots)
- Quadrature (integrals, expectations, averages)
- Graph-structured estimation (shortest paths)
- Sensor placement (function value at set of locations)



Local optima



Top-k optima

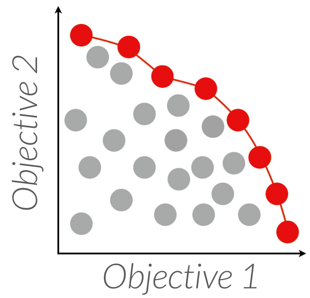
Applications

- *ML model training*
- *HPO / NAS*
- *Systems tuning*

Estimating other properties

In a variety of real-world tasks, there are **many other properties** of black-box functions that we also want to estimate:

- Optimization variations (global/local/top-k optima)
- **Multi-objective optimization (Pareto frontiers)**
- Level set estimation (sublevel sets, superlevel sets)
- Search (subset w/ value matching some criteria)
- Phase identification (boundaries / partitions)
- Root finding / noisy bisection (roots)
- Quadrature (integrals, expectations, averages)
- Graph-structured estimation (shortest paths)
- Sensor placement (function value at set of locations)



Applications

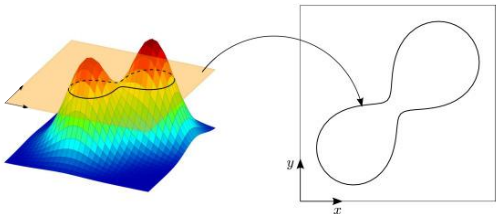
- *HPO / NAS*
- *Systems tuning*
- *Laboratory equipment / machines*

BEYOND GLOBAL OPTIMA

Estimating other properties

In a variety of real-world tasks, there are **many other properties** of black-box functions that we also want to estimate:

- Optimization variations (global/local/top-k optima)
- Multi-objective optimization (Pareto frontiers)
- **Level set estimation (sublevel sets, superlevel sets)**
- Search (subset w/ value matching some criteria)
- Phase identification (boundaries / partitions)
- Root finding / noisy bisection (roots)
- Quadrature (integrals, expectations, averages)
- Graph-structured estimation (shortest paths)
- Sensor placement (function value at set of locations)



Applications

- *Catalyst design*
- *Active learning / weak supervision*
- *Environmental monitoring*

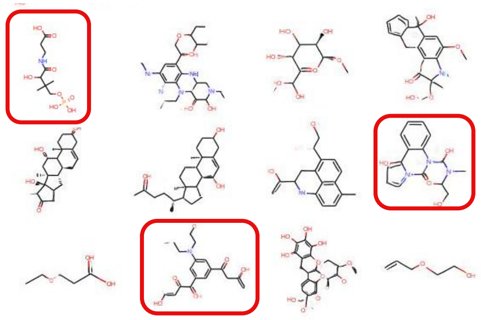
BEYOND GLOBAL OPTIMA

Estimating other properties

In a variety of real-world tasks, there are **many other properties** of black-box functions that we also want to estimate:

- Optimization variations (global/local/top-k optima)
- Multi-objective optimization (Pareto frontiers)
- Level set estimation (sublevel sets, superlevel sets)
- **Search (subset w/ value matching some criteria)**
- Phase identification (boundaries / partitions)
- Root finding / noisy bisection (roots)
- Quadrature (integrals, expectations, averages)
- Graph-structured estimation (shortest paths)
- Sensor placement (function value at set of locations)

(molecules)



Applications

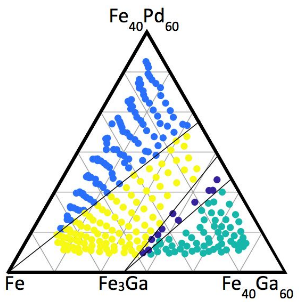
- *Drug discovery*
- *Fraud detection*
- *Targeted opinion polling*

Estimating other properties

In a variety of real-world tasks, there are **many other properties** of black-box functions that we also want to estimate:

- Optimization variations (global/local/top-k optima)
- Multi-objective optimization (Pareto frontiers)
- Level set estimation (sublevel sets, superlevel sets)
- Search (subset w/ value matching some criteria)
- **Phase identification (boundaries / partitions)**
- Root finding / noisy bisection (roots)
- Quadrature (integrals, expectations, averages)
- Graph-structured estimation (shortest paths)
- Sensor placement (function value at set of locations)

(compounds)



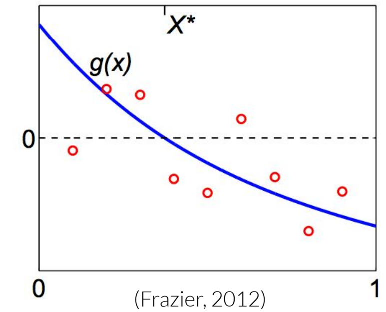
Applications

- High throughput materials design/discovery

Estimating other properties

In a variety of real-world tasks, there are **many other properties** of black-box functions that we also want to estimate:

- Optimization variations (global/local/top-k optima)
- Multi-objective optimization (Pareto frontiers)
- Level set estimation (sublevel sets, superlevel sets)
- Search (subset w/ value matching some criteria)
- Phase identification (boundaries / partitions)
- **Root finding / noisy bisection (roots)**
- Quadrature (integrals, expectations, averages)
- Graph-structured estimation (shortest paths)
- Sensor placement (function value at set of locations)



(Frazier, 2012)

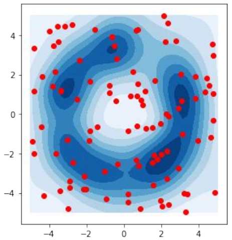
Applications

- *Target localization* (airborne radar)
- *Edge detection* (computer vision)
- *Biology / microbiology* (phase shifts)

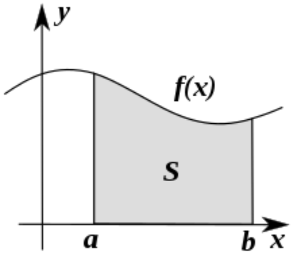
Estimating other properties

In a variety of real-world tasks, there are **many other properties** of black-box functions that we also want to estimate:

- Optimization variations (global/local/top-k optima)
- Multi-objective optimization (Pareto frontiers)
- Level set estimation (sublevel sets, superlevel sets)
- Search (subset w/ value matching some criteria)
- Phase identification (boundaries / partitions)
- Root finding / noisy bisection (roots)
- **Quadrature (integrals, expectations, averages)**
- Graph-structured estimation (shortest paths)
- Sensor placement (function value at set of locations)



(Emukit, 2021)



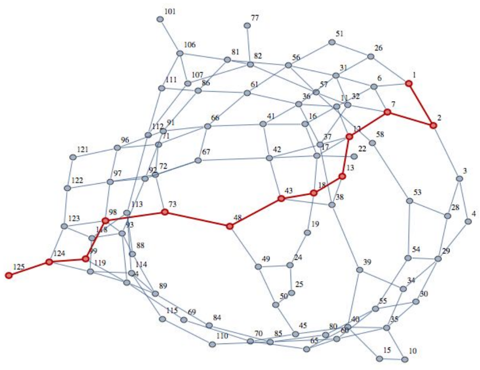
Applications

- *Probabilistic modeling (marginal distributions, normalization constants)*
- *Estimating center of mass (and centroids)*

Estimating other properties

In a variety of real-world tasks, there are **many other properties** of black-box functions that we also want to estimate:

- Optimization variations (global/local/top-k optima)
- Multi-objective optimization (Pareto frontiers)
- Level set estimation (sublevel sets, superlevel sets)
- Search (subset w/ value matching some criteria)
- Phase identification (boundaries / partitions)
- Root finding / noisy bisection (roots)
- Quadrature (integrals, expectations, averages)
- **Graph-structured estimation (shortest paths)**
- Sensor placement (function value at set of locations)



Applications

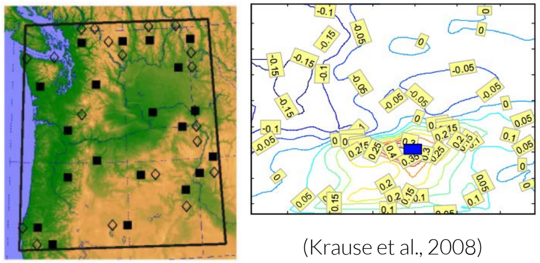
- *Transportation networks*
- *Shipping networks*
- *Social networks*

BEYOND GLOBAL OPTIMA

Estimating other properties

In a variety of real-world tasks, there are **many other properties** of black-box functions that we also want to estimate:

- Optimization variations (global/local/top-k optima)
- Multi-objective optimization (Pareto frontiers)
- Level set estimation (sublevel sets, superlevel sets)
- Search (subset w/ value matching some criteria)
- Phase identification (boundaries / partitions)
- Root finding / noisy bisection (roots)
- Quadrature (integrals, expectations, averages)
- Graph-structured estimation (shortest paths)
- **Sensor placement (function value at set of locations)**



(Krause et al., 2008)

Applications

- *Water distribution systems*
- *Outbreak detection in networks*
- *Weather monitoring*

Our goal

To develop methods to estimate a broad set of function properties within a limited budget, using probabilistic models.

⇒ Can view this as a generalization of Bayesian optimization to other function properties... *beyond global optima*.

First question:

How do we formalize *"other function properties"*?

ALGORITHMS w/o BUDGET CONSTRAINTS

Note that, given a function property of interest...

Often exists effective algorithms for computing (or numerically approximating) the property, ***if you ignore budget constraint***

ALGORITHMS w/o BUDGET CONSTRAINTS

Note that, given a function property of interest...

Often exists effective algorithms for computing (or numerically approximating) the property, ***if you ignore budget constraint***

Property: *local optima (close to some initial point) of f*

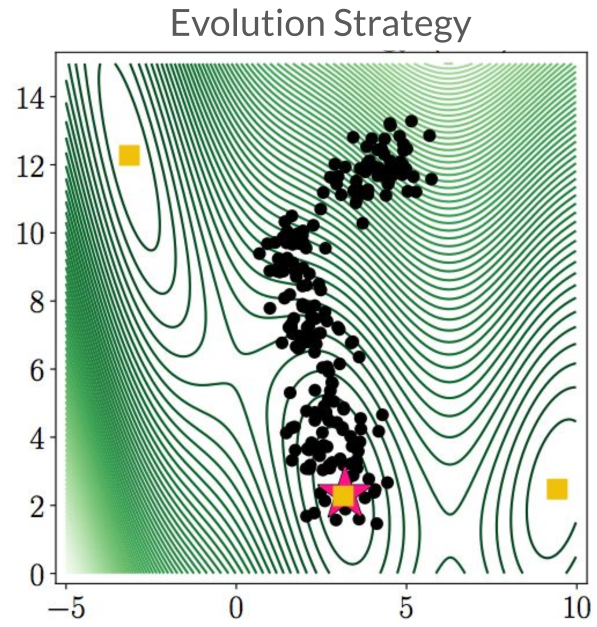
ALGORITHMS w/o BUDGET CONSTRAINTS

Note that, given a function property of interest...

Often exists effective algorithms for computing (or numerically approximating) the property, ***if you ignore budget constraint***

Property: *local optima (close to some initial point) of f*

Algorithm: *local optimization algorithm* → e.g. gradient descent, Nelder-Mead method, evolutionary algorithm, etc.



Note that, given a function property of interest...

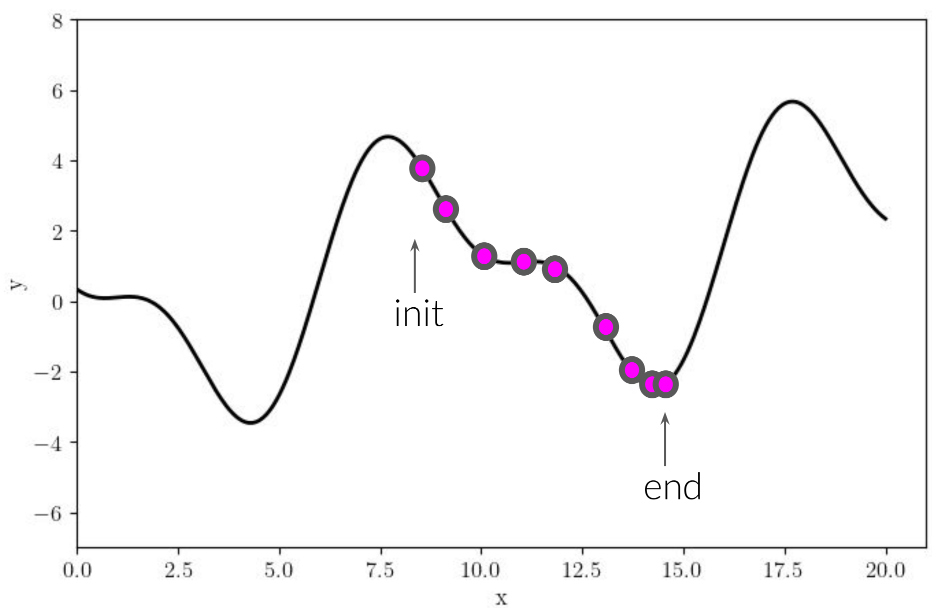
Often exists effective algorithms for computing (or numerically approximating) the property, ***if you ignore budget constraint***

Property: *local optima (close to some initial point) of f*

Algorithm: *local optimization algorithm*

e.g. gradient descent, Nelder-Mead method, evolutionary algorithm, etc.

⇒ initialize at some location, then run local minimizer. Return final query as output.



ALGORITHMS w/o BUDGET CONSTRAINTS

Note that, given a function property of interest...

Often exists effective algorithms for computing (or numerically approximating) the property, ***if you ignore budget constraint***

Property: *superlevel set of* f *(e.g. over a discrete space of items).*

ALGORITHMS w/o BUDGET CONSTRAINTS

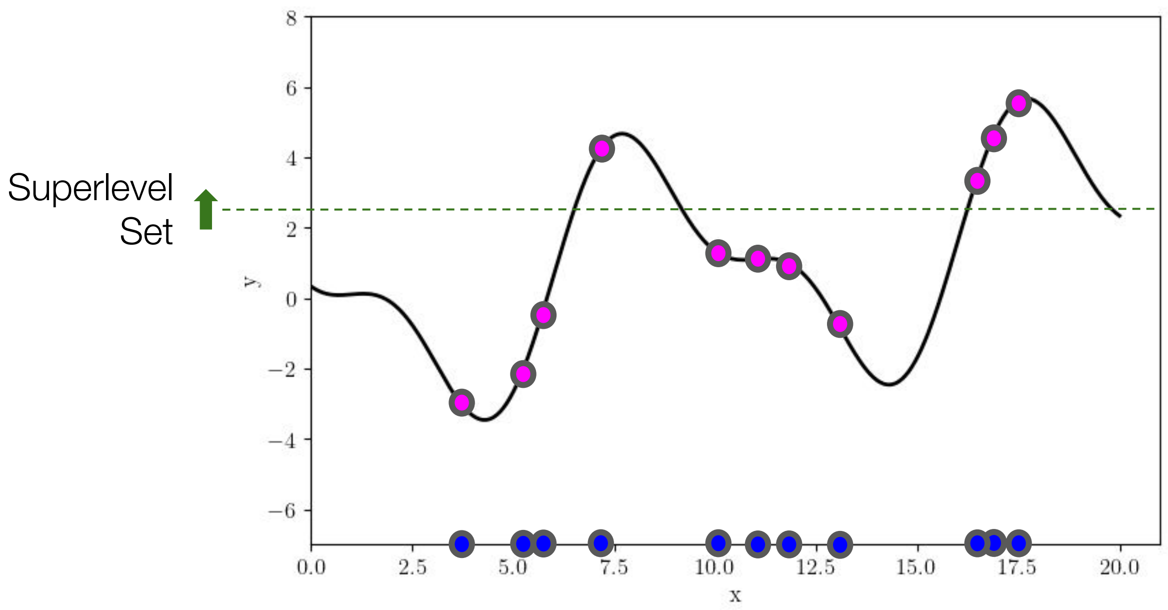
Note that, given a function property of interest...

Often exists effective algorithms for computing (or numerically approximating) the property, ***if you ignore budget constraint***

Property: *superlevel set of f (e.g. over a discrete space of items).*

Algorithm: *scan and threshold.*

⇒ Scan through each item ● , query its value ● , return subset of items above threshold.



ALGORITHMS w/o BUDGET CONSTRAINTS

Note that, given a function property of interest...

Often exists effective algorithms for computing (or numerically approximating) the property, ***if you ignore budget constraint***

Property: *integral or expectation of* f.

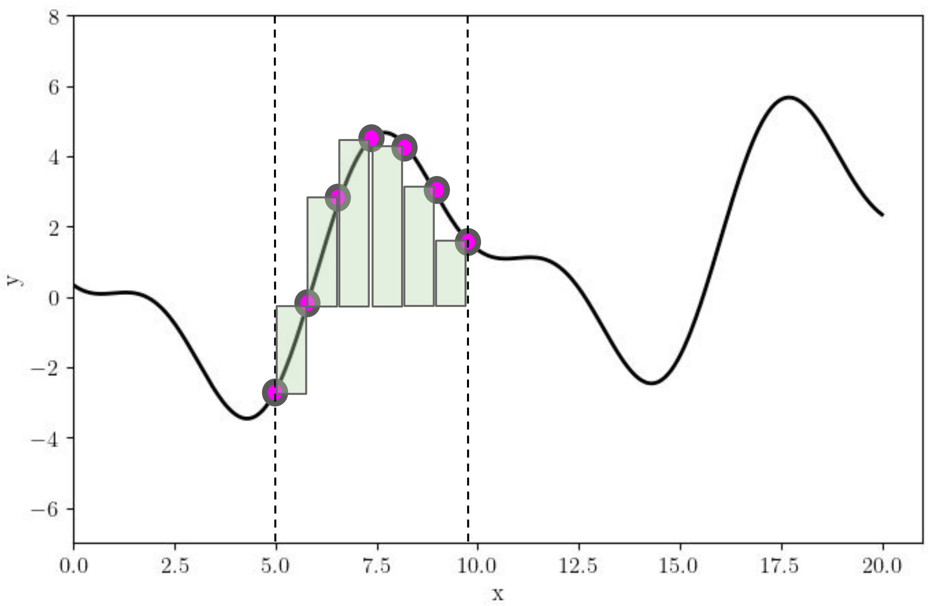
Note that, given a function property of interest...

Often exists effective algorithms for computing (or numerically approximating) the property, ***if you ignore budget constraint***

Property: *integral or expectation of f.*

Algorithm: *numerical integration (e.g. rectangle/trapezoidal approximation).*

⇒ Run numerical integration *(e.g. rectangle/trapezoidal approximation)*. Return approximate integral over region.



Definition: *computable function property*

The output of a given algorithm A, if it were run on our black-box function f.

⇒ E.g. previous properties are all *computable function properties*:
local optima, integrals, level sets, Pareto frontiers, partitions — and many others, defined by an algorithm!

DEFINITIONS

Definition: *computable function property*

The output of a given algorithm A, if it were run on our black-box function f.

⇒ E.g. previous properties are all *computable function properties*:
local optima, integrals, level sets, Pareto frontiers, partitions — and many others, defined by an algorithm!

Definition: *Bayesian algorithm execution (BAX)*

The task of estimating a *computable function property* (output of an algorithm A), using a budget of only T queries to f.

(Even if algorithm A requires far more than T queries.)

DEFINITIONS

Definition: *computable function property*

The output of a given algorithm A, if it were run on our black-box function f.

⇒ E.g. previous properties are all *computable function properties*:
local optima, integrals, level sets, Pareto frontiers, partitions — and many others, defined by an algorithm!

Definition: *Bayesian algorithm execution (BAX)*

The task of estimating a *computable function property* (output of an algorithm A), using a budget of only T queries to f.

(Even if algorithm A requires far more than T queries.)

Note: two main reasons to frame *function property* in terms of an algorithm:

(1) gives a flexible way to define function properties.

(2) we will use algorithm in our procedure to estimate these properties.

Methods for BAX

Information-based method for BAX

InfoBAX — an algorithm for BAX, based on info-theoretic methods for BO.

Information-based method for BAX

InfoBAX — an algorithm for BAX, based on info-theoretic methods for BO.

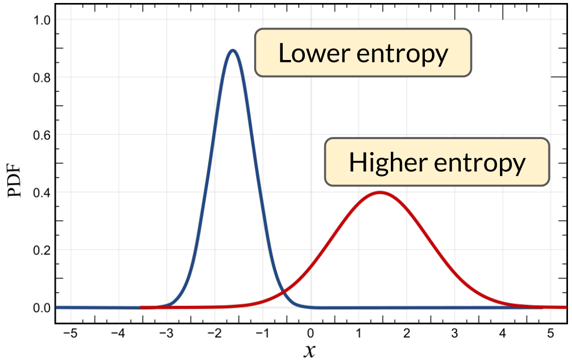
Some relevant background

There exist a few popular info-based methods for BO:

- E.g. entropy search (ES), predictive ES, max-value ES.
- Rooted in **Bayesian optimal experimental design (BOED).**

BOED: have model with an (unknown) parameter of interest.

- Choose experiments that most reduce uncertainty about parameter.
- *Uncertainty*: **entropy** of posterior distribution over parameter.



Some BOED History

Reducing theodolite measurements for surveying.

Information-based method for BAX

InfoBAX — an algorithm for BAX, based on info-theoretic methods for BO.

Some relevant background

There exist a few popular info-based methods for BO:

- E.g. entropy search (ES), predictive ES, max-value ES.
- Rooted in **Bayesian optimal experimental design (BOED).**

BOED: have model with an (unknown) parameter of interest.

- Choose experiments that most reduce uncertainty about parameter.
- *Uncertainty:* **entropy** of posterior distribution over parameter.

Some BOED History

Reducing theodolite measurements for surveying.

To describe *InfoBAX*:

(1) Describe info-based BO.
(2) Extend it to info-based BAX.

Information-based Bayesian Optimization

Algorithm 1 BAYESIAN OPTIMIZATION

Information-based Bayesian Optimization

Algorithm 1 BAYESIAN OPTIMIZATION

Input: dataset $\mathcal{D}_1$, prior distribution $p(f)$ → Initial dataset of (x, y) pairs (function observations) — can be empty set.

Information-based Bayesian Optimization

Algorithm 1 BAYESIAN OPTIMIZATION

Input: dataset $\mathcal{D}_1$, prior distribution $p(f)$

1: **for** $t = 1, \ldots, T$ **do**

Over a sequence of T iterations:

Information-based Bayesian Optimization

Algorithm 1 BAYESIAN OPTIMIZATION

Input: dataset $\mathcal{D}_1$, prior distribution $p(f)$

1: **for** $t = 1, \ldots, T$ **do**

2: $\quad x_t \leftarrow \arg\max_{x \in \mathcal{X}} \alpha_t(x)$

Optimize an *acquisition function*.

- aims to capture value of querying f at an x.
- defined using our probabilistic model.

$\Rightarrow$ Chooses next x to query.

Information-based Bayesian Optimization

Algorithm 1 BAYESIAN OPTIMIZATION

Input: dataset $\mathcal{D}_1$, prior distribution $p(f)$

1: **for** $t = 1, \ldots, T$ **do**

2: $x_t \leftarrow \arg\max_{x \in \mathcal{X}} \alpha_t(x)$

3: $y_{x_t} \sim f(x_t) + \epsilon$

Query f on chosen x, observe y

Information-based Bayesian Optimization

Algorithm 1 BAYESIAN OPTIMIZATION

Input: dataset $\mathcal{D}_1$, prior distribution $p(f)$

1: **for** $t = 1, \ldots, T$ **do**

2: $\quad x_t \leftarrow \arg\max_{x \in \mathcal{X}} \alpha_t(x)$

3: $\quad y_{x_t} \sim f(x_t) + \epsilon$

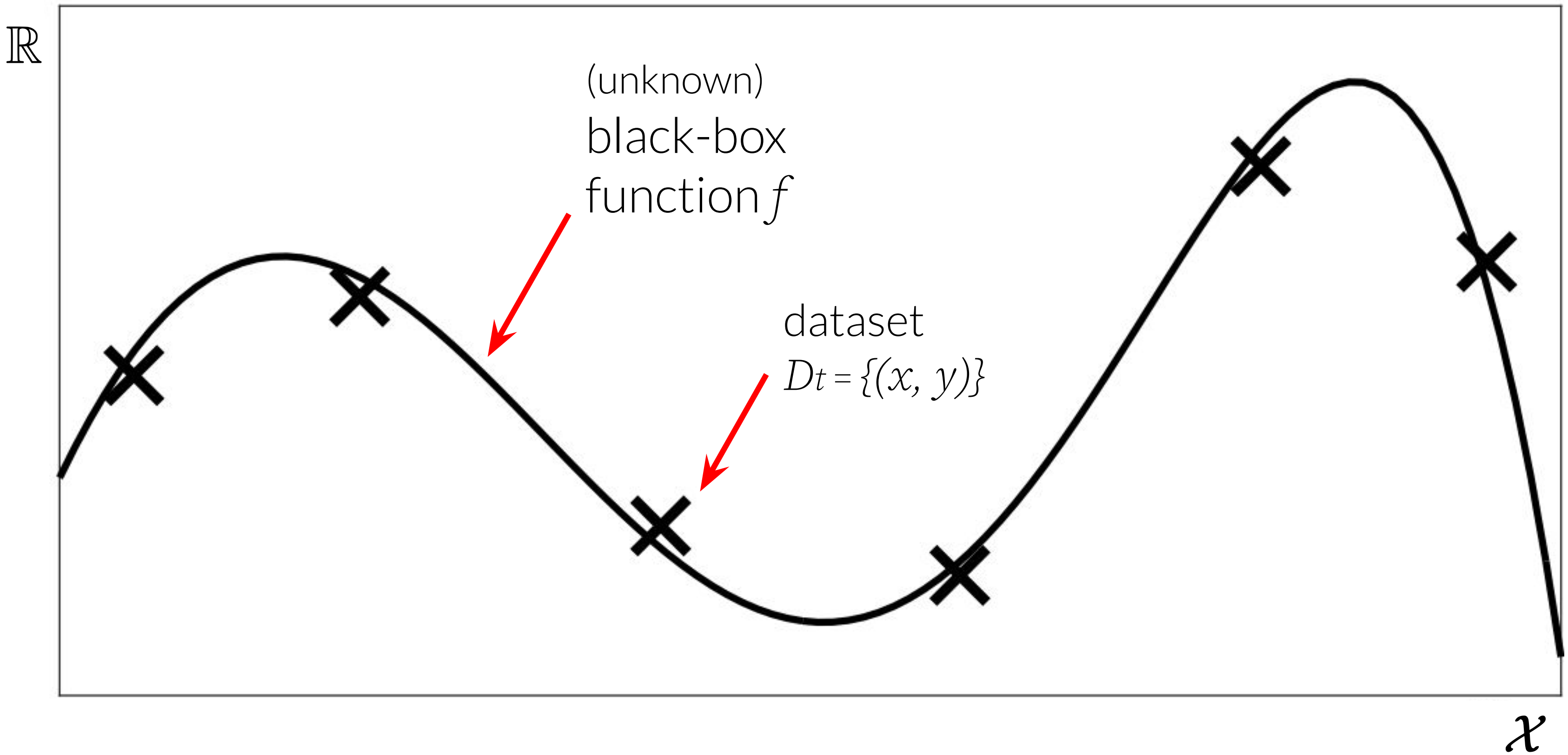
4: $\quad \mathcal{D}_{t+1} \leftarrow \mathcal{D}_t \cup \{(x_t, y_{x_t})\}$ → Update dataset with new *(x, y)* pair

Output: final dataset $\mathcal{D}_{T+1}$

Information-based Bayesian Optimization

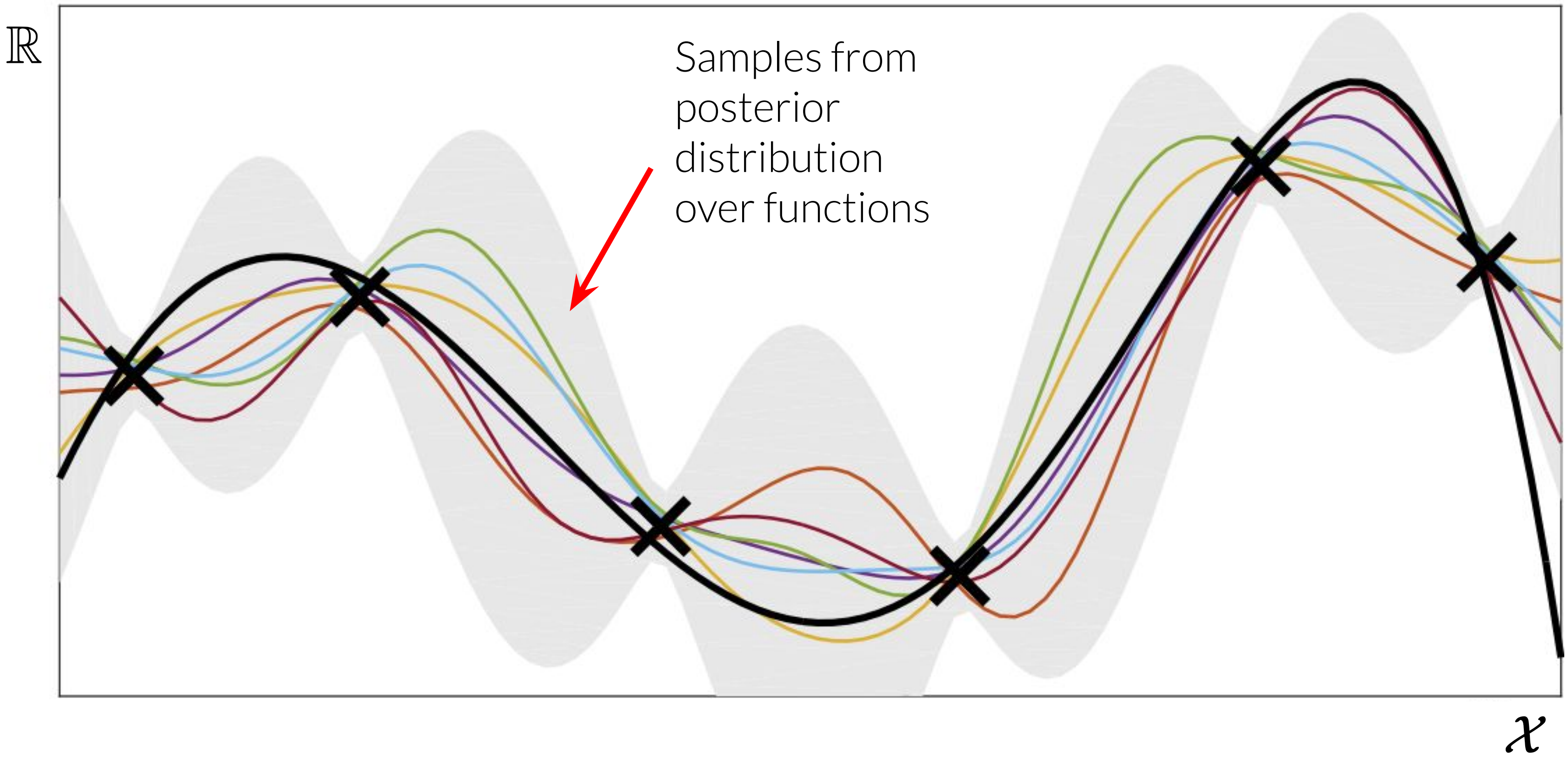
Visualizing this ...

BACKGROUND



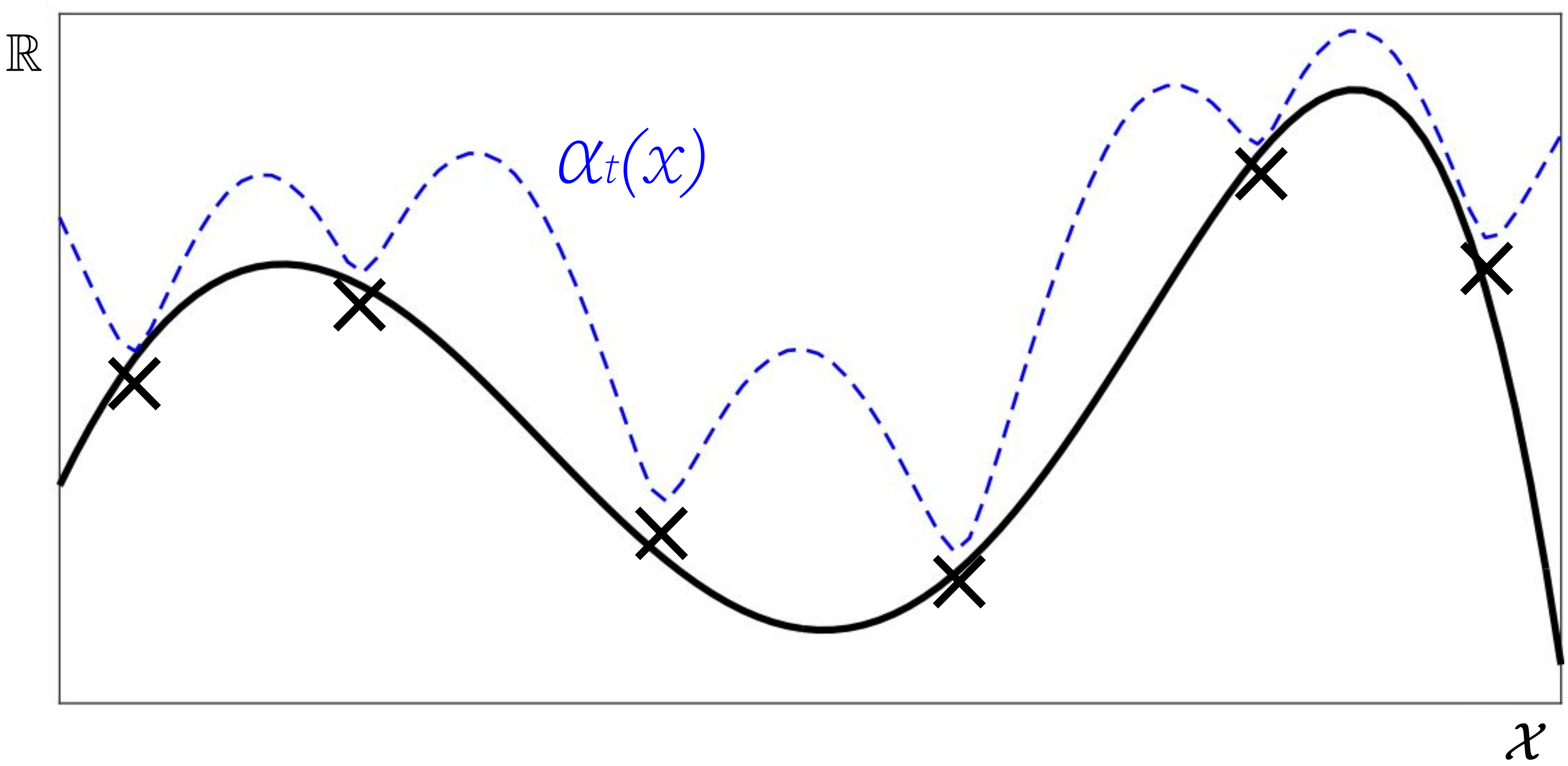
Unknown black-box f, and dataset of (x, y) pairs.

BACKGROUND

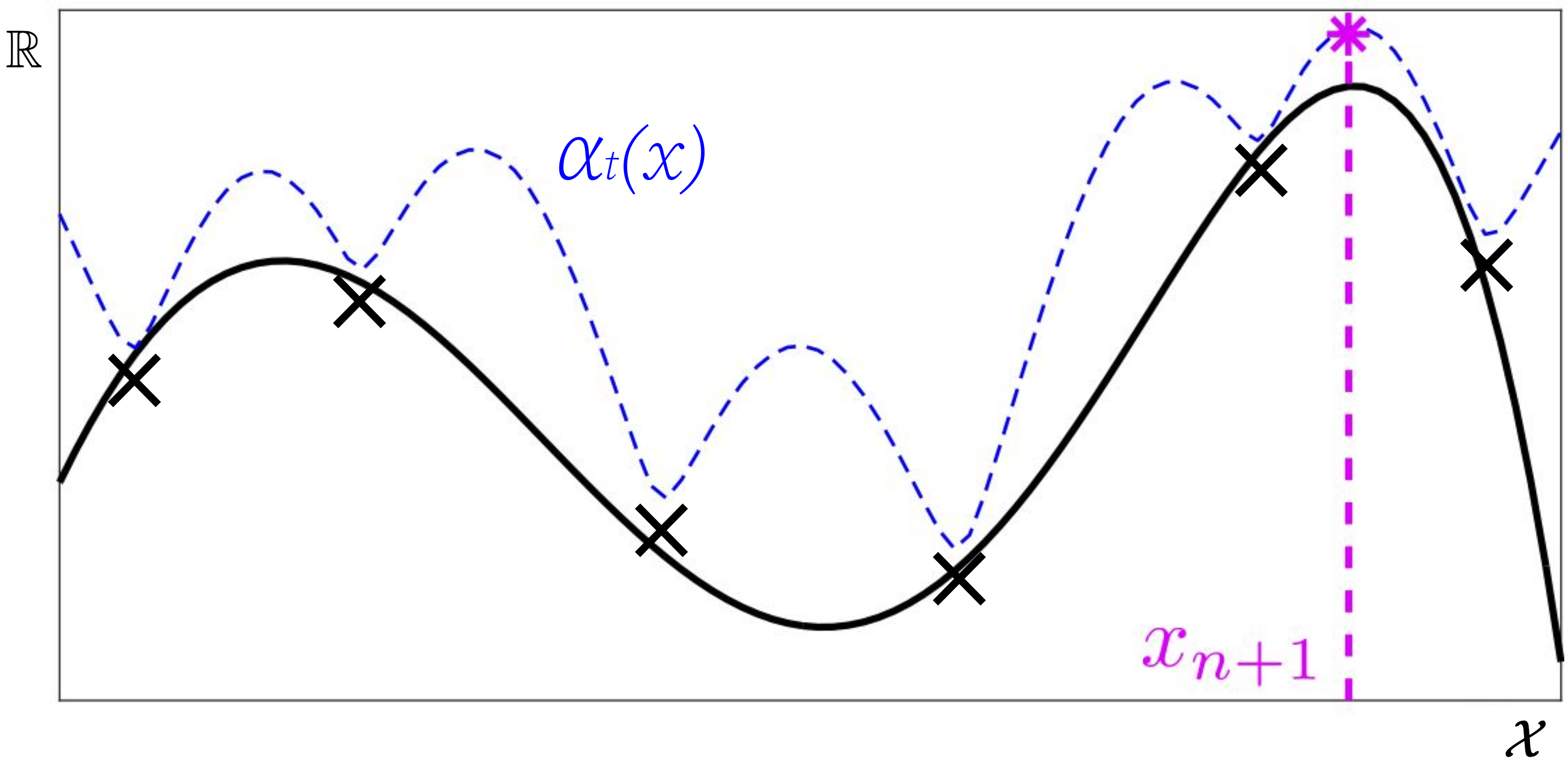


Given this dataset, can infer f using a probabilistic model.

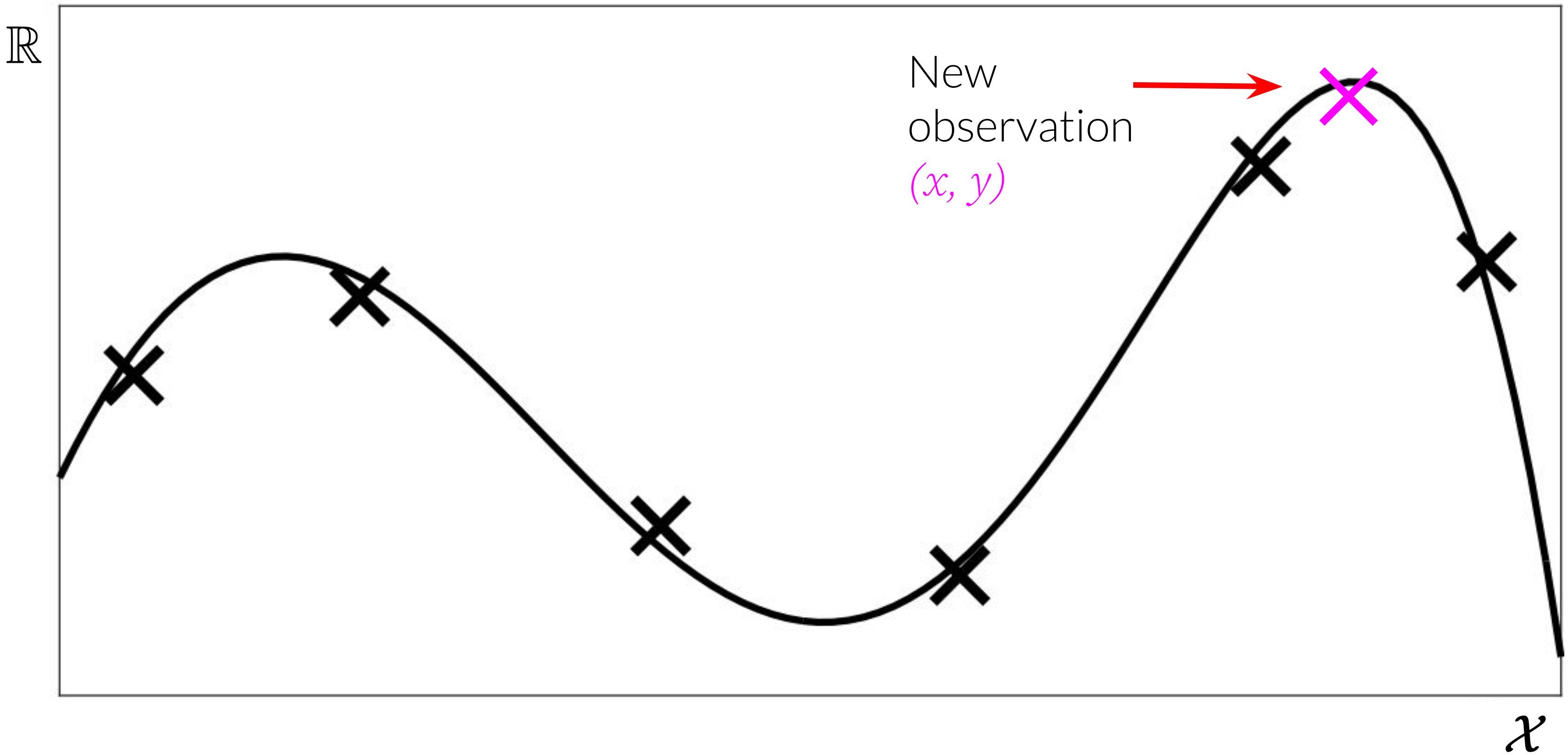
BACKGROUND



Define acquisition function using probabilistic model.



Optimize acquisition function ⇒ yields next point to query.



Query black-box f at x, observe y, and update dataset.

Information-based Bayesian Optimization

… Key step is line 2: *defining and optimizing acquisition function.*

Algorithm 1 BAYESIAN OPTIMIZATION

Input: dataset $\mathcal{D}_1$, prior distribution $p(f)$, algorithm $\mathcal{A}$

1: **for** $t = 1, \ldots, T$ **do**

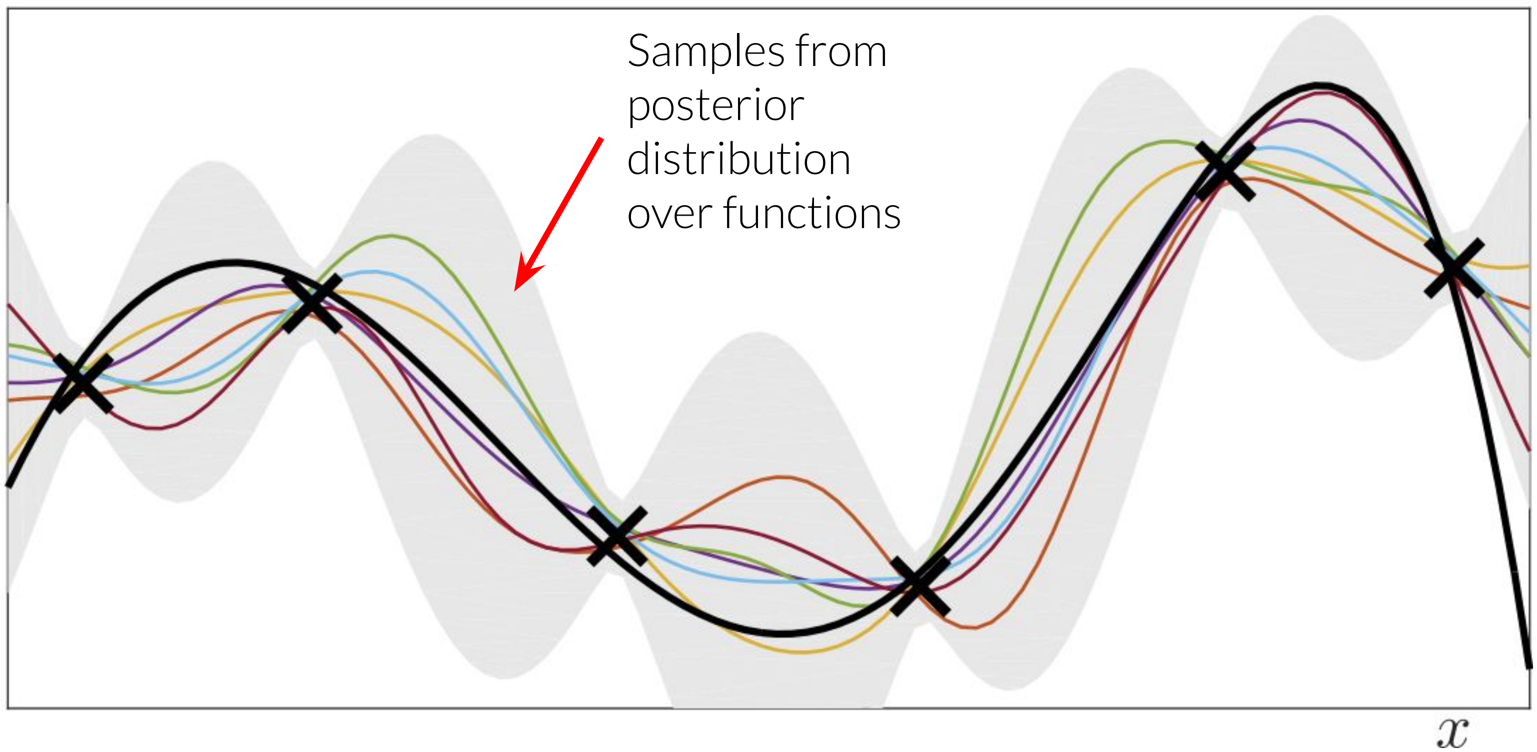
2: $\quad x_t \leftarrow \arg\max_{x \in \mathcal{X}} \alpha_t(x)$

3: $\quad y_{x_t} \sim f(x_t) + \epsilon$

4: $\quad \mathcal{D}_{t+1} \leftarrow \mathcal{D}_t \cup \{(x_t, y_{x_t})\}$

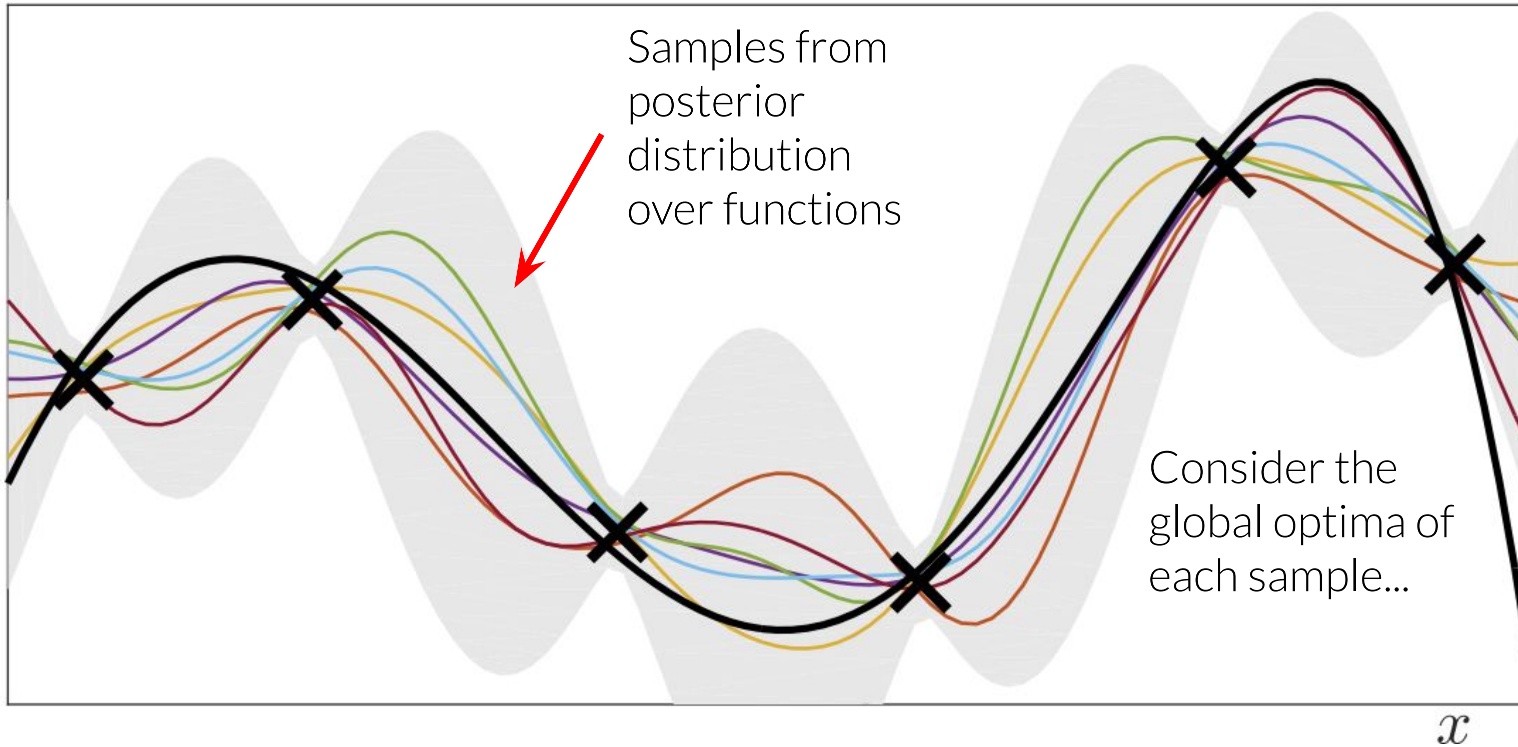
Output: final dataset $\mathcal{D}_{T+1}$

Acquisition function — info-based BO



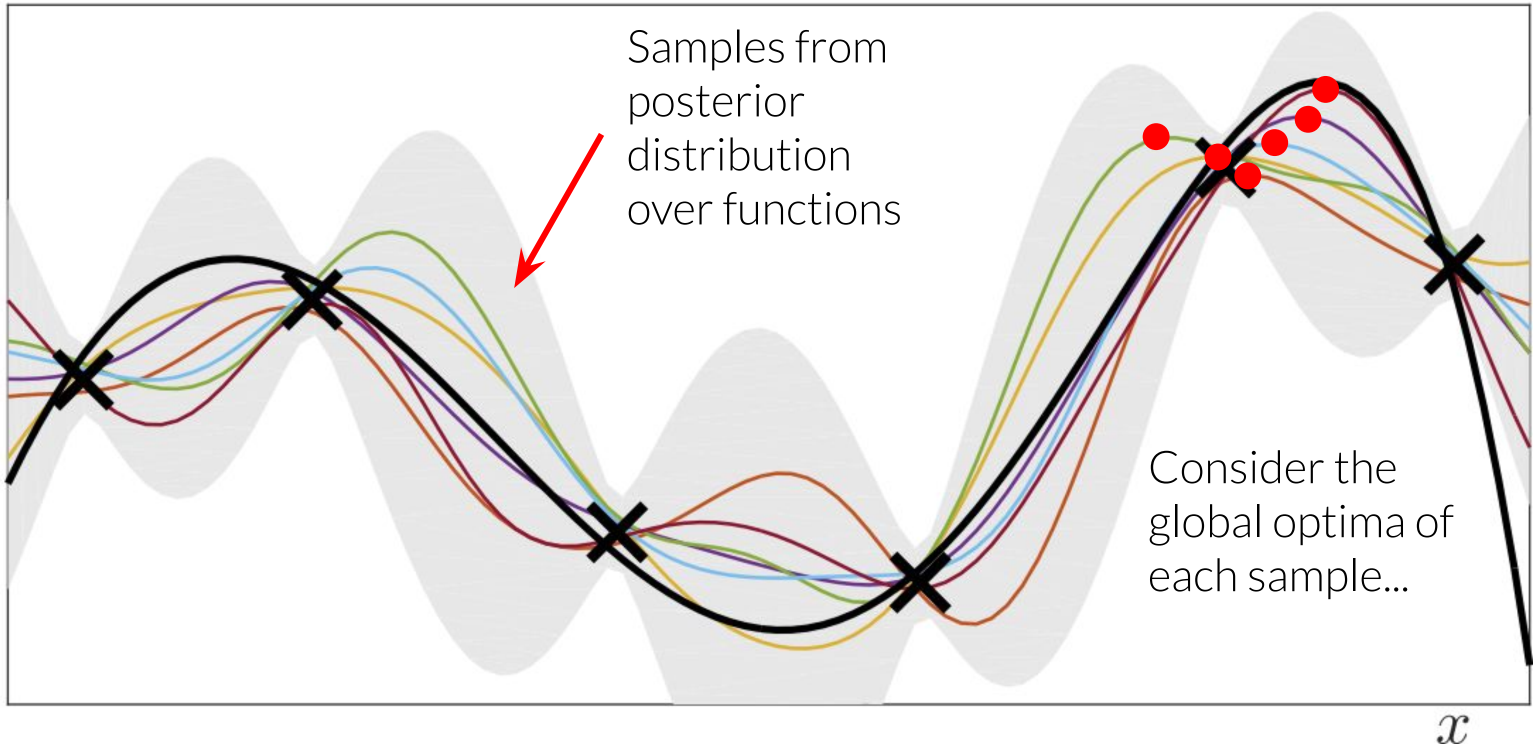
Probabilistic model of f, given dataset.

Acquisition function — info-based BO



Probabilistic model of f, given dataset.

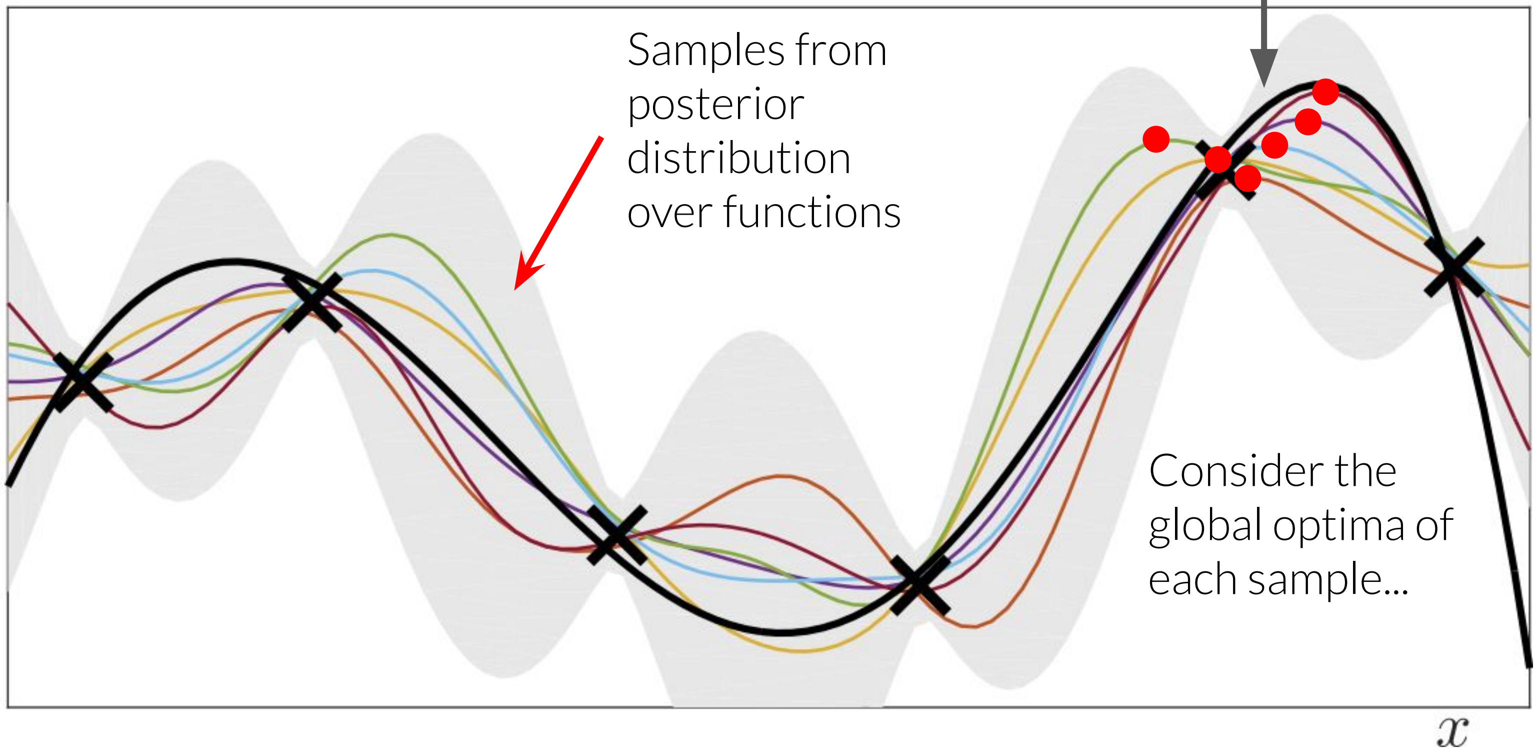
Acquisition function — info-based BO



Probabilistic model of f, given dataset.

Acquisition function — info-based BO

There is a posterior distribution over global optima induced by probabilistic model: $p(x^* \mid \mathcal{D}_t)$



Probabilistic model of f, given dataset.

Acquisition function — info-based BO

This leads us to the acquisition function:

e.g. used in entropy search (ES), predictive entropy search (PES)

Acquisition function — info-based BO

This leads us to the acquisition function:

e.g. used in entropy search (ES), predictive entropy search (PES)

$$\alpha_t(x) = H[p(x^* \mid \mathcal{D}_t)] - \mathbb{E}_{p(y_x \mid \mathcal{D}_t)}[H[p(x^* \mid \mathcal{D}_t \cup \{(x, y_x)\})]]$$

entropy of posterior distribution over global optima x^*

minus

expected entropy of posterior distribution over global optima x^*, *... if we were to make a query at x*

Acquisition function — info-based BO

This leads us to the acquisition function:

e.g. used in entropy search (ES), predictive entropy search (PES)

$$\alpha_t(x) = H[p(x^* \mid \mathcal{D}_t)] - \mathbb{E}_{p(y_x \mid \mathcal{D}_t)}[H[p(x^* \mid \mathcal{D}_t \cup \{(x, y_x)\})]]$$

entropy of posterior distribution over global optima x^*

minus

expected entropy of posterior distribution over global optima x^*, … *if we were to make a query at* x

"*Expected information gain*" (EIG) — expected decrease in entropy if we were to query f at x.

There exists a clever way to compute/optimize this (from work on PES)…

Acquisition function — info-based BO

How to compute and optimize it? **Two stages:**

Acquisition function — info-based BO

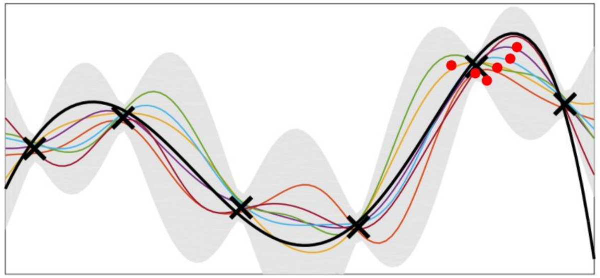
How to compute and optimize it? **Two stages:**

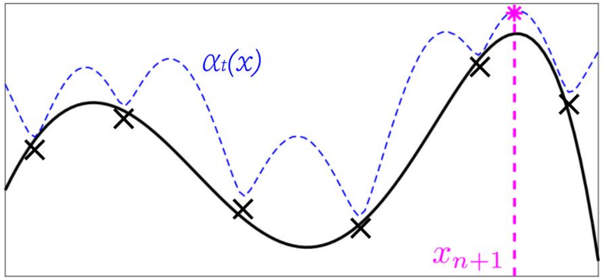
1) Before acquisition optimization:

- Generate posterior samples of global optima x^*.
- (⇒ run optimization algorithm on function samples to get optima).

2) Acquisition optimization:

- For any x, approximate EIG $\alpha_t(x)$ using these samples.
- Allows us to optimize acquisition function.





Benefits: generate samples (of x^*) only once. Then cheaper during iterative acquisition opt.

(*previous was existing work, following is new*)

What acquisition function do we use for *InfoBAX*?

Recall goal of BAX:

- Estimate a computable function property using a limited budget of queries.
- (*equivalently:* Estimate output of algorithm $\mathcal{A}$.)

(previous was existing work, following is new)

What acquisition function do we use for *InfoBAX*?

Recall goal of BAX:

- Estimate a computable function property using a limited budget of queries.
- (*equivalently*: Estimate output of algorithm A.)

Similar to info-based BO, *take a BOED strategy*:

- Denote the output of algorithm A (computable function property): $O_{\mathcal{A}}$
- We care about posterior over output: $p(O_{\mathcal{A}} \mid \mathcal{D}_t)$
- And its entropy: $H[p(O_{\mathcal{A}} \mid \mathcal{D}_t)]$

Want to make queries to best reduce this uncertainty over algorithm output.

InfoBAX acquisition function

Can also define an expected information gain (EIG) acquisition function:

InfoBAX acquisition function

Can also define an expected information gain (EIG) acquisition function:

$$\alpha_t(x) = H[p(O_{\mathcal{A}} \mid \mathcal{D}_t)] - \mathbb{E}_{p(y_x \mid \mathcal{D}_t)}[H[p(O_{\mathcal{A}} \mid \mathcal{D}_t \cup \{(x, y_x)\})]]$$

entropy of posterior distribution over algorithm output

minus

expected entropy of posterior distribution over algorithm output, *... if we were to make a query at x*

"Expected decrease in entropy on the ***algorithm output****, if we were to query f at x."*

InfoBAX acquisition function

Can also define an expected information gain (EIG) acquisition function:

$$\alpha_t(x) = H[p(O_{\mathcal{A}} \mid \mathcal{D}_t)] - \mathbb{E}_{p(y_x \mid \mathcal{D}_t)}[H[p(O_{\mathcal{A}} \mid \mathcal{D}_t \cup \{(x, y_x)\})]]$$

entropy of posterior distribution over algorithm output

minus

expected entropy of posterior distribution over algorithm output,
… *if we were to make a query at x*

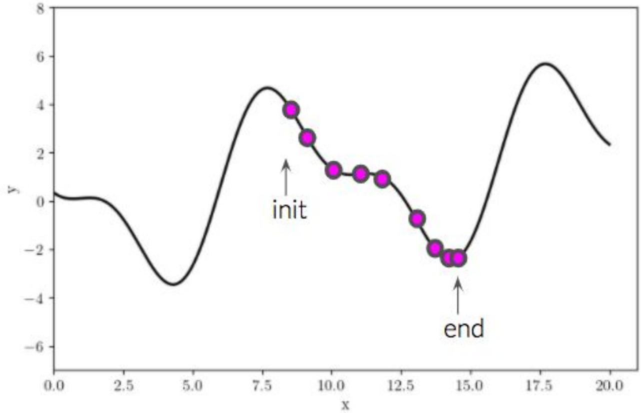
"Expected decrease in entropy on the ***algorithm output****, if we were to query f at x."*

… how can we compute (and optimize) this?

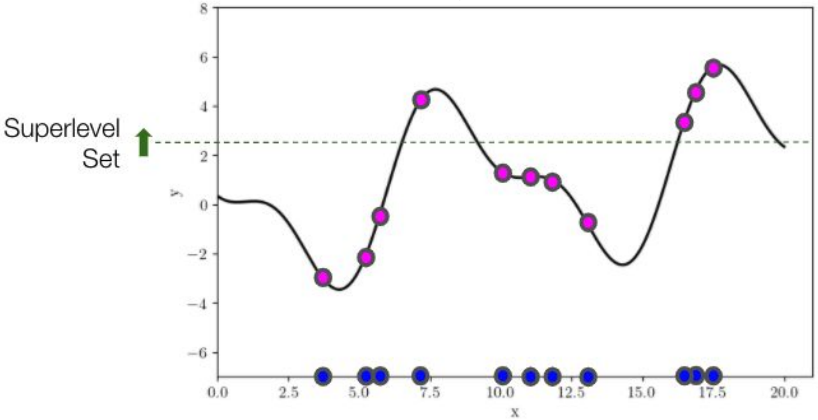
Definition:

Define the ***execution path of algorithm*** A as the sequence of queries ((x, y) pairs) that A would make on the black-box function f.

Local Optima



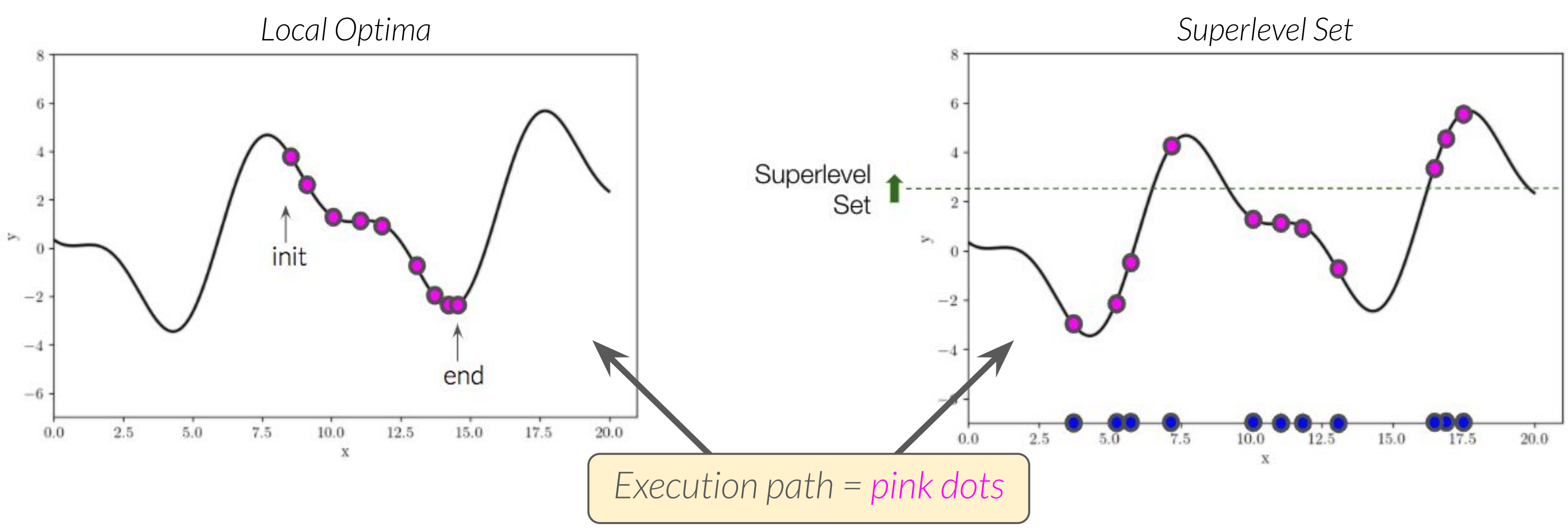
Superlevel Set



BAYESIAN ALGORITHM EXECUTION

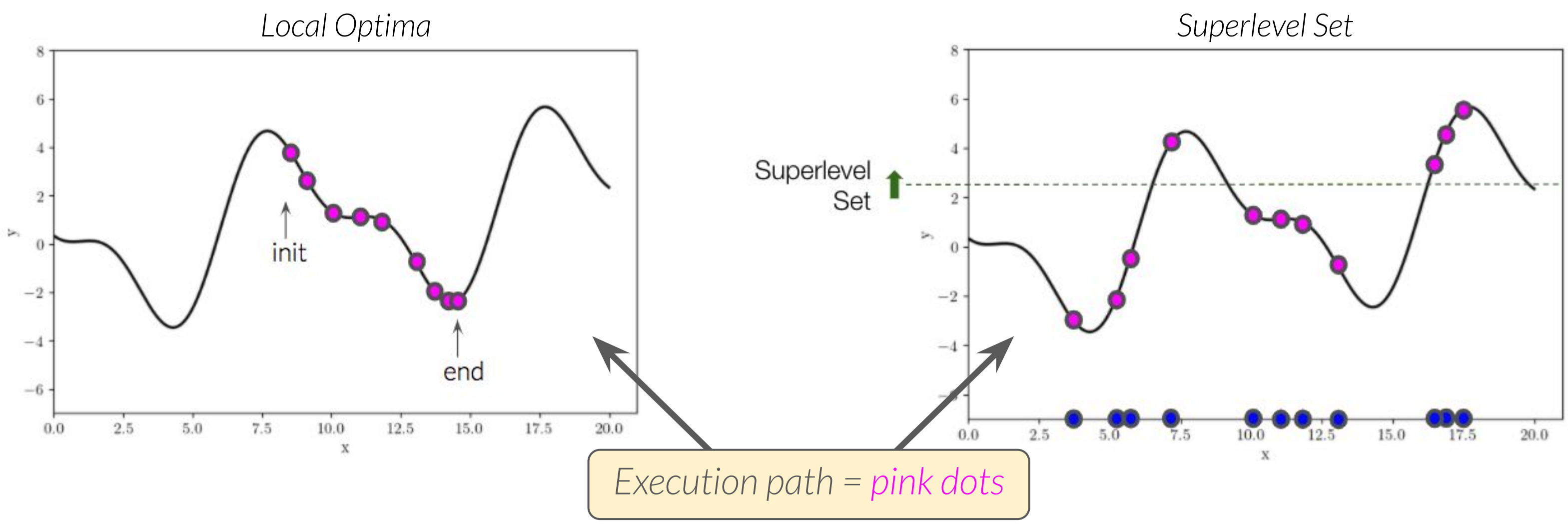
Definition:

Define the ***execution path of algorithm*** A as the sequence of queries ((x, y) pairs) that A would make on the black-box function f.



Definition:

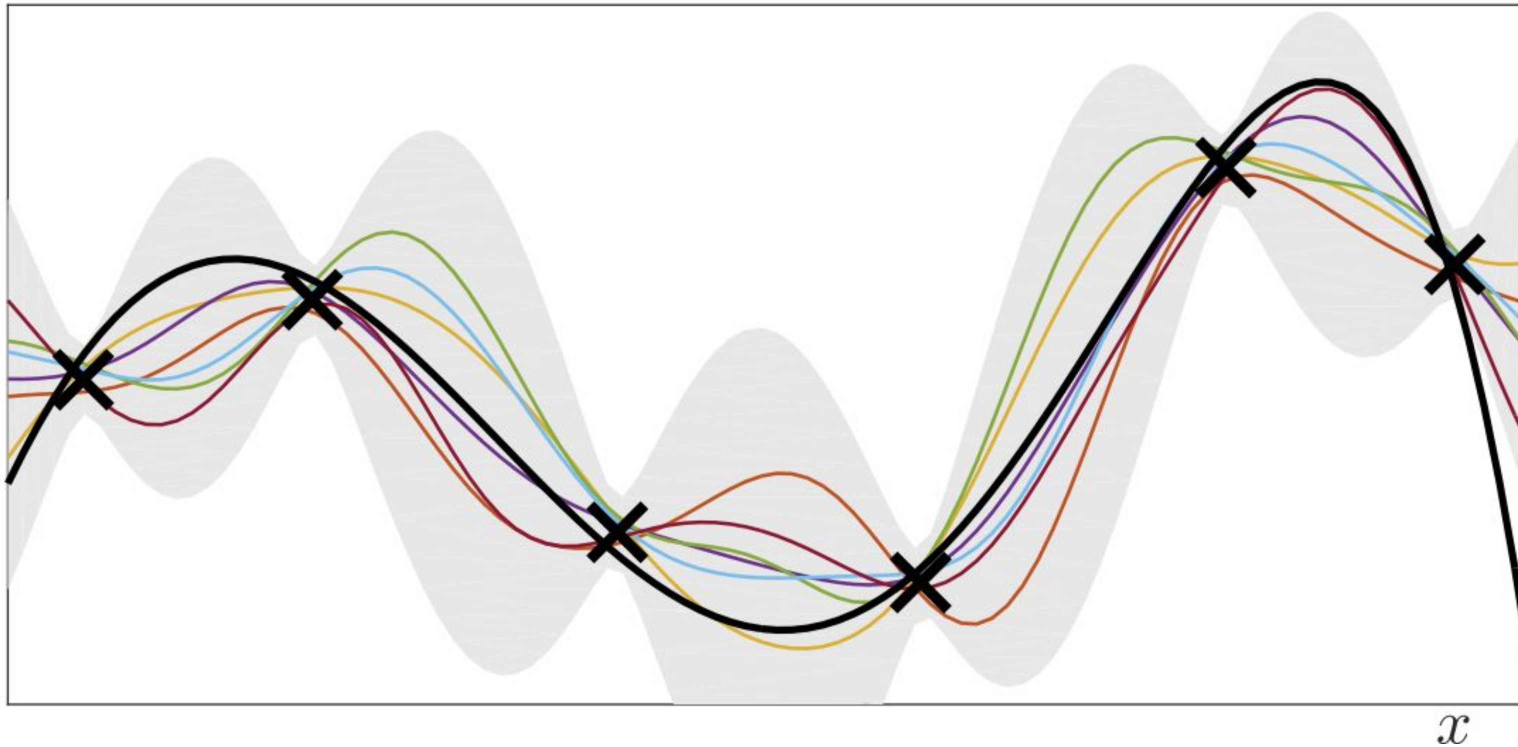
Define the ***execution path of algorithm*** A as the sequence of queries ((x, y) pairs) that A would make on the black-box function f.



Note: we don't know true execution path.

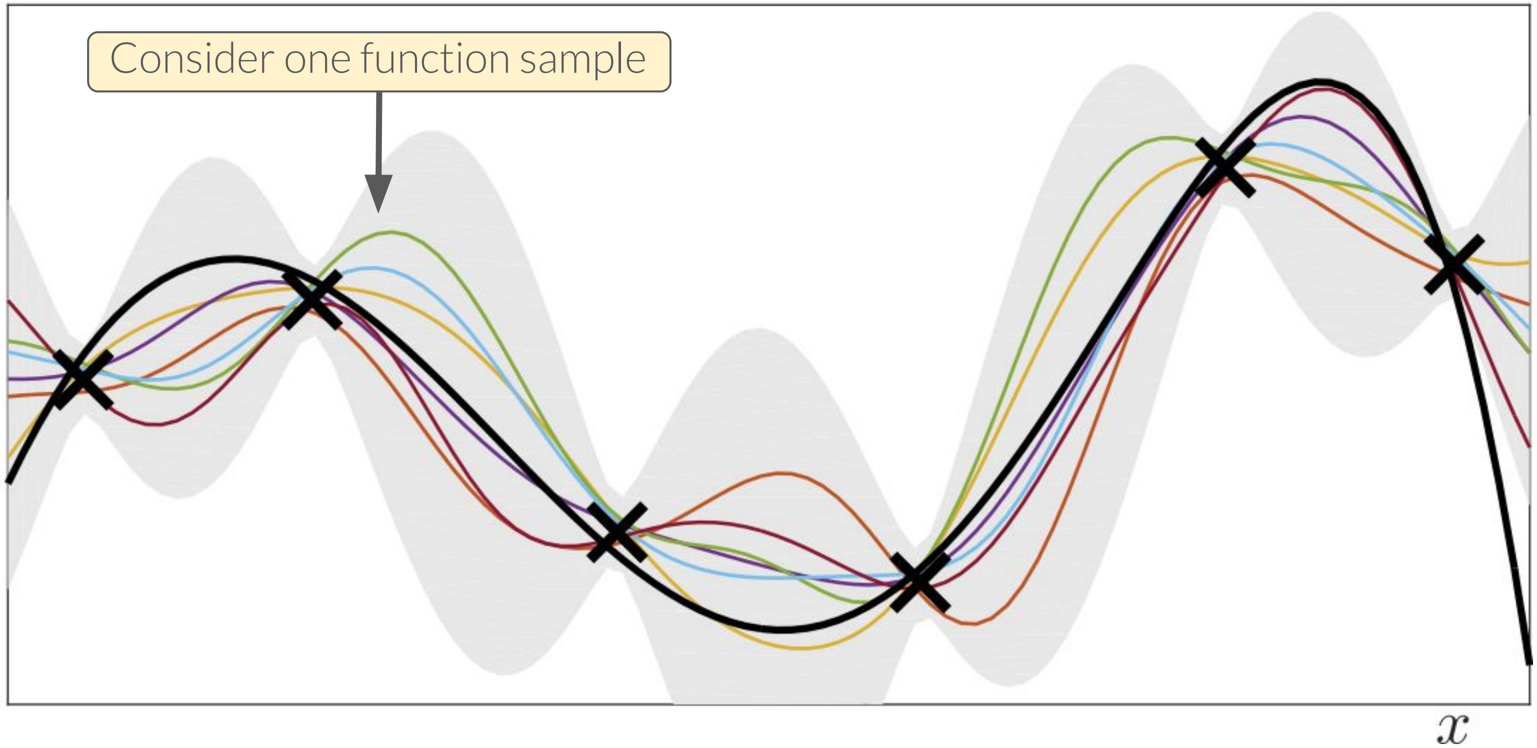
- (Since we are not running A on $f \Rightarrow$ this require too many queries)
- But given a model for f, we have a *posterior distribution over execution paths*

InfoBAX acquisition function

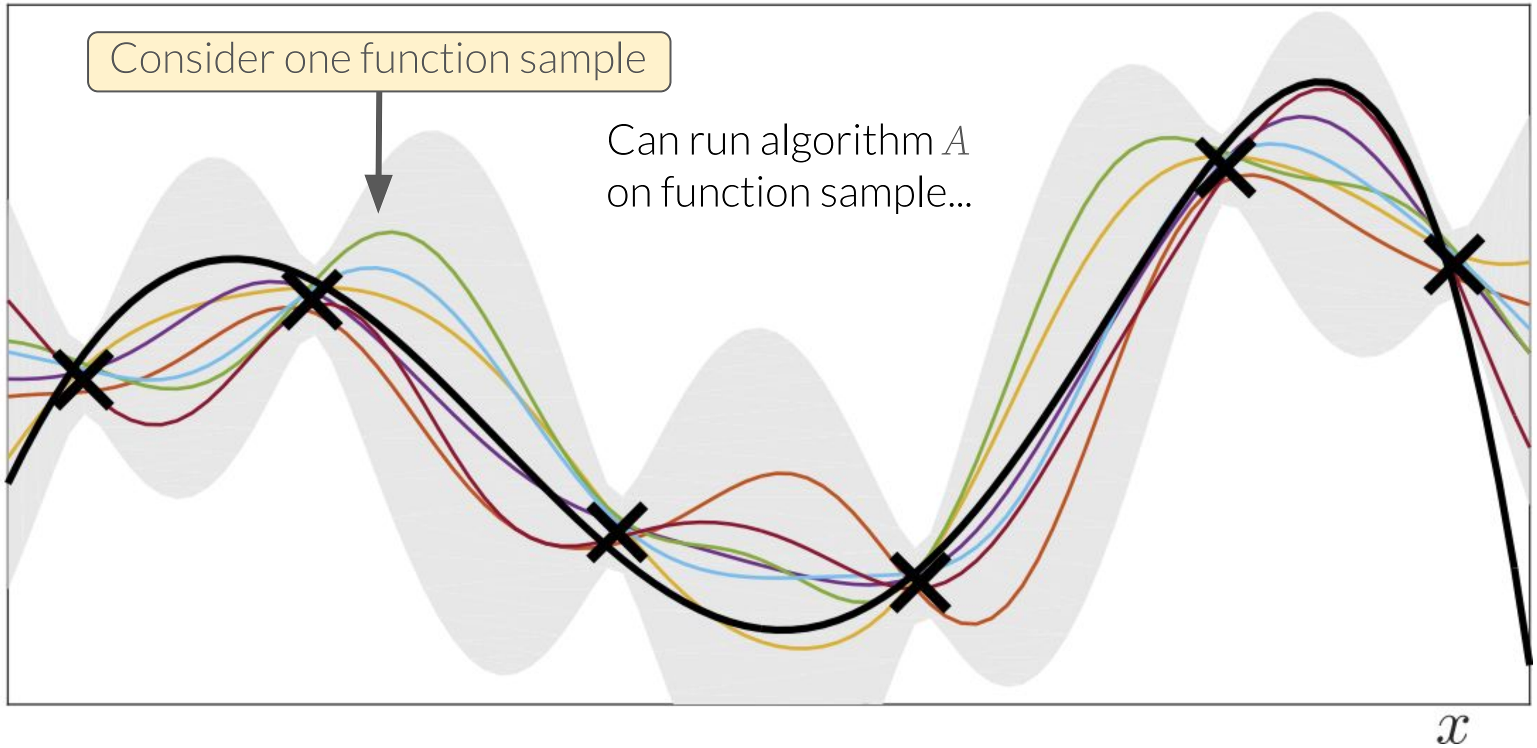


Probabilistic model of f, given dataset.

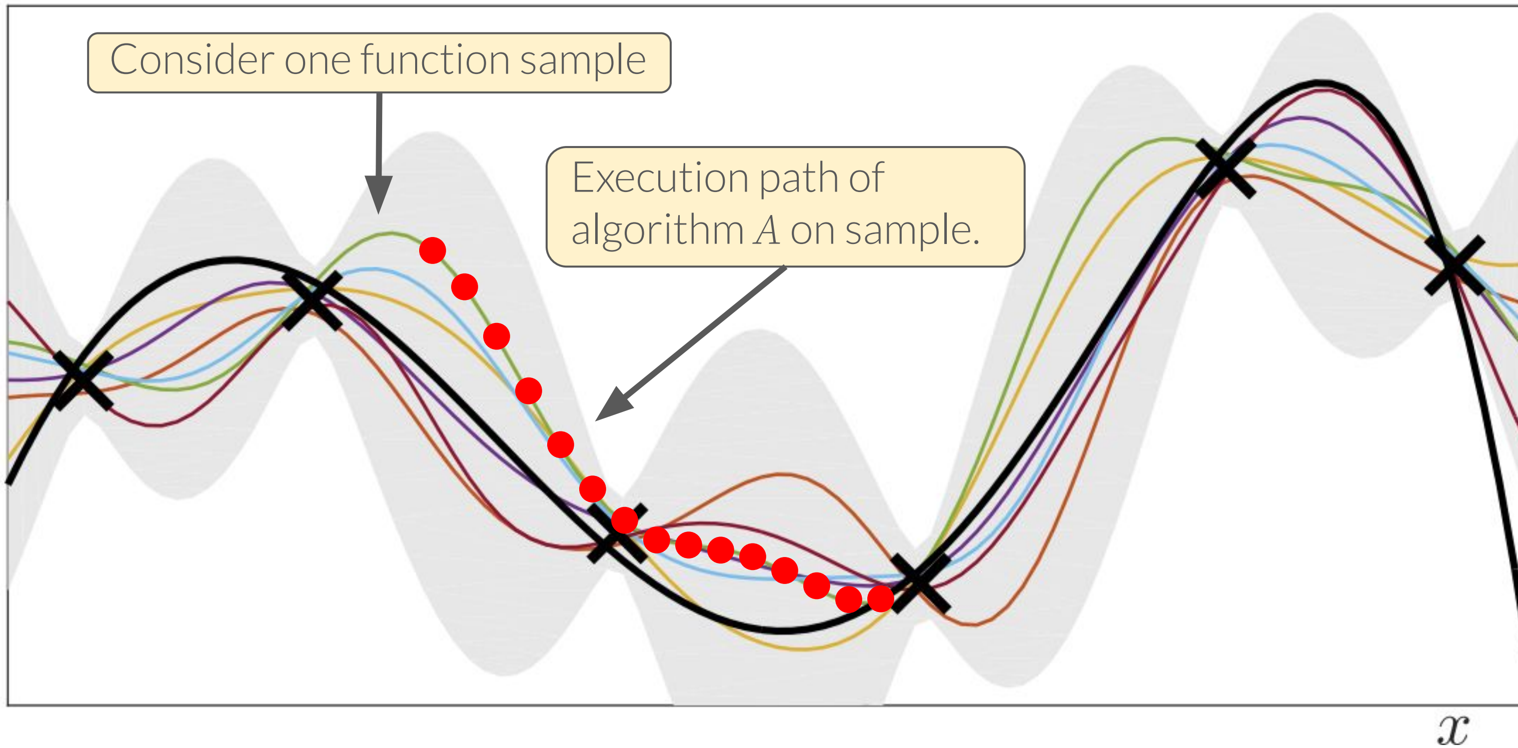
InfoBAX acquisition function



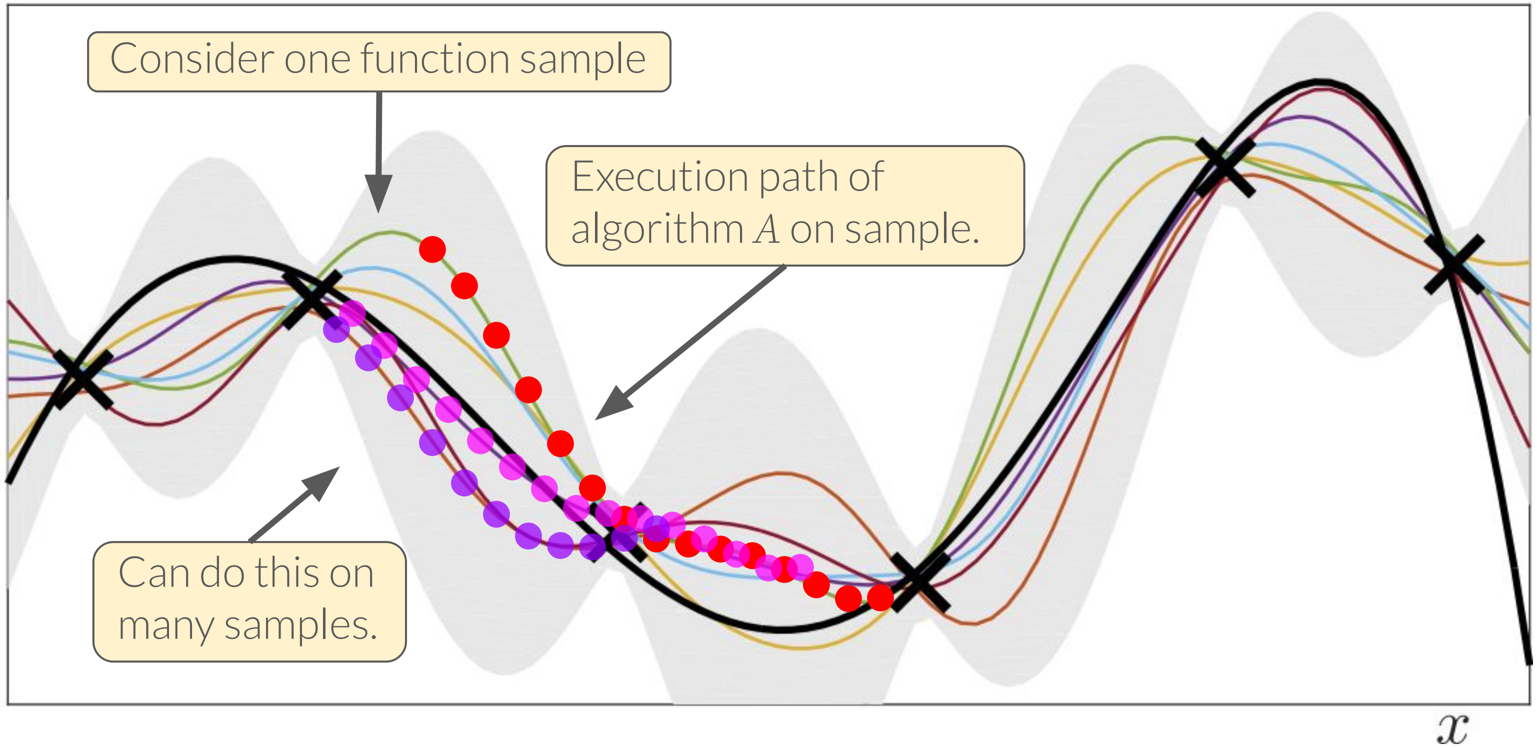
InfoBAX acquisition function



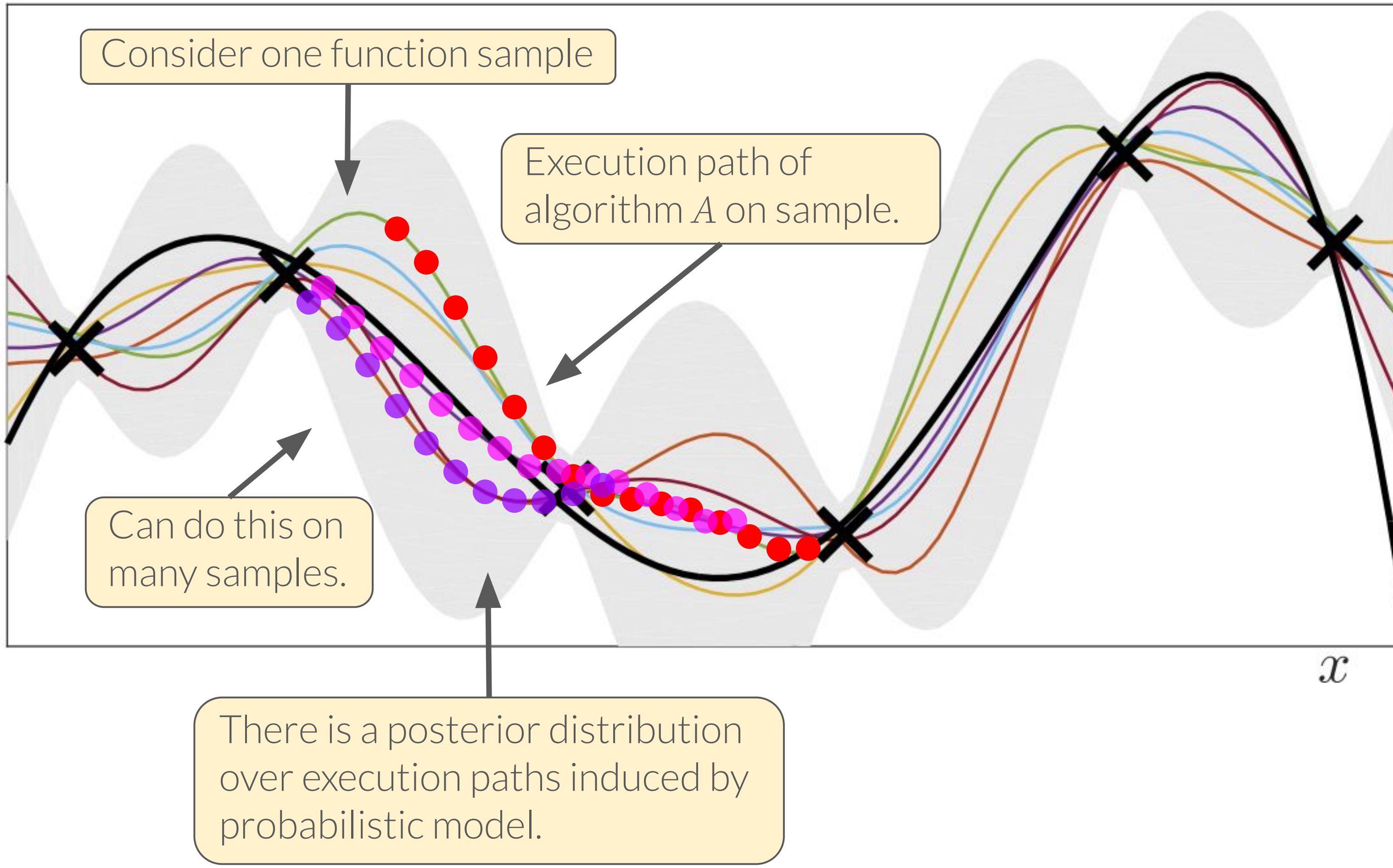
InfoBAX acquisition function



InfoBAX acquisition function



InfoBAX acquisition function



InfoBAX acquisition function

How to compute and optimize it? **Two stages:**

Similar to info-based BO!

InfoBAX acquisition function

How to compute and optimize it? **Two stages:**

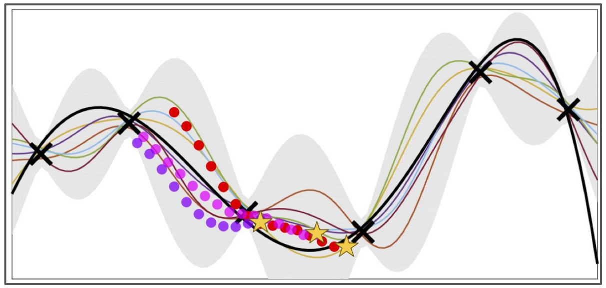
Similar to info-based BO!

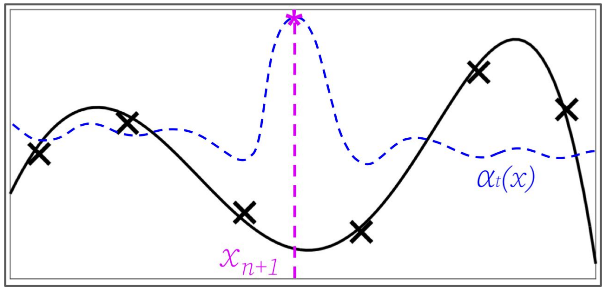
1) Before acquisition optimization:

- Run algorithm A on posterior function samples to get posterior samples of execution path.

2) Acquisition optimization:

- For any x, approximate EIG $\alpha_t(x)$ using these samples
- Allows us to optimize acquisition function





⇒ Similar structure as info-based BO, but replace global opt algorithm with A.

- *Same benefits*: generate samples only once. Then cheaper during iterative acquisition opt.

⇒ Look in paper for math on computing EIG $\alpha_t(x)$ with samples.

Information-based Bayesian algorithm execution

Algorithm 1 InfoBAX

Input: dataset $\mathcal{D}_1$, prior distribution $p(f)$, algorithm $\mathcal{A}$

1: **for** $t = 1, \ldots, T$ **do**

2: $\quad x_t \leftarrow \arg\max_{x \in \mathcal{X}} \alpha_t(x, \mathcal{A})$

3: $\quad y_{x_t} \sim f(x_t) + \epsilon$

4: $\quad \mathcal{D}_{t+1} \leftarrow \mathcal{D}_t \cup \{(x_t, y_{x_t})\}$

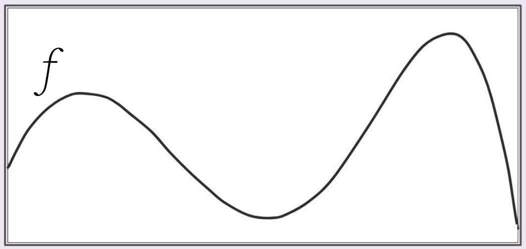
Output: final dataset $\mathcal{D}_{T+1}$

(1) Run algorithm on posterior function samples.
(2) Optimize $\alpha_t(x)$ using resulting execution paths.

InfoBAX — one-slide summary of the full story

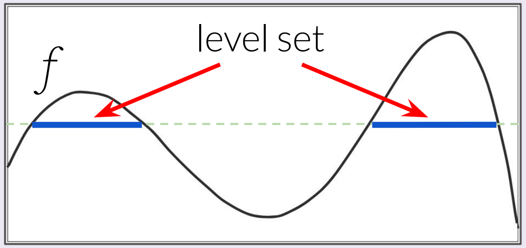
InfoBAX — one-slide summary of the full story

Suppose we have a black-box function f and a ***property of interest***



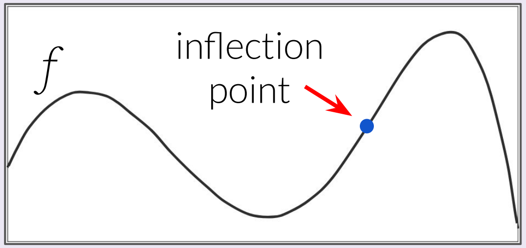
InfoBAX — one-slide summary of the full story

Suppose we have a black-box function f and a ***property of interest***



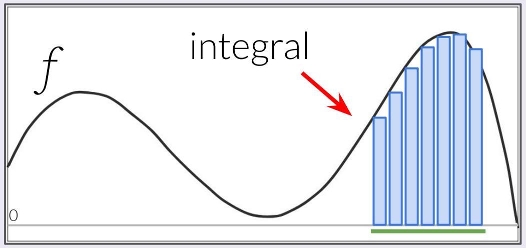
InfoBAX — one-slide summary of the full story

Suppose we have a black-box function f and a ***property of interest***



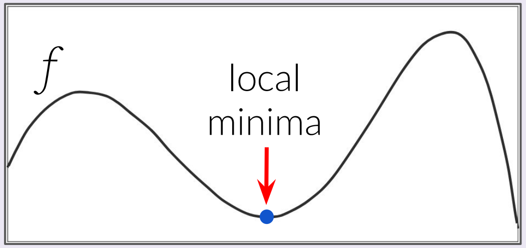
InfoBAX — one-slide summary of the full story

Suppose we have a black-box function f and a ***property of interest***



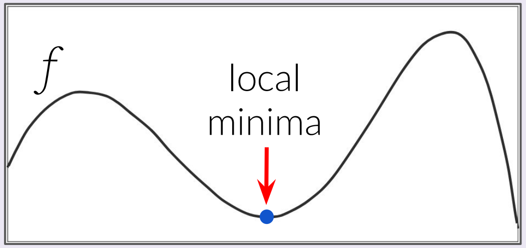
InfoBAX — one-slide summary of the full story

Suppose we have a black-box function f and a ***property of interest***

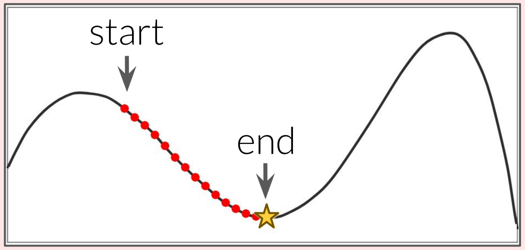


InfoBAX — one-slide summary of the full story

Suppose we have a black-box function f and a ***property of interest***

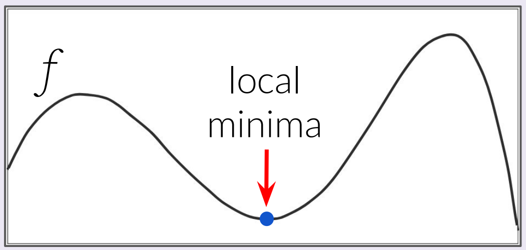


Suppose property is ***computable*** $\Rightarrow$ *there exists an algorithm A (of any budget)*

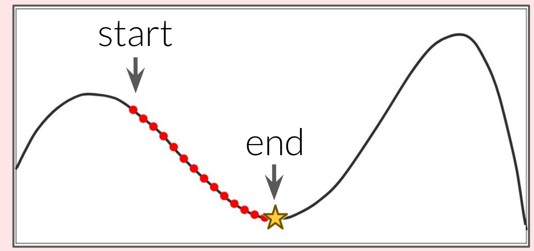


InfoBAX — one-slide summary of the full story

Suppose we have a black-box function f and a ***property of interest***



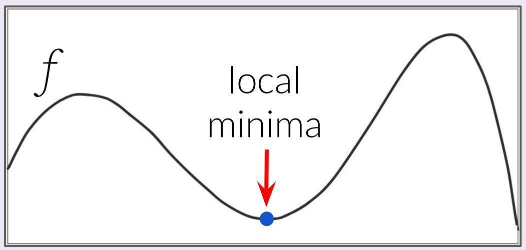
Suppose property is ***computable*** ⇒ *there exists an algorithm A (of any budget)*



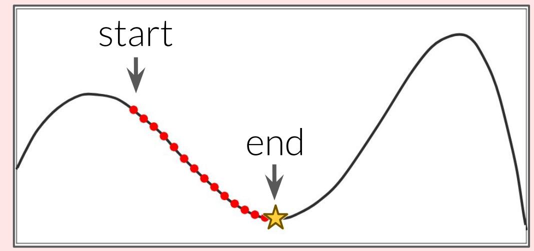
⇒ ***Goal:*** *estimate the property (i.e. output of A) with minimal function queries*

InfoBAX — one-slide summary of the full story

Suppose we have a black-box function f and a ***property of interest***



Suppose property is ***computable*** ⇒ *there exists an algorithm A (of any budget)*



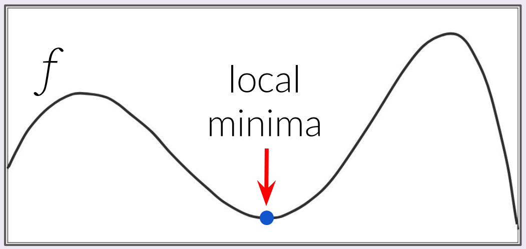
⇒ ***Goal:*** *estimate the property (i.e. output of A) with minimal function queries*

Run ***InfoBAX****, a sequential algorithm*
(similar in structure to BO)

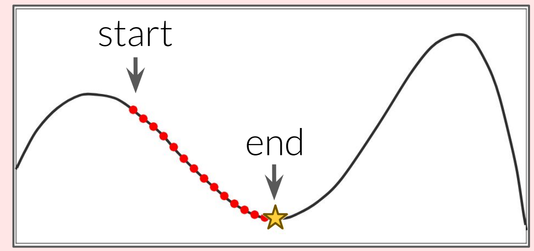
for $t = 1, \ldots, T$ **do**
  $x_t \leftarrow \arg\max_{x \in \mathcal{X}} \alpha_t(x, \mathcal{A})$
  $y_{x_t} \sim f(x_t) + \epsilon$
  $\mathcal{D}_{t+1} \leftarrow \mathcal{D}_t \cup \{(x_t, y_{x_t})\}$

InfoBAX — one-slide summary of the full story

*Suppose we have a black-box function f and a **property of interest***



*Suppose property is **computable** ⇒ there exists an algorithm $\mathcal{A}$ (of any budget)*

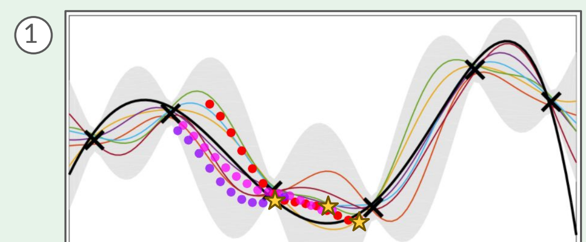


⇒ ***Goal:*** *estimate the property (i.e. output of $\mathcal{A}$) with minimal function queries*

*Run **InfoBAX**, a sequential algorithm*
(similar in structure to BO)

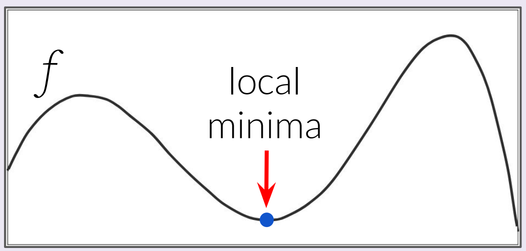
for $t = 1, \ldots, T$ **do**
$x_t \leftarrow \arg\max_{x \in \mathcal{X}} \alpha_t(x, \mathcal{A})$
$y_{x_t} \sim f(x_t) + \epsilon$
$\mathcal{D}_{t+1} \leftarrow \mathcal{D}_t \cup \{(x_t, y_{x_t})\}$

*(at each iteration) To optimize InfoBAX acquisition function: **two stages***

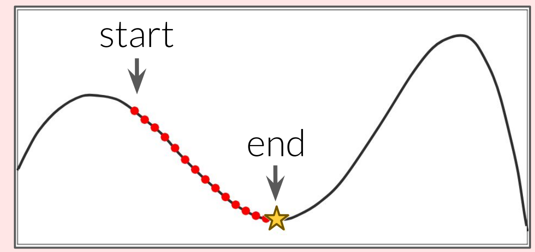


InfoBAX — one-slide summary of the full story

Suppose we have a black-box function f and a ***property of interest***



Suppose property is ***computable*** $\Rightarrow$ *there exists an algorithm A (of any budget)*

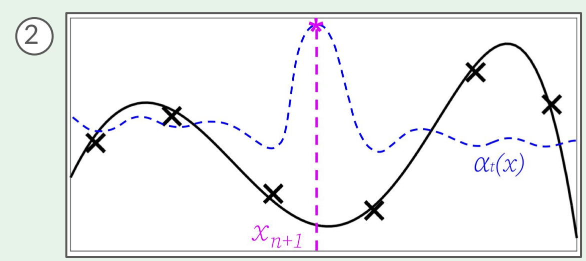


$\Rightarrow$ ***Goal:*** *estimate the property (i.e. output of A) with minimal function queries*

Run ***InfoBAX****, a sequential algorithm*
(similar in structure to BO)

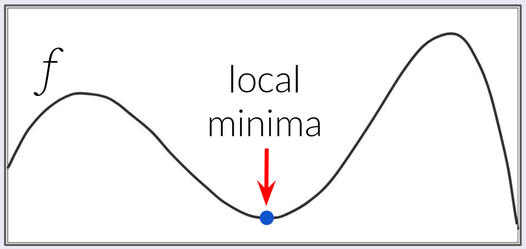
$$
\begin{array}{l}
\textbf{for } t = 1, \ldots, T \textbf{ do} \\
\quad x_t \leftarrow \arg\max_{x \in \mathcal{X}} \alpha_t(x, \mathcal{A}) \\
\quad y_{x_t} \sim f(x_t) + \epsilon \\
\quad \mathcal{D}_{t+1} \leftarrow \mathcal{D}_t \cup \{(x_t, y_{x_t})\}
\end{array}
$$

(at each iteration) To optimize InfoBAX acquisition function: ***two stages***

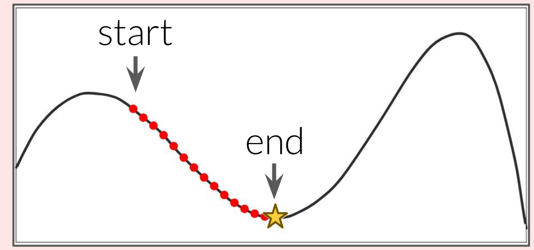


InfoBAX — one-slide summary of the full story

Suppose we have a black-box function f and a ***property of interest***



Suppose property is ***computable*** ⇒ *there exists an algorithm A (of any budget)*

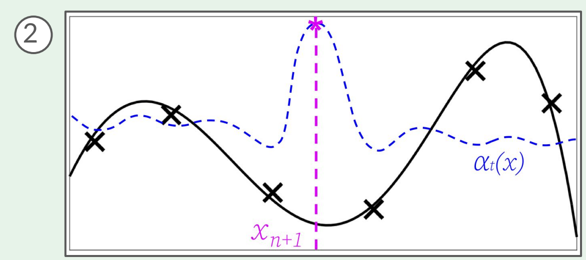


⇒ ***Goal:*** *estimate the property (i.e. output of A) with minimal function queries*

Run ***InfoBAX****, a sequential algorithm (similar in structure to BO)*

for $t = 1, \ldots, T$ **do**
$x_t \leftarrow \arg\max_{x \in \mathcal{X}} \alpha_t(x, \mathcal{A})$
$y_{x_t} \sim f(x_t) + \epsilon$
$\mathcal{D}_{t+1} \leftarrow \mathcal{D}_t \cup \{(x_t, y_{x_t})\}$

(at each iteration) To optimize InfoBAX acquisition function: ***two stages***



⇒ ***Output:*** *posterior estimate of property (i.e. output of A)*

BAX: Demos and Applications

Applications

We demo BAX to estimate a few different properties of black-box functions
(trying to show the breadth of what we can estimate)

Three applications:

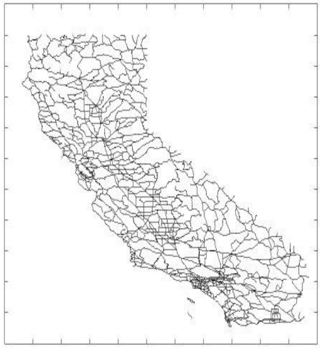
- Estimating shortest paths in graphs ← Black-box function over edges in a network
- Bayesian local optimization ← Optimization variants
- Estimating top-*k* optima ← Optimization variants

Application: *estimating shortest paths in graphs*

Graph traversal/search algorithms can define properties of a black-box function f defined on edge weights in a graph.

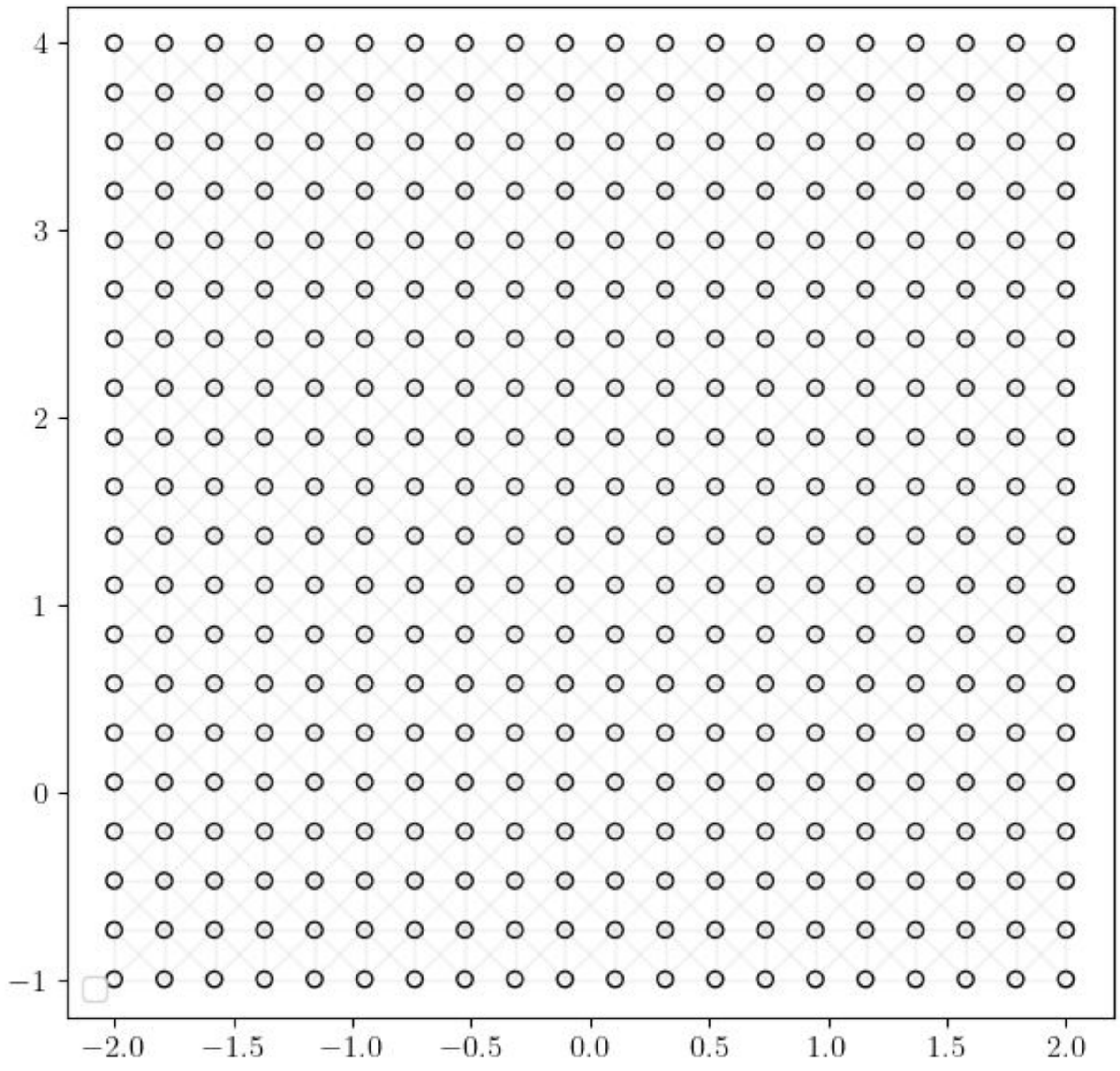
Example: *real-world transportation network*
(e.g. road, railway, shipping, air)

- Suppose we want to find shortest path from location A to location B.
- Shortest path depends on edge weights.
  - e.g. traffic, road conditions, weather, etc.
- It can be ***expensive to query edge weights***
  - e.g. measure traffic/road/weather conditions via satellite.
  - e.g. determine/access shipping costs.
- **Goal**: adaptively query edge weights to estimate shortest path.



California road network.

APPLICATIONS of BAX — *estimating shortest paths in graphs*

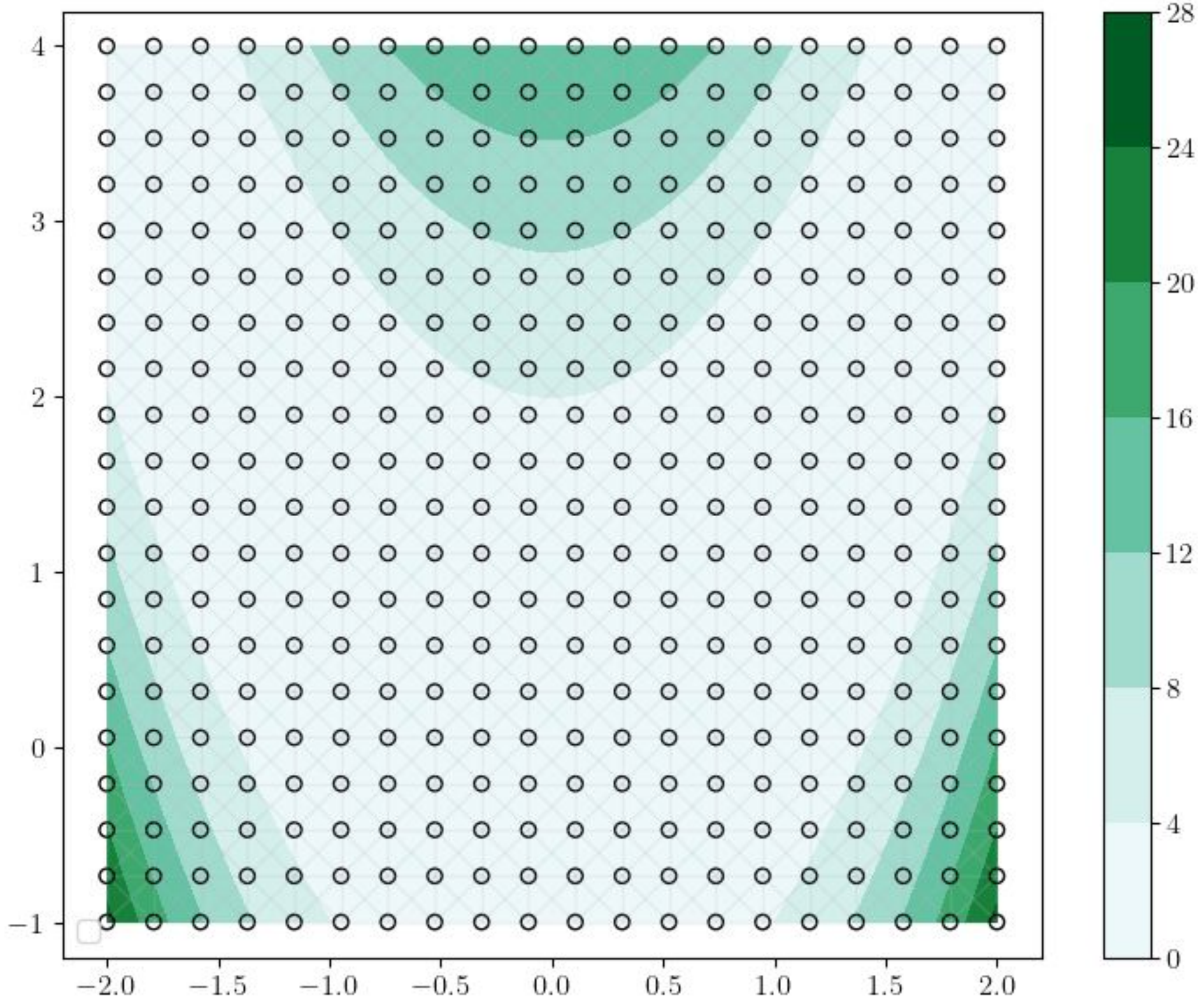


Grid-shaped graph:

- 400 nodes
- 2964 edges

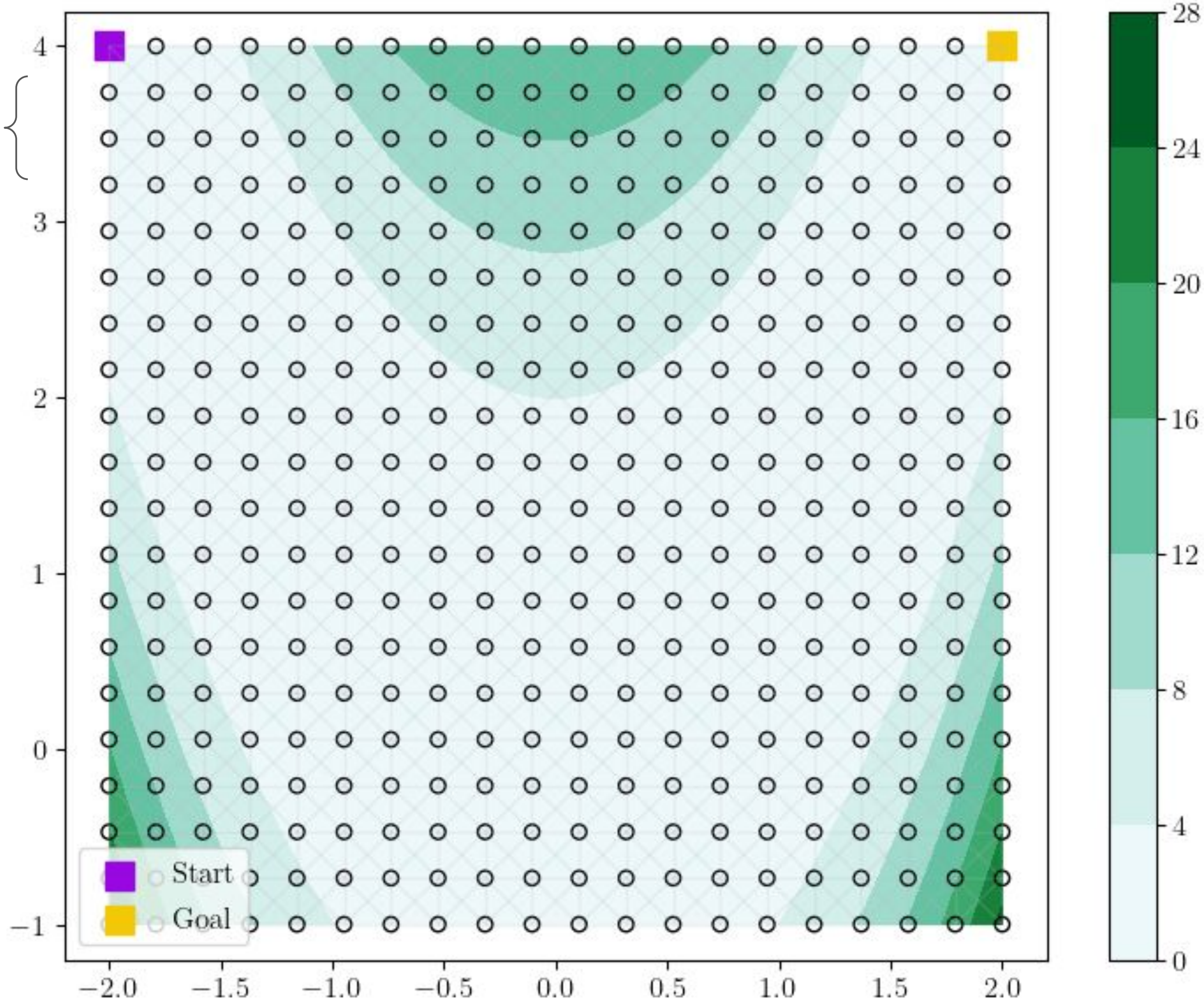
APPLICATIONS of BAX — *estimating shortest paths in graphs*

Edge weights are given by function f (green).



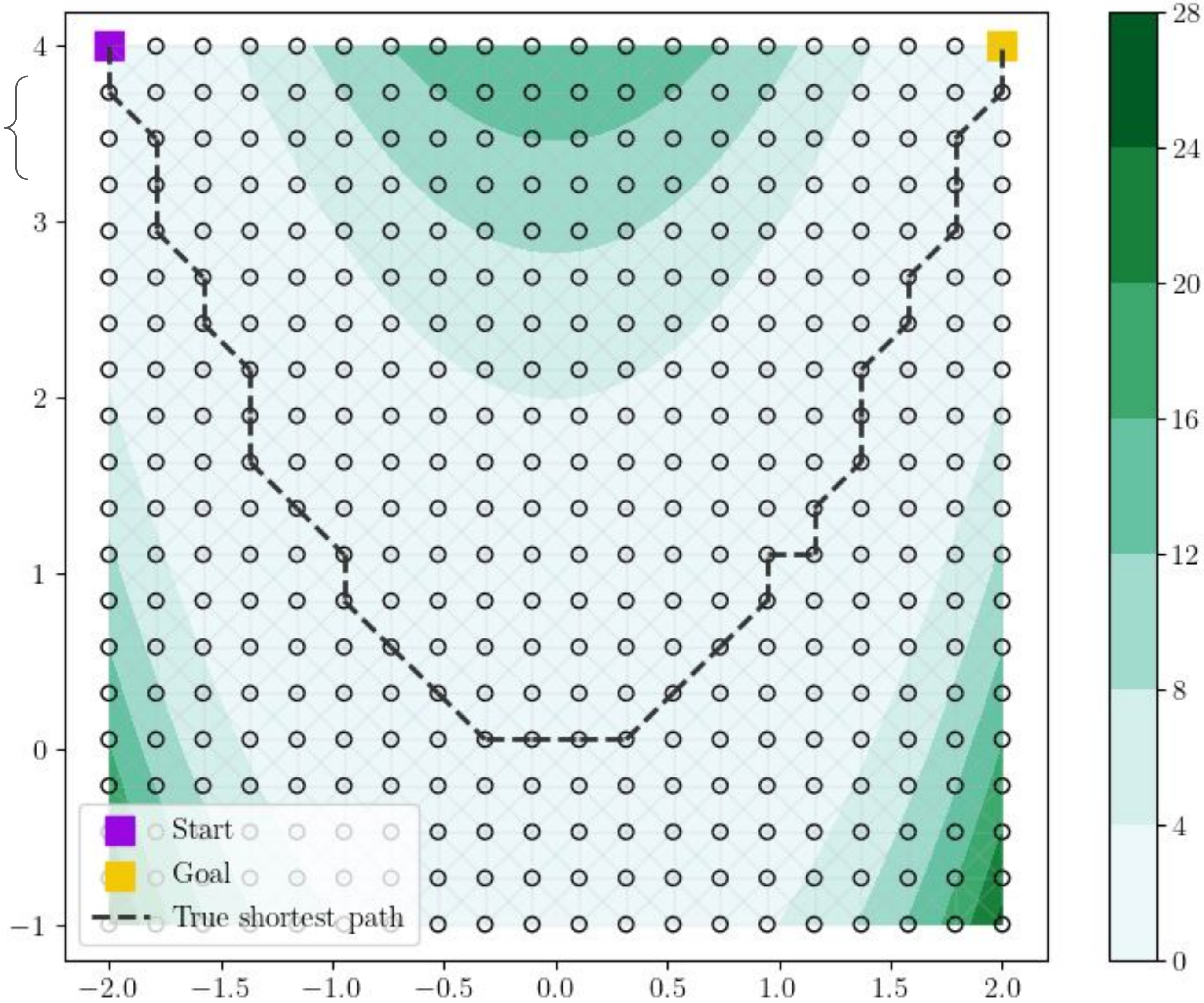
APPLICATIONS of BAX — *estimating shortest paths in graphs*

Want to know shortest path between start and goal.



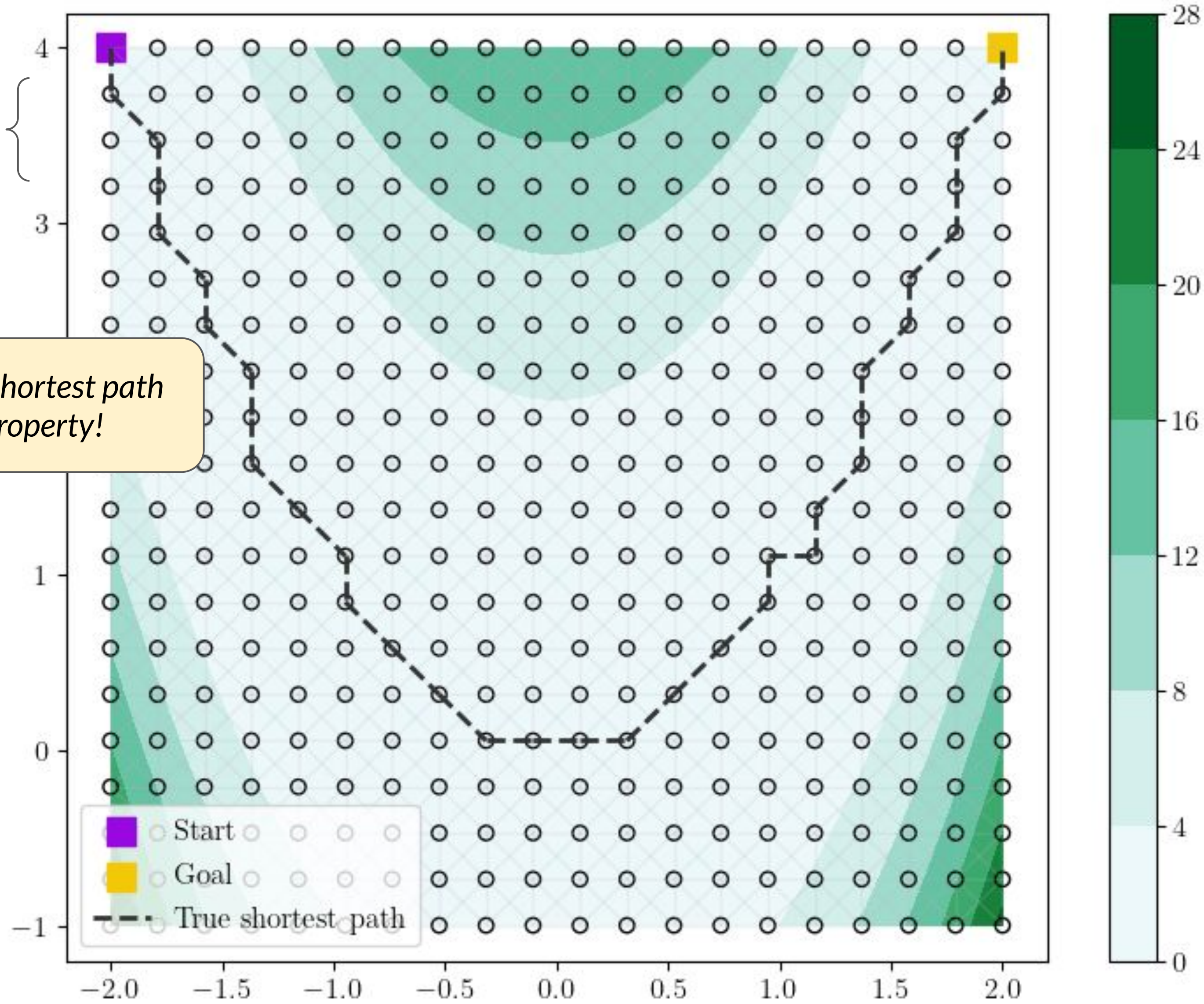
APPLICATIONS of BAX — *estimating shortest paths in graphs*

Want to know shortest path between start and goal.



APPLICATIONS of BAX — *estimating shortest paths in graphs*

Want to know shortest path between start and goal.

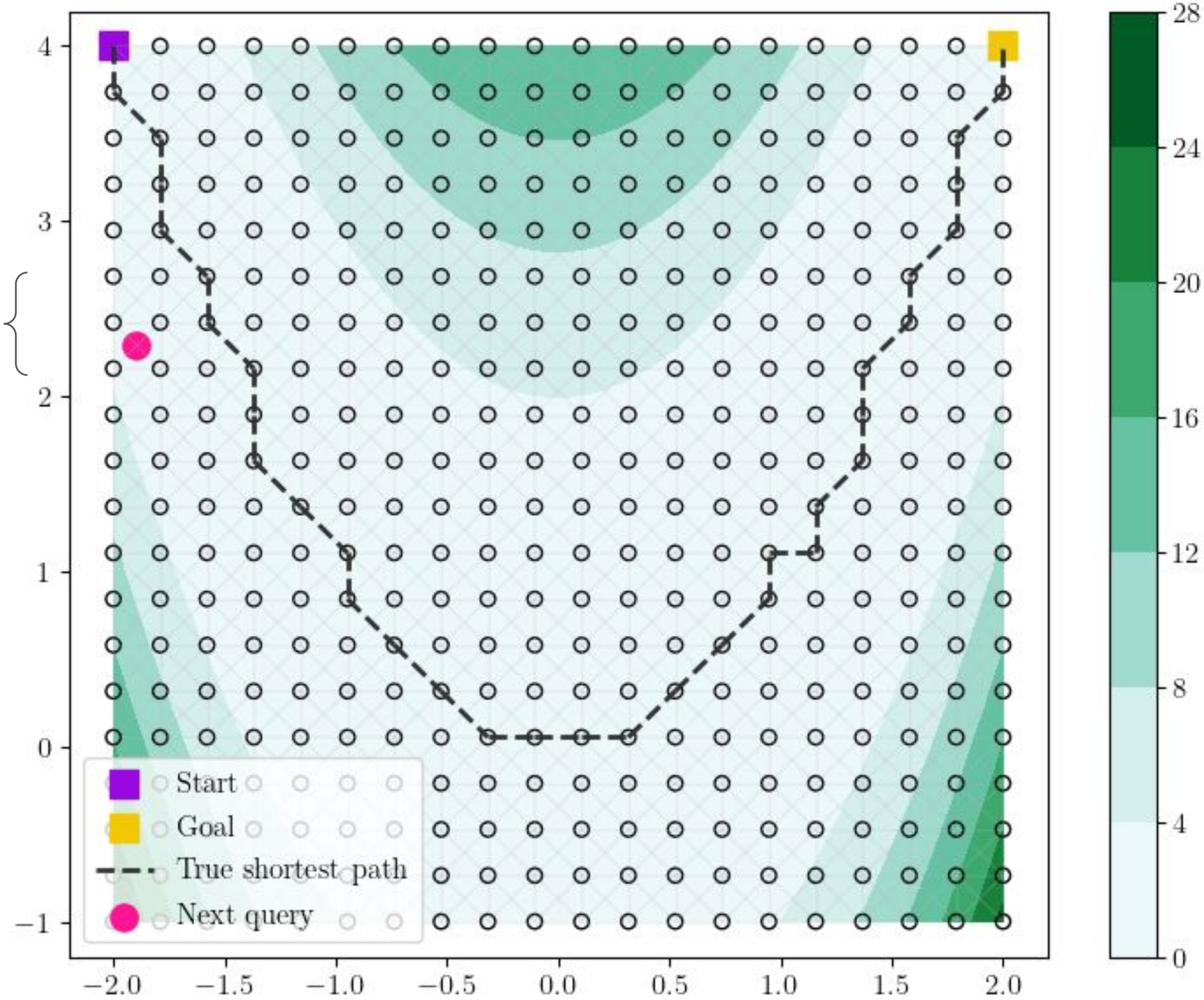


Can view this shortest path as a function property!

APPLICATIONS of BAX — *estimating shortest paths in graphs*

Suppose that accessing (*querying*) edge weights is expensive.

→ e.g. in transportation network examples



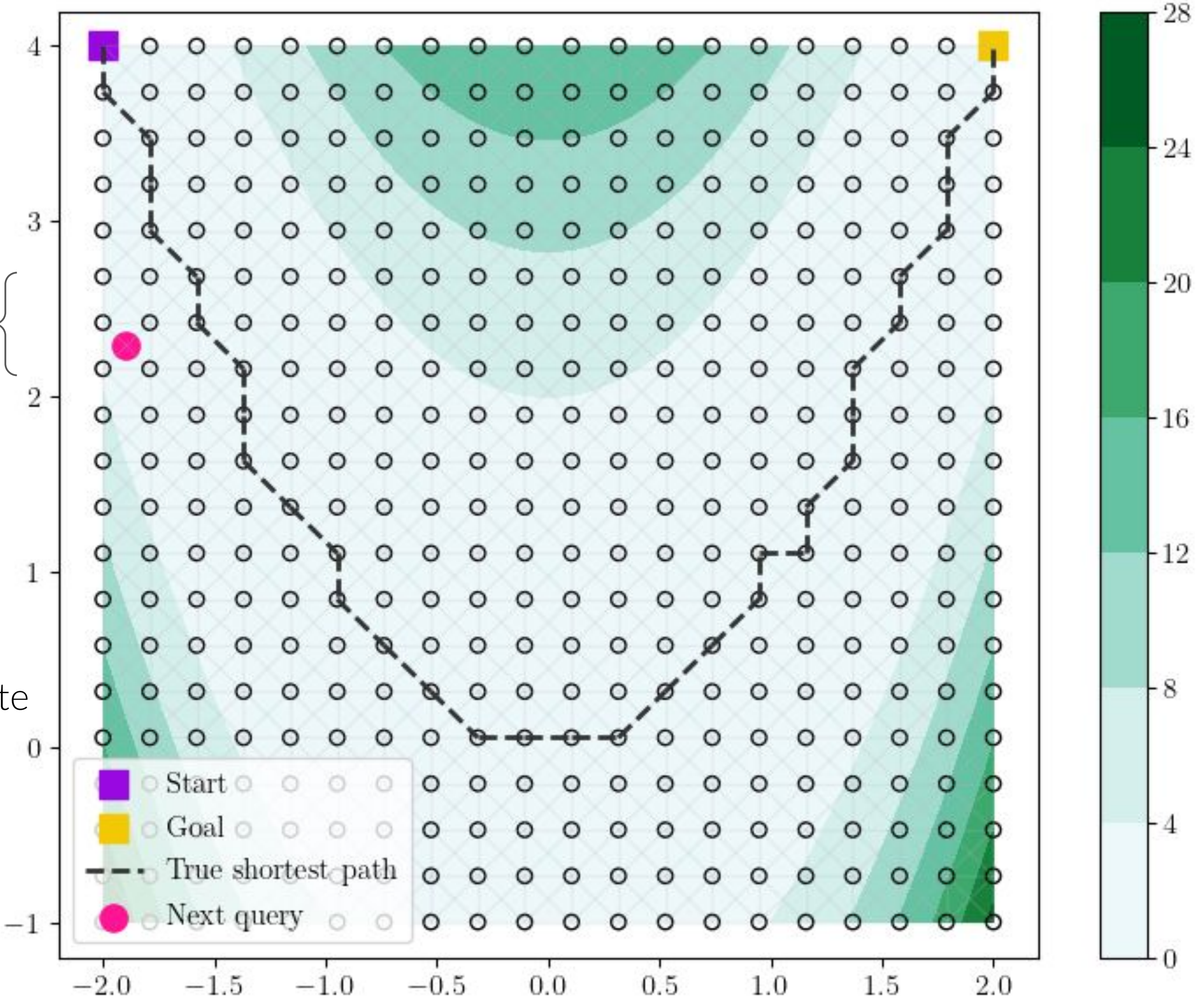
APPLICATIONS of BAX — *estimating shortest paths in graphs*

Suppose that accessing (*querying*) edge weights is expensive.

→ e.g. in transportation network examples

How to estimate shortest path?

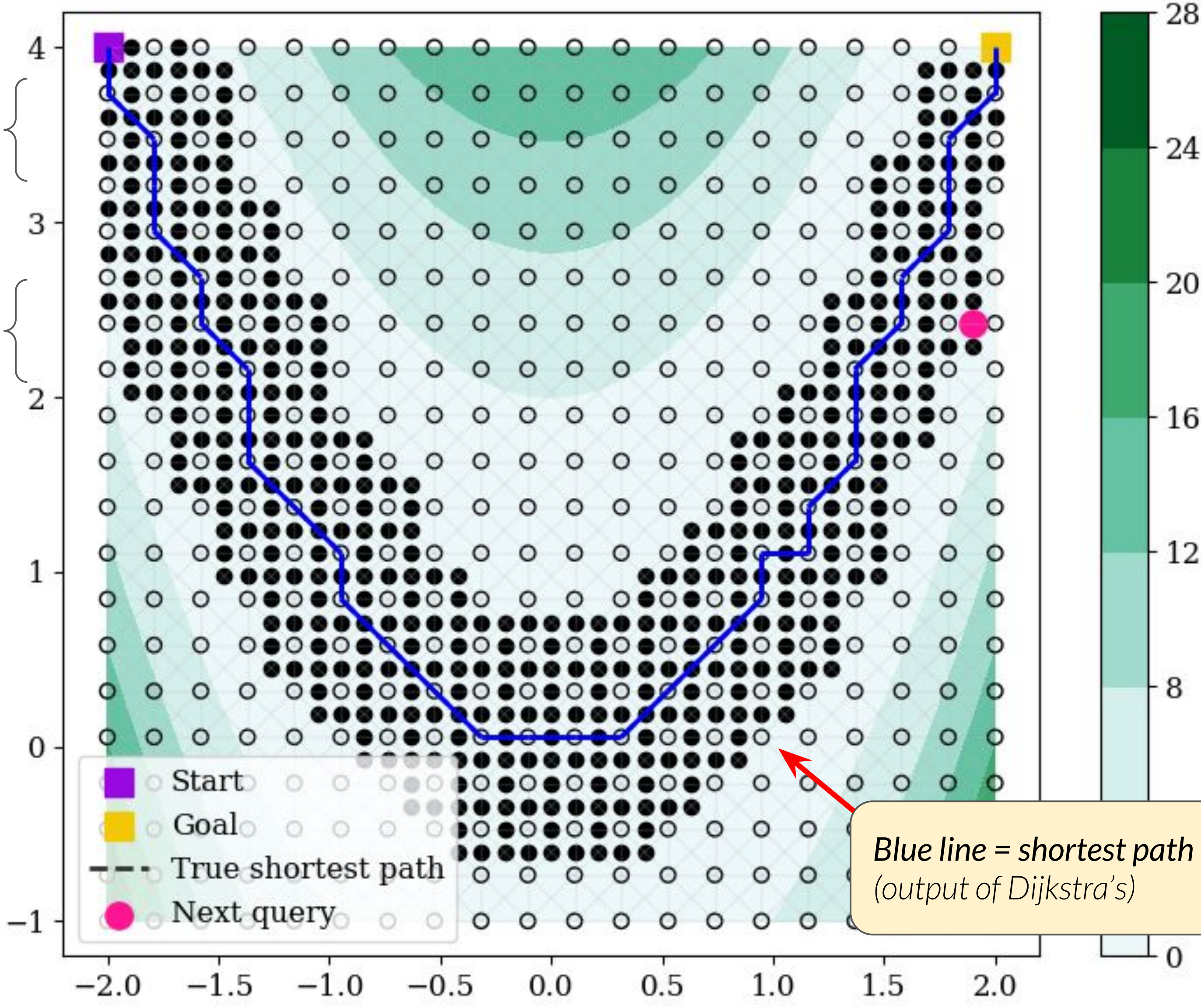
One strategy: use *Dijkstra's algorithm*



APPLICATIONS of BAX — *estimating shortest paths in graphs*

One strategy: use *Dijkstra's algorithm*

Exactly computes the shortest path



Blue line = shortest path
(output of Dijkstra's)

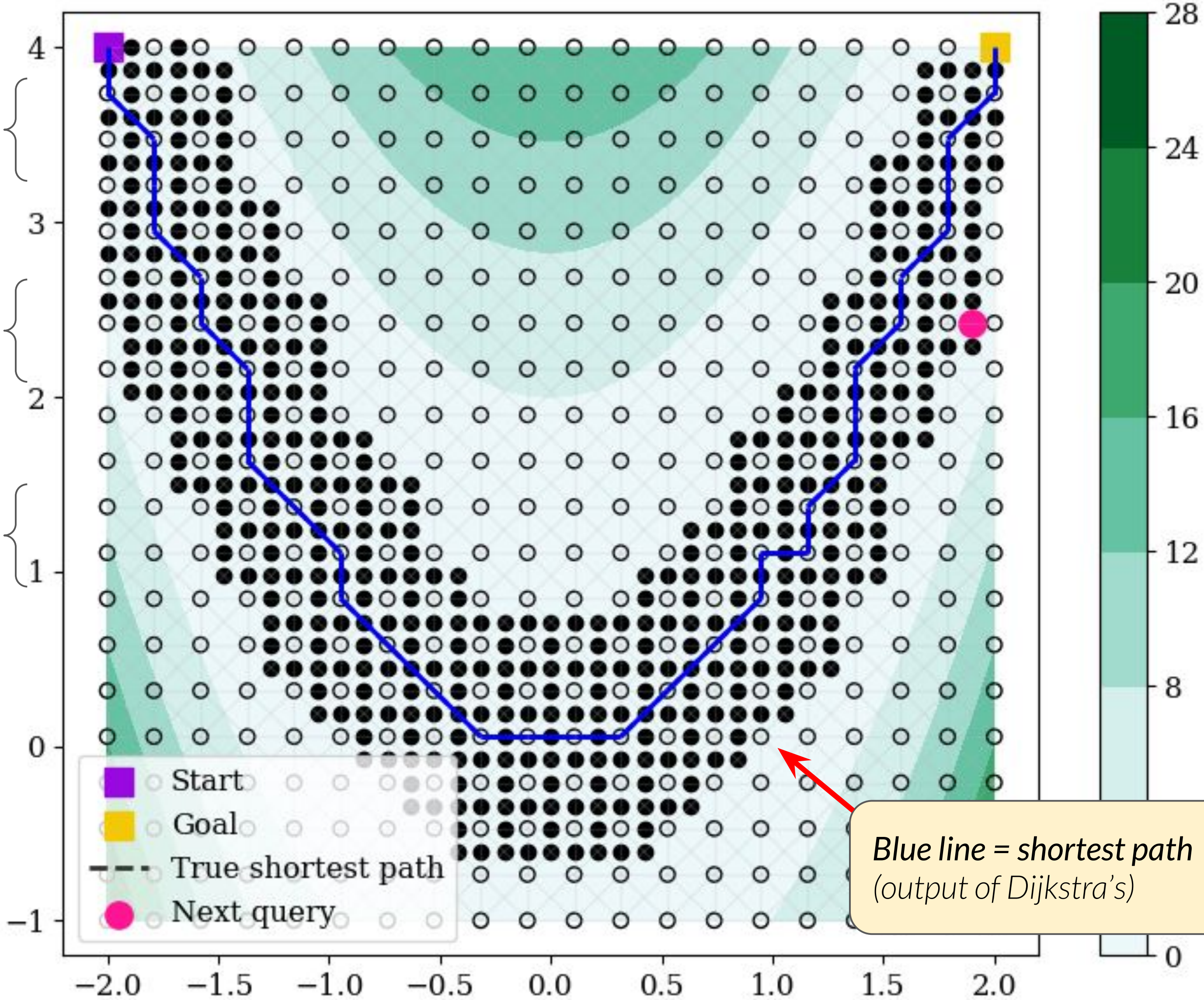
APPLICATIONS of BAX — *estimating shortest paths in graphs*

One strategy: use *Dijkstra's algorithm*

Exactly computes the shortest path

However… requires over 430 queries!

Can try a different strategy…

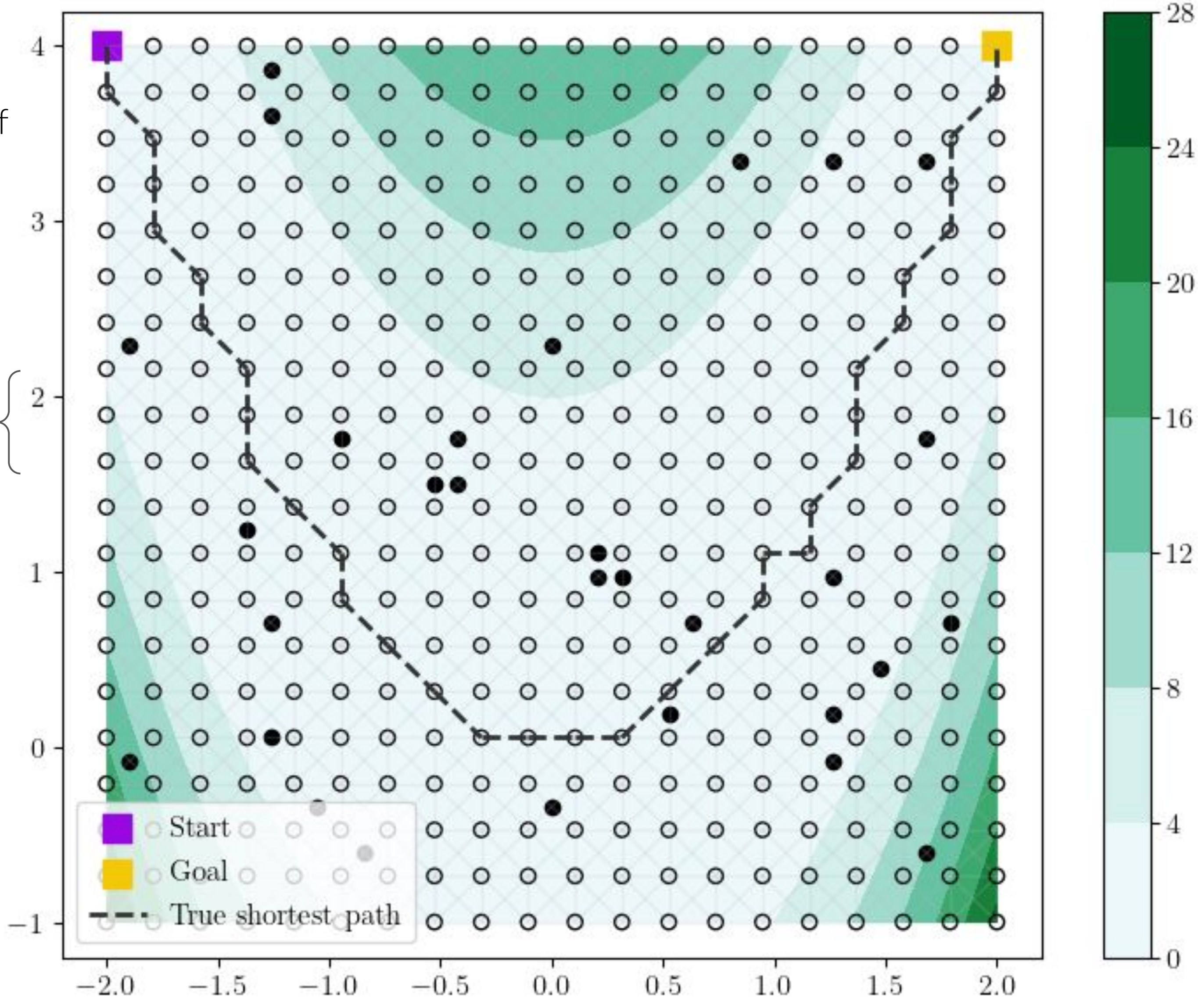


Blue line = shortest path
(output of Dijkstra's)

APPLICATIONS of BAX — *estimating shortest paths in graphs*

Suppose we take small set of edge weight queries
(e.g. 30 unif-random)

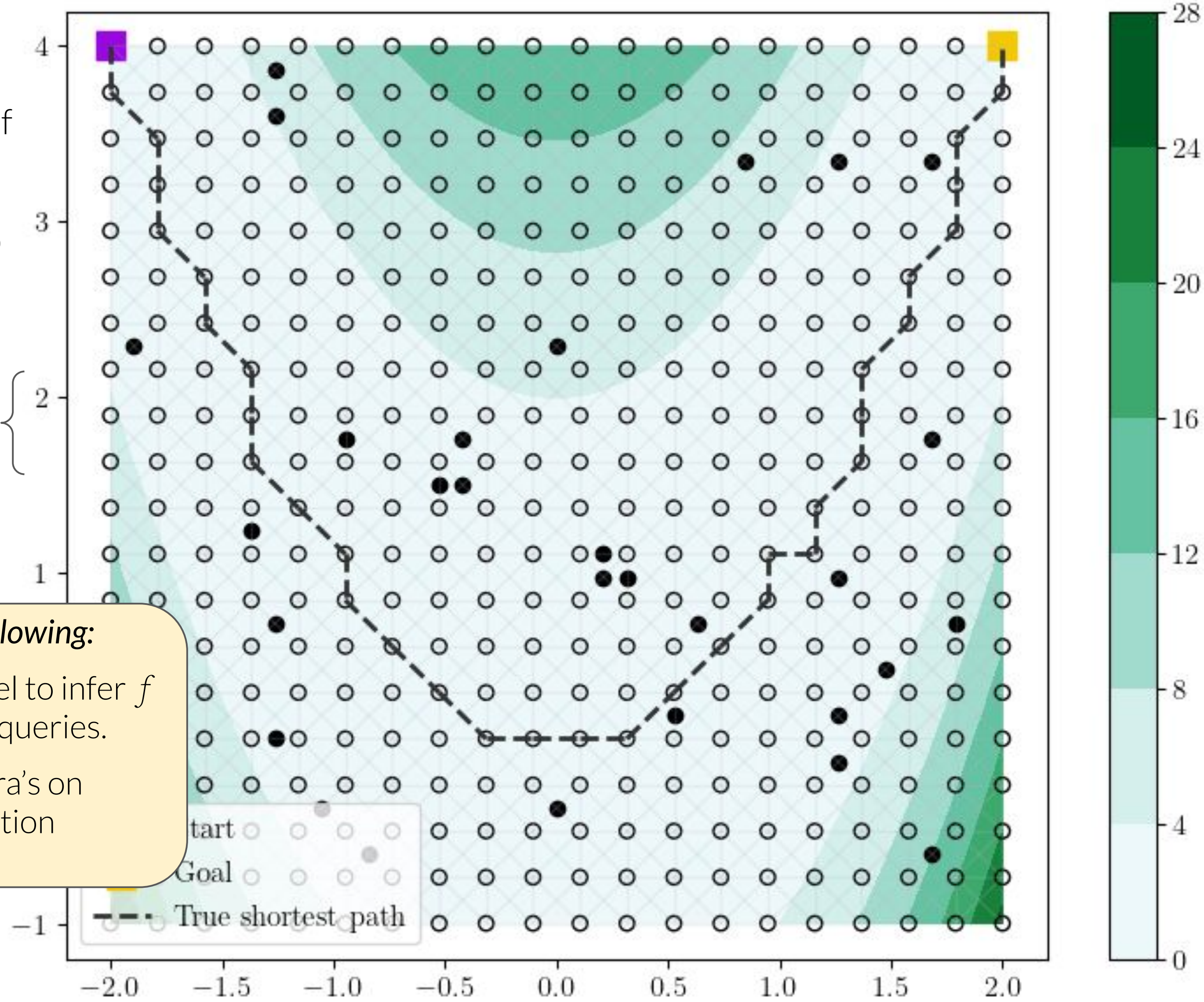
Could try to infer shortest path from these



APPLICATIONS of BAX — *estimating shortest paths in graphs*

Suppose we take small set of edge weight queries
(e.g. 30 unif-random)

Could try to infer shortest path from these



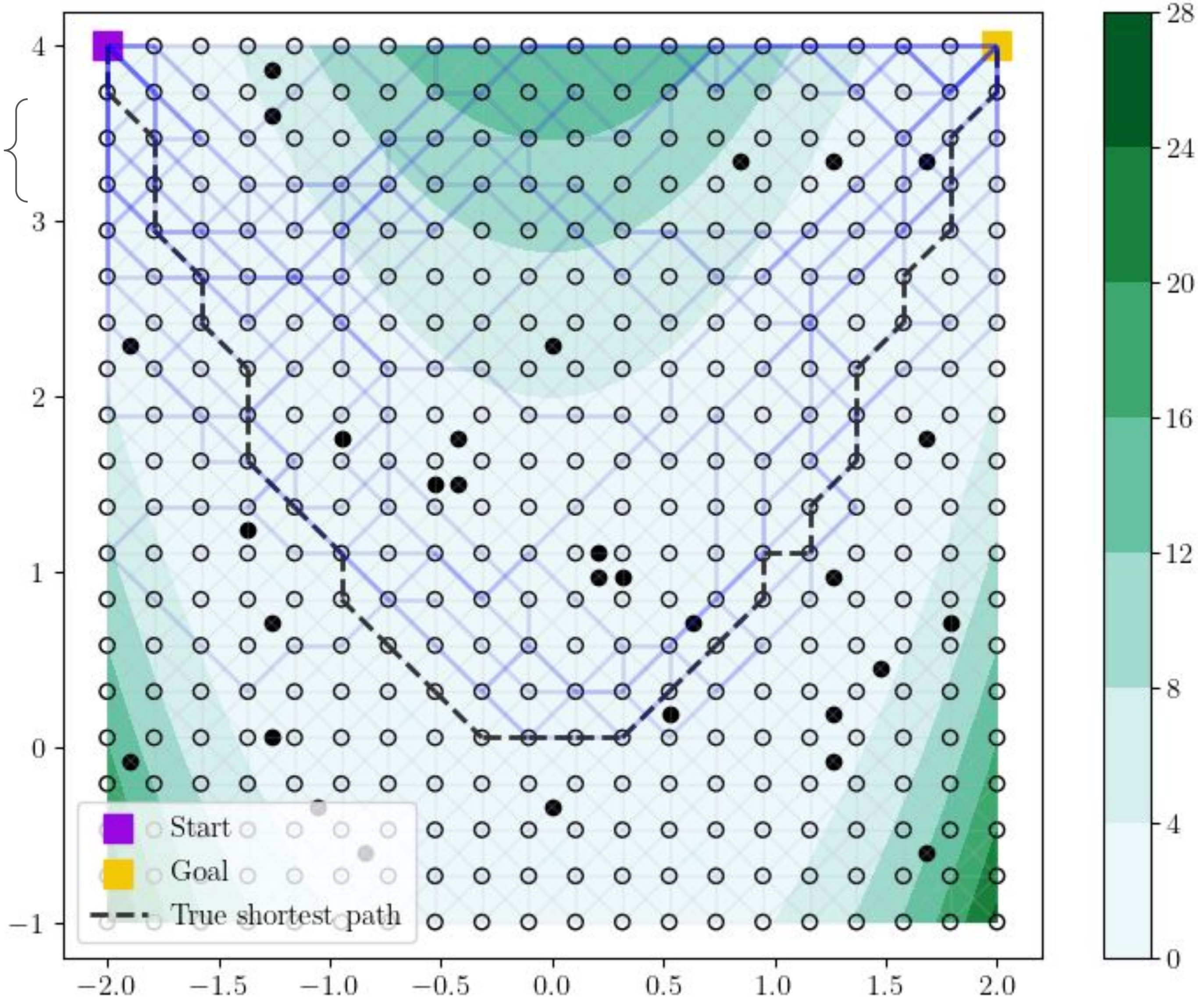
We'll do the following:

(1) Use a model to infer f (e.g. GP), given queries.

(2) Run Dijkstra's on posterior function samples.

APPLICATIONS of BAX — *estimating shortest paths in graphs*

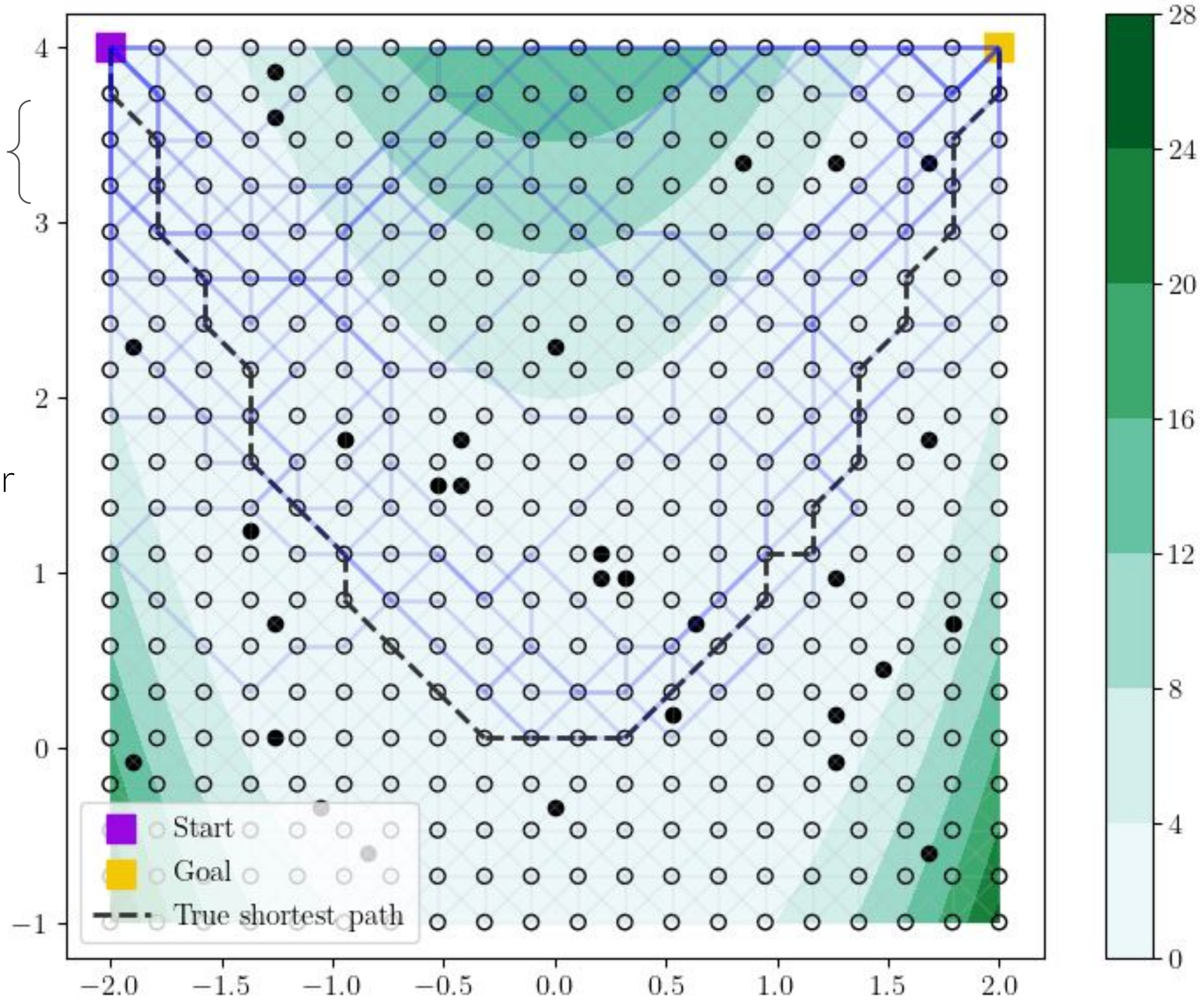
Posterior samples of shortest path (output of Dijkstra's) via GP model



APPLICATIONS of BAX — *estimating shortest paths in graphs*

Posterior samples of shortest path (output of Dijkstra's) via GP model

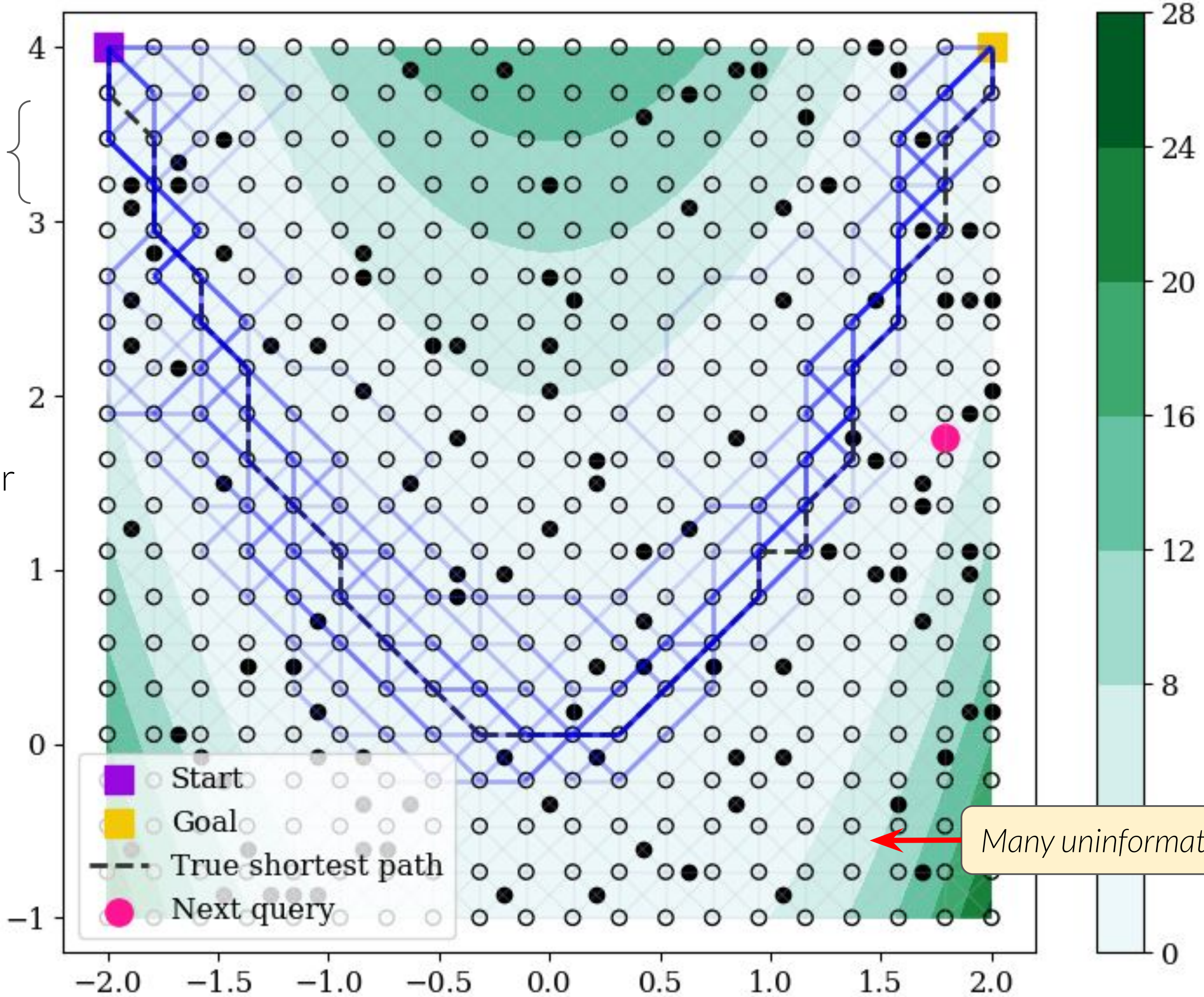
Estimate after 100 queries?



APPLICATIONS of BAX — *estimating shortest paths in graphs*

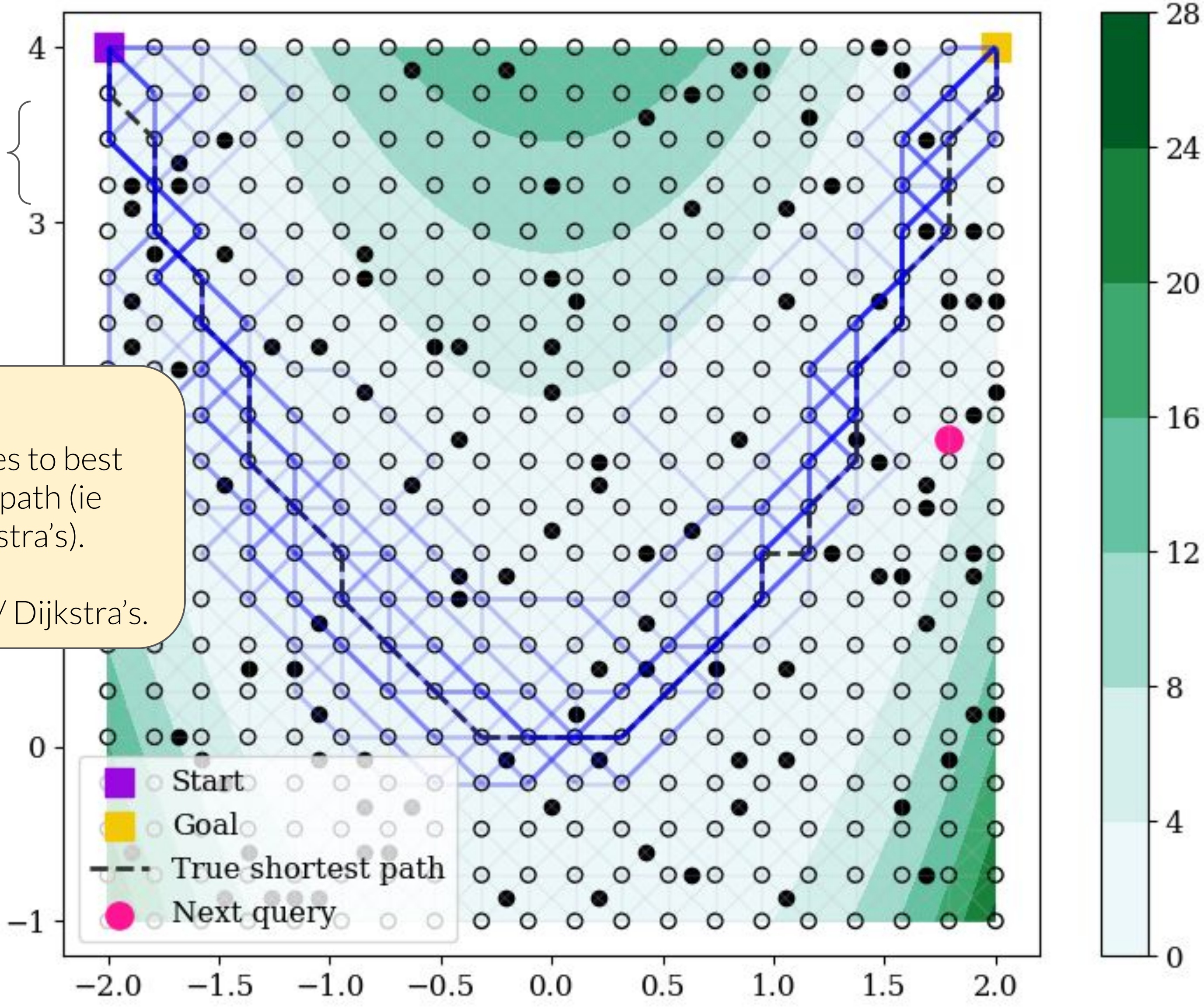
Posterior samples of shortest path (output of Dijkstra's) via GP model

Estimate after 100 queries?



APPLICATIONS of BAX — *estimating shortest paths in graphs*

Posterior samples of shortest path (output of Dijkstra's) via GP model



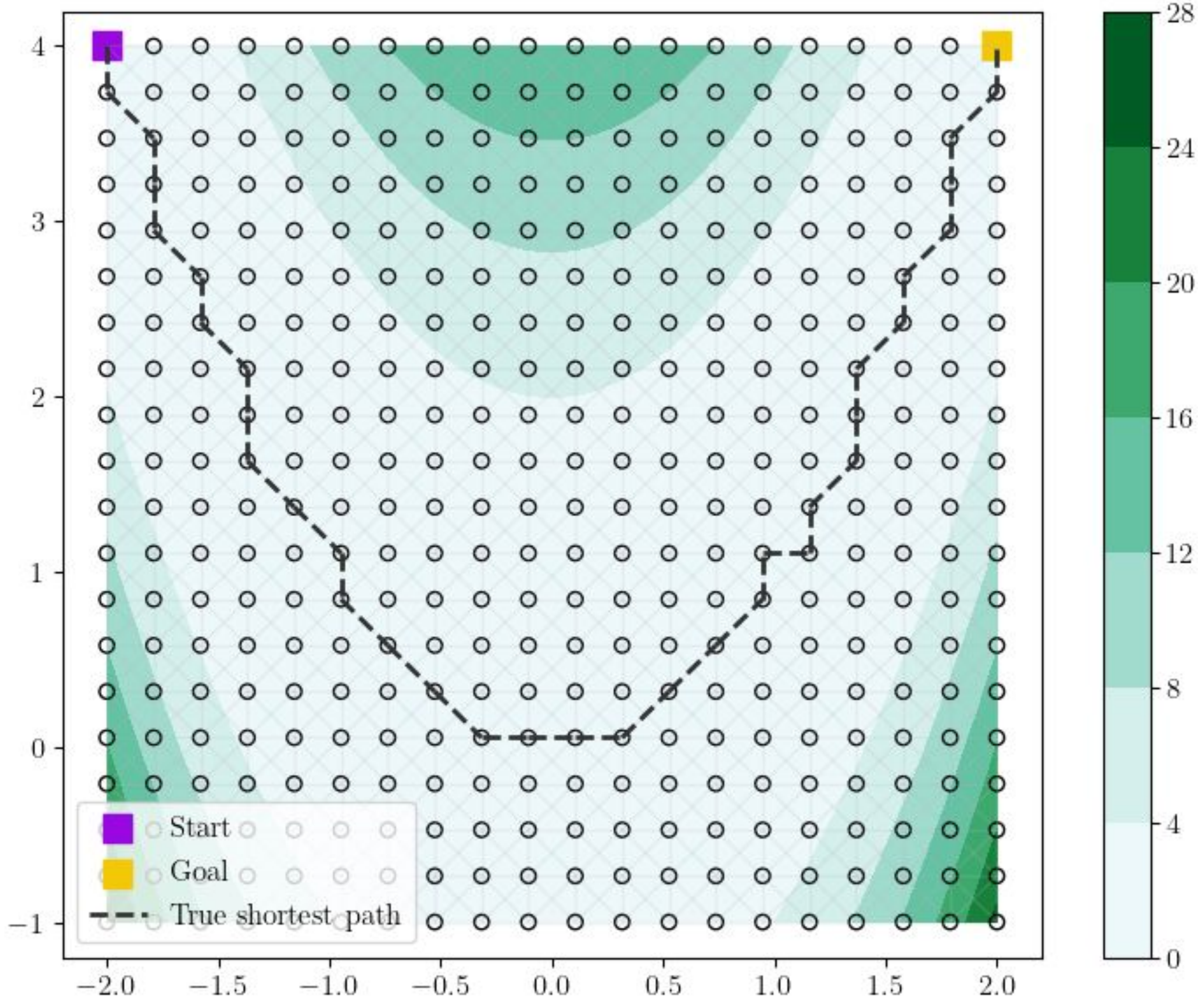
Goal:

Choose queries to best infer shortest path (ie output of Dijkstra's).

⇒ *InfoBAX* w/ Dijkstra's.

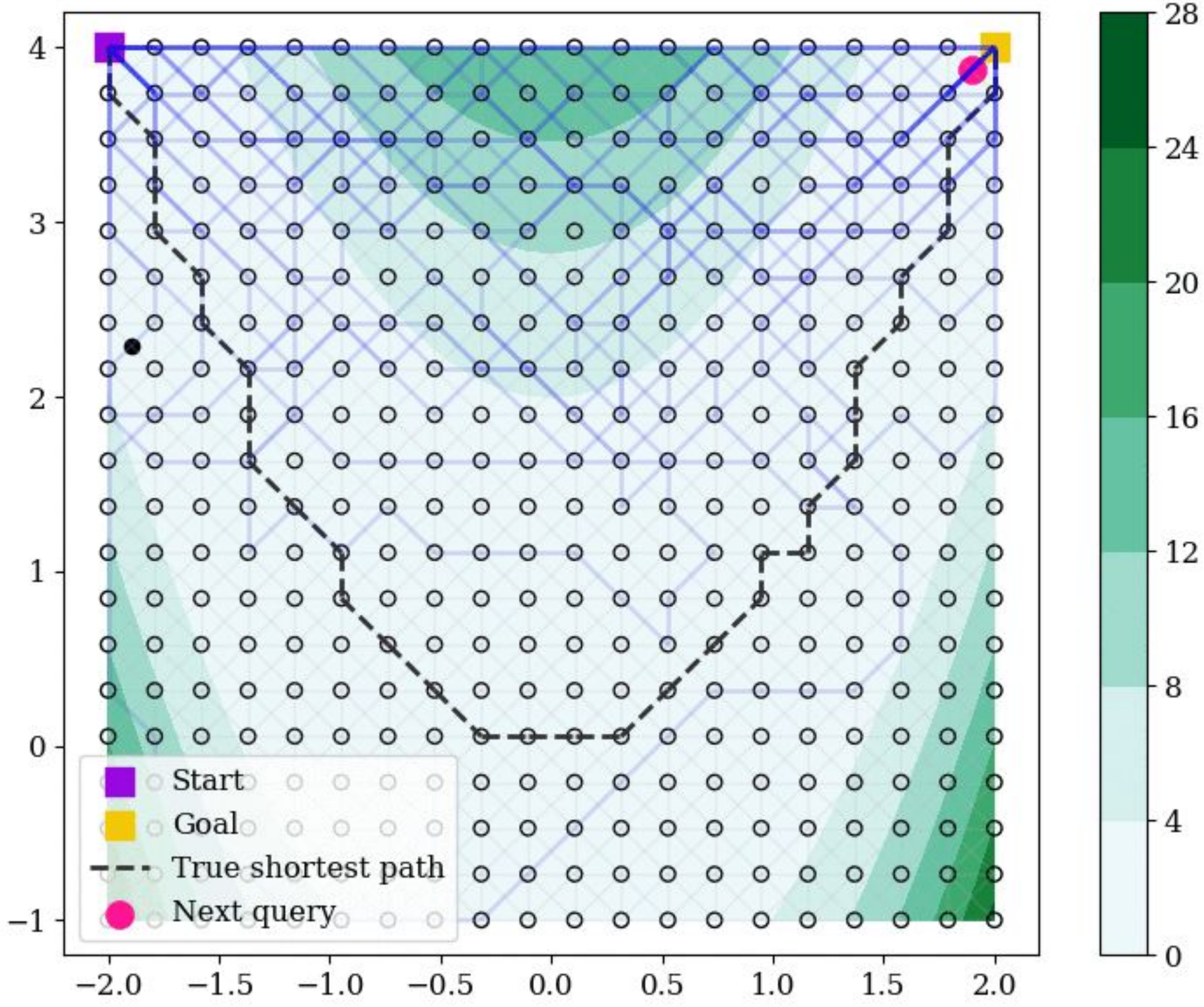
APPLICATIONS of BAX — *estimating shortest paths in graphs*

InfoBAX in action ⟶



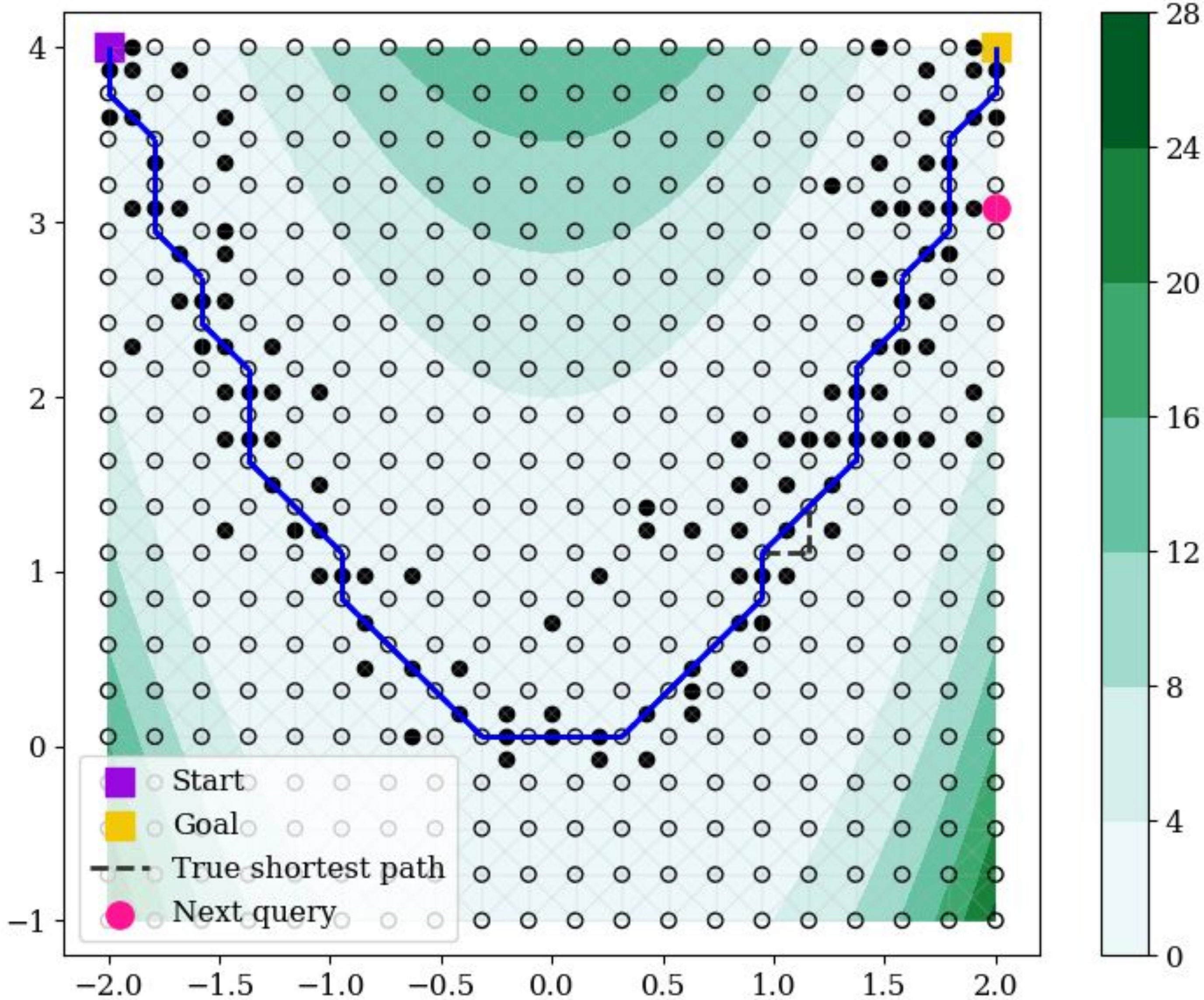
APPLICATIONS of BAX — *estimating shortest paths in graphs*

InfoBAX in action ⟶



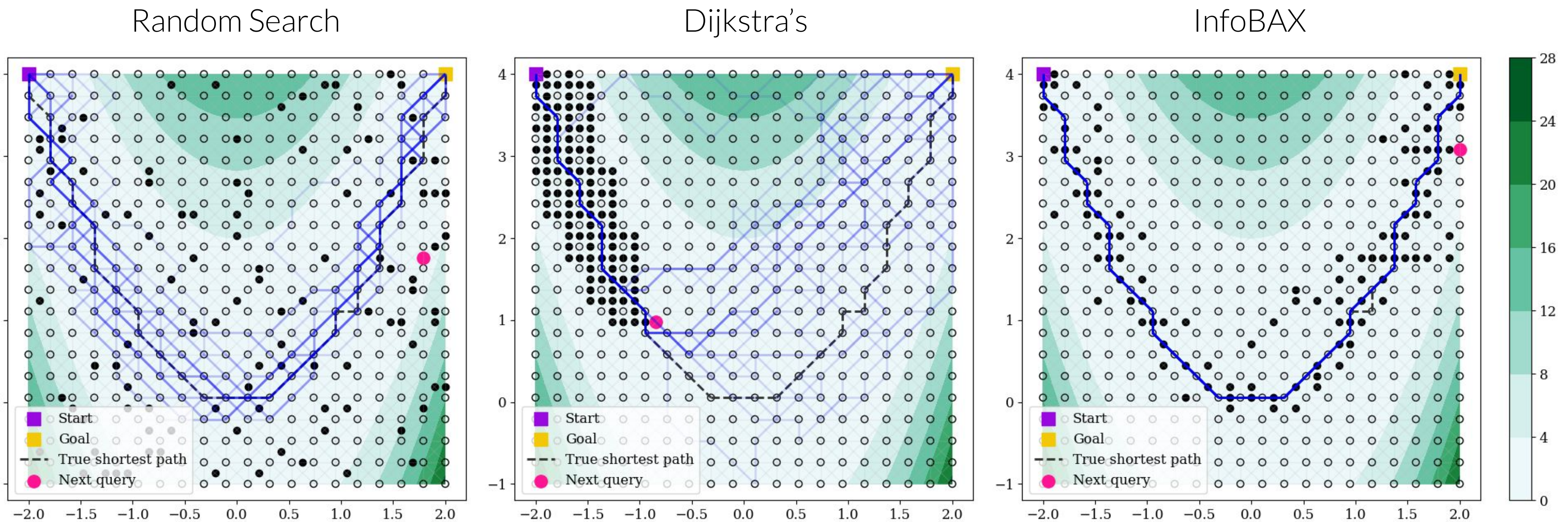
InfoBAX in action ⟶

After 100 queries.



APPLICATIONS of BAX — *estimating shortest paths in graphs*

Comparison after **100 queries**:



Application: *Bayesian local optimization*

Application: *Bayesian local optimization*

BO (typically) aims to estimate global optima.

However, many *local optimization* algorithms only aim to find a local optima (nearby some initial point)

- e.g. gradient descent, evolutionary algorithms, nelder-mead/simplex, etc.

Local opt can be very effective for certain settings (e.g. high dimensions), but can require large numbers of queries.

- Sometimes many redundant queries.
- Not effective if each query is very expensive.

We can use the local opt algorithms in *InfoBAX* procedure.

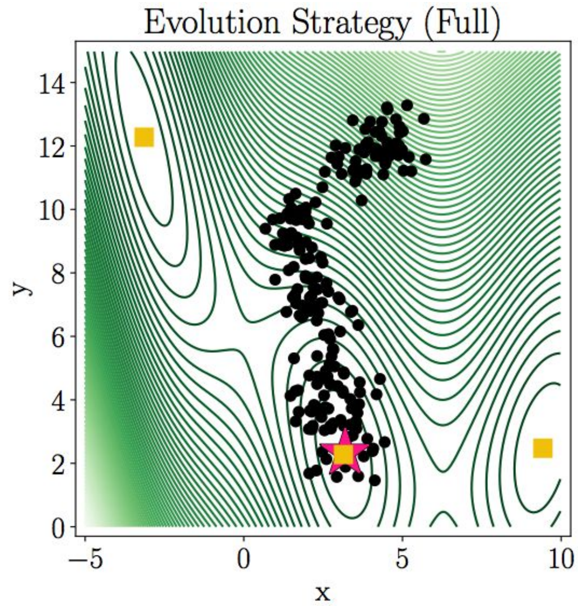
- ⇒ Yields local variants of BO parameterized by a local opt algorithm.

Overall intuition — view optimization as:

- Trying to estimate the output of a local opt algo, given limited budget of queries.

APPLICATIONS of BAX — *Bayesian local optimization*

After *208* queries



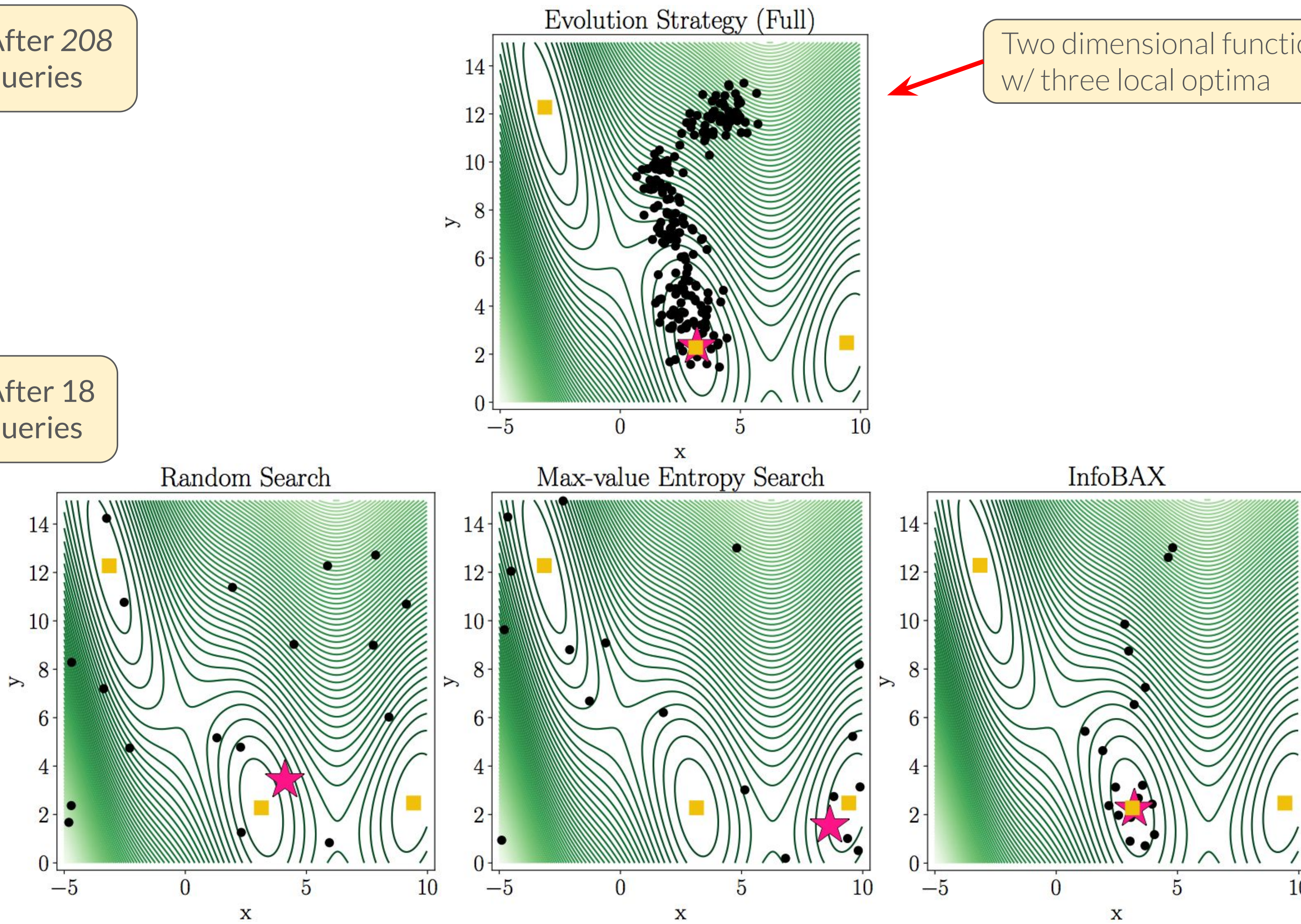
Two dimensional function, w/ three local optima

APPLICATIONS of BAX — *Bayesian local optimization*

After 208 queries

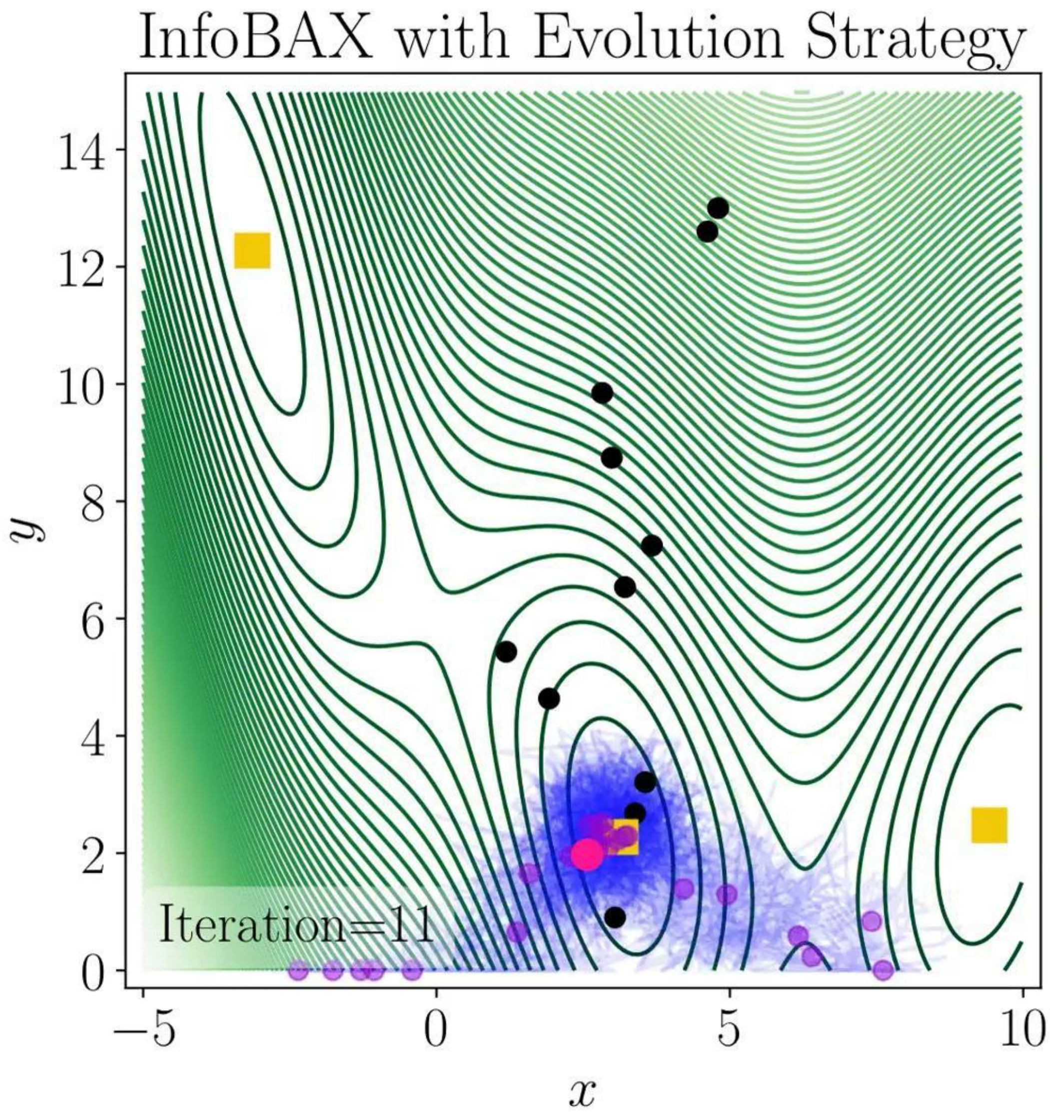
Two dimensional function, w/ three local optima

After 18 queries



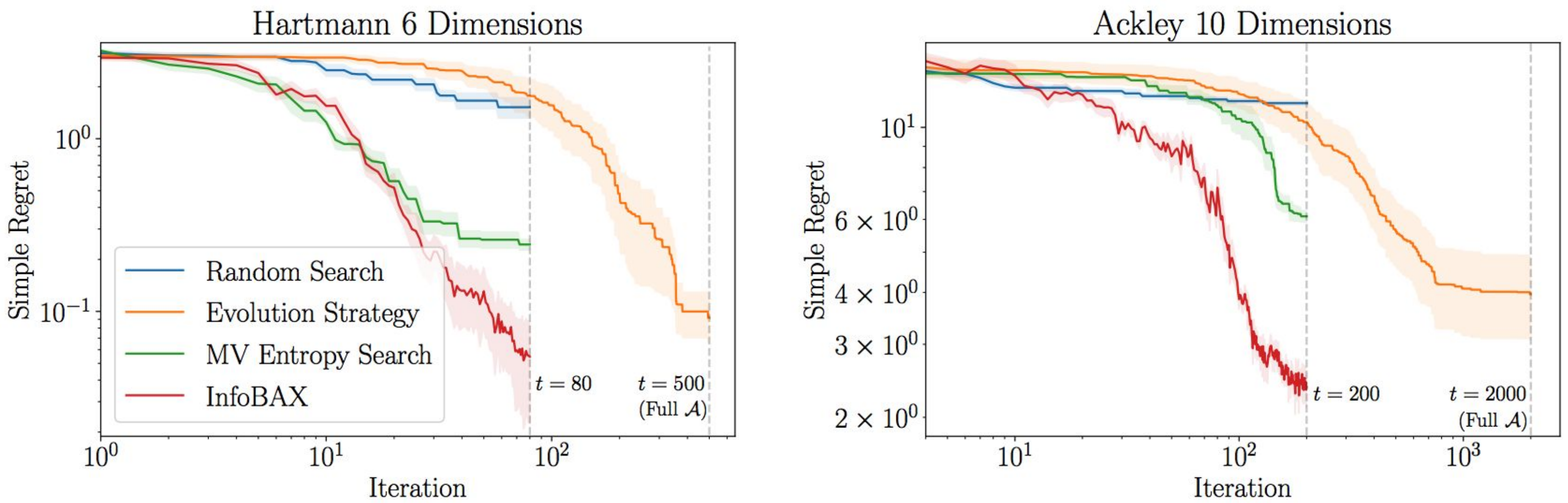
APPLICATIONS of BAX — *estimating shortest paths in graphs*

InfoBAX in action →



APPLICATIONS of BAX — *Bayesian local optimization*

InfoBAX matches performance of Evolution Strategy, using ***<10%*** of the queries.



***Future steps*:** try this out with a variety of local optimizers.

Application: *top-k estimation*

Suppose we have a large set of items.

- E.g. set of 500 catalyst materials / bulks.

Each item has a value under an expensive black-box function f.

- E.g. each catalyst bulk has an *activity level*, which is expensive to measure (simulate).

Suppose we want to determine the ***top-k*** items in the set.

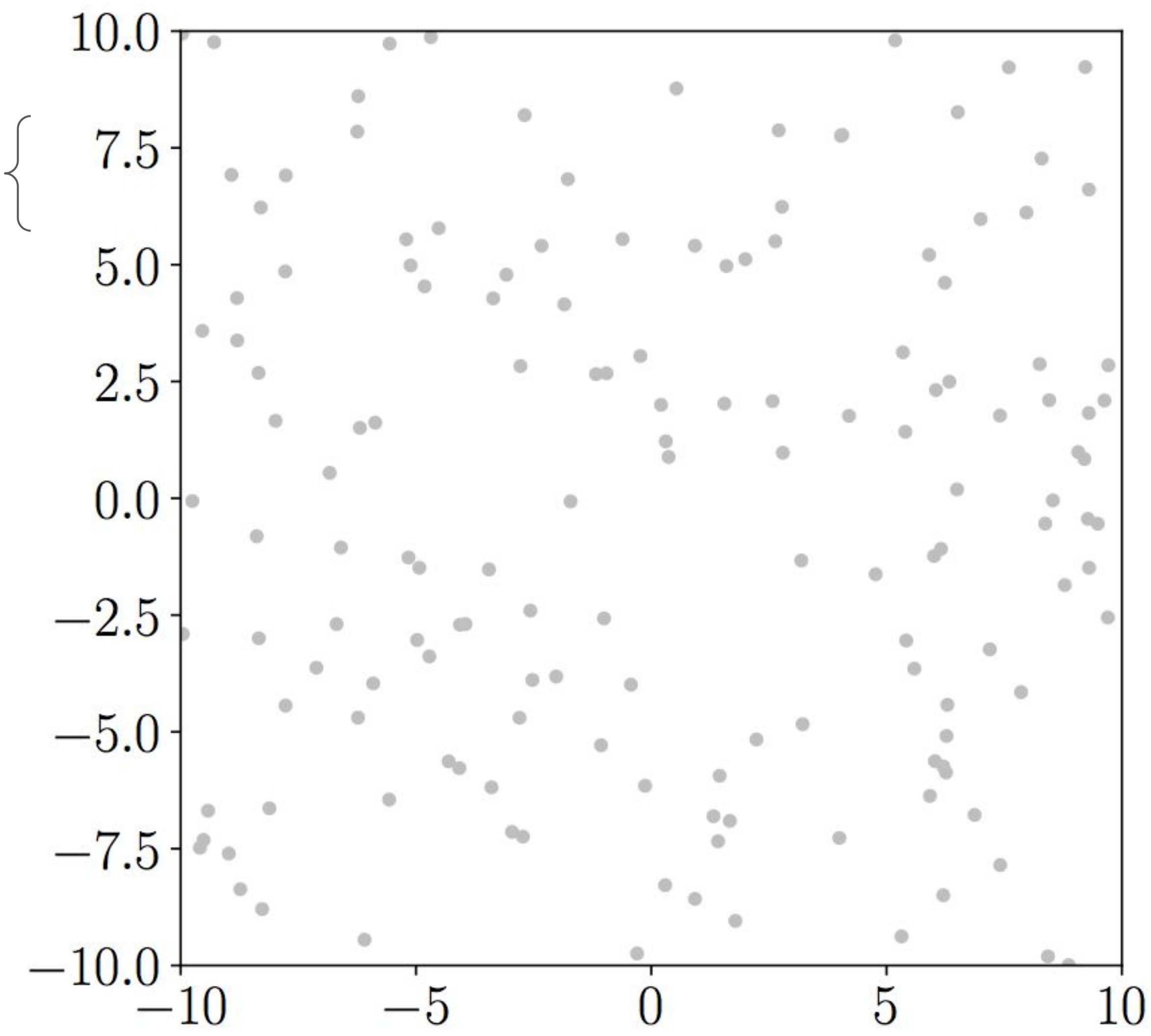
- E.g. the *top-10* catalysts, with highest activity ⇒ for experimental evaluation.
- (These ***top-k*** might then be filtered further based on additional tests)

⇒ distinct from both global optimization ($k=1$) and level set estimation.

Visualizing this…

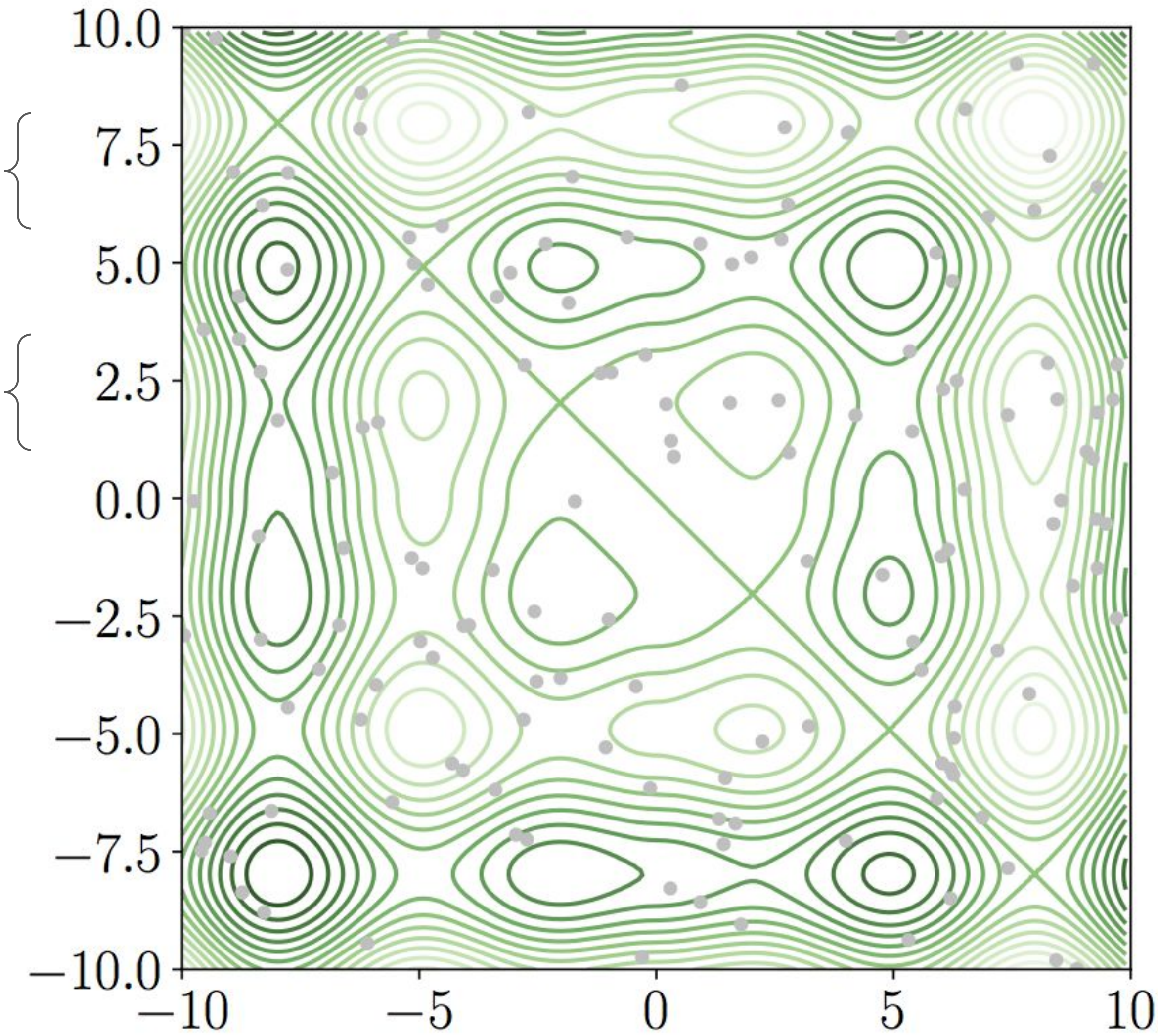
APPLICATIONS of BAX — *top-k estimation*

250 items



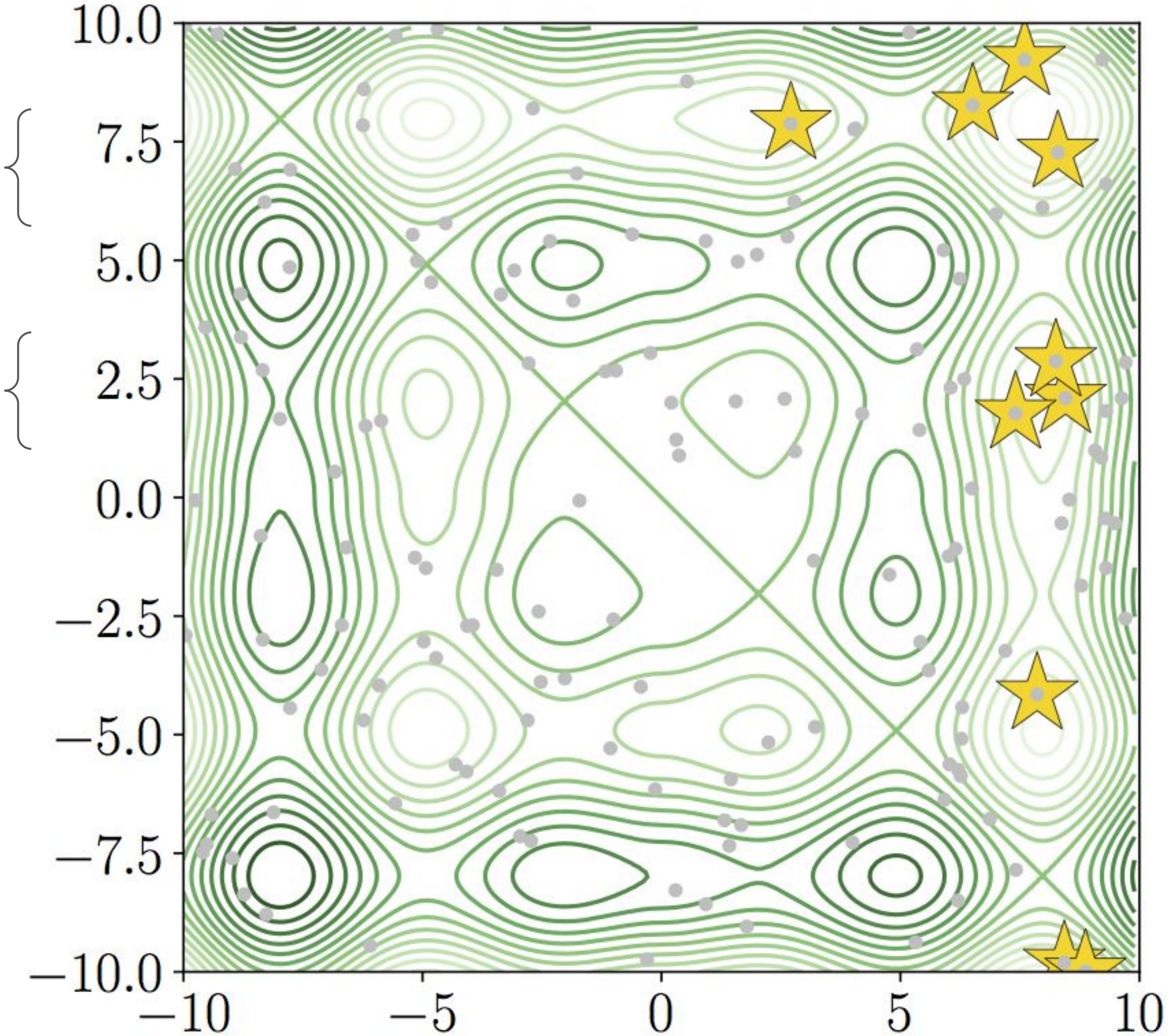
250 items

Black-box function f (green).



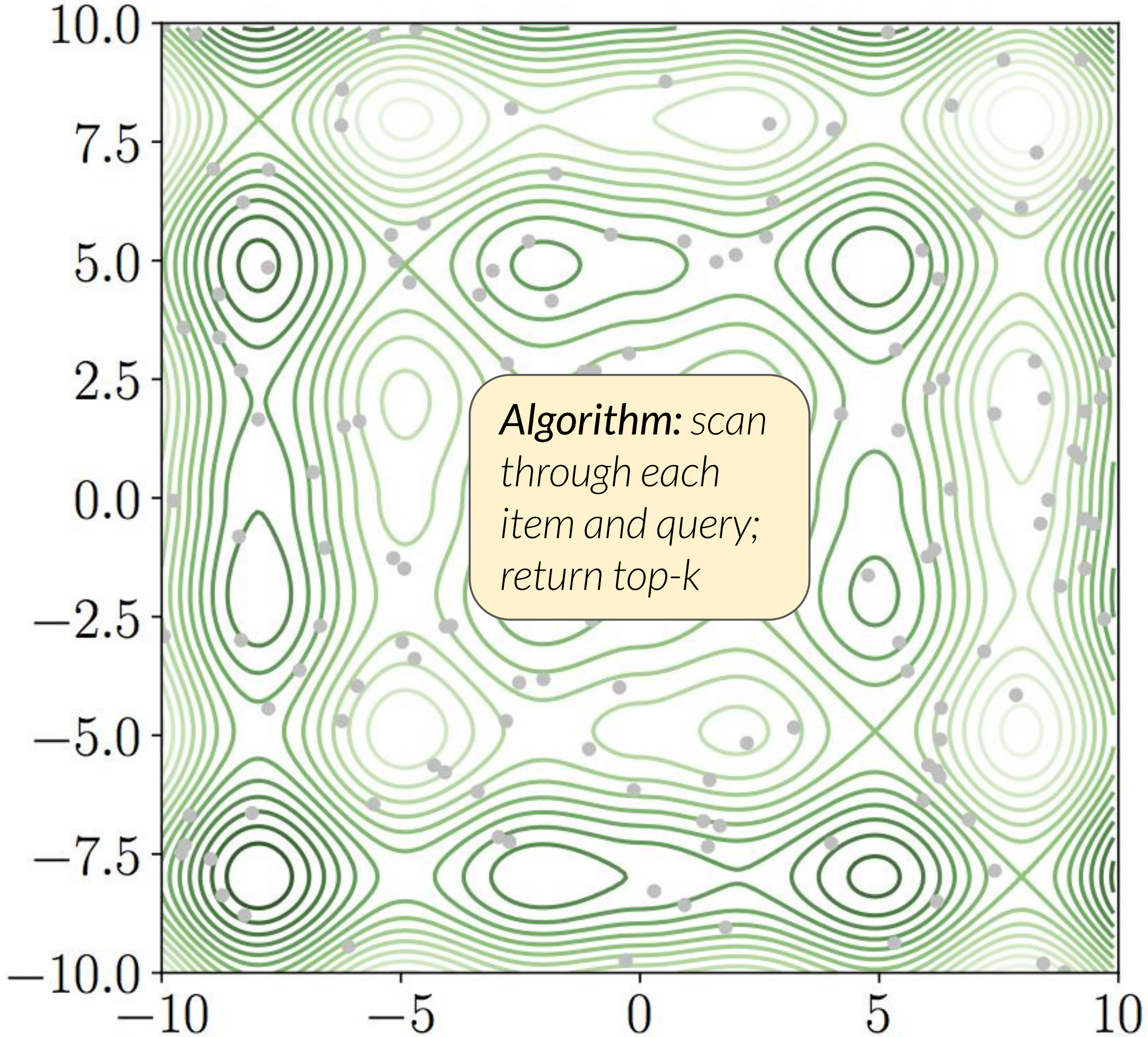
250 items

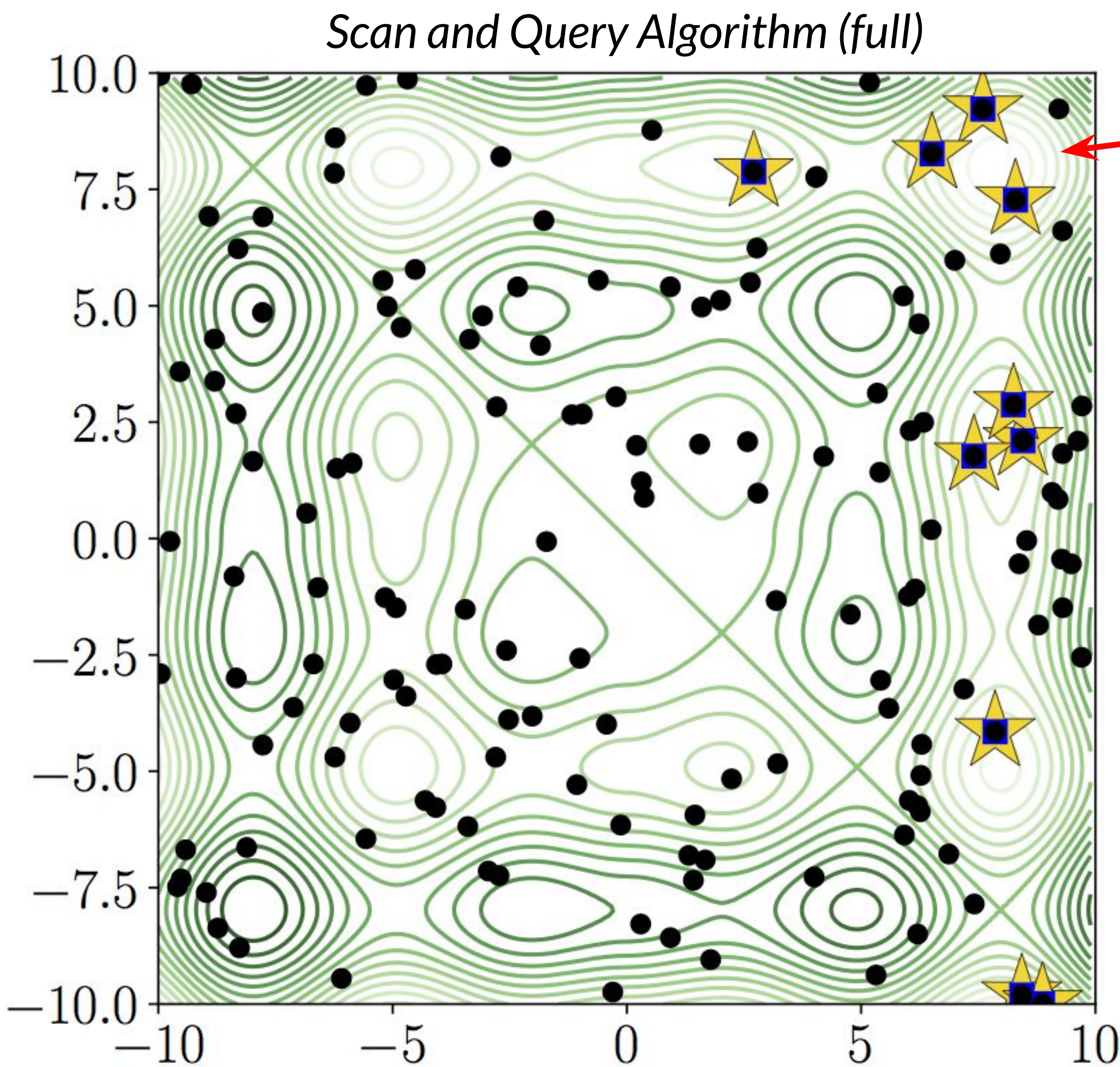
Black-box function f (green).



Stars denote *top-10* items

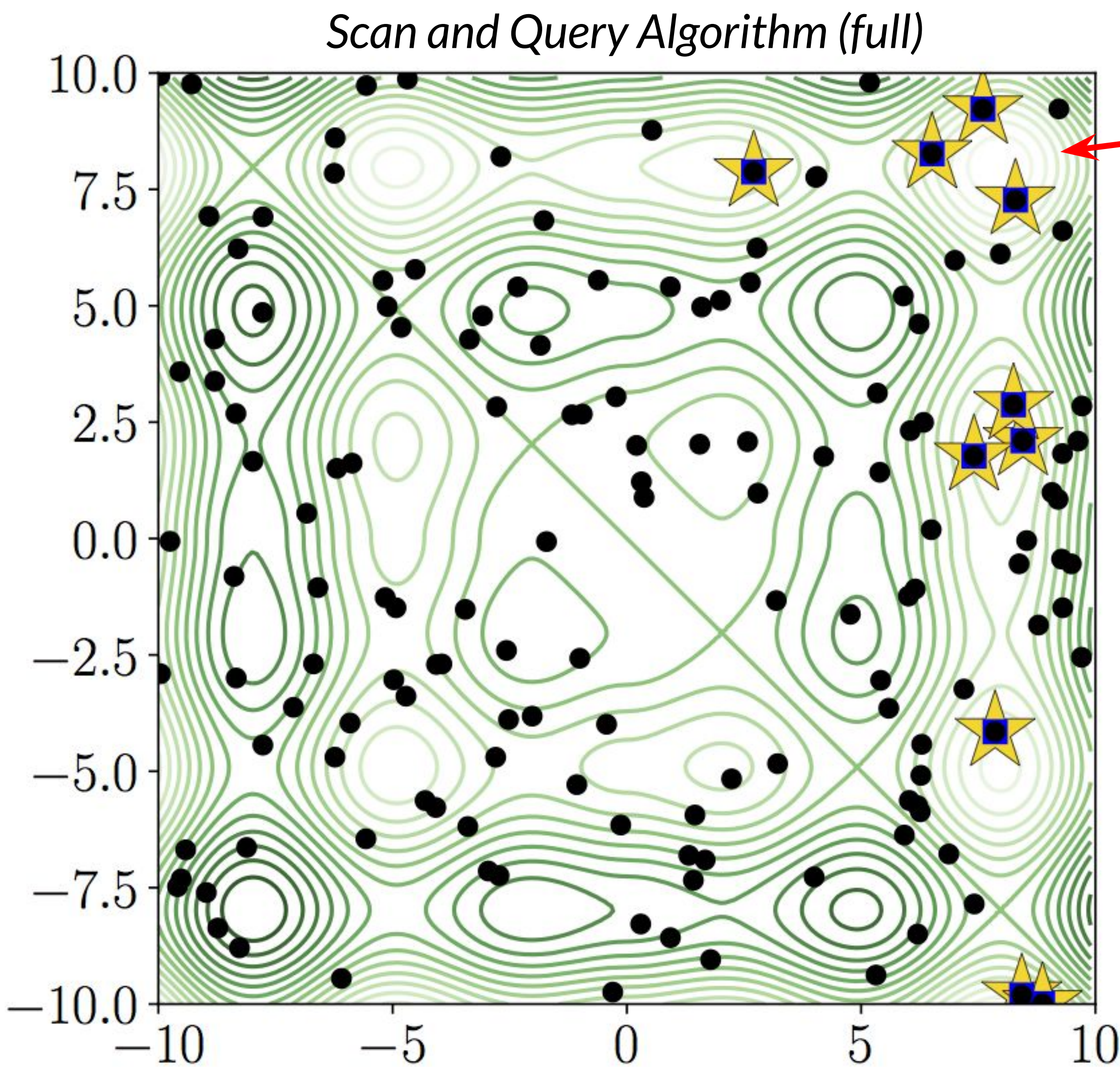
APPLICATIONS of BAX — *top-k estimation*





Algorithm yields perfect estimate

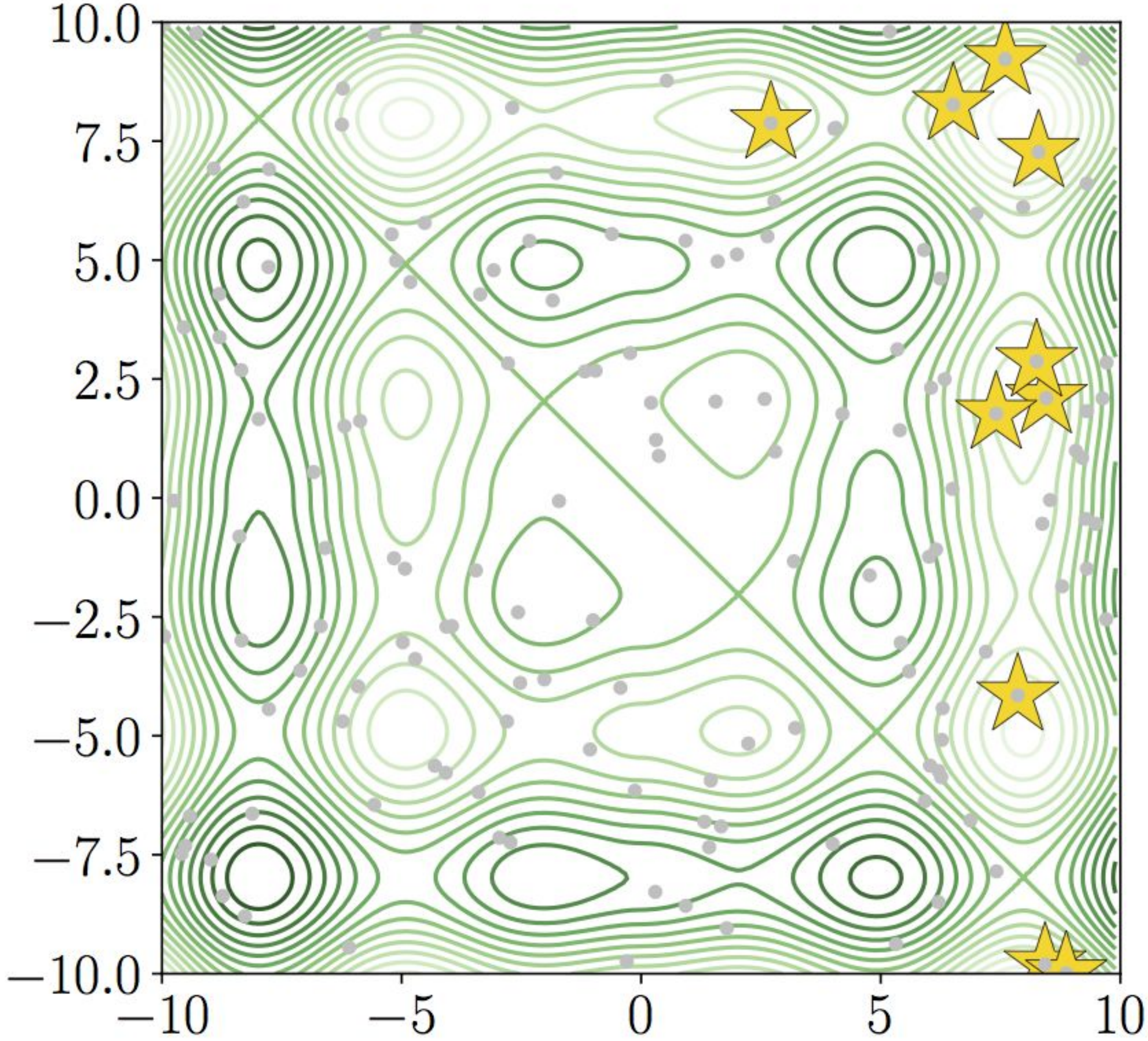
APPLICATIONS of BAX — *top-k estimation*



Algorithm yields perfect estimate

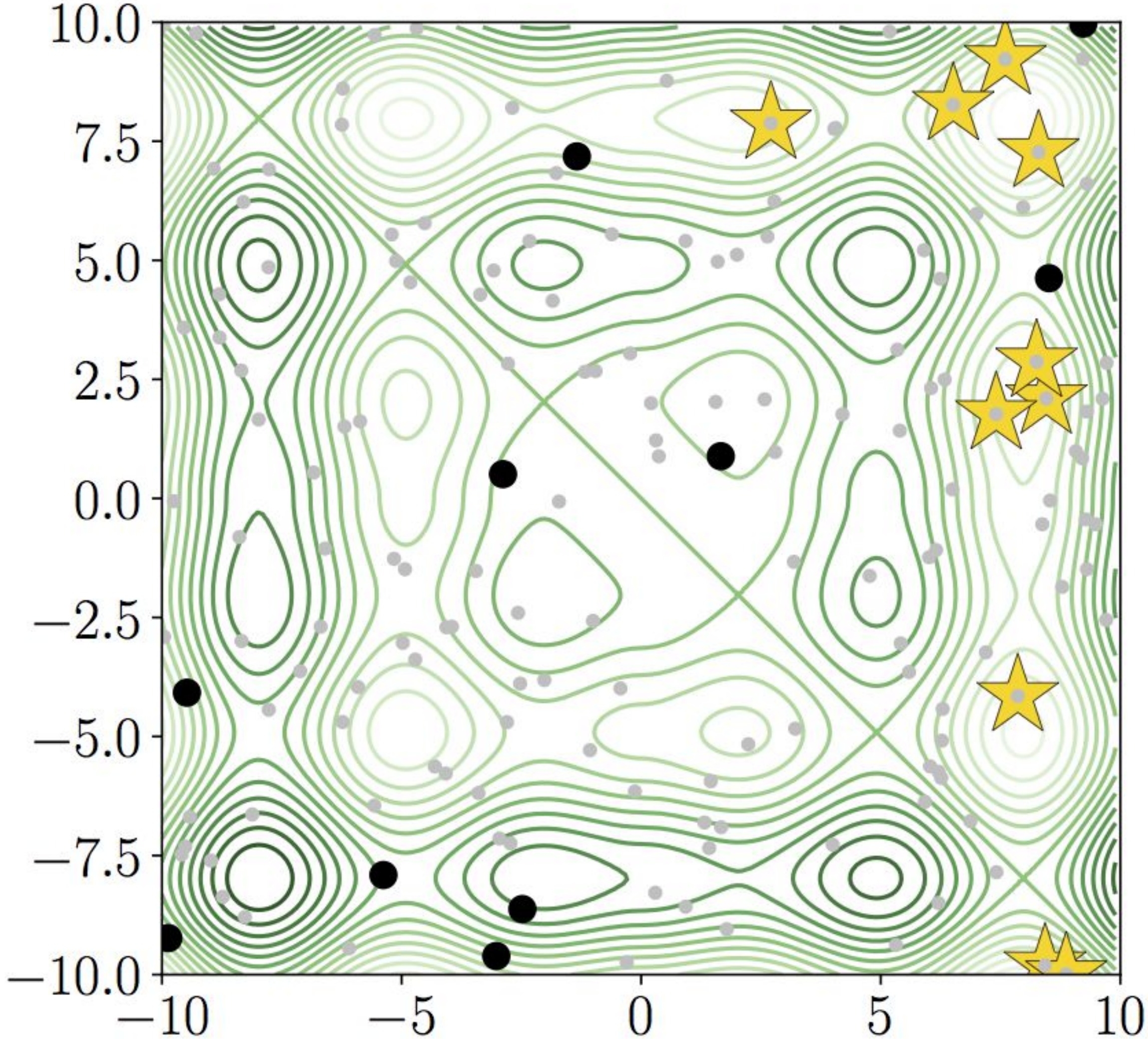
However: *uses 250 queries*

APPLICATIONS of BAX — *top-k estimation*



New strategy:
Make a few queries and infer top-10 items

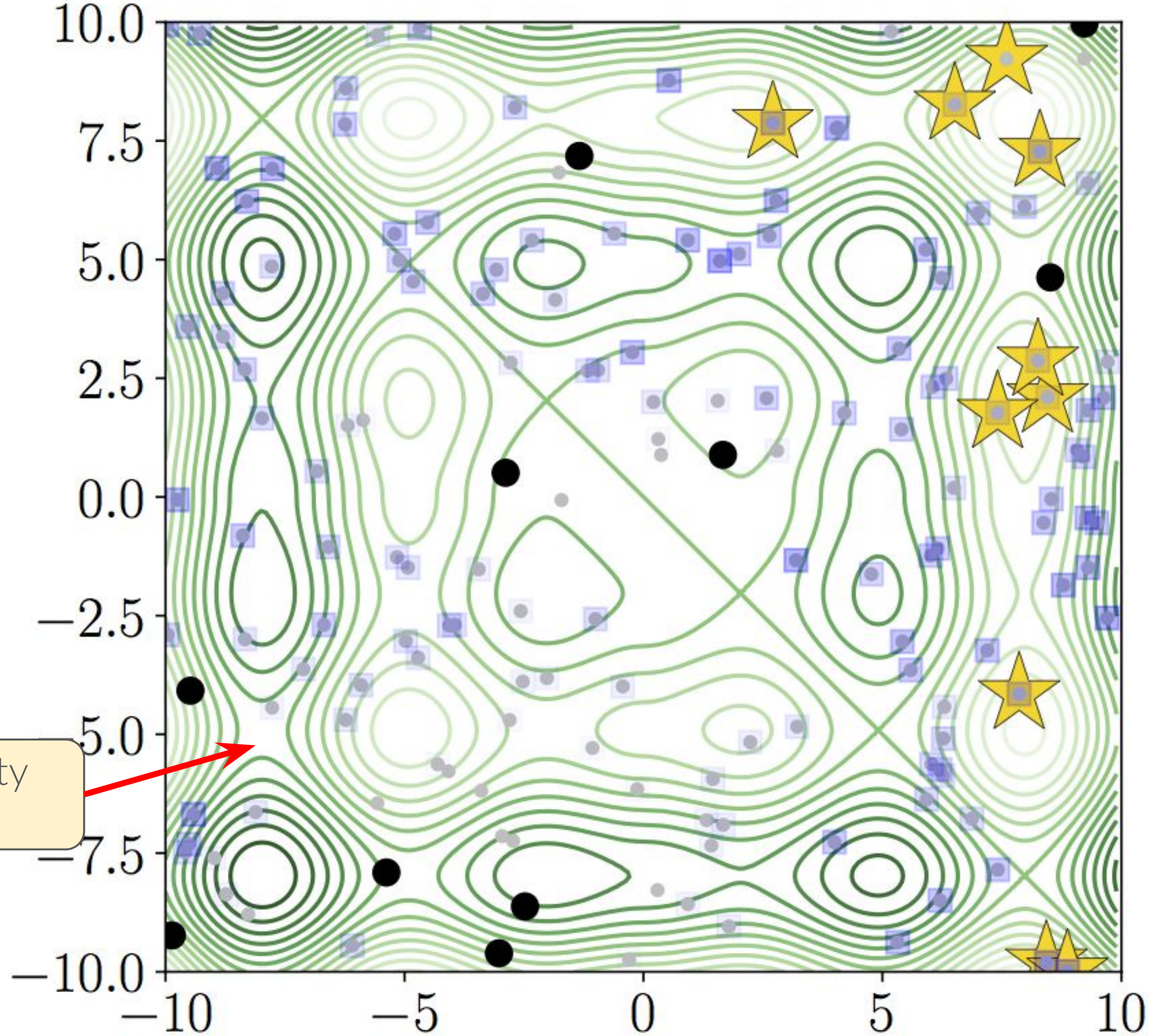
APPLICATIONS of BAX — *top-k estimation*



New strategy:
Make a few queries and infer top-10 items

10 queries (black dots)

APPLICATIONS of BAX — *top-k estimation*



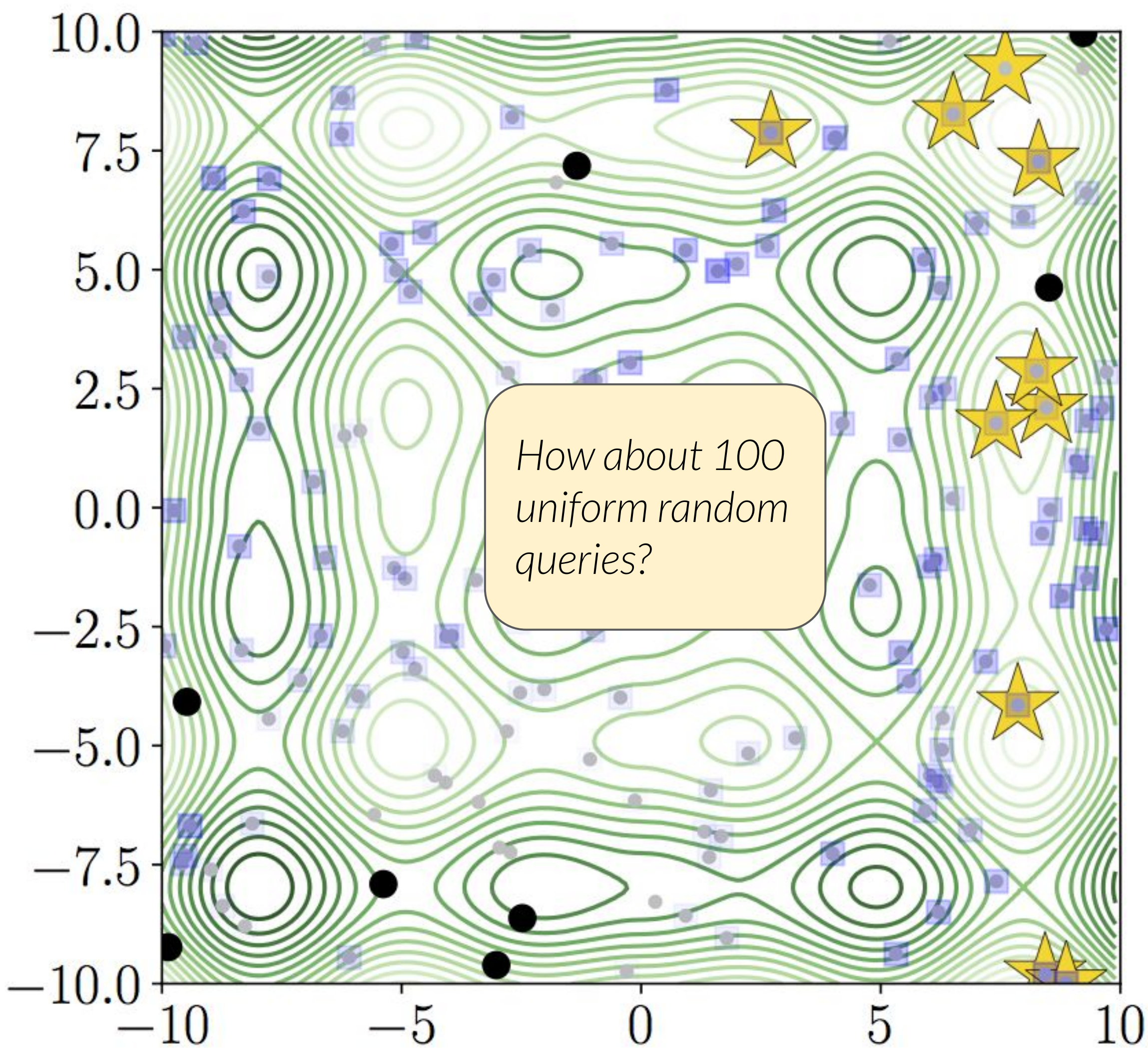
New strategy:
Make a few queries and infer top-10 items

Posterior samples of *top-10* items (blue squares)

10 queries (black dots)

Estimate is pretty bad

APPLICATIONS of BAX — *top-k estimation*

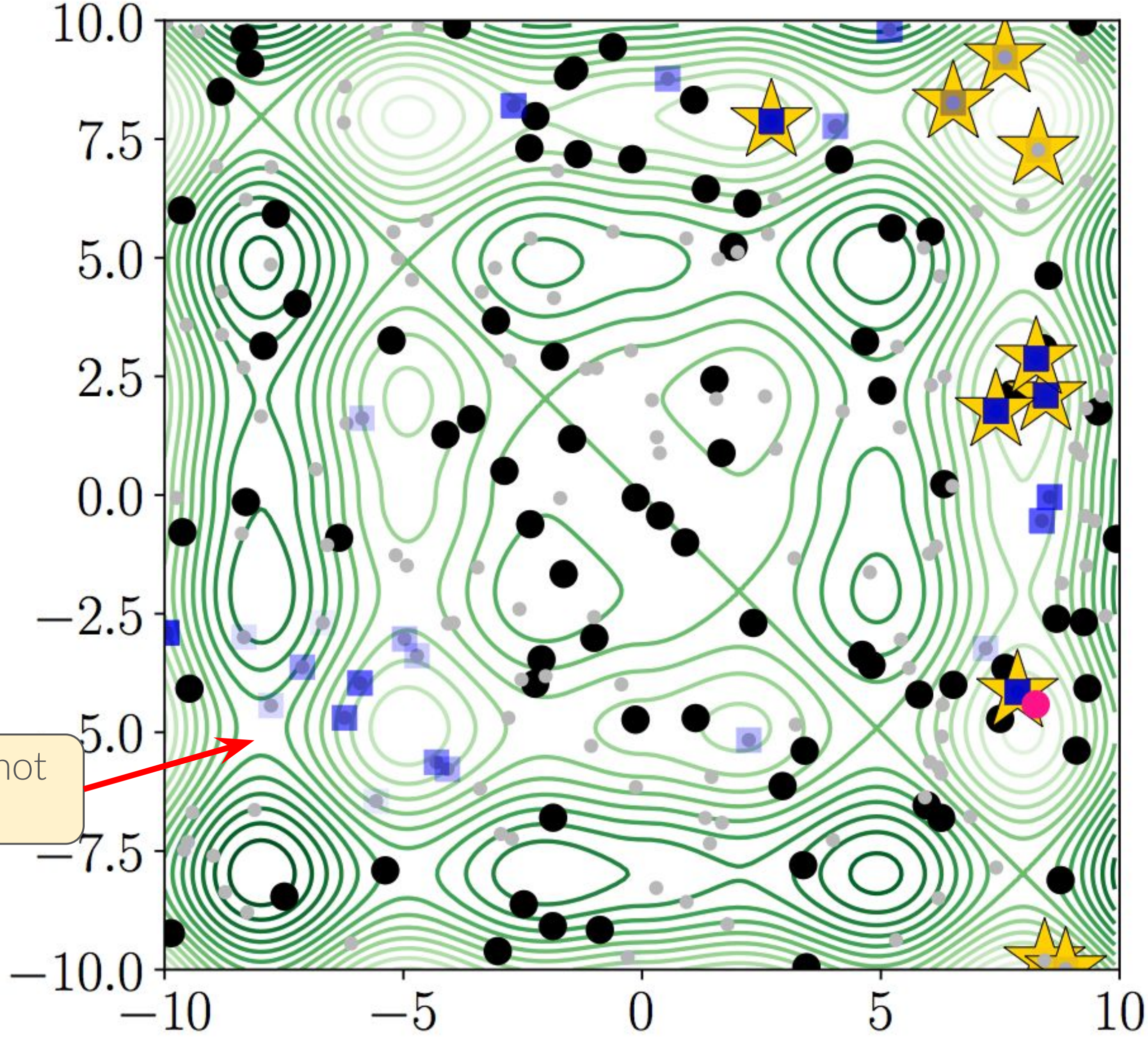


New strategy:
Make a few queries and infer top-10 items

Posterior samples of *top-10* items (blue squares)

10 queries (black dots)

APPLICATIONS of BAX — *top-k estimation*



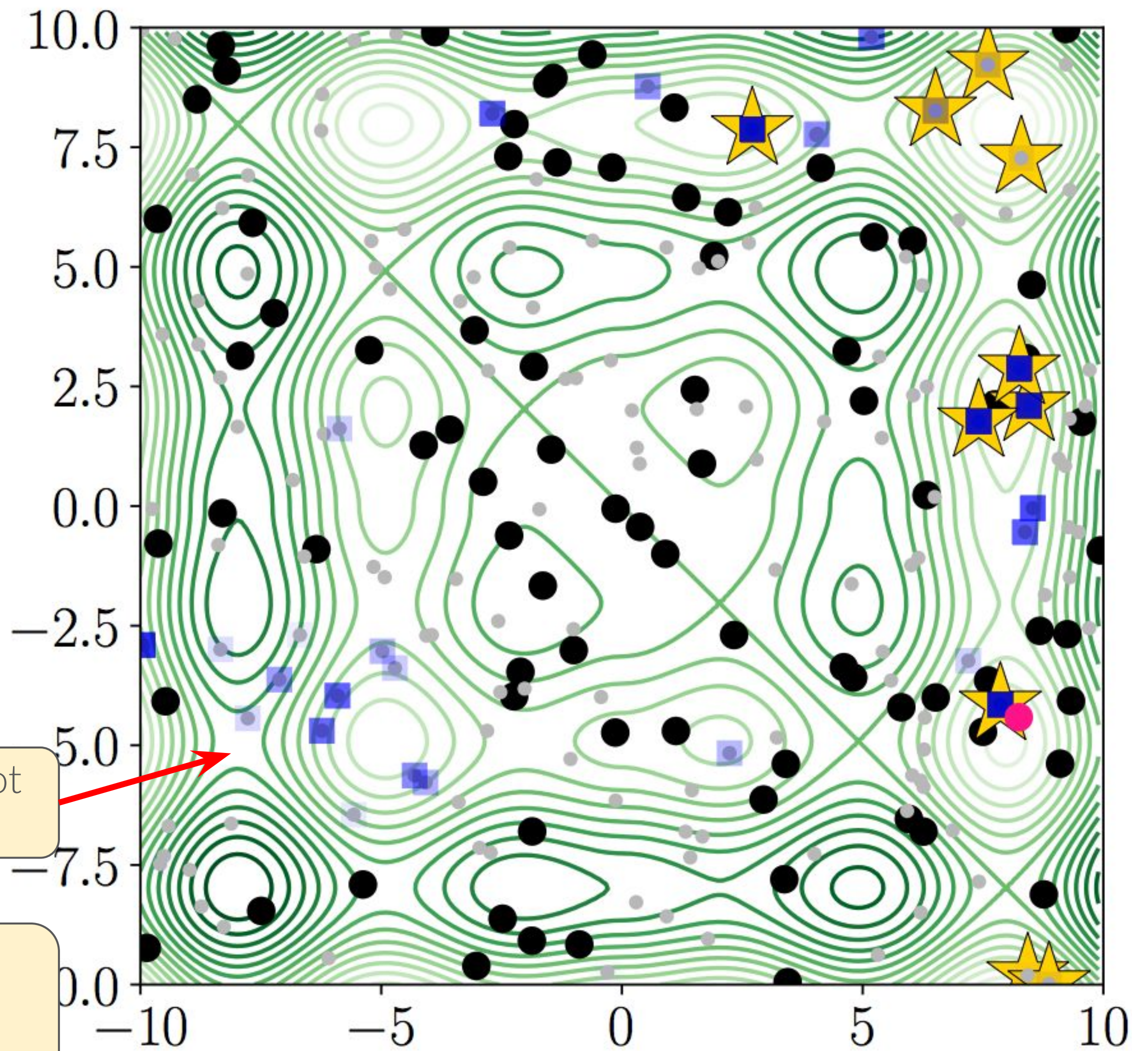
New strategy: *Make a few queries and infer top-10 items*

Posterior samples of *top-10* items (blue squares)

100 queries (black dots)

Estimate is still not great

APPLICATIONS of BAX — *top-k estimation*



New strategy:
Make a few queries and infer top-10 items

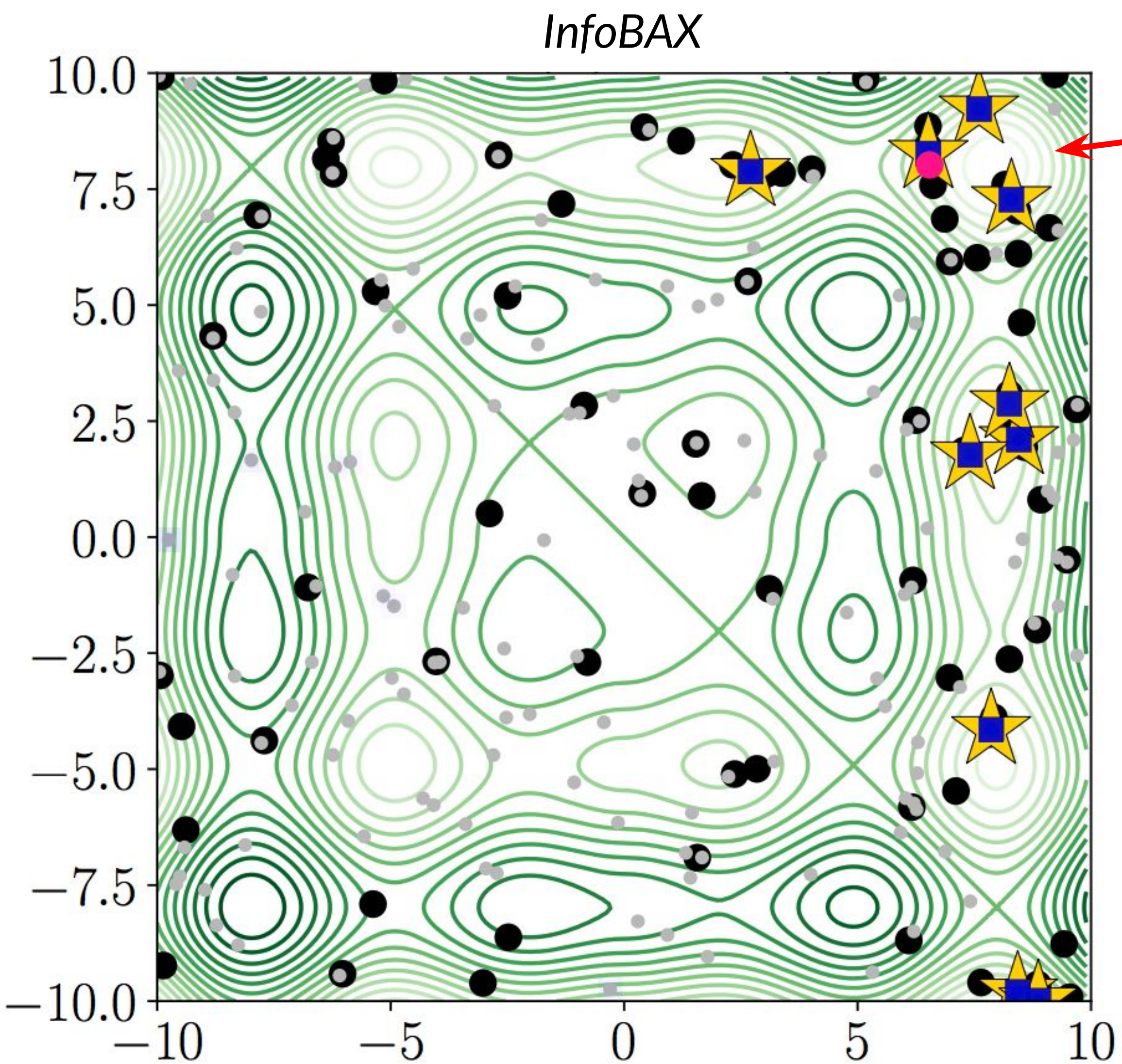
Posterior samples of *top-10* items (blue squares)

100 queries (black dots)

Estimate is still not great

Try ***InfoBAX*** *with "scan and query" algorithm*

APPLICATIONS of BAX — *top-k estimation*



Good estimate in <*100* queries

InfoBAX explores the space but samples densely around high-value items.

UNCERTAINTY MODELS

Final topic: software tools for uncertainty models

The BAX/BO procedures discussed all use *predictive uncertainty models*.

"A model of the conditional distribution over output y given an input x"

UNCERTAINTY MODELS

Final topic: software tools for uncertainty models

The BAX/BO procedures discussed all use *predictive uncertainty models*.

> *"A model of the conditional distribution over output y given an input x"*

⇒ in BAX we focus on GP models, but we may wish to run similar procedures on a variety of probabilistic models (*and to know if our models are good)*

Final topic: software tools for uncertainty models

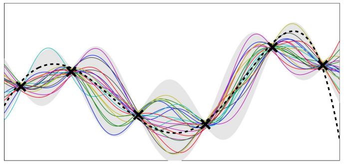
The BAX/BO procedures discussed all use *predictive uncertainty models*.

"A model of the conditional distribution over output y given an input x"

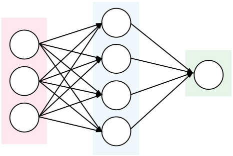
⇒ in BAX we focus on GP models, but we may wish to run similar procedures on a variety of probabilistic models (*and to know if our models are good)*

Types of uncertainty models:

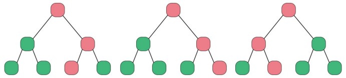
"Classic" Bayesian models: GPs, various (non)lin/hier/add or other Bayesian models



Neural models: probabilistic neural networks, BNN, neural processes, deep generative models



Also: ensembles, quantile regression, conformal prediction, etc.



ASSESSING UNCERTAINTY

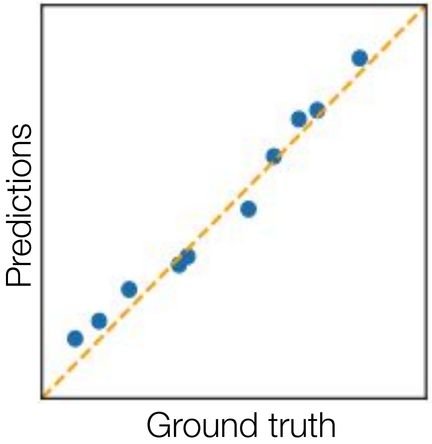
How can we assess quality of predictive uncertainty?

How can we assess quality of predictive uncertainty?

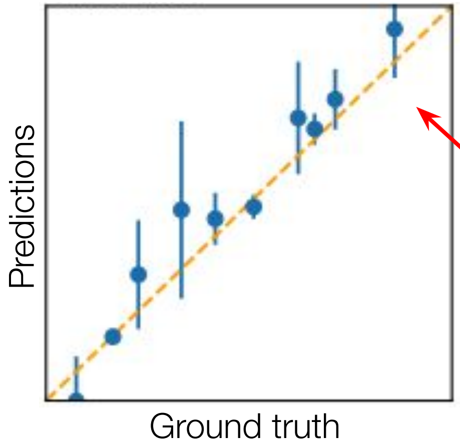
We can visualize point-predictions and predictive uncertainties on a given test set.

For each test point, plot ***predictions*** *vs.* ***ground truth values****:*

Point prediction model



Predictive uncertainty model



Uncertainty for point prediction: *interval, parametric distribution, samples, etc.*

ASSESSING UNCERTAINTY

How can we empirically assess predictive uncertainty?

Three important criteria are...

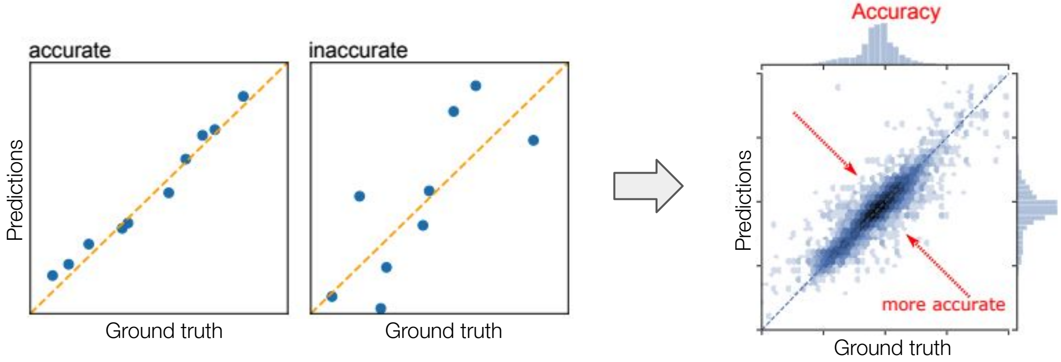
ASSESSING UNCERTAINTY

How can we empirically assess predictive uncertainty?

Three important criteria are...

Accuracy:

"How good is mean prediction? (agnostic to uncertainty)"



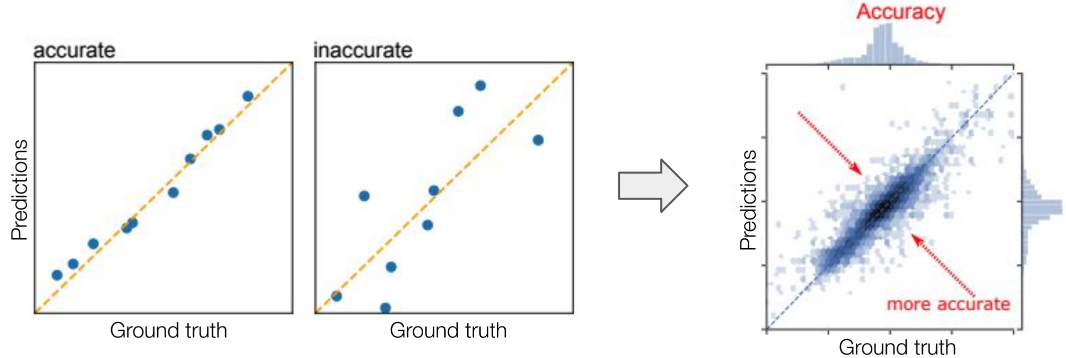
ASSESSING UNCERTAINTY

How can we empirically assess predictive uncertainty?

Three important criteria are...

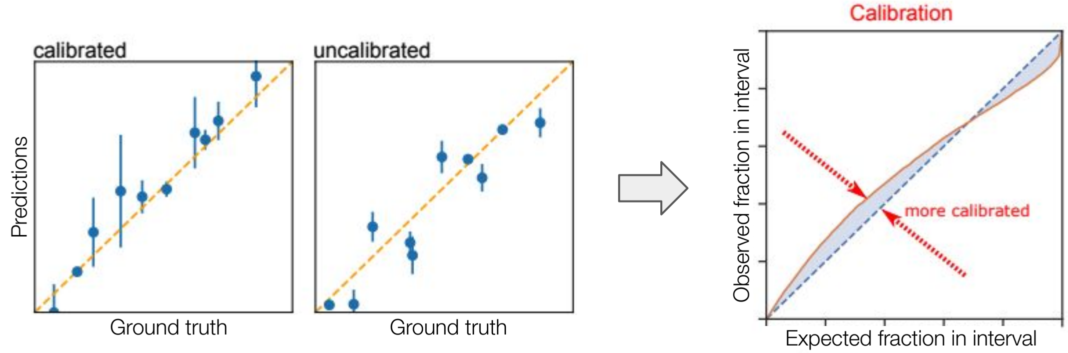
Accuracy:

"How good is mean prediction? (agnostic to uncertainty)"



Calibration:

"Is predictive uncertainty distribution under/over confident? (ignoring prediction accuracy)"

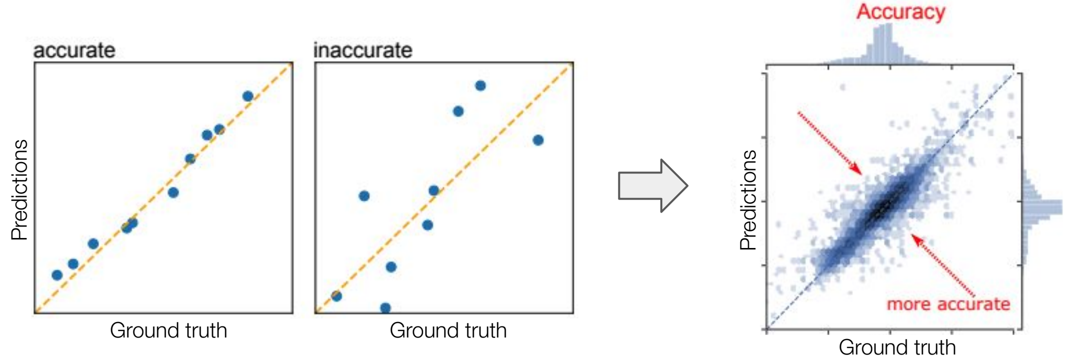


How can we empirically assess predictive uncertainty?

Three important criteria are...

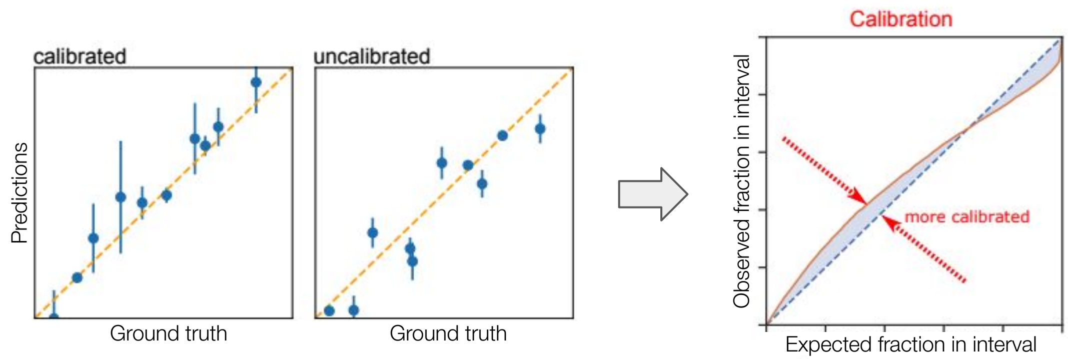
Accuracy:

"How good is mean prediction? (agnostic to uncertainty)"



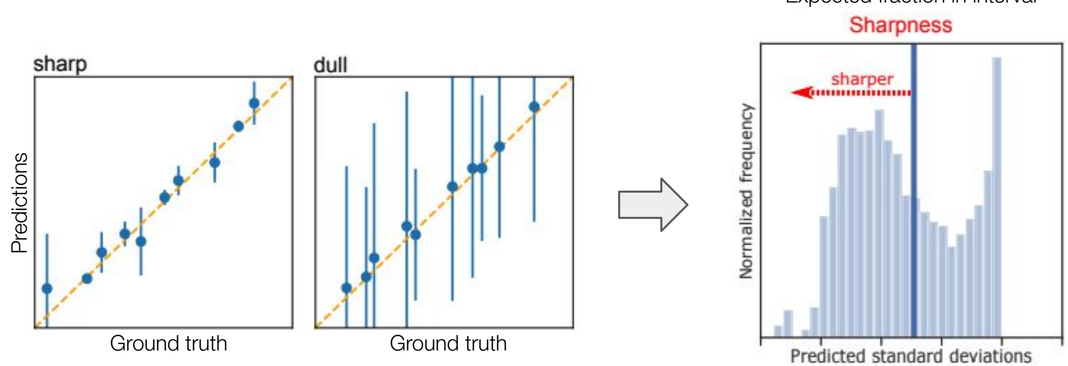
Calibration:

"Is predictive uncertainty distribution under/over confident? (ignoring prediction accuracy)"



Sharpness:

"On average, how confident are the predictions? (ignoring both of the above)"



Metrics for calibration

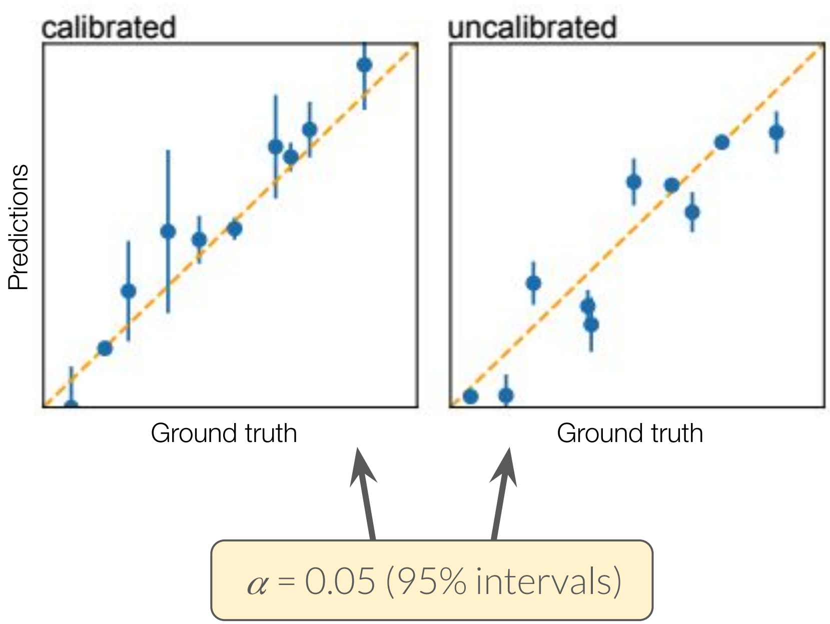
Suppose for each test point, our predictive uncertainty model returns a (1-α)-interval (e.g. 95% interval) of the predictive distribution.

Well-calibrated ⇒ *"the (1-α)-interval covers the true value (1-α)-proportion of the time, for all α"*

Metrics for calibration

Suppose for each test point, our predictive uncertainty model returns a (1-α)-interval (e.g. 95% interval) of the predictive distribution.

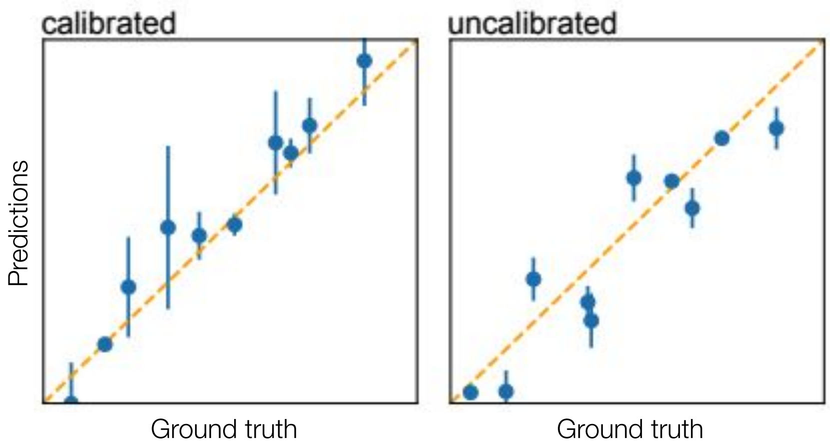
Well-calibrated **⇒** *"the (1-α)-interval covers the true value (1-α)-proportion of the time, for all α"*

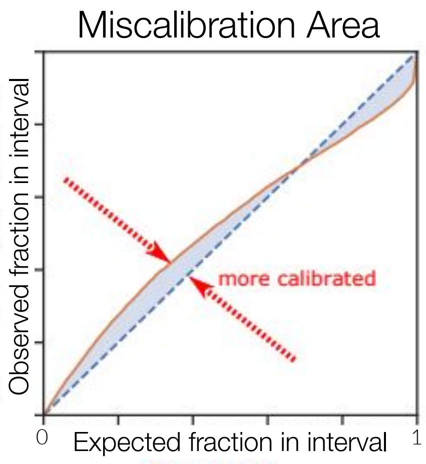


Metrics for calibration

Suppose for each test point, our predictive uncertainty model returns a ($1-\alpha$)-interval (e.g. 95% interval) of the predictive distribution.

Well-calibrated ⇒ *"the ($1-\alpha$)-interval covers the true value ($1-\alpha$)-proportion of the time, for all α"*





Can scan from α=0 to α=1, and compute:

(1) expected fraction of true values contained in interval

(2) observed fraction of true values contained in interval

Uncertainty Toolbox

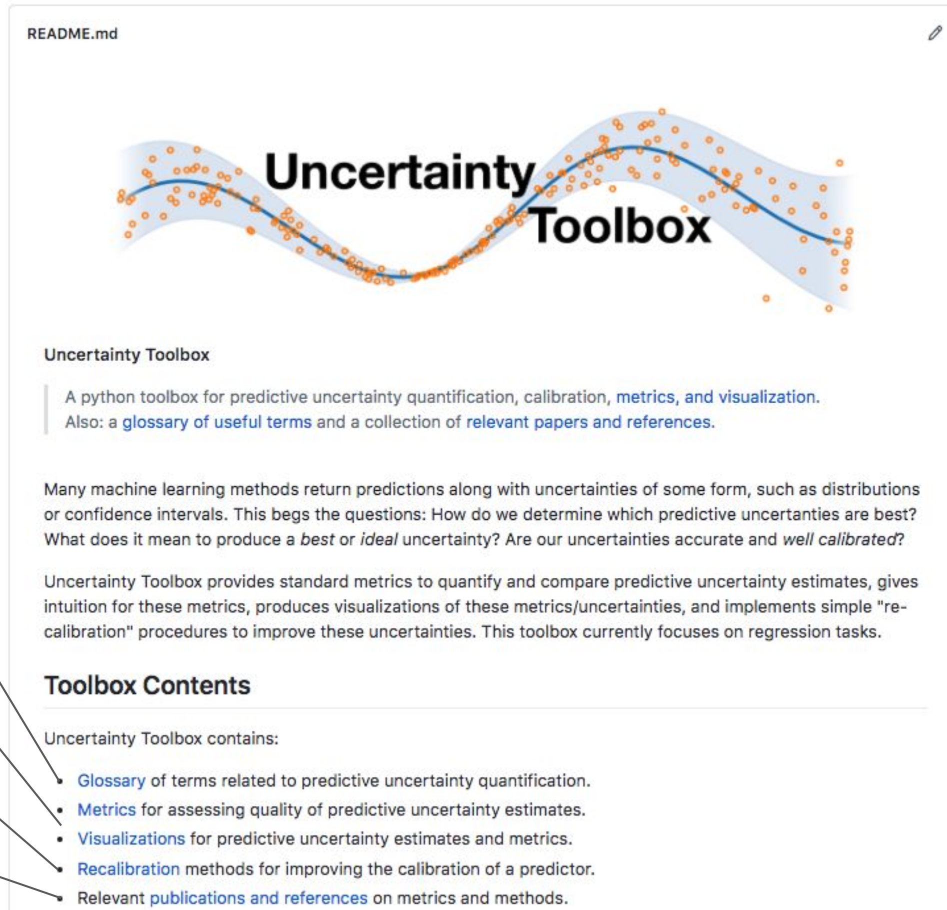
- To help assess uncertainty quantification methods, we released Uncertainty Toolbox.
- "A python toolbox for predictive uncertainty quantification, calibration, metrics, and visualization" → github.com/uncertainty-toolbox/uncertainty-toolbox

Glossary

Metrics / Viz

Recalibration

Relevant Papers


README.md

Uncertainty Toolbox

Uncertainty Toolbox

> A python toolbox for predictive uncertainty quantification, calibration, metrics, and visualization.
> Also: a glossary of useful terms and a collection of relevant papers and references.

Many machine learning methods return predictions along with uncertainties of some form, such as distributions or confidence intervals. This begs the questions: How do we determine which predictive uncertanties are best? What does it mean to produce a *best* or *ideal* uncertainty? Are our uncertainties accurate and *well calibrated*?

Uncertainty Toolbox provides standard metrics to quantify and compare predictive uncertainty estimates, gives intuition for these metrics, produces visualizations of these metrics/uncertainties, and implements simple "re-calibration" procedures to improve these uncertainties. This toolbox currently focuses on regression tasks.

Toolbox Contents

Uncertainty Toolbox contains:

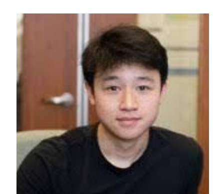
- Glossary of terms related to predictive uncertainty quantification.
- Metrics for assessing quality of predictive uncertainty estimates.
- Visualizations for predictive uncertainty estimates and metrics.
- Recalibration methods for improving the calibration of a predictor.
- Relevant publications and references on metrics and methods.

Collaborators


Kevin Tran


Young Chung


Ian Char


Han Guo

CONCLUSION

In summary…

We extend Bayesian optimization from targeting *global optima* to targeting *other function properties defined by algorithms*.

⇒ Introduce the task of **BAX**,
and the information-based procedure ***InfoBAX***

BAX Paper: *arxiv.org/abs/2104.09460*

Uncertainty Toolbox: *github.com/uncertainty-toolbox/uncertainty-toolbox*

Thanks for listening!

ACQUISITION FUNCTION DETAILS

ACQUISITION FUNCTION DETAILS

InfoBAX expected info gain (EIG) acquisition function

"Expected decrease in entropy on the ***algorithm output****, if we were to query f at x."*

$$\mathrm{EIG}_t(x) = H[p(O_{\mathcal{A}} \mid \mathcal{D}_t)] - \mathbb{E}_{p(y_x \mid \mathcal{D}_t)}[H[p(O_{\mathcal{A}} \mid \mathcal{D}_t \cup \{(x, y_x)\})]]$$

- entropy of posterior distribution over algorithm output
- minus
- expected entropy of posterior distribution over algorithm output, *... if we were to make a query at x*

ACQUISITION FUNCTION DETAILS

InfoBAX expected info gain (EIG) acquisition function

*"Expected decrease in entropy on the **algorithm output**, if we were to query f at x."*

$$\mathrm{EIG}_t(x) = H[p(O_{\mathcal{A}} \mid \mathcal{D}_t)] - \mathbb{E}_{p(y_x \mid \mathcal{D}_t)}[H[p(O_{\mathcal{A}} \mid \mathcal{D}_t \cup \{(x, y_x)\})]]$$

entropy of posterior distribution over algorithm output

minus

expected entropy of posterior distribution over algorithm output, *... if we were to make a query at x*

Can consider the following related objective...

ACQUISITION FUNCTION DETAILS

Related objective: EIG on execution path

"Expected decrease in entropy on the ***execution path****, if we were to query f at x."*

$$\mathrm{EIG}_t^e(x) = \mathrm{H}\left[e_\mathcal{A} \mid \mathcal{D}_t\right] - \mathbb{E}_{p(y_x|\mathcal{D}_t)}\left[\mathrm{H}\left[e_\mathcal{A} \mid D_t \cup \{(x, y_x)\}\right]\right]$$

ACQUISITION FUNCTION DETAILS

Related objective: EIG on execution path

*"Expected decrease in entropy on the **execution path**, if we were to query f at x."*

$$\mathrm{EIG}_t^e(x) = \mathrm{H}\left[e_{\mathcal{A}} \mid \mathcal{D}_t\right] - \mathbb{E}_{p(y_x \mid \mathcal{D}_t)}\left[\mathrm{H}\left[e_{\mathcal{A}} \mid D_t \cup \{(x, y_x)\}\right]\right]$$

entropy of posterior distribution over **execution path**

minus

expected entropy of posterior distribution over **execution path**, *… if we were to make a query at x*

Related objective: EIG on execution path

*"Expected decrease in entropy on the **execution path**, if we were to query f at x."*

$$\mathrm{EIG}_t^e(x) = \mathrm{H}\left[e_{\mathcal{A}} \mid \mathcal{D}_t\right] - \mathbb{E}_{p(y_x \mid \mathcal{D}_t)}\left[\mathrm{H}\left[e_{\mathcal{A}} \mid D_t \cup \{(x, y_x)\}\right]\right]$$

entropy of posterior distribution over **execution path**

minus

expected entropy of posterior distribution over **execution path**, *... if we were to make a query at x*

If we know function value at each point in execution path ⇒ we know property.

However: this acquisition function may be effective, but *suboptimal in general*.

(Parts of execution path may not be very important for final output)

(E.g. consider case where subsequence in execution path has no influence on later steps/output)

ACQUISITION FUNCTION DETAILS

Related objective: EIG on execution path

We can efficiently estimate this via following trick:

$$\text{EIG}_t^e(x) = \text{H}\left[e_{\mathcal{A}} \mid \mathcal{D}_t\right] - \mathbb{E}_{p(y_x|\mathcal{D}_t)}\left[\text{H}\left[e_{\mathcal{A}} \mid D_t \cup \{(x, y_x)\}\right]\right]$$

ACQUISITION FUNCTION DETAILS

Related objective: EIG on execution path

We can efficiently estimate this via following trick:

$$\mathrm{EIG}_t^e(x) = \mathrm{H}\left[e_{\mathcal{A}} \mid \mathcal{D}_t\right] - \mathbb{E}_{p(y_x \mid \mathcal{D}_t)}\left[\mathrm{H}\left[e_{\mathcal{A}} \mid D_t \cup \{(x, y_x)\}\right]\right]$$

$$= \mathrm{H}\left[y_x \mid \mathcal{D}_t\right] - \mathbb{E}_{p(e_{\mathcal{A}} \mid \mathcal{D}_t)}\left[\mathrm{H}\left[y_x \mid \mathcal{D}_t, e_{\mathcal{A}}\right]\right]$$

entropy of posterior predictive at x.

minus

expected entropy of posterior predictive **given execution path**, *… if we were to make a query at* x

ACQUISITION FUNCTION DETAILS

Related objective: EIG on execution path

We can efficiently estimate this via following trick:

$$\mathrm{EIG}_t^e(x) = \mathrm{H}\left[e_{\mathcal{A}} \mid \mathcal{D}_t\right] - \mathbb{E}_{p(y_x \mid \mathcal{D}_t)}\left[\mathrm{H}\left[e_{\mathcal{A}} \mid D_t \cup \{(x, y_x)\}\right]\right]$$

$$= \mathrm{H}\left[y_x \mid \mathcal{D}_t\right] - \mathbb{E}_{p(e_{\mathcal{A}} \mid \mathcal{D}_t)}\left[\mathrm{H}\left[y_x \mid \mathcal{D}_t, e_{\mathcal{A}}\right]\right]$$

entropy of posterior predictive at x.

minus

expected entropy of posterior predictive **given execution path**,
... if we were to make a query at x

And note that

$$p\left(y_x \mid \mathcal{D}_t, \tilde{e}_{\mathcal{A}}\right) = p\left(y_x \,\middle|\, \mathcal{D}_t, \{\tilde{f}_{z_s}\}_{s=1}^{S}\right)$$

Gaussian

ACQUISITION FUNCTION DETAILS

Back to the: EIG on algorithm output

We can similarity write:

$$\mathrm{EIG}_t(x) = H[p(O_{\mathcal{A}} \mid \mathcal{D}_t)] - \mathbb{E}_{p(y_x \mid \mathcal{D}_t)}[H[p(O_{\mathcal{A}} \mid \mathcal{D}_t \cup \{(x, y_x)\})]]$$

entropy of posterior distribution over algorithm output

minus

expected entropy of posterior distribution over algorithm output,
… if we were to make a query at x

ACQUISITION FUNCTION DETAILS

Back to the: EIG on algorithm output

We can similarity write:

$$\mathrm{EIG}_t(x) = H[p(O_{\mathcal{A}} \mid \mathcal{D}_t)] - \mathbb{E}_{p(y_x \mid \mathcal{D}_t)}[H[p(O_{\mathcal{A}} \mid \mathcal{D}_t \cup \{(x, y_x)\})]]$$

$$= \mathrm{H}\left[y_x \mid \mathcal{D}_t\right] - \mathbb{E}_{p(O_{\mathcal{A}} \mid \mathcal{D}_t)}\left[\mathrm{H}\left[y_x \mid \mathcal{D}_t, O_{\mathcal{A}}\right]\right]$$

ACQUISITION FUNCTION DETAILS

Back to the: EIG on algorithm output

We can similarity write:

$$\mathrm{EIG}_t(x) = H[p(O_{\mathcal{A}} \mid \mathcal{D}_t)] - \mathbb{E}_{p(y_x \mid \mathcal{D}_t)}[H[p(O_{\mathcal{A}} \mid \mathcal{D}_t \cup \{(x, y_x)\})]]$$

$$= \mathrm{H}\left[y_x \mid \mathcal{D}_t\right] - \mathbb{E}_{p(O_{\mathcal{A}} \mid \mathcal{D}_t)}\left[\mathrm{H}\left[y_x \mid \mathcal{D}_t, O_{\mathcal{A}}\right]\right]$$

Difficult

ACQUISITION FUNCTION DETAILS

Back to the: EIG on algorithm output

We can similarity write:

$$\mathrm{EIG}_t(x) = H[p(O_{\mathcal{A}} \mid \mathcal{D}_t)] - \mathbb{E}_{p(y_x \mid \mathcal{D}_t)}[H[p(O_{\mathcal{A}} \mid \mathcal{D}_t \cup \{(x, y_x)\})]]$$

$$= \mathrm{H}[y_x \mid \mathcal{D}_t] - \mathbb{E}_{p(O_{\mathcal{A}} \mid \mathcal{D}_t)}[\mathrm{H}[y_x \mid \mathcal{D}_t, O_{\mathcal{A}}]]$$

Note that:

$$\begin{aligned} p(y_x \mid \mathcal{D}_t, O_{\mathcal{A}}) &= \int p(y_x, e_{\mathcal{A}} \mid \mathcal{D}_t, O_{\mathcal{A}}) \, \mathrm{d}e_{\mathcal{A}} \\ &= \int p(y_x \mid \mathcal{D}_t, e_{\mathcal{A}}, O_{\mathcal{A}}) \, p(e_{\mathcal{A}} \mid O_{\mathcal{A}}, \mathcal{D}_t) \, \mathrm{d}e_{\mathcal{A}} \\ &= \mathbb{E}_{p(e_{\mathcal{A}} \mid O_{\mathcal{A}}, \mathcal{D}_t)} [p(y_x \mid \mathcal{D}_t, e_{\mathcal{A}})]. \end{aligned}$$

ACQUISITION FUNCTION DETAILS

Back to the: EIG on algorithm output

We can similarity write:

$$\mathrm{EIG}_t(x) = H[p(O_{\mathcal{A}} \mid \mathcal{D}_t)] - \mathbb{E}_{p(y_x \mid \mathcal{D}_t)}[H[p(O_{\mathcal{A}} \mid \mathcal{D}_t \cup \{(x, y_x)\})]]$$

$$= \mathrm{H}\left[y_x \mid \mathcal{D}_t\right] - \mathbb{E}_{p(O_{\mathcal{A}} \mid \mathcal{D}_t)}\left[\mathrm{H}\left[y_x \mid \mathcal{D}_t, O_{\mathcal{A}}\right]\right]$$

Note that:

$$p(y_x \mid \mathcal{D}_t, O_{\mathcal{A}}) = \int p(y_x, e_{\mathcal{A}} \mid \mathcal{D}_t, O_{\mathcal{A}})\, \mathrm{d}e_{\mathcal{A}}$$

$$= \int p(y_x \mid \mathcal{D}_t, e_{\mathcal{A}}, O_{\mathcal{A}})\, p(e_{\mathcal{A}} \mid O_{\mathcal{A}}, \mathcal{D}_t)\, \mathrm{d}e_{\mathcal{A}}$$

$$= \mathbb{E}_{p(e_{\mathcal{A}} \mid O_{\mathcal{A}}, \mathcal{D}_t)}\left[p\left(y_x \mid \mathcal{D}_t, e_{\mathcal{A}}\right)\right].$$

Gaussian

Back to the: EIG on algorithm output

We can similarity write:

$$\mathrm{EIG}_t(x) = H[p(O_{\mathcal{A}} \mid \mathcal{D}_t)] - \mathbb{E}_{p(y_x \mid \mathcal{D}_t)}[H[p(O_{\mathcal{A}} \mid \mathcal{D}_t \cup \{(x, y_x)\})]]$$

$$= \mathrm{H}\left[y_x \mid \mathcal{D}_t\right] - \mathbb{E}_{p(O_{\mathcal{A}} \mid \mathcal{D}_t)}\left[\mathrm{H}\left[y_x \mid \mathcal{D}_t, O_{\mathcal{A}}\right]\right]$$

$$= \mathrm{H}\left[y_x \mid \mathcal{D}_t\right] - \mathbb{E}_{p(O_{\mathcal{A}} \mid \mathcal{D}_t)}\left[\mathrm{H}\left[\mathbb{E}_{p(e_{\mathcal{A}} \mid O_{\mathcal{A}}, \mathcal{D}_t)}\left[p\left(y_x \mid \mathcal{D}_t, e_{\mathcal{A}}\right)\right]\right]\right]$$