Going Beyond Global Optima with Bayesian Algorithm Execution

Willie Neiswanger
Going Beyond Global Optima with Bayesian Algorithm Execution

Willie Neiswanger
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 on \textit{Black-box Global Optimization}

Suppose we have a noisy \textit{"black-box"} function $f$.

Assume:

- Observations are \textbf{noisy}: $y \sim f(x) + \epsilon$
- Each function query is \textbf{costly} - E.g. in money, time, labor, etc.
- \textbf{Goal}: estimate the location of \textbf{global optima} of $f$
- \textbf{Budget} of $T$ queries
Black-box Global Optimization — many applications

**Hyperparameter Opt & Neural Architecture Search**

**Systems Auto-tuning**

**Optimizing Laboratory Equipment & Machines**

**Materials Discovery & Protocols**

---


(4) Attia et al., "Closed-loop optimization of fast-charging protocols for batteries with machine learning", Nature, 2020


(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., Poczos, B., & Xing, E. P. "Neural architecture search with Bayesian optimization and optimal transport". NeurIPS 2018.

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.
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)
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
- 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
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
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
- 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)

- High throughput materials design/discovery

Applications
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**

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

(Frazier, 2012)
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

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

(Krause et al., 2008)
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”?
Note that, given a function property of interest...

Often exists effective algorithms for computing (or numerically approximating) the property, *if you ignore budget constraint*
Note that, given a function property of interest...

Often exists effective algorithms for computing (or numerically approximating) the property, \textit{if you ignore budget constraint}

\textbf{Property:} local optima (close to some initial point) 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:** local optima (close to some initial point) of $f$

**Algorithm:** local optimization algorithm

![Evolution Strategy](image)

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

$\Rightarrow$ initialize at some location, then run local minimizer. Return final query as output.
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).
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.
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!
**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.)
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.
BAYESIAN ALGORITHM EXECUTION

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.
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.

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 $D_1$, prior distribution $p(f)$

1. for $t = 1, \ldots, T$ do
2. \hspace{1cm} $x_t \leftarrow \arg \max_{x \in \mathcal{X}} \alpha_t(x)$

Optimize an *acquisition function*.
- aims to capture value of querying $f$ at an $x$.
- defined using our probabilistic model.
⇒ Chooses next $x$ to query.
Information-based Bayesian Optimization

Algorithm 1 Bayesian Optimization

Input: dataset $D_1$, prior distribution $p(f)$

1: for $t = 1, \ldots, T$ do
2: \hspace{1em} $x_t \leftarrow \arg\max_{x \in \mathcal{X}} \alpha_t(x)$
3: \hspace{1em} $y_t \sim f(x_t) + \epsilon$

Query $f$ on chosen $x$, observe $y$
Information-based Bayesian Optimization

Algorithm 1 Bayesian Optimization

**Input:** dataset $D_1$, prior distribution $p(f)$

1: for $t = 1, \ldots, T$ do
2: \hspace{1em} $x_t \leftarrow \arg \max_{x \in X} \alpha_t(x)$
3: \hspace{1em} $y_{x_t} \sim f(x_t) + \epsilon$
4: \hspace{1em} $D_{t+1} \leftarrow D_t \cup \{(x_t, y_{x_t})\}$

**Update dataset with new $(x, y)$ pair**

**Output:** final dataset $D_{T+1}$
Information-based Bayesian Optimization

Visualizing this ...
Unknown black-box $f$, and dataset of $(x, y)$ pairs.
Can use probabilistic model to infer $f$, given dataset.
Define acquisition function using probabilistic model.
Optimize acquisition function $\Rightarrow$ 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**

<table>
<thead>
<tr>
<th><strong>Input:</strong> dataset $D_1$, prior distribution $p(f)$, algorithm $A$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: for $t = 1, \ldots, T$ do</td>
</tr>
<tr>
<td>2: $x_t \leftarrow \arg\max_{x \in X} \alpha_t(x)$</td>
</tr>
<tr>
<td>3: $y_{x_t} \sim f(x_t) + \epsilon$</td>
</tr>
<tr>
<td>4: $D_{t+1} \leftarrow D_t \cup {(x_t, y_{x_t})}$</td>
</tr>
</tbody>
</table>

**Output:** final dataset $D_{T+1}$
Acquisition function – info-based BO

Samples from posterior distribution over functions

Probabilistic model of $f$, given dataset.
Acquisition function – info-based BO

Samples from posterior distribution over functions

Consider the global optima of each sample...

Probabilistic model of $f$, given dataset.
Acquisition function — info-based BO

Consider the global optima of each sample...

Probabilistic model of $f$, given dataset.
There is a posterior distribution over global optima induced by probabilistic model: \( p(x^* \mid D_t) \)

Samples from posterior distribution over functions

Consider the global optima of each sample...

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:

\[ \alpha_t(x) = H[p(x^* | D_t)] - \mathbb{E}_{p(y_x | D_t)}[H[p(x^* | D_t \cup \{(x, y_x)\})]] \]

- *entropy of posterior distribution over global optima* \( x^* \)
- *expected entropy of posterior distribution over global optima* \( x^* \),
  *if we were to make a query at* \( x \)
- *minus*
Acquisition function — info-based BO

This leads us to the acquisition function:

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

- entropy of posterior distribution over global optima \( x^* \)
- 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:
**INFO-BASED BO**

**Acquisition function** — info-based BO

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

1) **Before acquisition optimization:**
   - Generate posterior samples of global optima.
   - (⇒ 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 only once. Then cheaper during iterative acquisition opt.
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.)
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): $O_A$
- We care about posterior over output: $p(O_A \mid D_t)$
- And its entropy: $H[p(O_A \mid D_t)]$

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:

\[
\alpha_t(x) = H[p(O_A | D_t)] - \mathbb{E}_{p(y|x|D_t)}[H[p(O_A | D_t \cup \{(x, y_x)\})]]
\]

"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:

\[
\alpha_t(x) = H[p(O_A | D_t)] - \mathbb{E}_{p(y_x | D_t)}[H[p(O_A | D_t \cup \{(x, y_x)\})]]
\]

"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 \(f\).
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 $f$. 

*Execution path = pink dots*
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 \(f\) ⇒ 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

Consider one function sample
InfoBAX acquisition function

Consider one function sample

Can run algorithm $A$ on function sample...
InfoBAX acquisition function

Consider one function sample

Execution path of algorithm $A$ on sample.
InfoBAX acquisition function

Consider one function sample

Execution path of algorithm A on sample.

Can do this on many samples.
InfoBAX acquisition function

Consider one function sample

Execution path of algorithm $A$ on sample.

Can do this on many samples.

There is a posterior distribution over execution paths induced by probabilistic model.
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:**

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 $\alpha_t(x)$ 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 Optimization

**Algorithm 1 INFOBAX**

**Input:** dataset $D_1$, prior distribution $p(f)$, algorithm $A$

1. for $t = 1, \ldots, T$ do
2.   $x_t \leftarrow \arg \max_{x \in X} \alpha_t(x, A)$
3.   $y_{x_t} \sim f(x_t) + \epsilon$
4.   $D_{t+1} \leftarrow D_t \cup \{(x_t, y_{x_t})\}$

**Output:** final dataset $D_{T+1}$

(1) Run algorithm on posterior function samples.

(2) Optimize $\alpha_t(x)$ using resulting execution paths.
InfoBAX — one-slide summary of the full story
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 we have a black-box function $f$ and a property of interest.

![Graph showing inflection point](image-url)
Suppose we have a black-box function \( f \) and a property of interest.
Suppose we have a black-box function $f$ and a property of interest.
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).
INFO-BASED BO

*InfoBAX — one-slide summary of the full story*

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

![Diagram of a function with local minima]

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

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

![Diagram of a search process from start to end]

⇒ **Goal:** estimate the property (i.e. output of $A$) with minimal function queries
**INFO-BASED BO**

InfoBAX — one-slide summary of the full story

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

![Graph showing a function $f$ with local minima](image)

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

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

![Graph showing a sequence from start to end](image)

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

1: for $t = 1, \ldots, T$ do
2: \[ x_t \leftarrow \arg \max_{x \in \mathcal{X}} \alpha_t(x, A) \]
3: \[ y_{x_t} \sim f(x_t) + \epsilon \]
4: \[ \mathcal{D}_{t+1} \leftarrow \mathcal{D}_t \cup \{(x_t, y_{x_t})\} \]
**InfoBAX — one-slide summary of the full story**

Suppose we have a black-box function \( f \) and a *property of interest*.

\[
\begin{align*}
    f & \quad \text{local minima}
\end{align*}
\]

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

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

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

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

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

\[
\begin{align*}
    \text{start} & \quad \text{end}
\end{align*}
\]
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)

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

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

1: for $t = 1, \ldots, T$ do
2: $x_t \leftarrow \arg \max_{x \in X} \alpha_t(x, A)$
3: $y_{x_t} \sim f(x_t) + \epsilon$
4: $D_{t+1} \leftarrow D_t \cup \{(x_t, y_{x_t})\}$

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

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

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

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

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

\[ \Rightarrow \text{Output: posterior estimate of property (i.e. output of } A) \]
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
- Bayesian local optimization
- Estimating top-$k$ optima
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.
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
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)
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 a 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?

Many uninformative 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 (i.e., 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 →
APPLICATIONS of BAX — *estimating shortest paths in graphs*

*InfoBAX in action →*

After 100 queries.
APPLICATIONS of BAX — *estimating shortest paths in graphs*

Comparison after **100 queries**:

**Random Search**

**Dijkstra’s**

**InfoBAX**
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 a BAX 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 — 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
APPLICATIONS of BAX — *top-k estimation*

250 items

Black-box function $f$ (green).
APPLICATIONS of BAX — *top-k estimation*

Black-box function $f$ (green).

250 items

Stars denote top-10 items
APPLICATIONS of BAX — *top-k estimation*

**Algorithm:** scan through each item and query; return top-k
APPLICATIONS of BAX — top-k estimation

Scan and Query Algorithm (full)

Algorithm yields perfect estimate
APPLICATIONS of BAX — *top-k estimation*

Scan and Query Algorithm (full)

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
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)

Estimate is pretty bad

10 queries (black dots)
APPLICATIONS of BAX — *top-k estimation*

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

How about 100 uniform random queries?

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

Estimate is still not great

Posterior samples of top-10 items (blue squares)

100 queries (black dots)
APPLICATIONS of BAX — *top-k estimation*

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

Estimate is still not great

Try *InfoBAX* with “scan and query” algorithm

Posterior samples of top-10 items (blue squares)

100 queries (black dots)
APPLICATIONS of BAX — top-k estimation

InfoBAX

Good estimate in <100 queries

InfoBAX explores the space but samples densely around high-value items.
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 $x$”
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.
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.
How can we empirically assess predictive uncertainty?
Three important criteria are...
How can we empirically assess predictive uncertainty? Three important criteria are...

Accuracy:

“How good is mean prediction? (agnostic to 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 \((1-\alpha)\)-interval (e.g. 95% interval) of the predictive distribution.

**Well-calibrated** \(\Rightarrow\) “the \((1-\alpha)\)-interval covers the true value \((1-\alpha)\)-proportion of the time, for all \(\alpha\)”
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 \(\Rightarrow\) “the \((1-\alpha)\)-interval covers the true value \((1-\alpha)\)-proportion of the time, for all \(\alpha\)”
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** $\Rightarrow$ “the \((1-\alpha)\)-interval covers the true value \((1-\alpha)\)-proportion of the time, for all $\alpha$”

Can scan from $\alpha=0$ to $\alpha=1$, and compute:

1. expected fraction of true values contained in interval
2. observed fraction of true values contained in interval
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](https://github.com/uncertainty-toolbox/uncertainty-toolbox)

**Glossary**

**Metrics / Viz**

**Recalibration**

**Relevant Papers**
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*


Uncertainty Toolbox: [github.com/uncertainty-toolbox/uncertainty-toolbox](https://github.com/uncertainty-toolbox/uncertainty-toolbox)

*Thanks for listening!*