Bayesian Optimization

fundamentals, implementations, and practice

Quan Nguyen

Who am I?

- **Quan Nguyen**
- Ph.D. student – Bayesian machine learning, decision-making under uncertainty

krisnguyen135.github.io

github.com/KrisNguyen135/Talks

@the_subtrahend

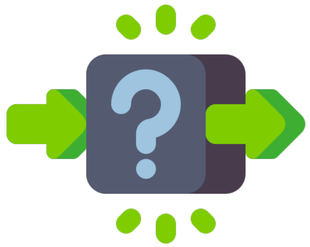
What you will learn today

Bayesian optimization (BayesOpt) essentials

What you will learn today

Bayesian optimization (BayesOpt) essentials

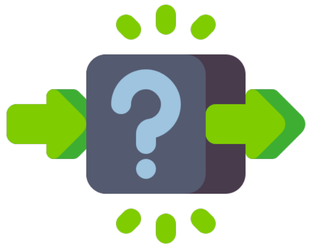
expensive black-box optimization



What you will learn today

Bayesian optimization (BayesOpt) essentials

expensive black-box optimization

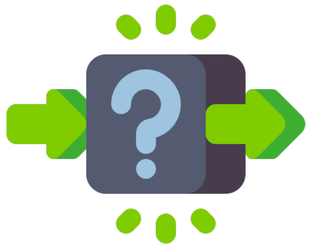


high-level overview of BayesOpt

What you will learn today

Bayesian optimization (BayesOpt) essentials

expensive black-box optimization



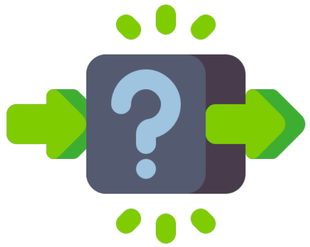
high-level overview of BayesOpt

implementing BayesOpt

What you will learn today

Bayesian optimization (BayesOpt) essentials

expensive black-box optimization



high-level overview of BayesOpt

implementing BayesOpt

BayesOpt in practice

Black-box optimization is everywhere

Black-box optimization is everywhere

Black-box optimization

Black-box optimization is everywhere

Black-box optimization

- Want to optimize an objective function $f(x)$

Black-box optimization is everywhere

Black-box optimization

- Want to optimize an objective function $f(x)$
- Can’t look inside, can only observe $y = f(x)$ at selected x

Black-box optimization is everywhere

Black-box optimization

- Want to optimize an objective function $f(x)$
- Can't look inside, can only observe $y = f(x)$ at selected x
- Evaluating $y = f(x)$ is expensive (time, money, resources)

Black-box optimization is everywhere

Black-box optimization

- Want to optimize an objective function $f(x)$
- Can't look inside, can only observe $y = f(x)$ at selected x
- Evaluating $y = f(x)$ is expensive (time, money, resources)

Examples

Black-box optimization is everywhere

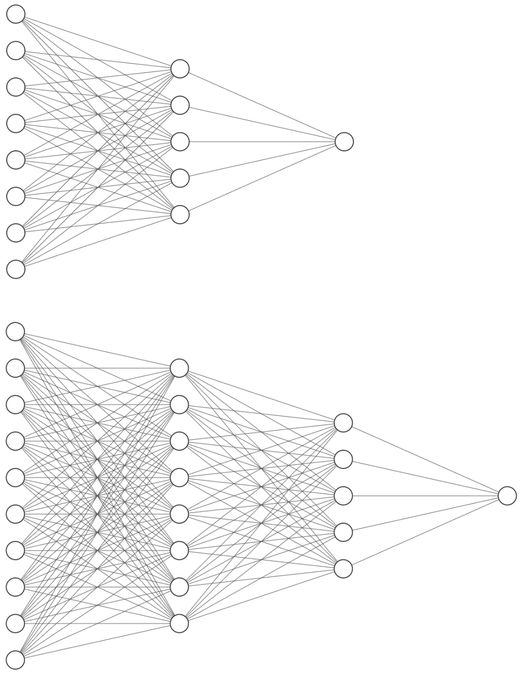
Black-box optimization

- Want to optimize an objective function $f(x)$
- Can't look inside, can only observe $y = f(x)$ at selected x
- Evaluating $y = f(x)$ is expensive (time, money, resources)

Examples

- *Hyperparameter tuning*: choose parameters for ML models

which model gives higher accuracy?



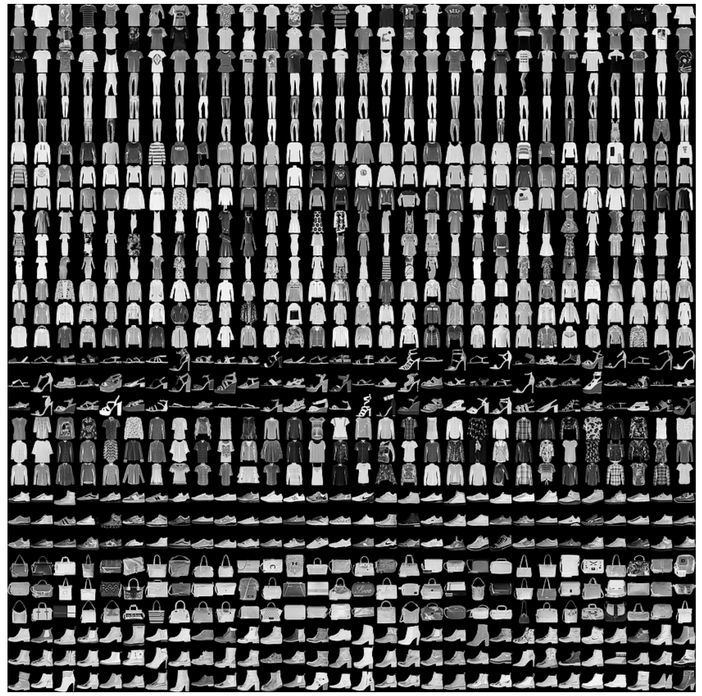
Black-box optimization is everywhere

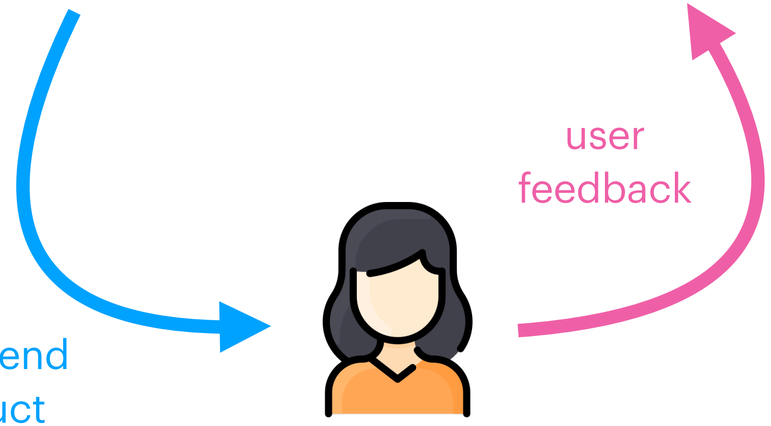
Black-box optimization

- Want to optimize an objective function $f(x)$
- Can't look inside, can only observe $y = f(x)$ at selected x
- Evaluating $y = f(x)$ is expensive (time, money, resources)

Examples

- *Hyperparameter tuning*: choose parameters for ML models
- *Product recommendation*: find products customers want





FashionMNIST https://github.com/zalandoresearch/fashion-mnist

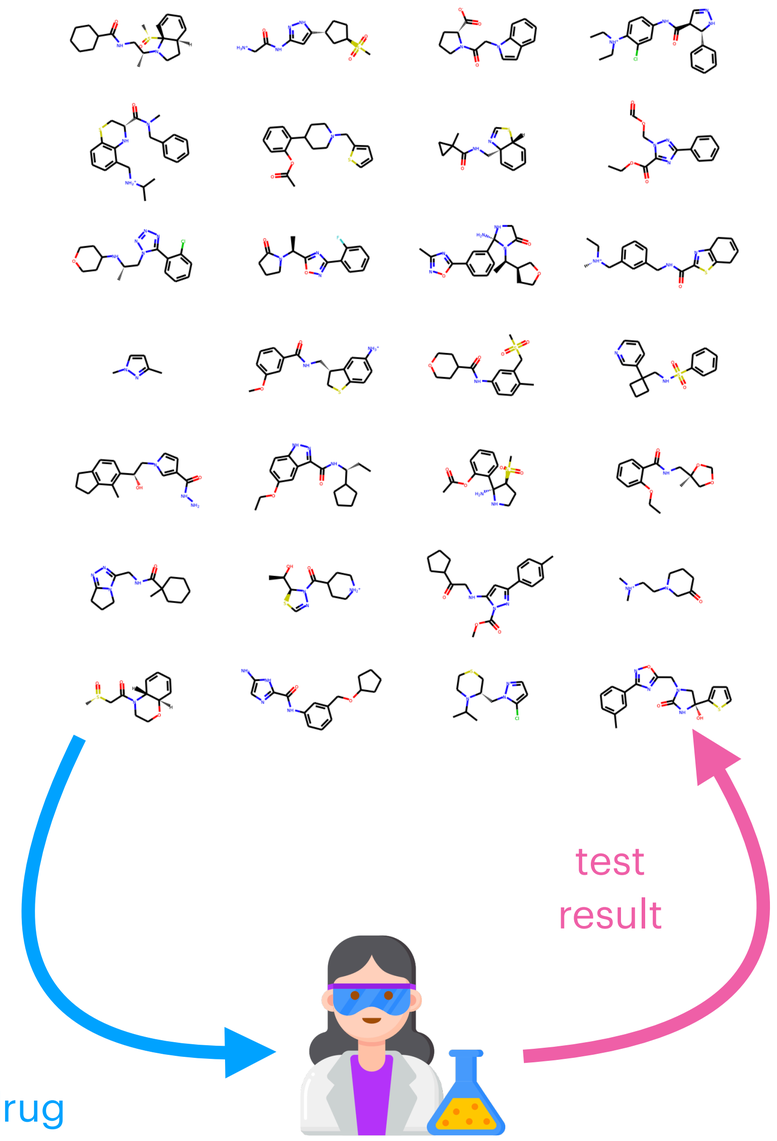
Black-box optimization is everywhere

Black-box optimization

- Want to optimize an objective function $f(x)$
- Can't look inside, can only observe $y = f(x)$ at selected x
- Evaluating $y = f(x)$ is expensive (time, money, resources)

Examples

- *Hyperparameter tuning*: choose parameters for ML models
- *Product recommendation*: find products customers want
- *Drug discovery*: discover effective drugs against diseases



Naïve strategies might waste resources

example: finding the best neural network architecture

Naïve strategies might waste resources

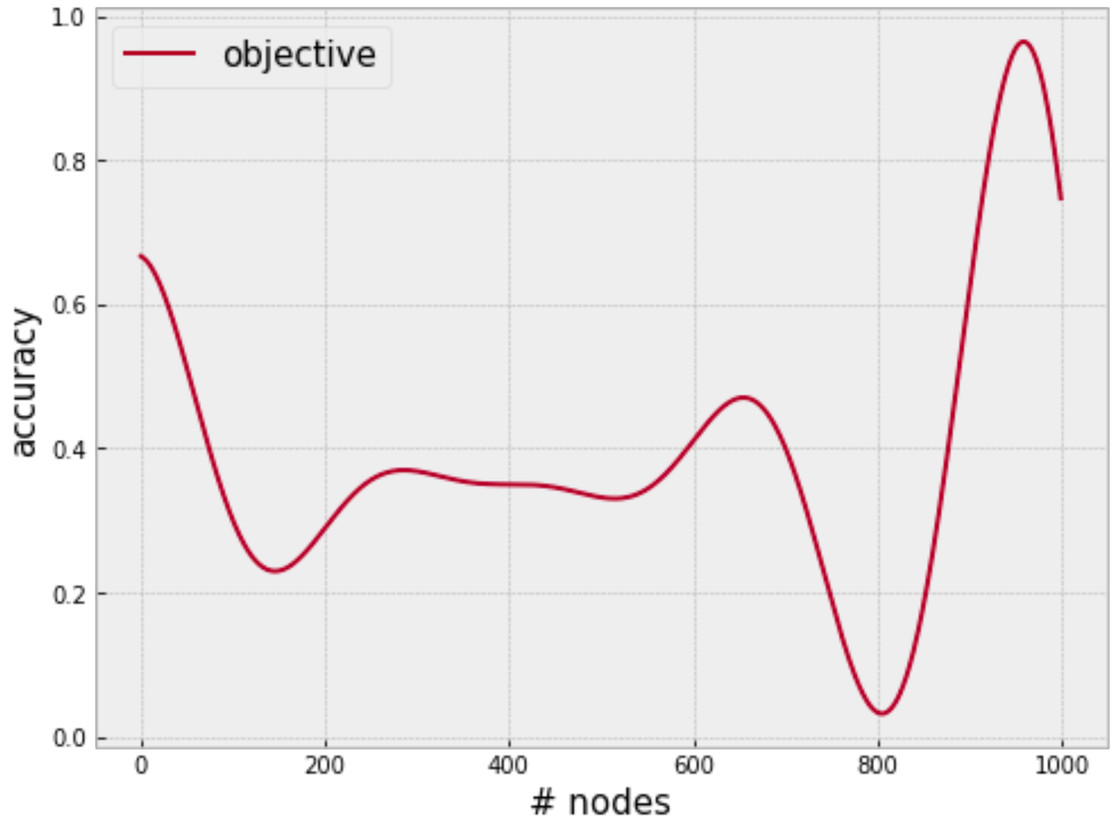
example: finding the best neural network architecture

- **Goal**: find the number of nodes the network has to *maximize accuracy*

Naïve strategies might waste resources

example: finding the best neural network architecture

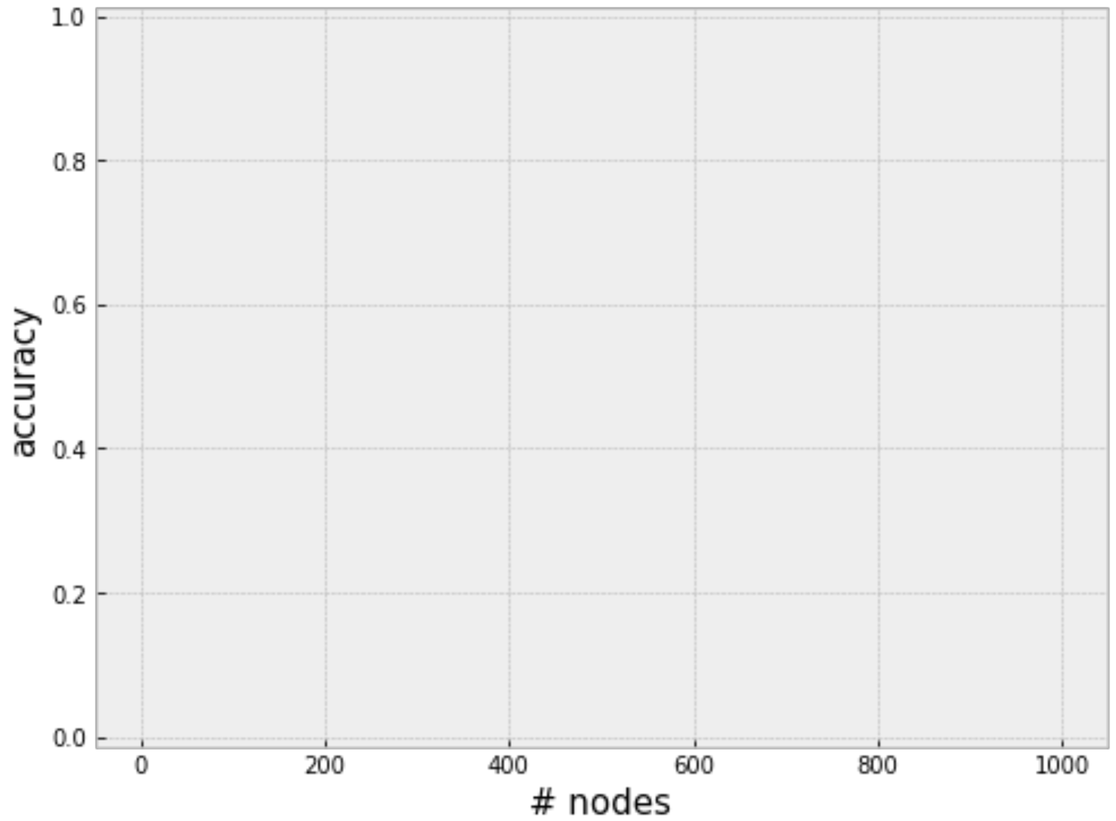
- **Goal**: find the number of nodes the network has to *maximize accuracy*



Naïve strategies might waste resources

example: finding the best neural network architecture

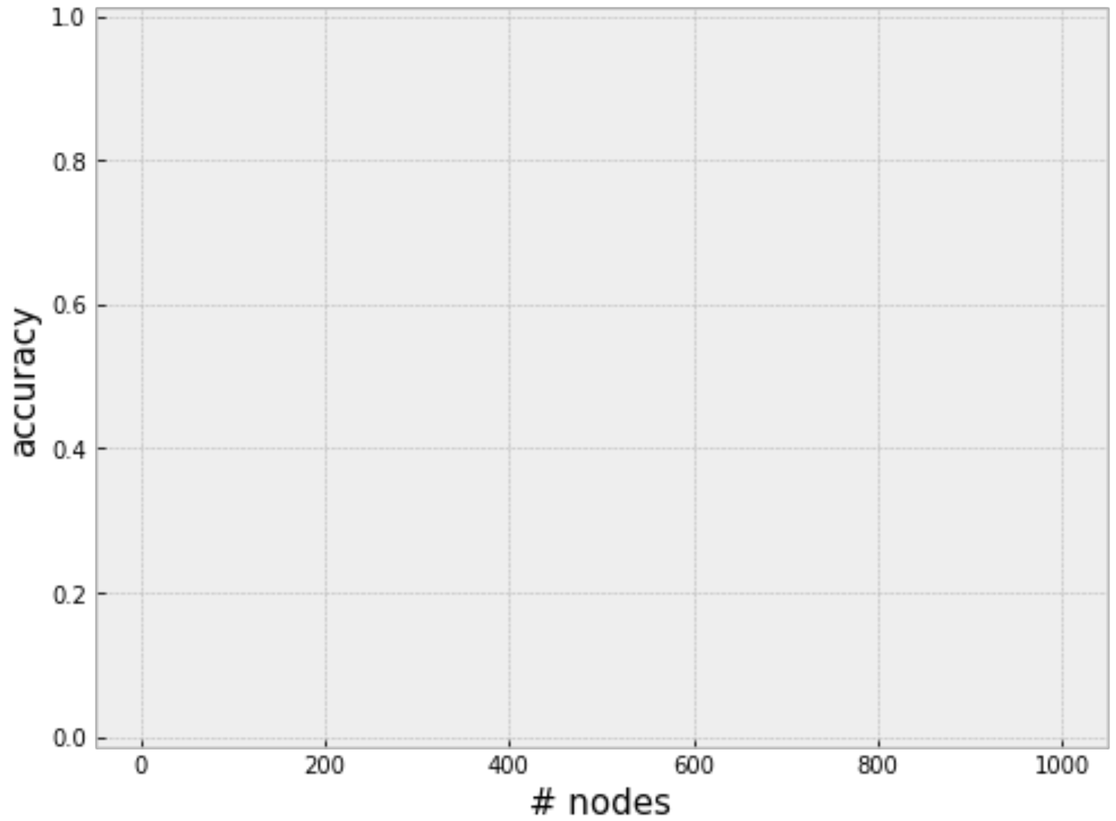
- **Goal**: find the number of nodes the network has to *maximize accuracy*
- **Challenge**: don't know how the objective function of accuracy behaves



Naïve strategies might waste resources

example: finding the best neural network architecture

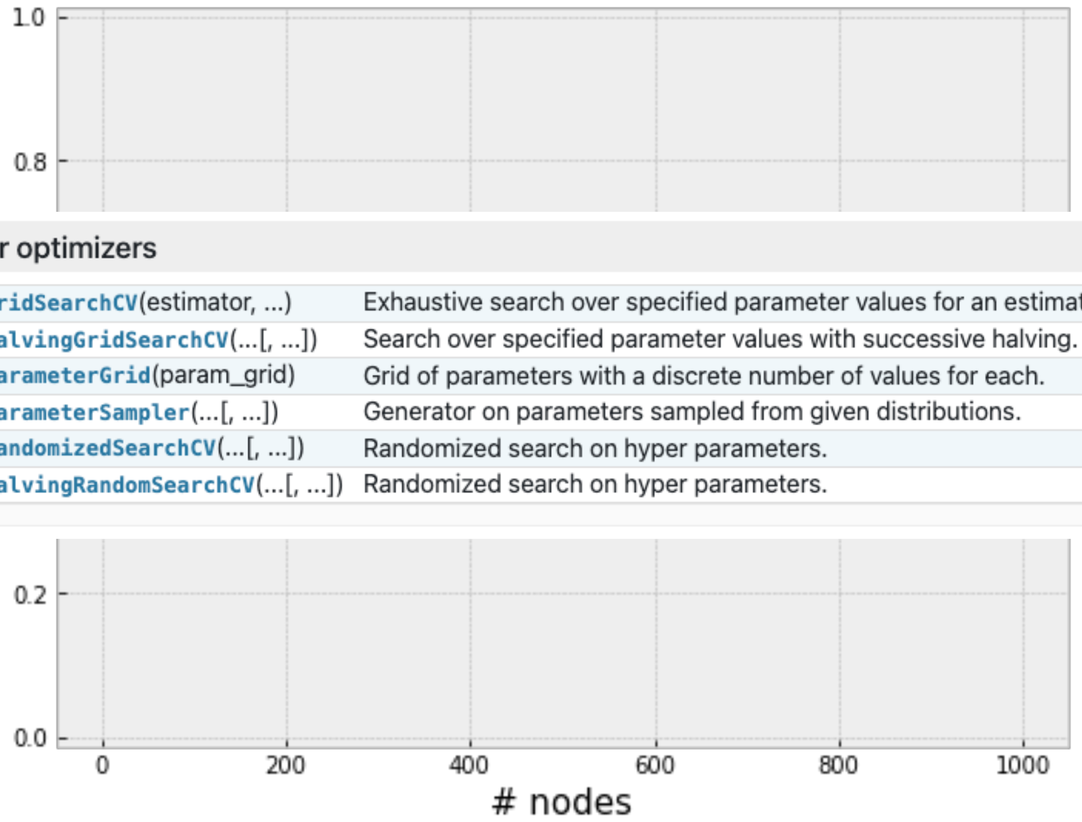
- **Goal**: find the number of nodes the network has to *maximize accuracy*
- **Challenge**: don't know how the objective function of accuracy behaves
- **Strategy**: *probe* the objective function to find the optimum



Naïve strategies might waste resources

example: finding the best neural network architecture

- **Goal**: find the number of nodes the network has to *maximize accuracy*
- **Challenge**: don't know how the objective function of accuracy behaves
- **Strategy**: *probe* the objective function to find the optimum
  - Scikit-learn to the rescue…right?



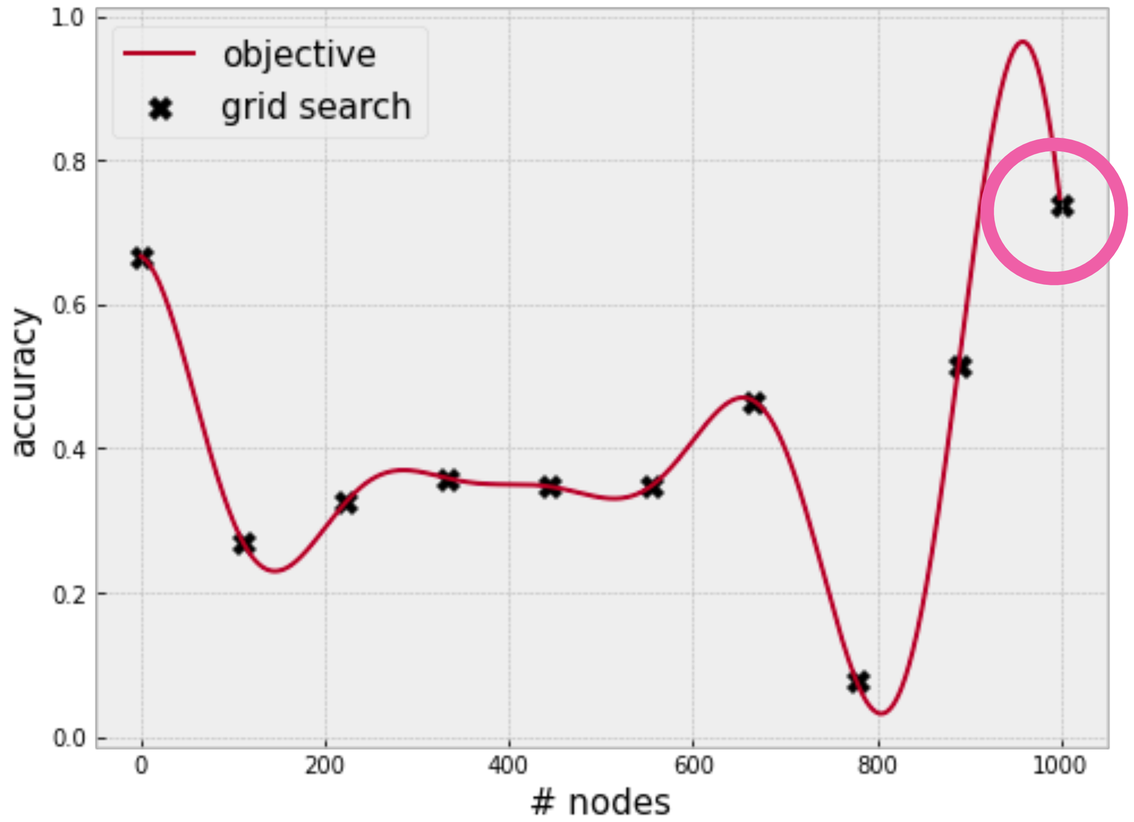
Hyper-parameter optimizers

`model_selection.GridSearchCV`(estimator, ...)	Exhaustive search over specified parameter values for an estimator.
`model_selection.HalvingGridSearchCV`(...[, ...])	Search over specified parameter values with successive halving.
`model_selection.ParameterGrid`(param_grid)	Grid of parameters with a discrete number of values for each.
`model_selection.ParameterSampler`(...[, ...])	Generator on parameters sampled from given distributions.
`model_selection.RandomizedSearchCV`(...[, ...])	Randomized search on hyper parameters.
`model_selection.HalvingRandomSearchCV`(...[, ...])	Randomized search on hyper parameters.

Naïve strategies might waste resources

example: finding the best neural network architecture

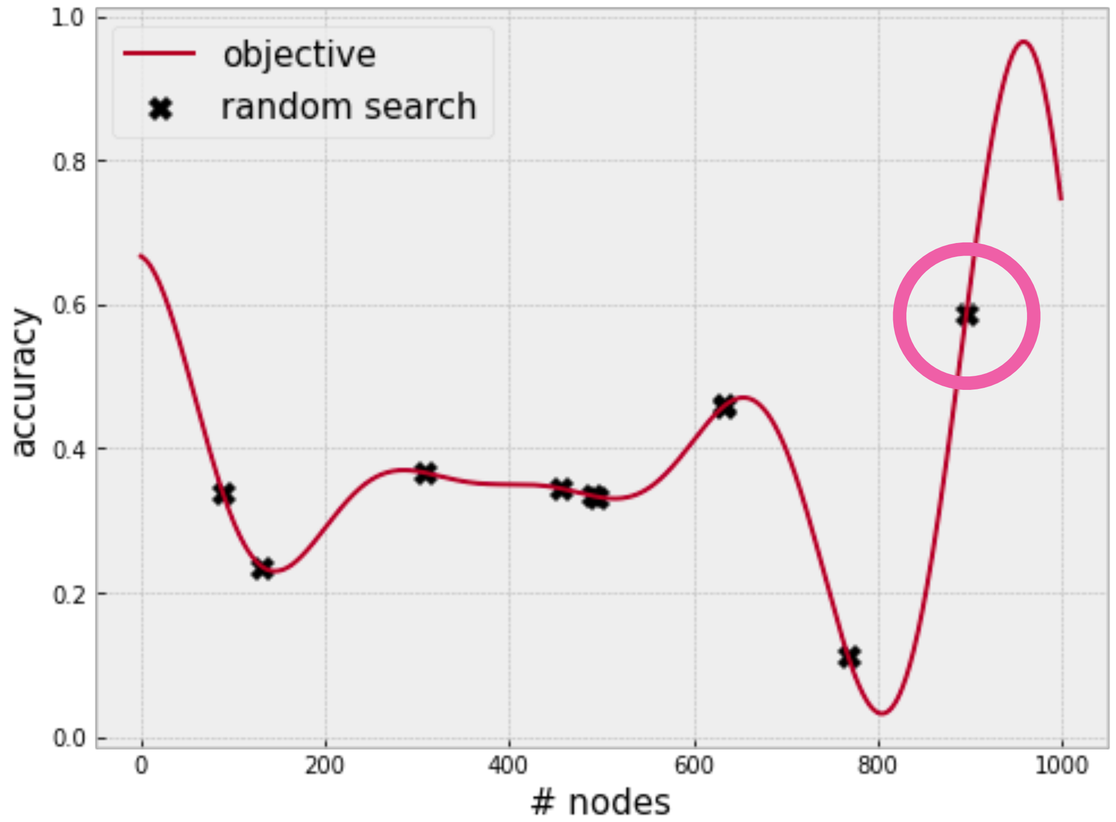
- **Goal**: find the number of nodes the network has to *maximize accuracy*
- **Challenge**: don't know how the objective function of accuracy behaves
- **Strategy**: *probe* the objective function to find the optimum
  - Scikit-learn to the rescue...right?



Naïve strategies might waste resources

example: finding the best neural network architecture

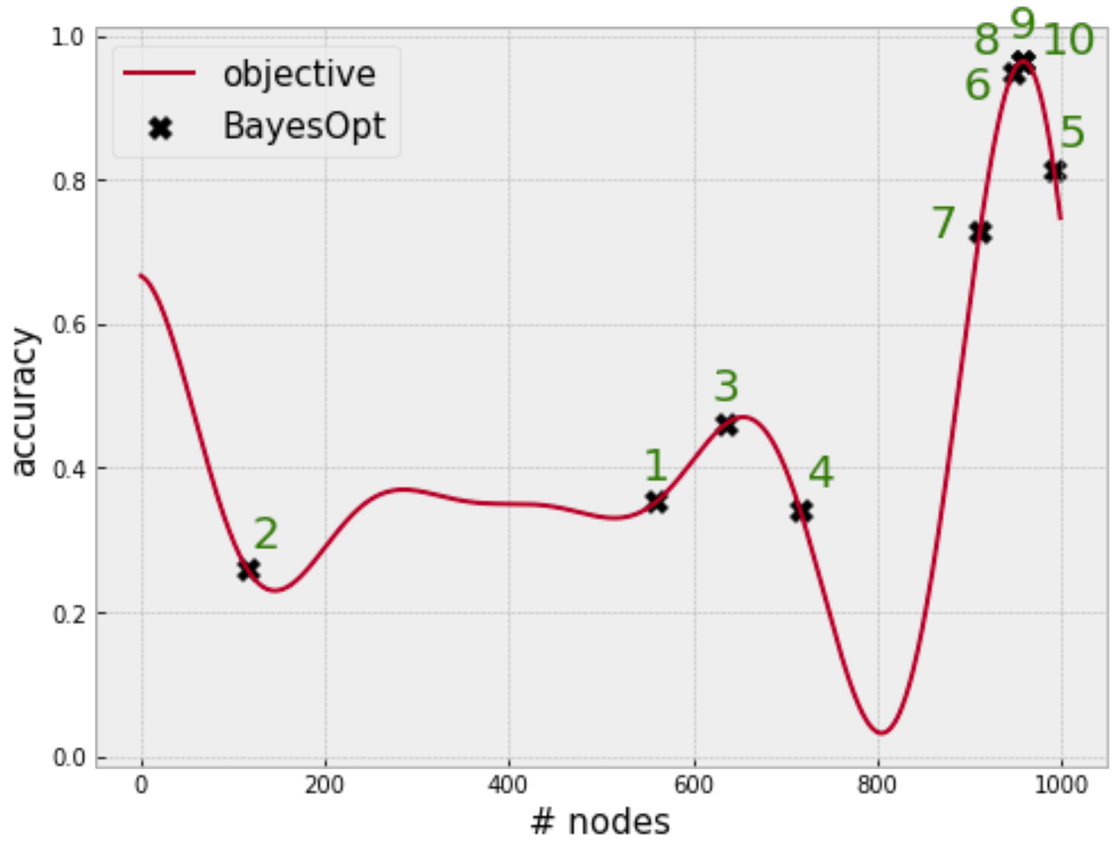
- **Goal**: find the number of nodes the network has to *maximize accuracy*
- **Challenge**: don't know how the objective function of accuracy behaves
- **Strategy**: *probe* the objective function to find the optimum
  - Scikit-learn to the rescue…right?



Naïve strategies might waste resources

example: finding the best neural network architecture

- **Goal**: find the number of nodes the network has to *maximize accuracy*
- **Challenge**: don't know how the objective function of accuracy behaves
- **Strategy**: *probe* the objective function to find the optimum
  - Scikit-learn to the rescue…right?
  - ~~Scikit-learn~~ *BayesOpt* to the rescue!



Introducing Bayesian optimization

the main components

and how they work

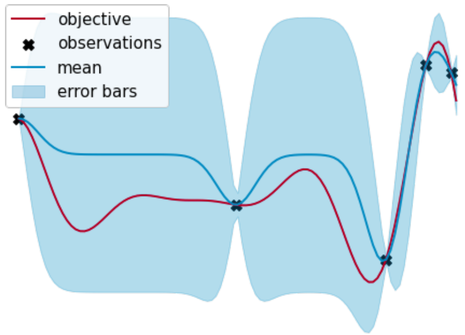
The Bayesian optimization loop

a virtuous cycle of optimization

The Bayesian optimization loop

a virtuous cycle of optimization

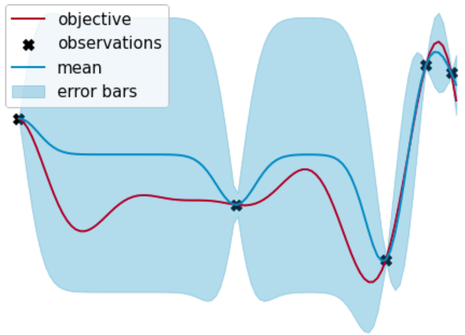
ML model (Gaussian process)
with uncertainty quantification



The Bayesian optimization loop

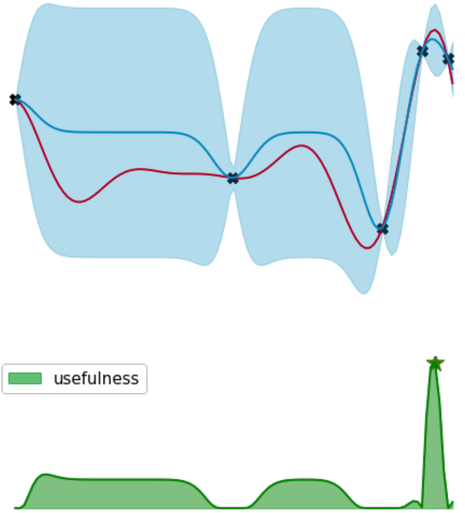
a virtuous cycle of optimization

ML model (Gaussian process)
with uncertainty quantification



inform decisions

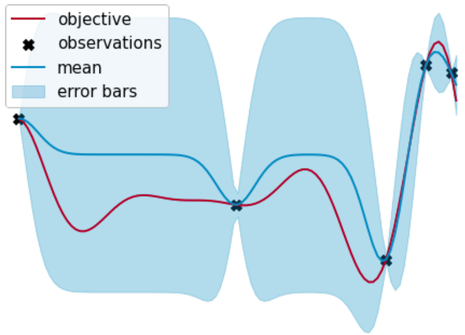
decision-making policy
finds the next point to label



The Bayesian optimization loop

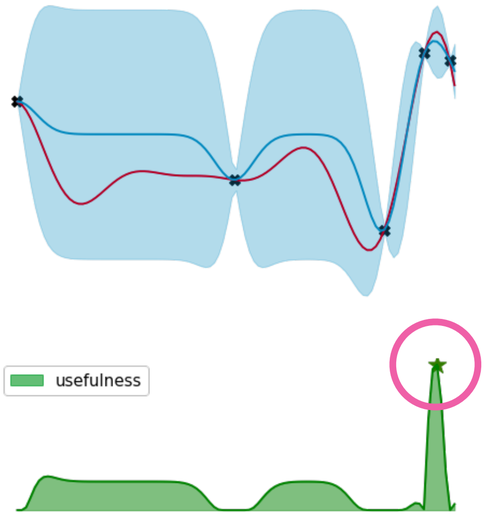
a virtuous cycle of optimization

ML model (Gaussian process)
with uncertainty quantification



inform decisions

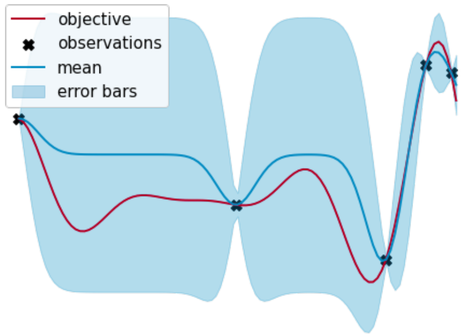
decision-making policy
finds the next point to label



The Bayesian optimization loop

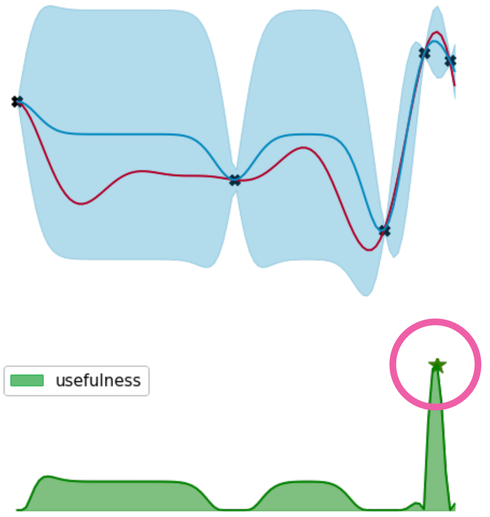
a virtuous cycle of optimization

ML model (Gaussian process)
with uncertainty quantification



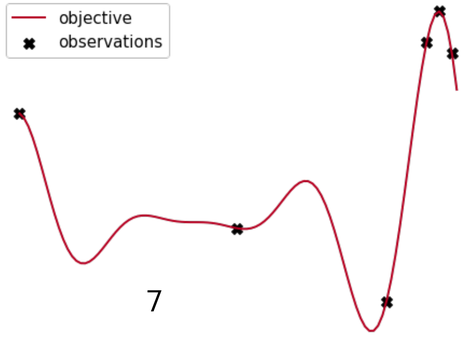
inform decisions

decision-making policy
finds the next point to label



label the
data point
suggested by
the policy

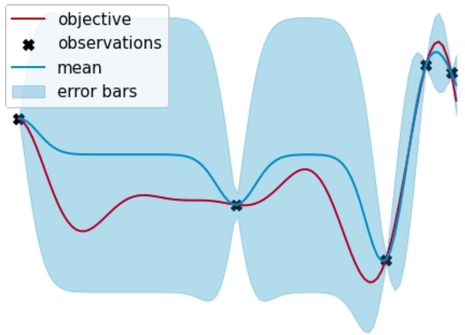
training data is updated



The Bayesian optimization loop

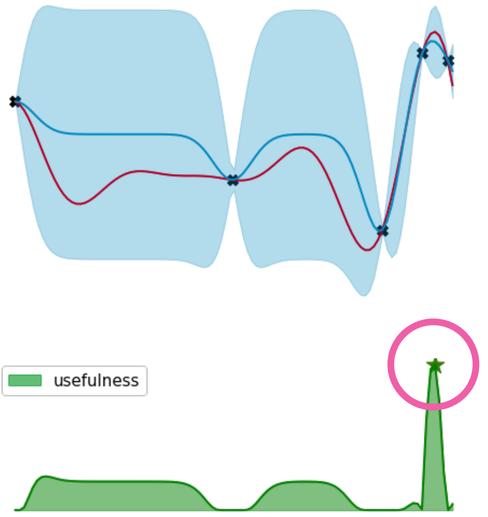
a virtuous cycle of optimization

ML model (Gaussian process)
with uncertainty quantification



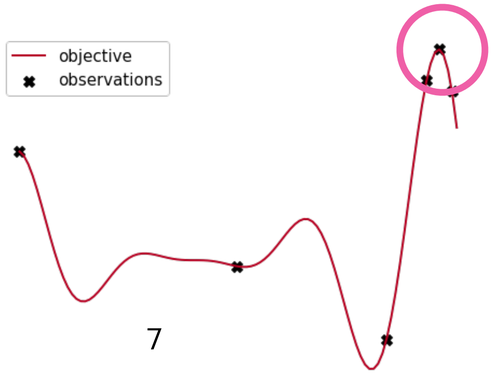
inform decisions

decision-making policy
finds the next point to label



label the
data point
suggested by
the policy

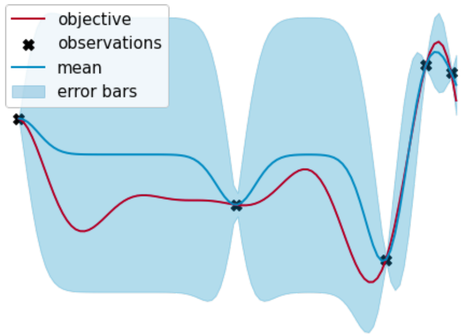
training data is updated



The Bayesian optimization loop

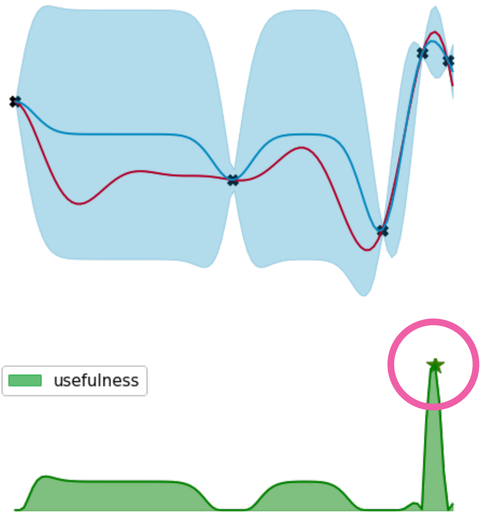
a virtuous cycle of optimization

ML model (Gaussian process)
with uncertainty quantification



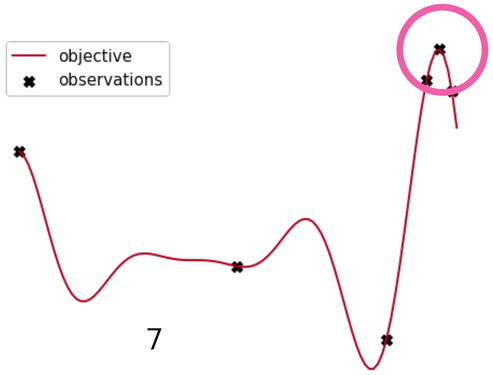
inform decisions

decision-making policy
finds the next point to label



label the
data point
suggested by
the policy

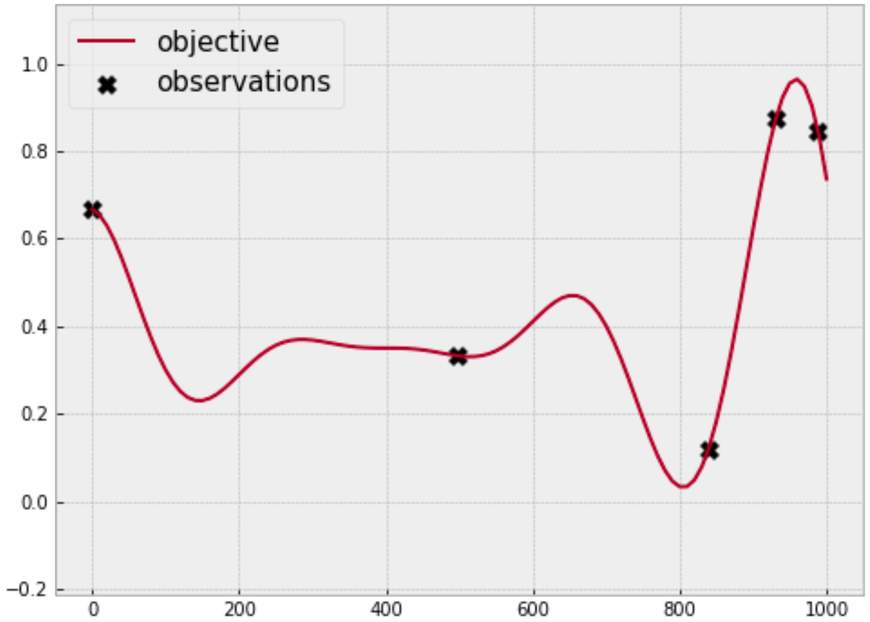
training data is updated



retrain the
ML model

Gaussian process as the predictive model

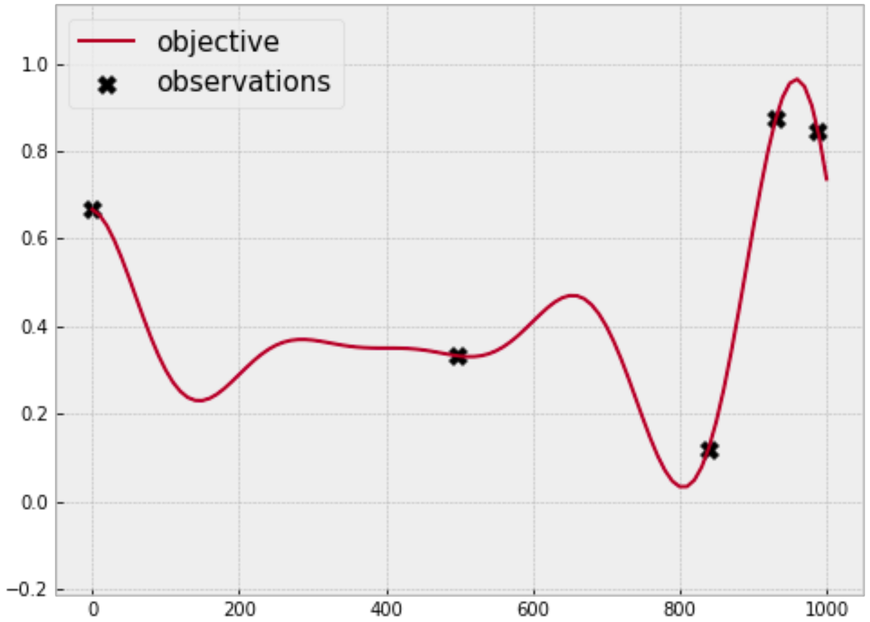
predictions with confidence levels



Gaussian process as the predictive model

predictions with confidence levels

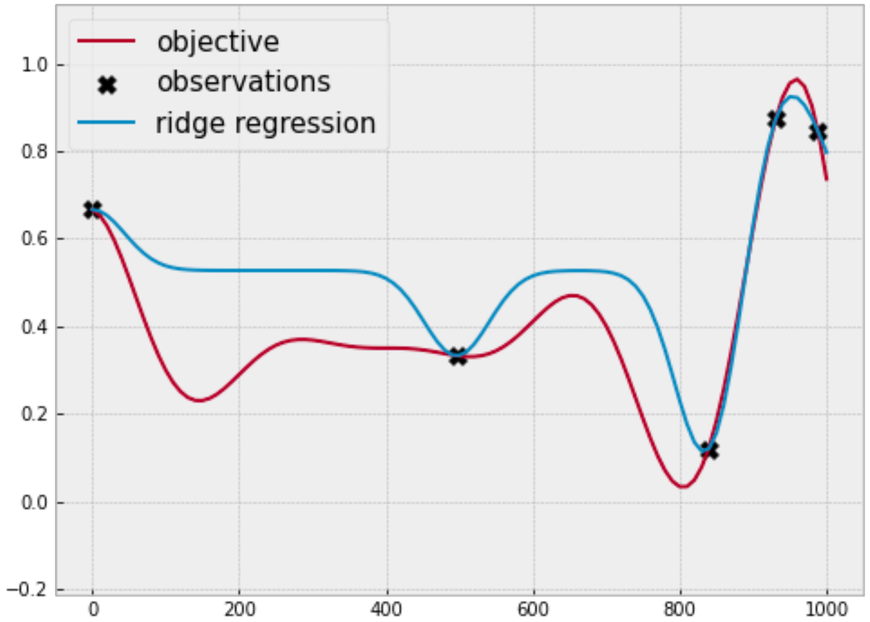
- Common ML models produce *single-valued* predictions



Gaussian process as the predictive model

predictions with confidence levels

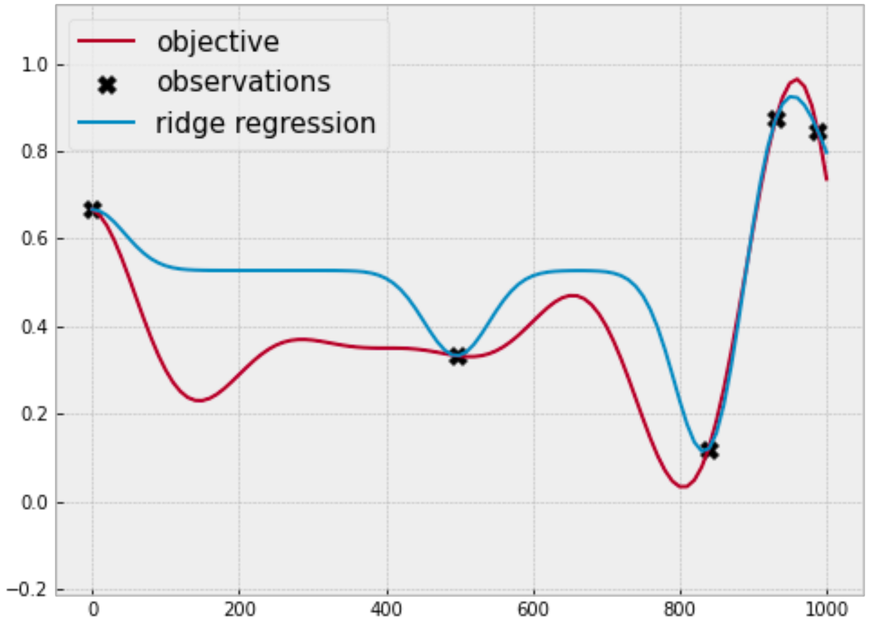
- Common ML models produce *single-valued* predictions



Gaussian process as the predictive model

predictions with confidence levels

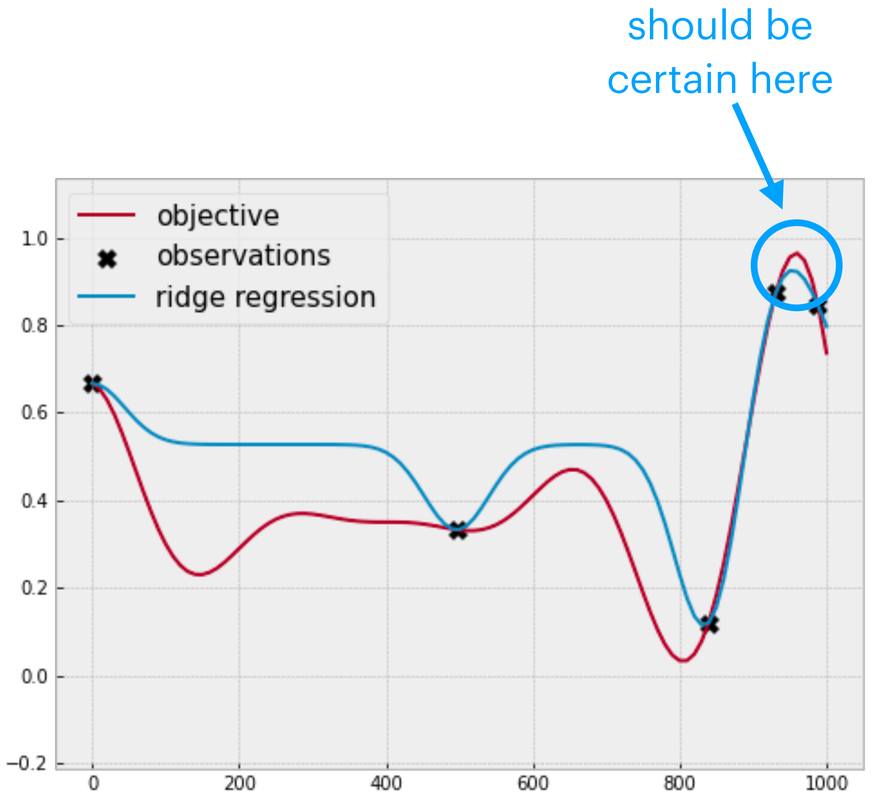
- Common ML models produce *single-valued* predictions
  - *No quantification of uncertainty* 😕



Gaussian process as the predictive model

predictions with confidence levels

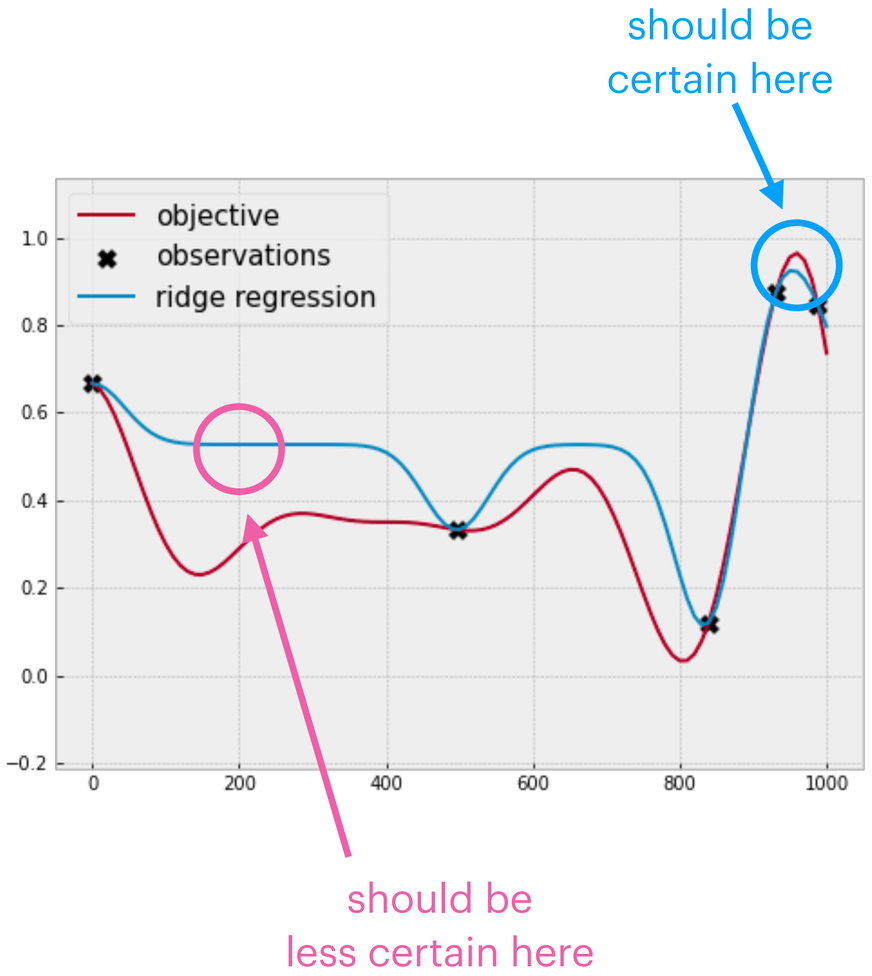
- Common ML models produce *single-valued* predictions
  - *No quantification of uncertainty* 😕



Gaussian process as the predictive model

predictions with confidence levels

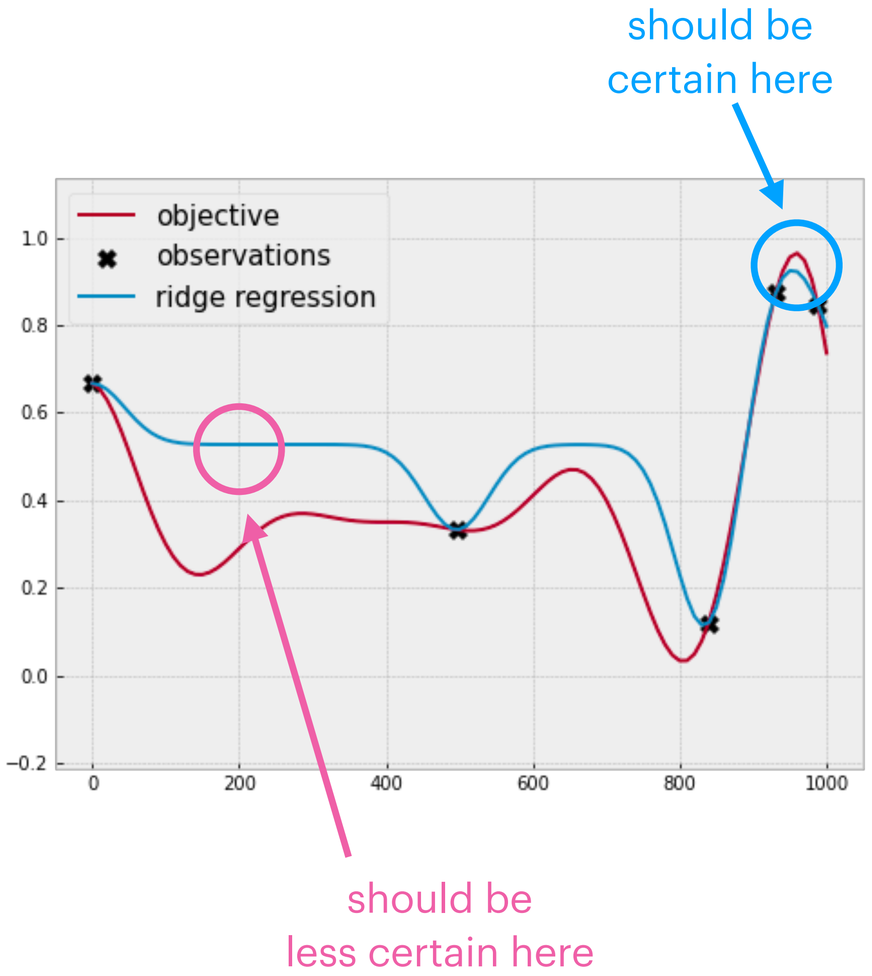
- Common ML models produce *single-valued* predictions
  - *No quantification of uncertainty* 😕



Gaussian process as the predictive model

predictions with confidence levels

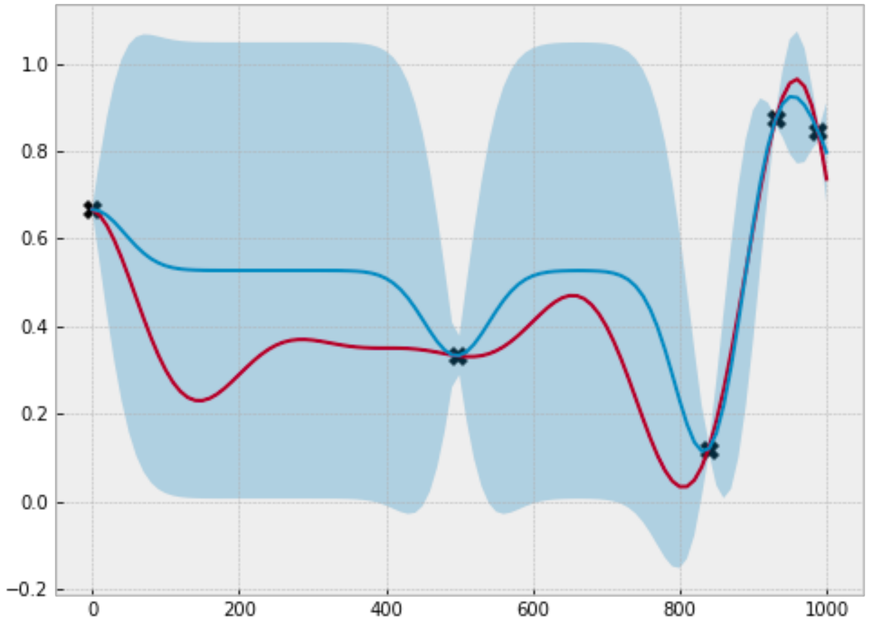
- Common ML models produce *single-valued* predictions
  - *No quantification of uncertainty* 😕
- The prediction of a Gaussian process at each point is a *normal distribution*



Gaussian process as the predictive model

predictions with confidence levels

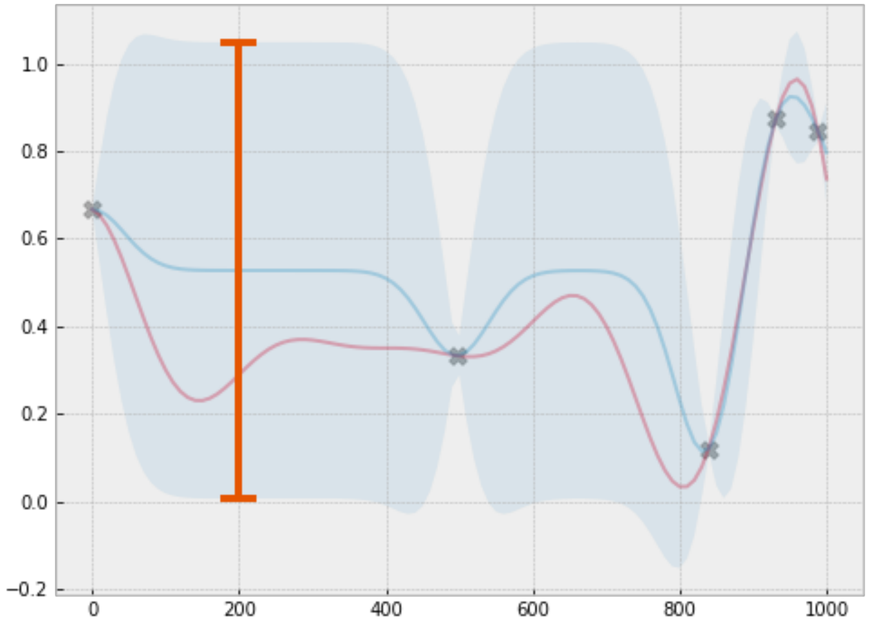
- Common ML models produce *single-valued* predictions
  - *No quantification of uncertainty* 😕
- The prediction of a Gaussian process at each point is a *normal distribution*



Gaussian process as the predictive model

predictions with confidence levels

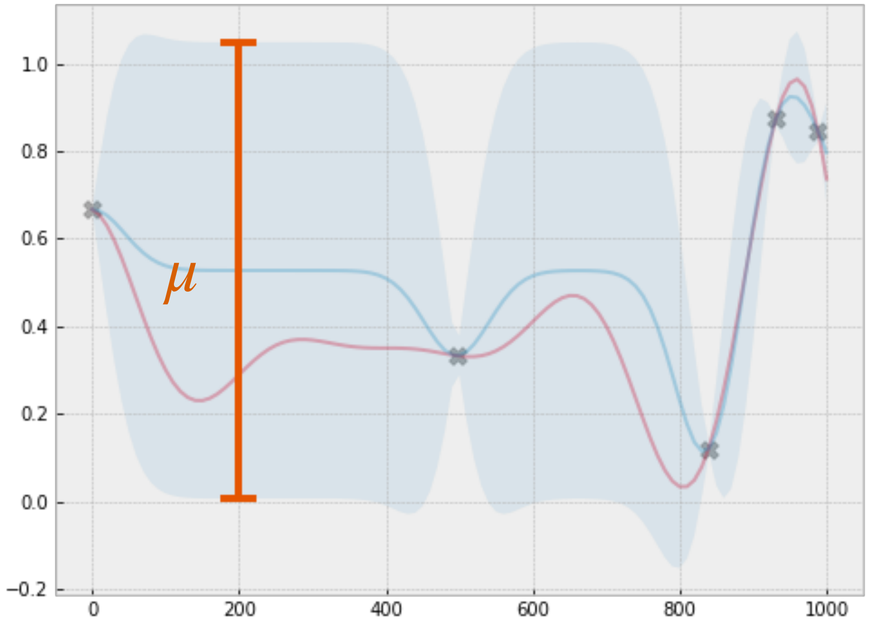
- Common ML models produce *single-valued* predictions
  - *No quantification of uncertainty* 😕
- The prediction of a Gaussian process at each point is a *normal distribution*



Gaussian process as the predictive model

predictions with confidence levels

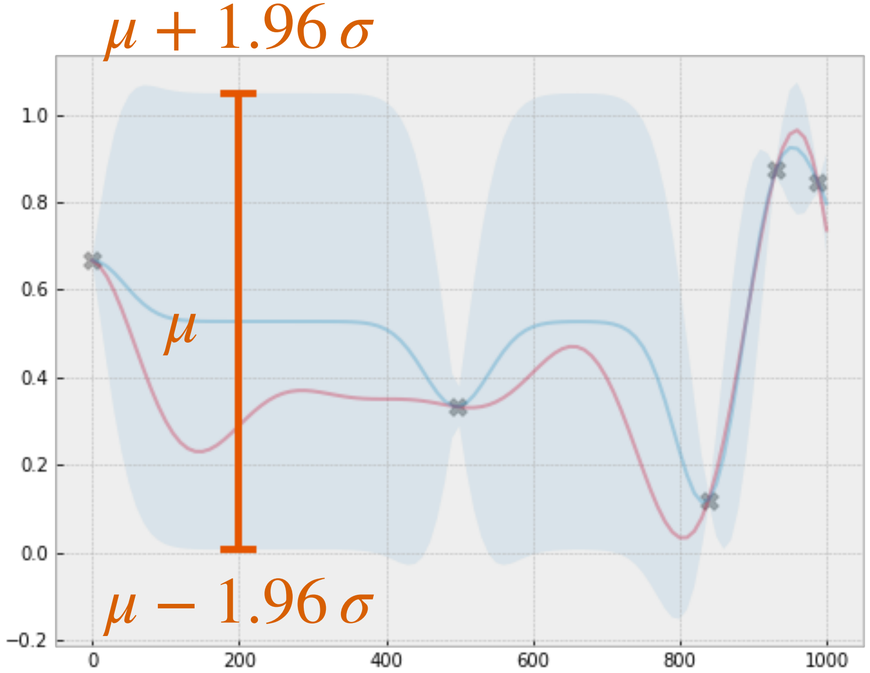
- Common ML models produce *single-valued* predictions
  - *No quantification of uncertainty* 😕
- The prediction of a Gaussian process at each point is a *normal distribution*



Gaussian process as the predictive model

predictions with confidence levels

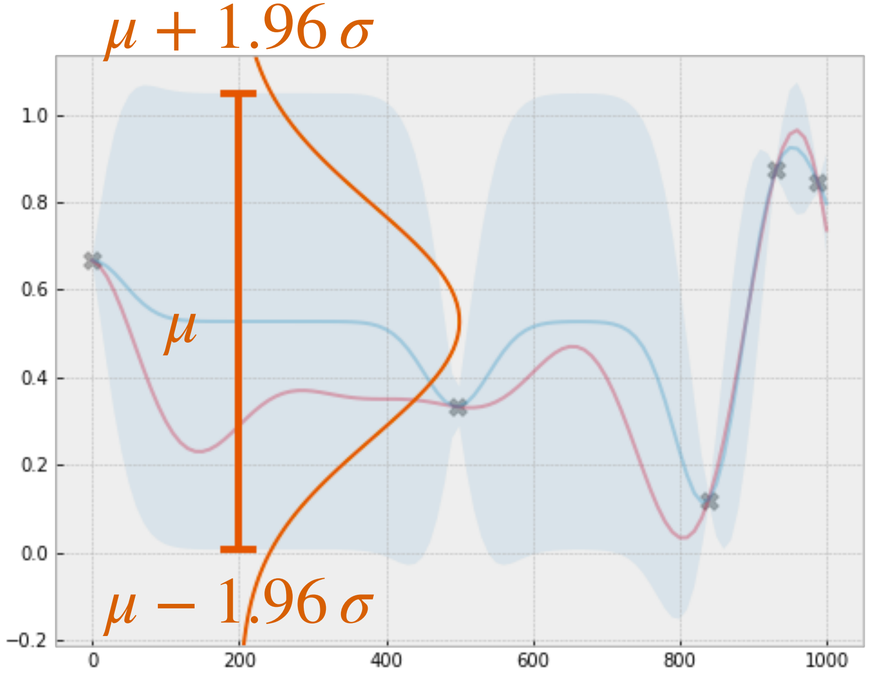
- Common ML models produce *single-valued* predictions
  - *No quantification of uncertainty* 😕
- The prediction of a Gaussian process at each point is a *normal distribution*



Gaussian process as the predictive model

predictions with confidence levels

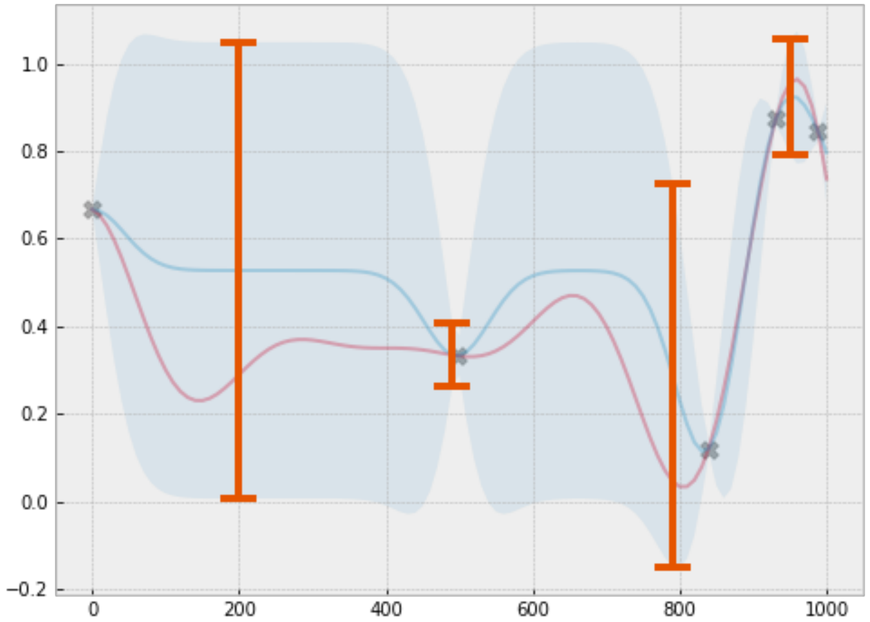
- Common ML models produce *single-valued* predictions
  - *No quantification of uncertainty* 😕
- The prediction of a Gaussian process at each point is a *normal distribution*



Gaussian process as the predictive model

predictions with confidence levels

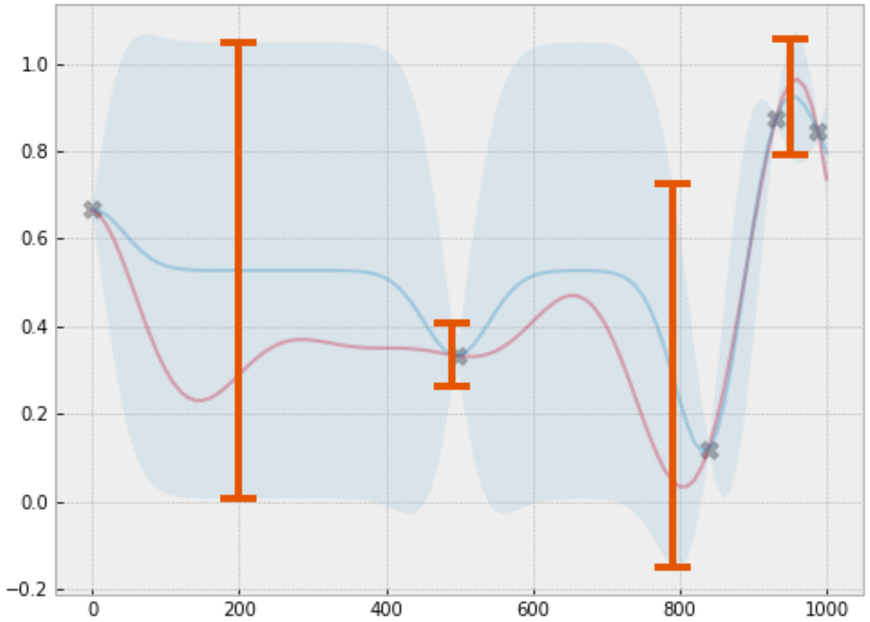
- Common ML models produce *single-valued* predictions
  - *No quantification of uncertainty* 😕
- The prediction of a Gaussian process at each point is a *normal distribution*



Gaussian process as the predictive model

predictions with confidence levels

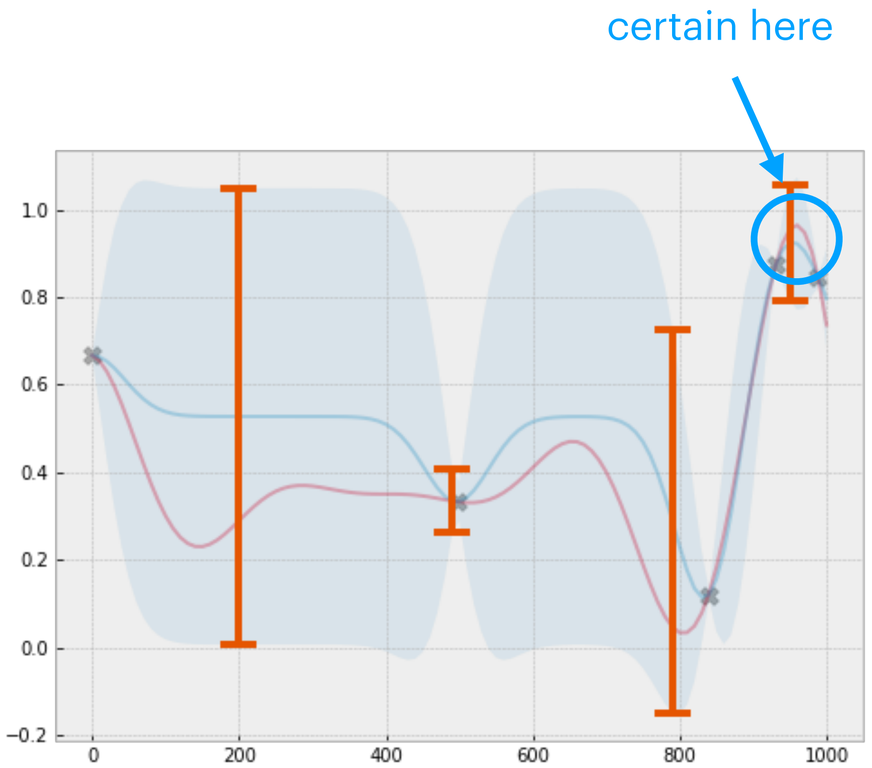
- Common ML models produce *single-valued* predictions
  - *No quantification of uncertainty* 😕
- The prediction of a Gaussian process at each point is a *normal distribution*
  - *Uncertainty* is quantified by the *spread* of the distribution 😊



Gaussian process as the predictive model

predictions with confidence levels

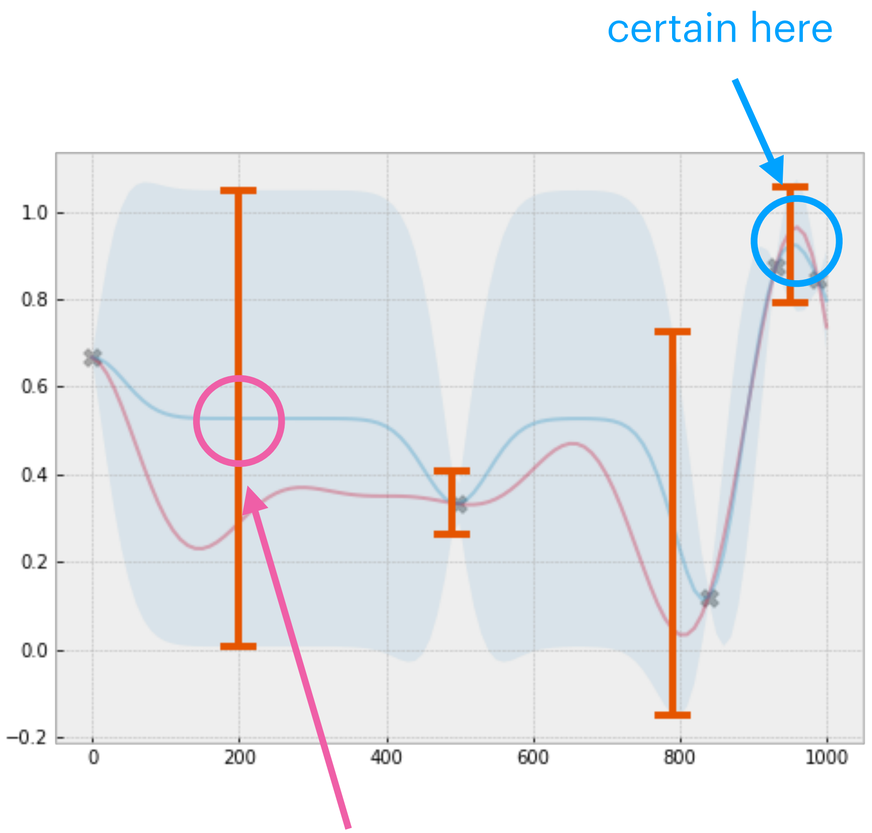
- Common ML models produce *single-valued* predictions
  - *No quantification of uncertainty* 😕
- The prediction of a Gaussian process at each point is a *normal distribution*
  - *Uncertainty* is quantified by the *spread* of the distribution 😊



Gaussian process as the predictive model

predictions with confidence levels

- Common ML models produce *single-valued* predictions
  - *No quantification of uncertainty* 😕
- The prediction of a Gaussian process at each point is a *normal distribution*
  - *Uncertainty* is quantified by the *spread* of the distribution 😊

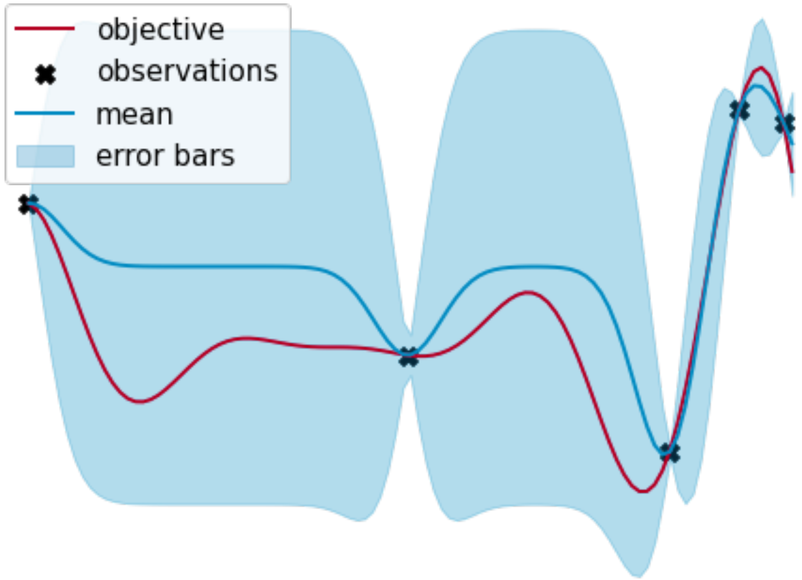


The exploitation–exploration trade-off

sticking to what we know vs. taking a risk

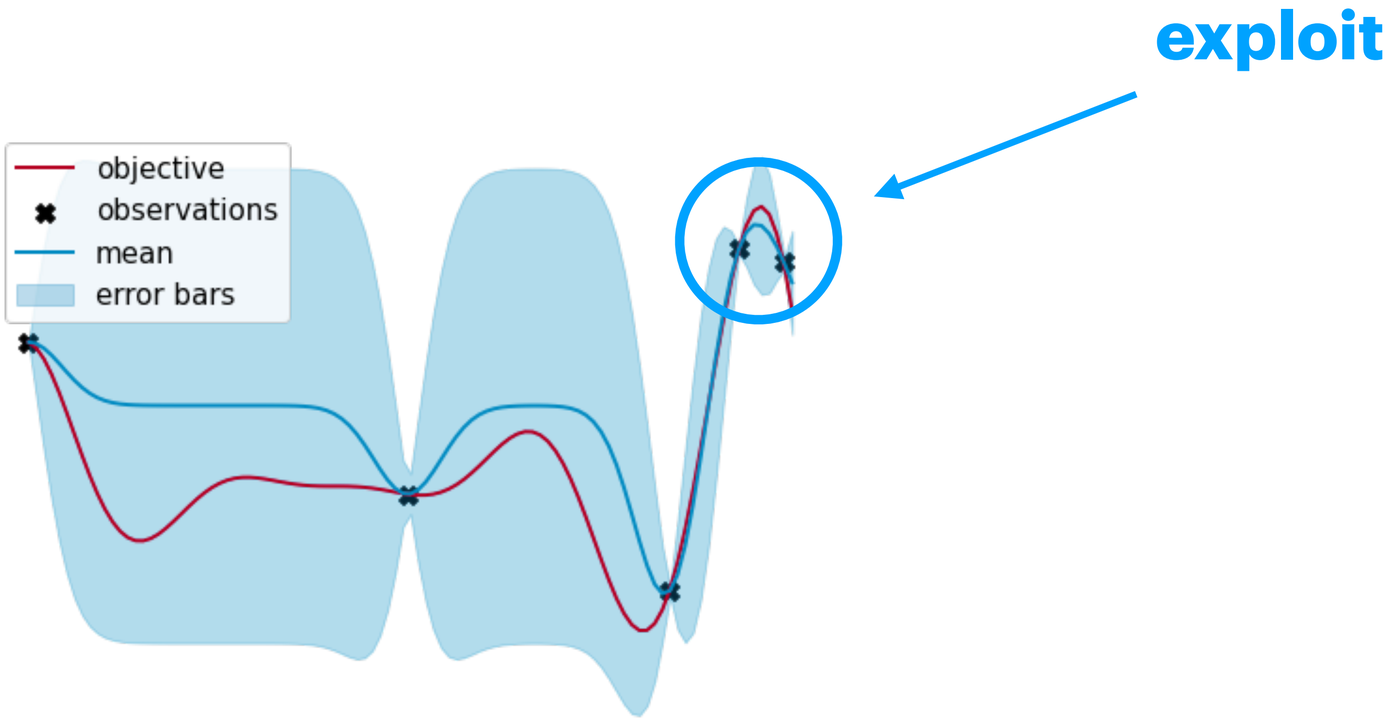
The exploitation–exploration trade-off

sticking to what we know vs. taking a risk



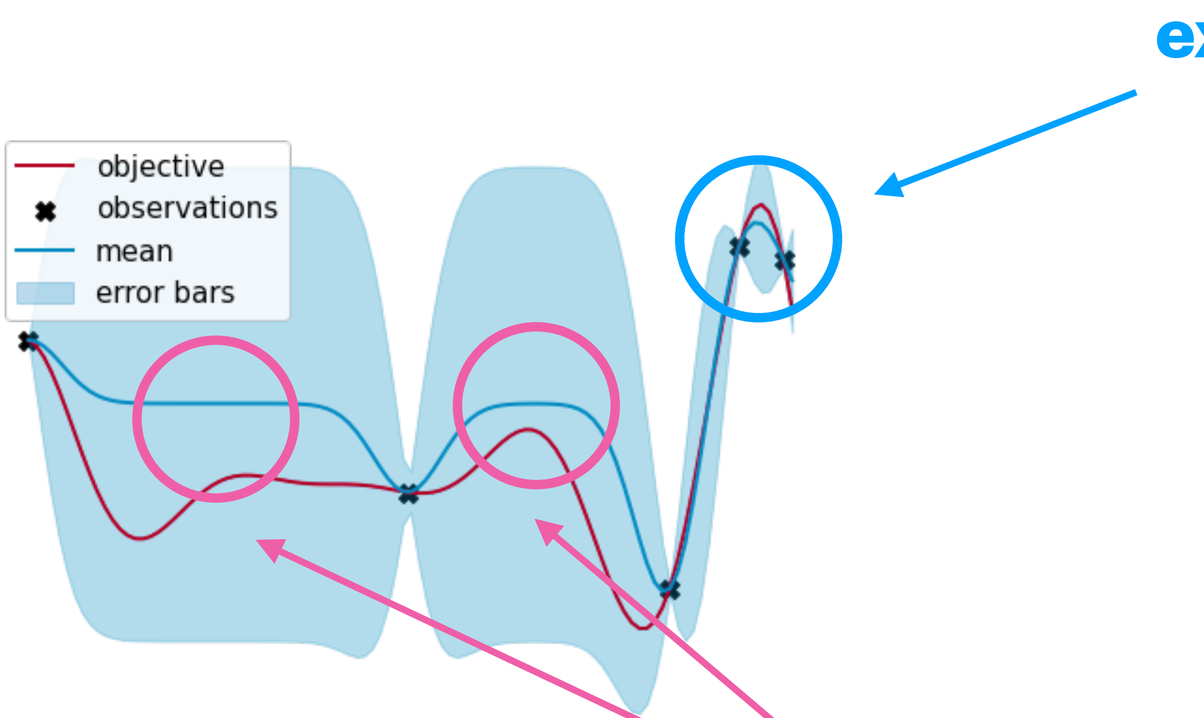
The exploitation–exploration trade-off

sticking to what we know vs. taking a risk



The exploitation–exploration trade-off

sticking to what we know vs. taking a risk

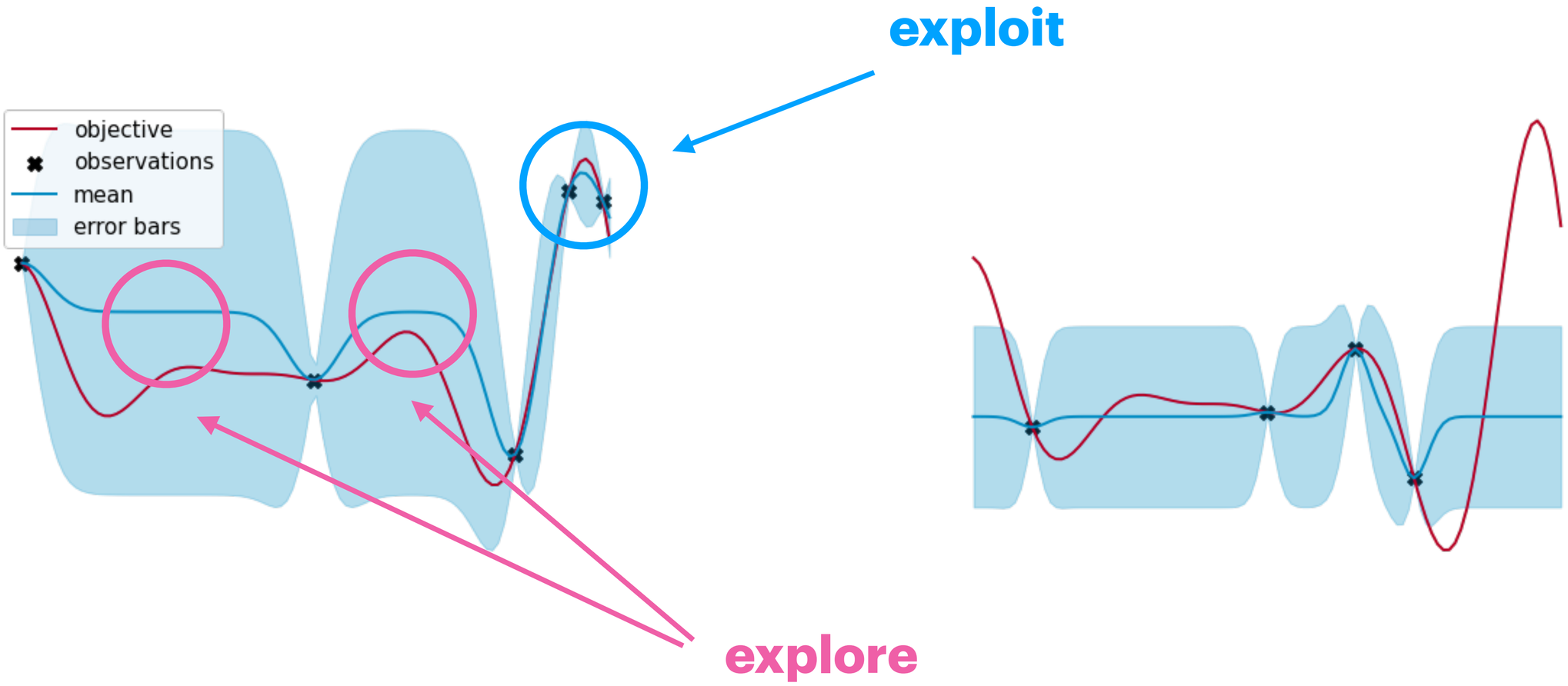


exploit

explore

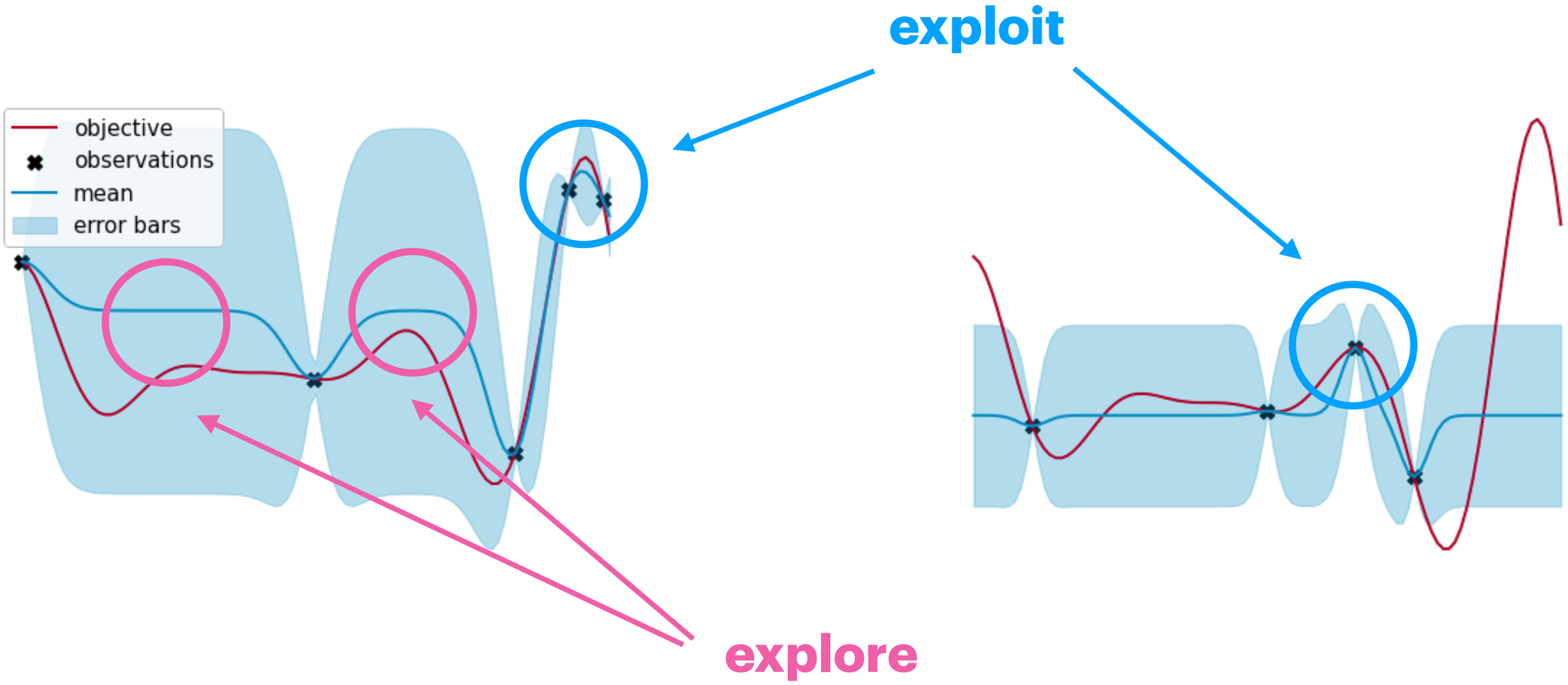
The exploitation–exploration trade-off

sticking to what we know vs. taking a risk



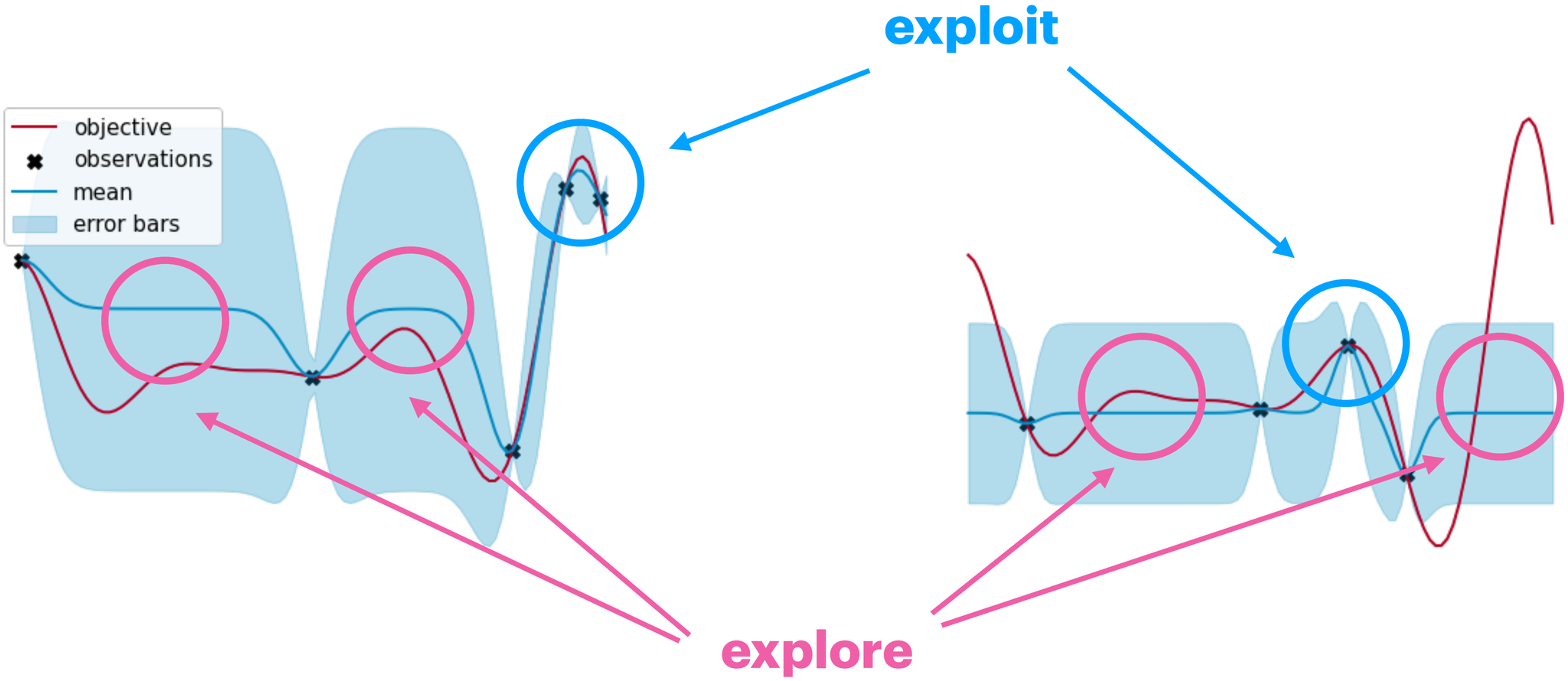
The exploitation–exploration trade-off

sticking to what we know vs. taking a risk



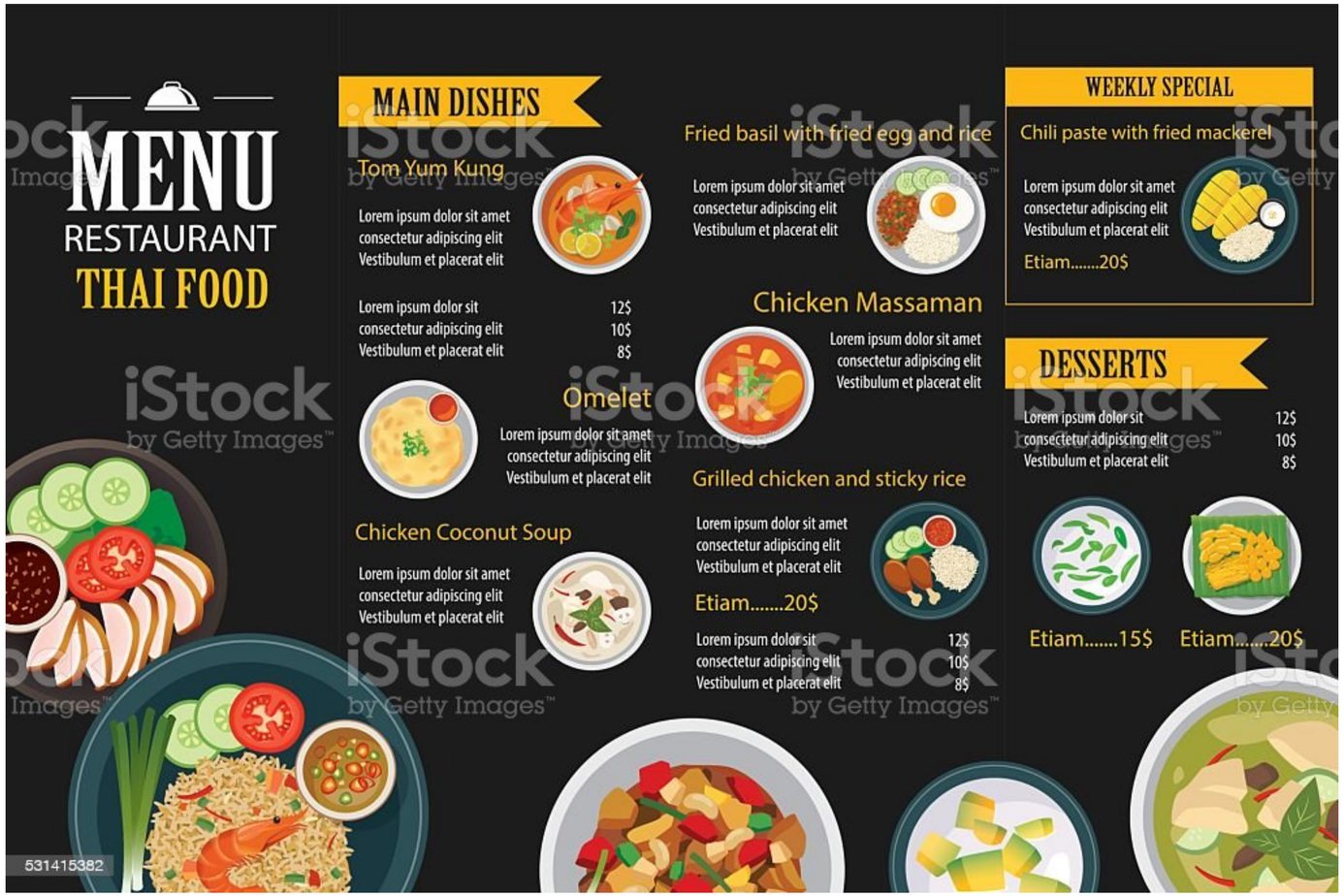
The exploitation–exploration trade-off

sticking to what we know vs. taking a risk



The exploitation–exploration trade-off

sticking to what we know vs. taking a risk

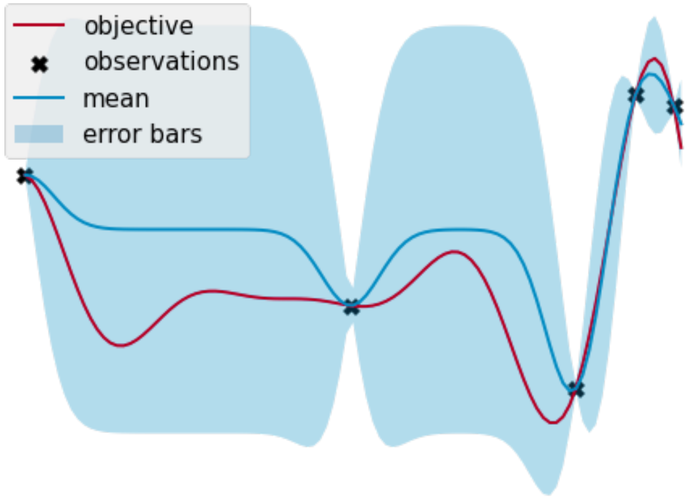


Optimization policy for decision making

choosing the best course of action, the Bayesian way

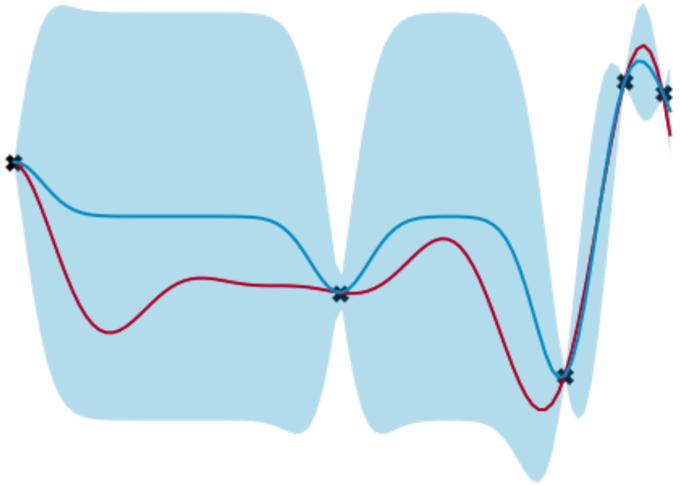
Optimization policy for decision making

choosing the best course of action, the Bayesian way



Optimization policy for decision making

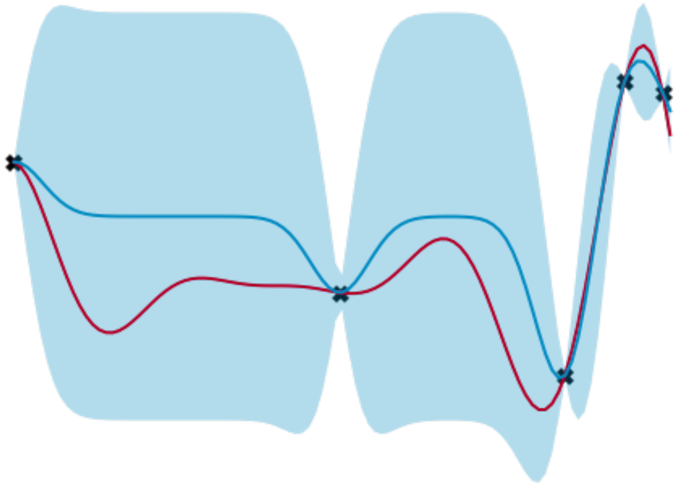
choosing the best course of action, the Bayesian way



Optimization policy for decision making

choosing the best course of action, the Bayesian way

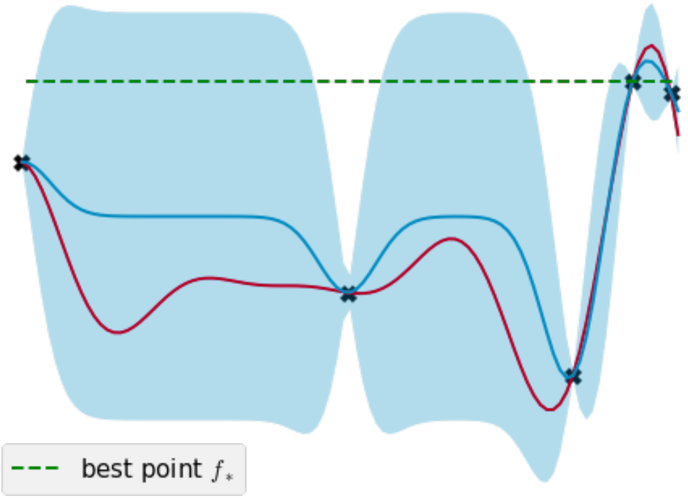
- **Goal**: pick x to improve from the best point f_*



Optimization policy for decision making

choosing the best course of action, the Bayesian way

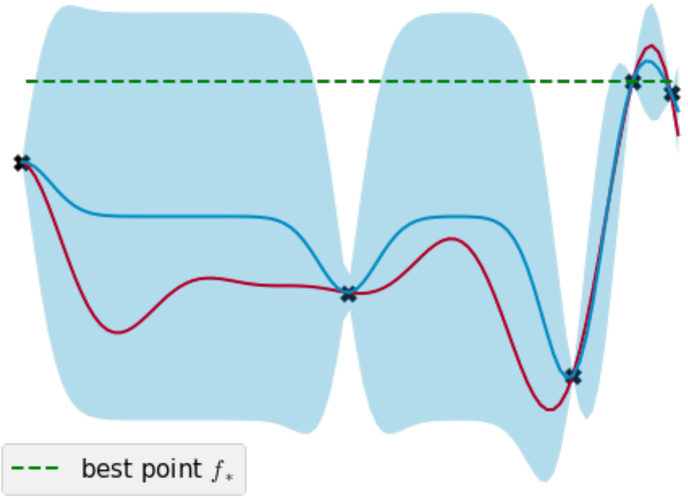
- **Goal**: pick x to improve from the best point f_*



Optimization policy for decision making

choosing the best course of action, the Bayesian way

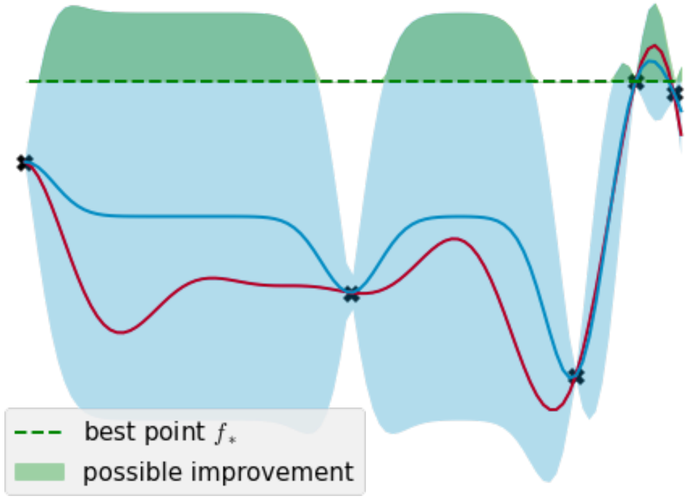
- **Goal**: pick x to improve from the best point f_*
  - Want to maximize $\max\{0, f(x) - f_*\}$



Optimization policy for decision making

choosing the best course of action, the Bayesian way

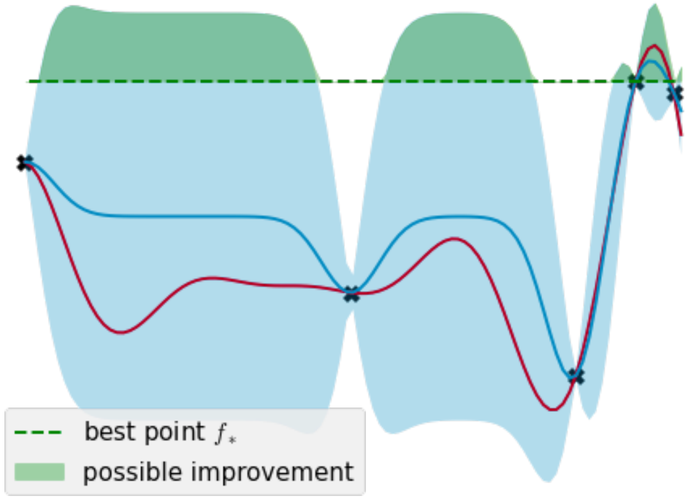
- **Goal**: pick x to improve from the best point f_*
  - Want to maximize $\max\{0, f(x) - f_*\}$



Optimization policy for decision making

choosing the best course of action, the Bayesian way

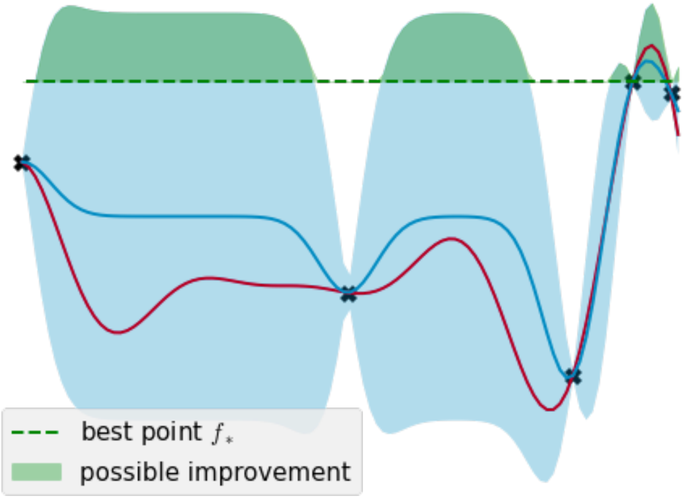
- **Goal**: pick x to improve from the best point f_*
  - Want to maximize $\max\{0, f(x) - f_*\}$
- **Problem**: $f(x)$ is unknown



Optimization policy for decision making

choosing the best course of action, the Bayesian way

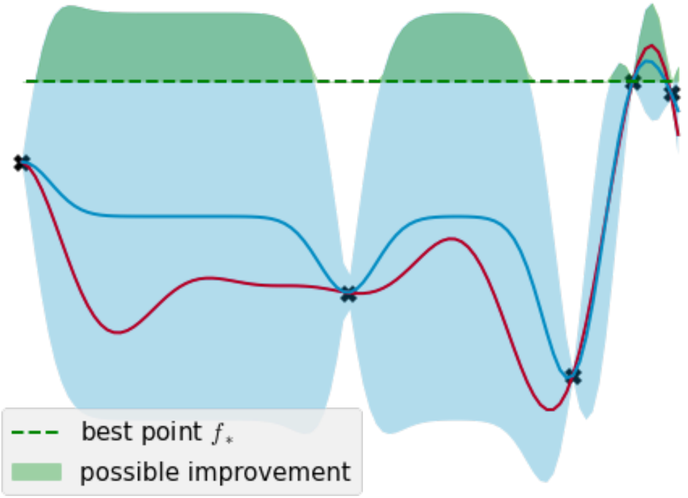
- **Goal**: pick x to improve from the best point f_*
  - Want to maximize $\max\{0, f(x) - f_*\}$
- **Problem**: $f(x)$ is unknown
- **Solution**: $f(x)$ follows a normal distribution



Optimization policy for decision making

choosing the best course of action, the Bayesian way

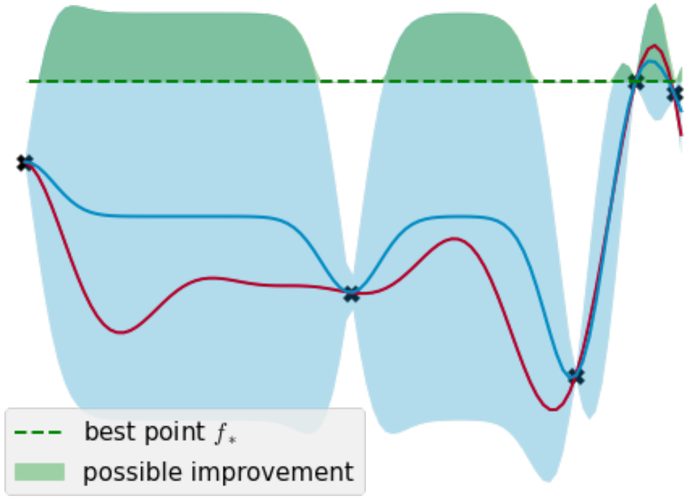
- **Goal**: pick x to improve from the best point f_*
  - Want to maximize $\max\{0, f(x) - f_*\}$
- **Problem**: $f(x)$ is unknown
- **Solution**: $f(x)$ follows a normal distribution
  - Can compute the expected improvement (**EI**) value $\mathbb{E}\left[\max\{0, f(x) - f_*\}\right]$

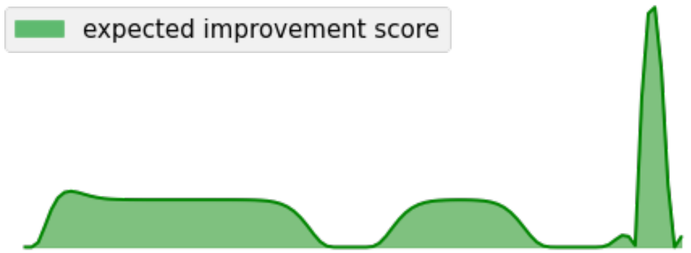


Optimization policy for decision making

choosing the best course of action, the Bayesian way

- **Goal**: pick x to improve from the best point f_*
  - Want to maximize $\max\{0, f(x) - f_*\}$
- **Problem**: $f(x)$ is unknown
- **Solution**: $f(x)$ follows a normal distribution
  - Can compute the expected improvement (**EI**) value $\mathbb{E}\left[\max\{0, f(x) - f_*\}\right]$

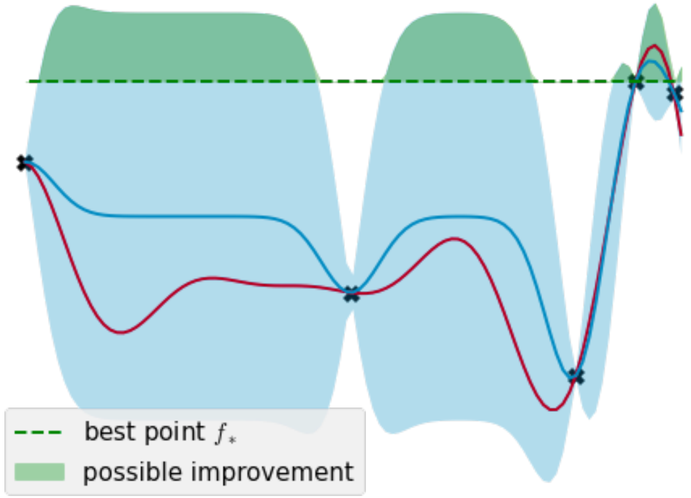


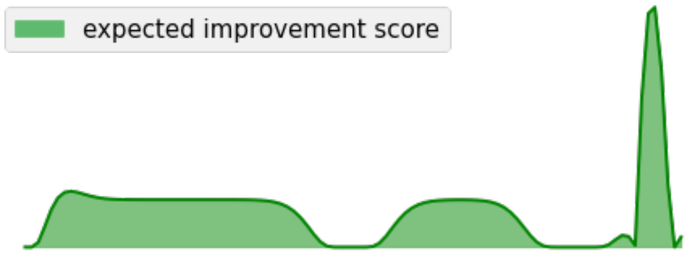


Optimization policy for decision making

choosing the best course of action, the Bayesian way

- **Goal**: pick x to improve from the best point f_*
  - Want to maximize $\max\{0, f(x) - f_*\}$
- **Problem**: $f(x)$ is unknown
- **Solution**: $f(x)$ follows a normal distribution
  - Can compute the expected improvement (**EI**) value $\mathbb{E}\left[\max\{0, f(x) - f_*\}\right]$
    - Balances large *means (exploit)* and *standard deviations (explore)*

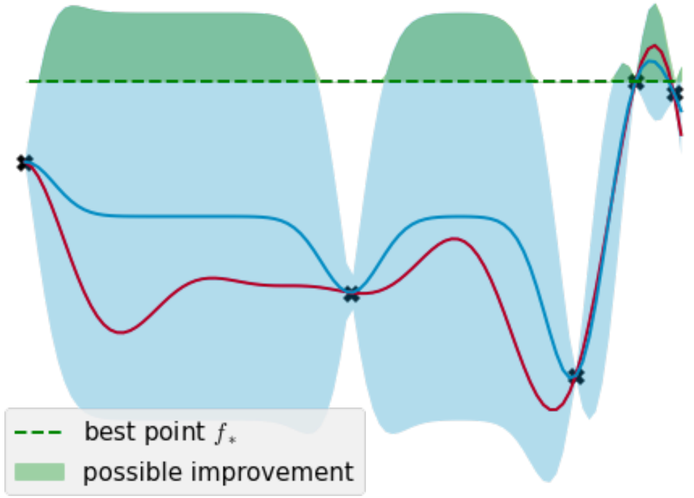


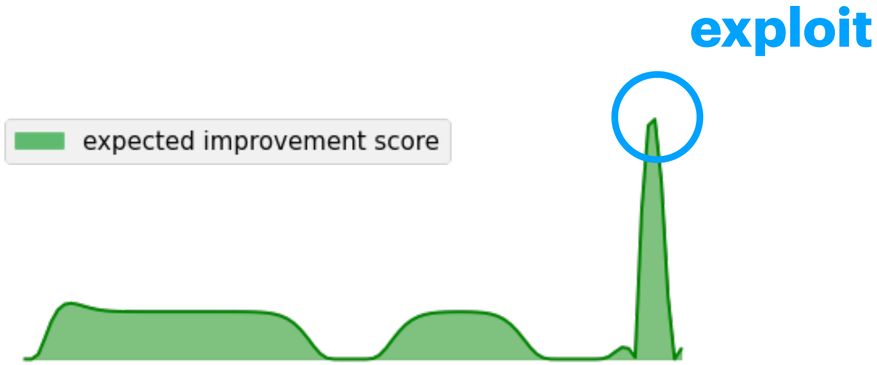


Optimization policy for decision making

choosing the best course of action, the Bayesian way

- **Goal**: pick x to improve from the best point f_*
  - Want to maximize $\max\{0, f(x) - f_*\}$
- **Problem**: $f(x)$ is unknown
- **Solution**: $f(x)$ follows a normal distribution
  - Can compute the expected improvement (**EI**) value $\mathbb{E}\left[\max\{0, f(x) - f_*\}\right]$
    - Balances large *means (exploit)* and *standard deviations (explore)*

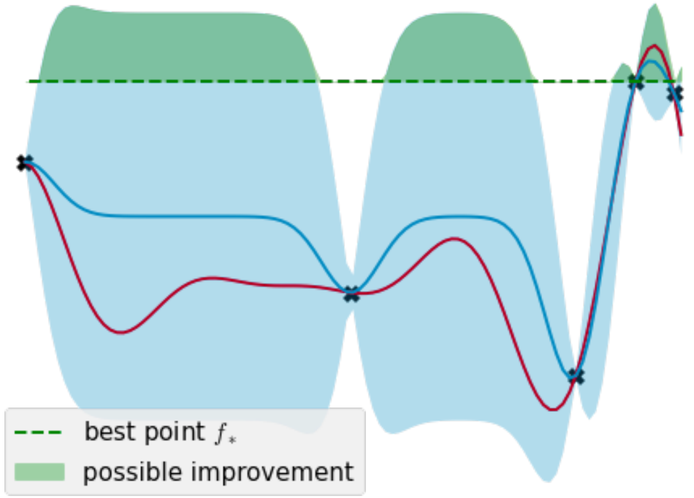


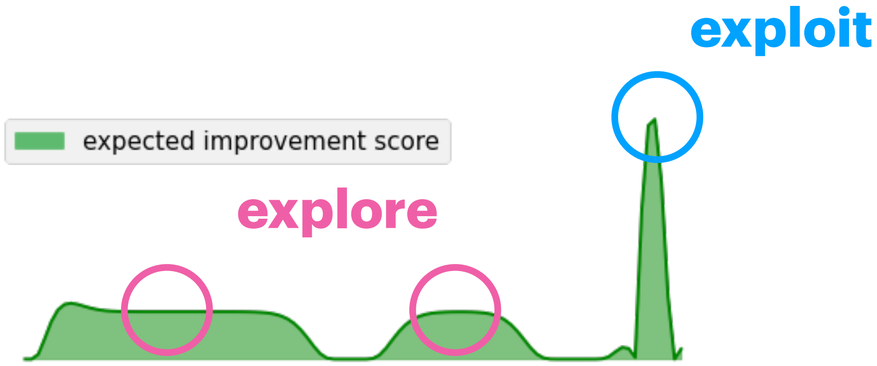


Optimization policy for decision making

choosing the best course of action, the Bayesian way

- **Goal**: pick x to improve from the best point f_*
  - Want to maximize $\max\{0, f(x) - f_*\}$
- **Problem**: $f(x)$ is unknown
- **Solution**: $f(x)$ follows a normal distribution
  - Can compute the expected improvement (**EI**) value $\mathbb{E}\left[\max\{0, f(x) - f_*\}\right]$
    - Balances large *means (exploit)* and *standard deviations (explore)*

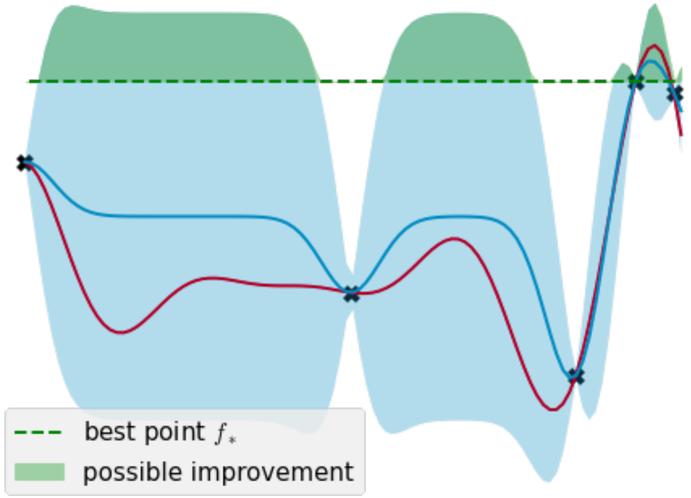


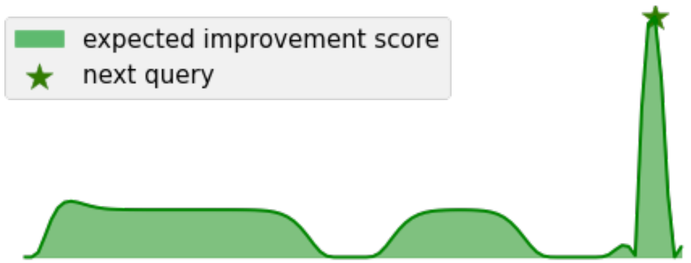


Optimization policy for decision making

choosing the best course of action, the Bayesian way

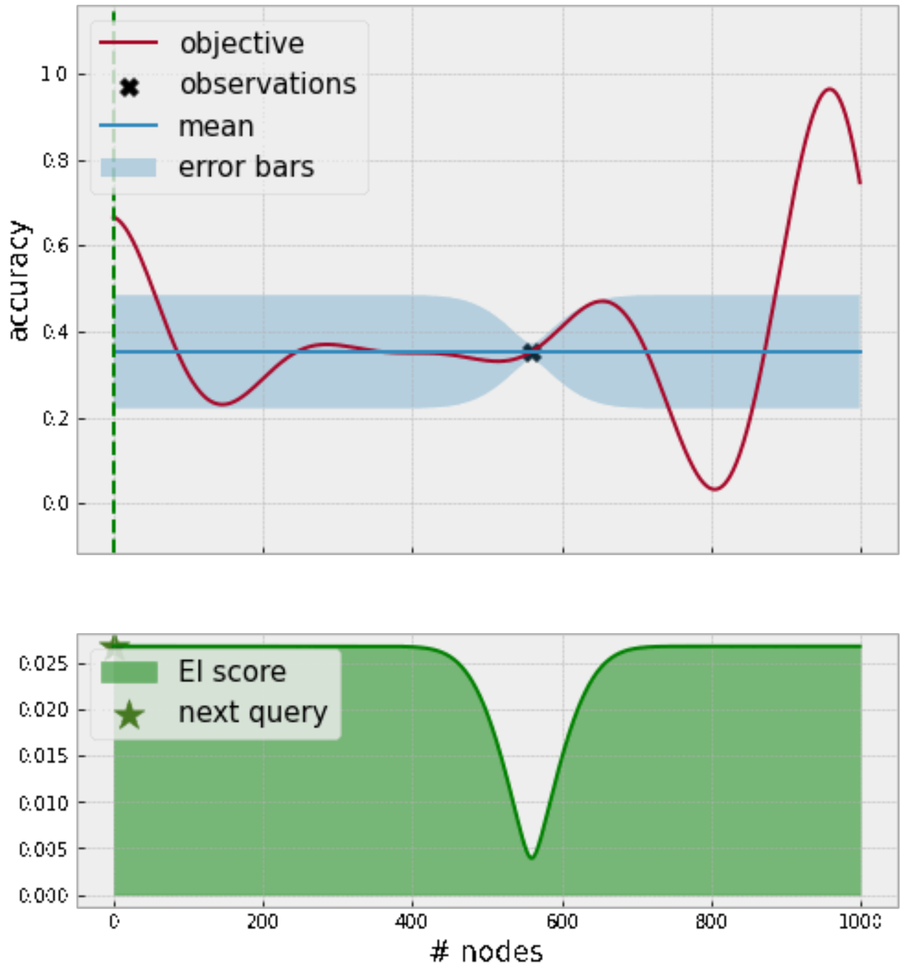
- **Goal**: pick x to improve from the best point f_*
  - Want to maximize $\max\{0, f(x) - f_*\}$
- **Problem**: $f(x)$ is unknown
- **Solution**: $f(x)$ follows a normal distribution
  - Can compute the expected improvement (**EI**) value $\mathbb{E}\left[\max\{0, f(x) - f_*\}\right]$
    - Balances large *means (exploit)* and *standard deviations (explore)*





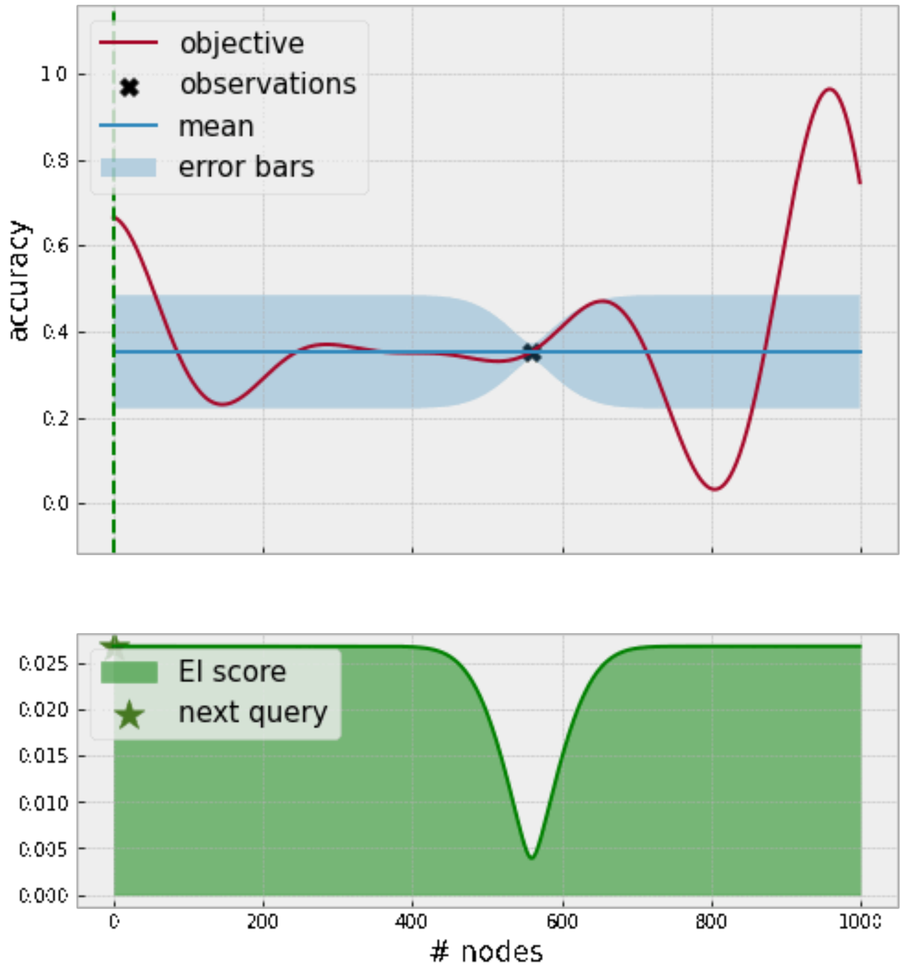
Bayesian optimization in action

benefits from adaptive decision-making



Bayesian optimization in action

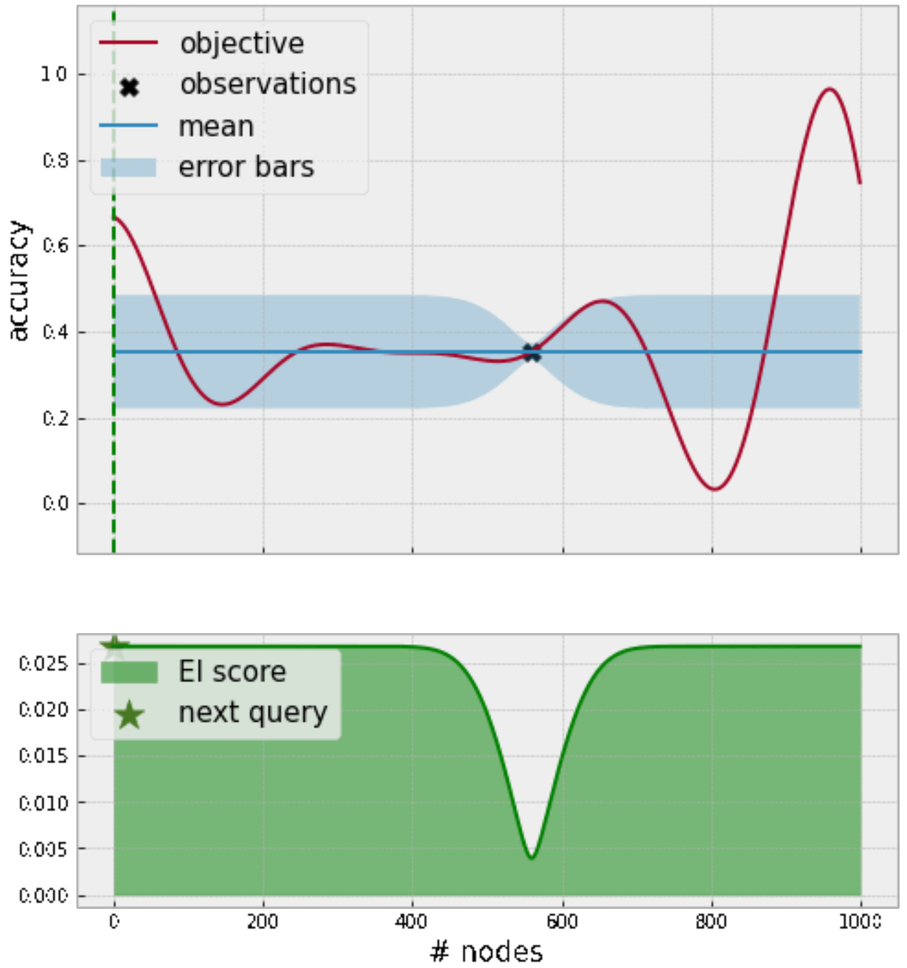
benefits from adaptive decision-making



Bayesian optimization in action

benefits from adaptive decision-making

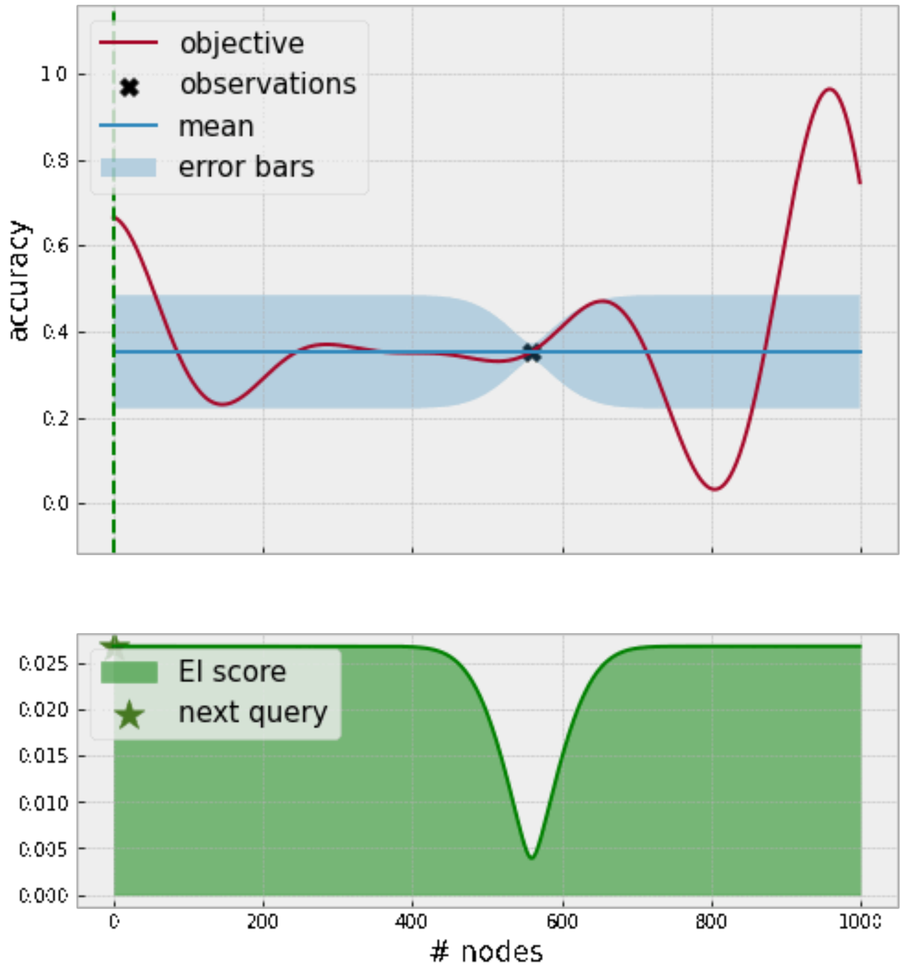
- Leverages *past observations to inform future decisions*



Bayesian optimization in action

benefits from adaptive decision-making

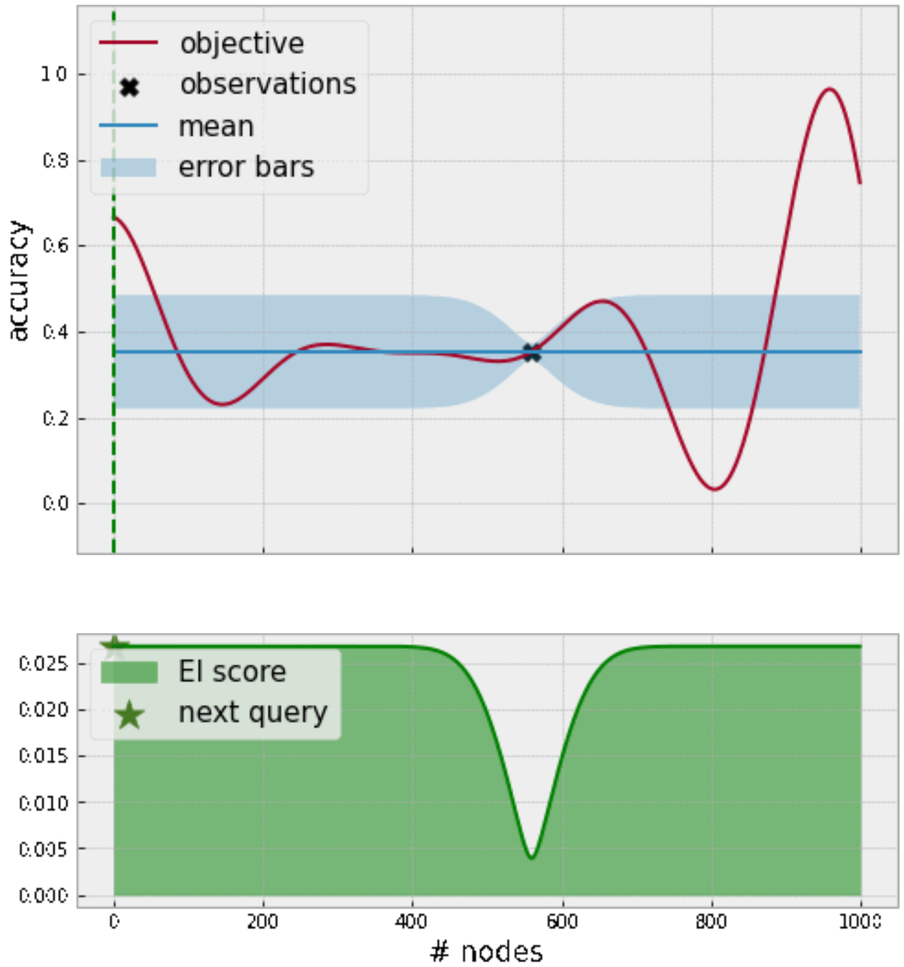
- Leverages *past observations to inform future decisions*
- Doesn't get *stuck in local optima*



Bayesian optimization in action

benefits from adaptive decision-making

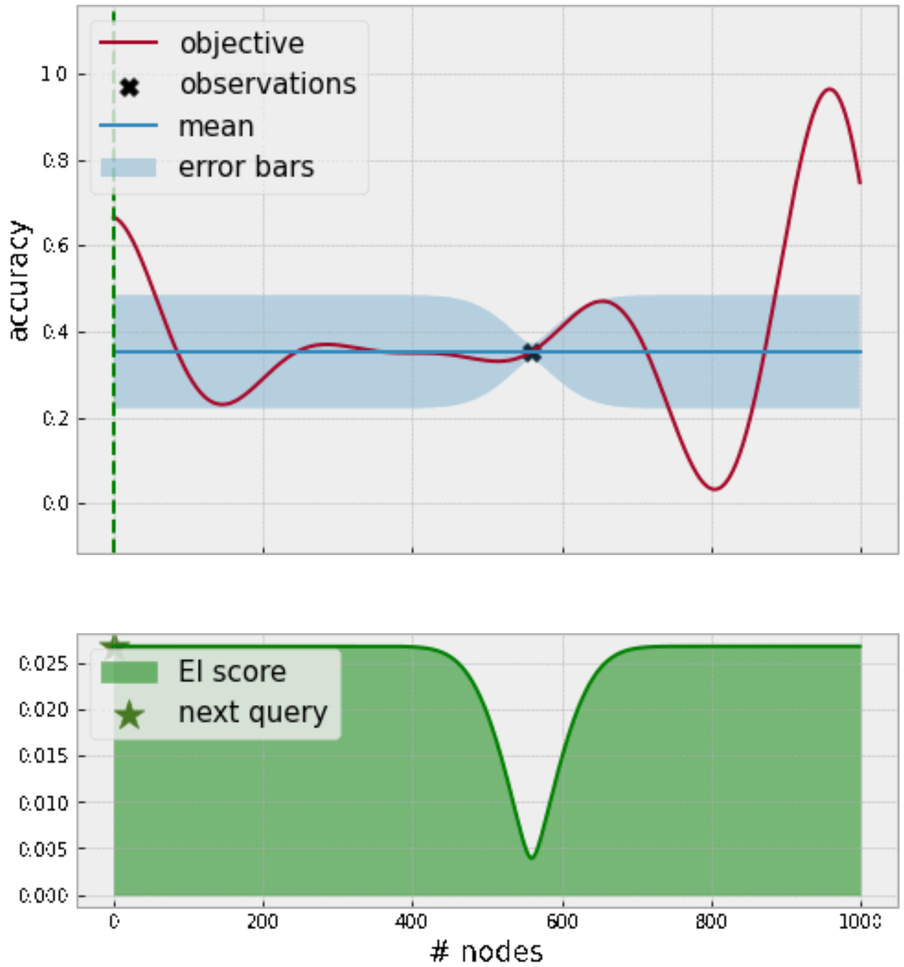
- Leverages *past observations to inform future decisions*
- Doesn't get *stuck in local optima*
- Doesn't explore *low-performing regions*



Bayesian optimization in action

benefits from adaptive decision-making

- Leverages *past observations to inform future decisions*
- Doesn't get *stuck in local optima*
- Doesn't explore *low-performing regions*
- Converges to the *global optimum*



Other Bayesian optimization policies

different heuristics to decision-making

Other Bayesian optimization policies

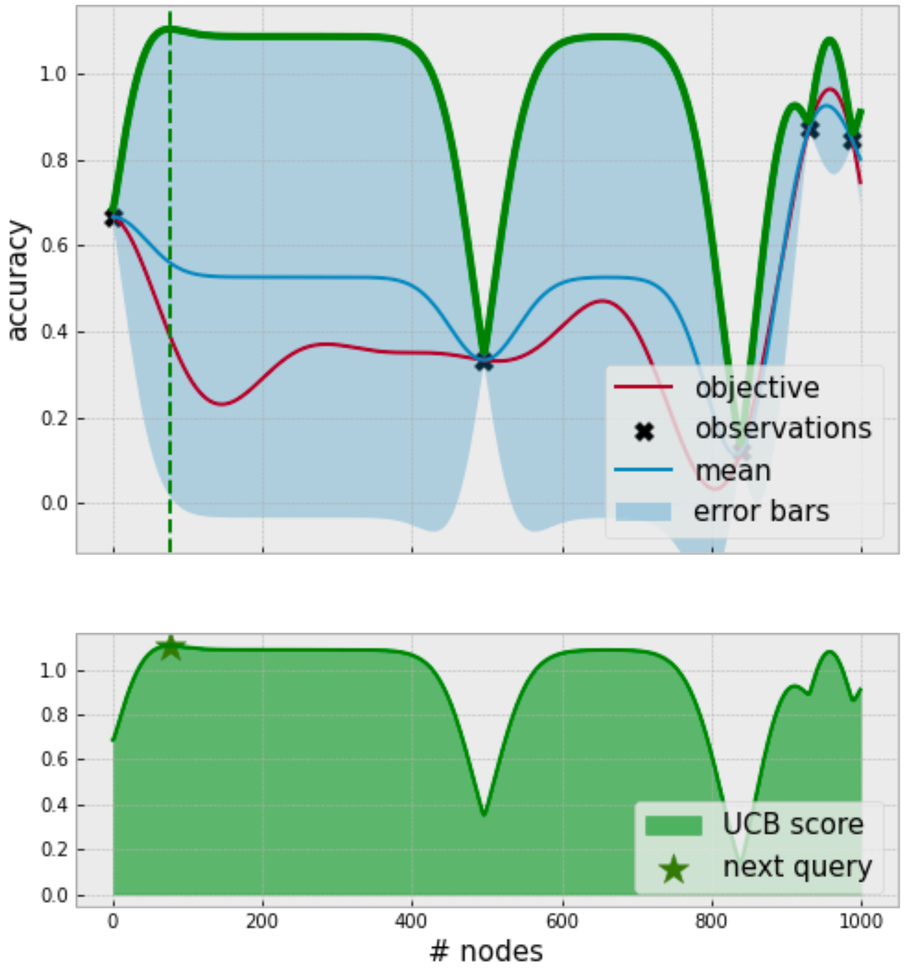
different heuristics to decision-making

- From reinforcement learning's multi-armed bandit:

Other Bayesian optimization policies

different heuristics to decision-making

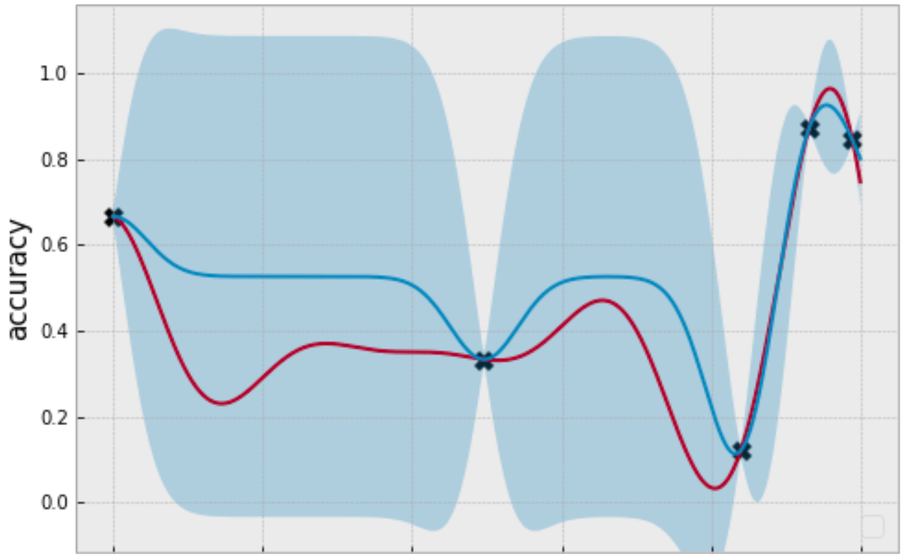
- From reinforcement learning's multi-armed bandit:
  - *Upper Confidence Bound (UCB)*



Other Bayesian optimization policies

different heuristics to decision-making

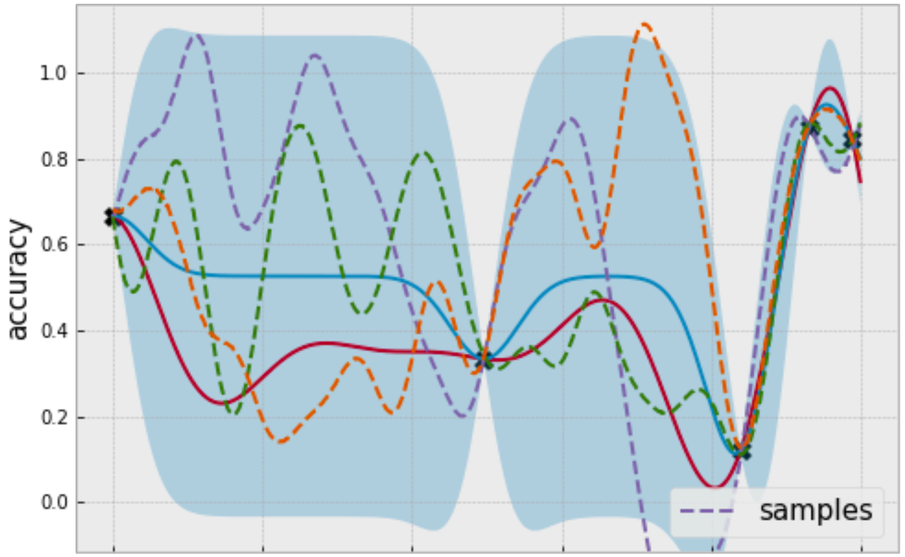
- From reinforcement learning's multi-armed bandit:
  - *Upper Confidence Bound (UCB)*
  - *Thompson Sampling*



Other Bayesian optimization policies

different heuristics to decision-making

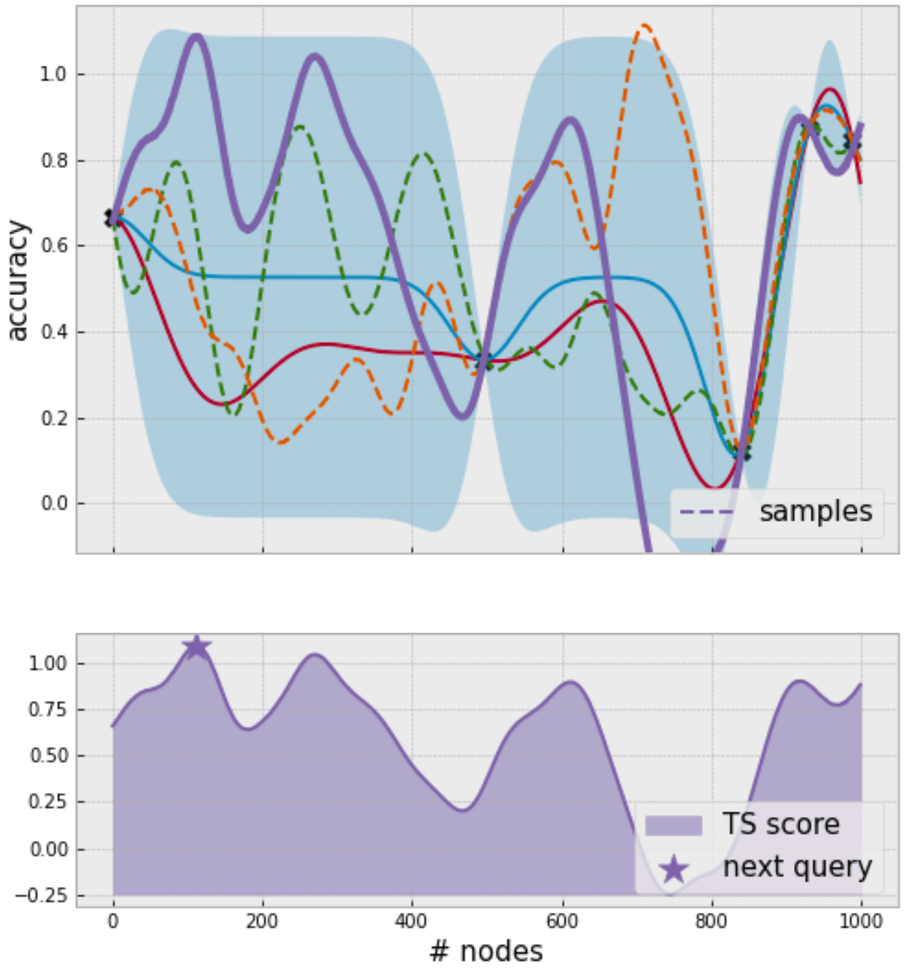
- From reinforcement learning's multi-armed bandit:
  - *Upper Confidence Bound (UCB)*
  - *Thompson Sampling*



Other Bayesian optimization policies

different heuristics to decision-making

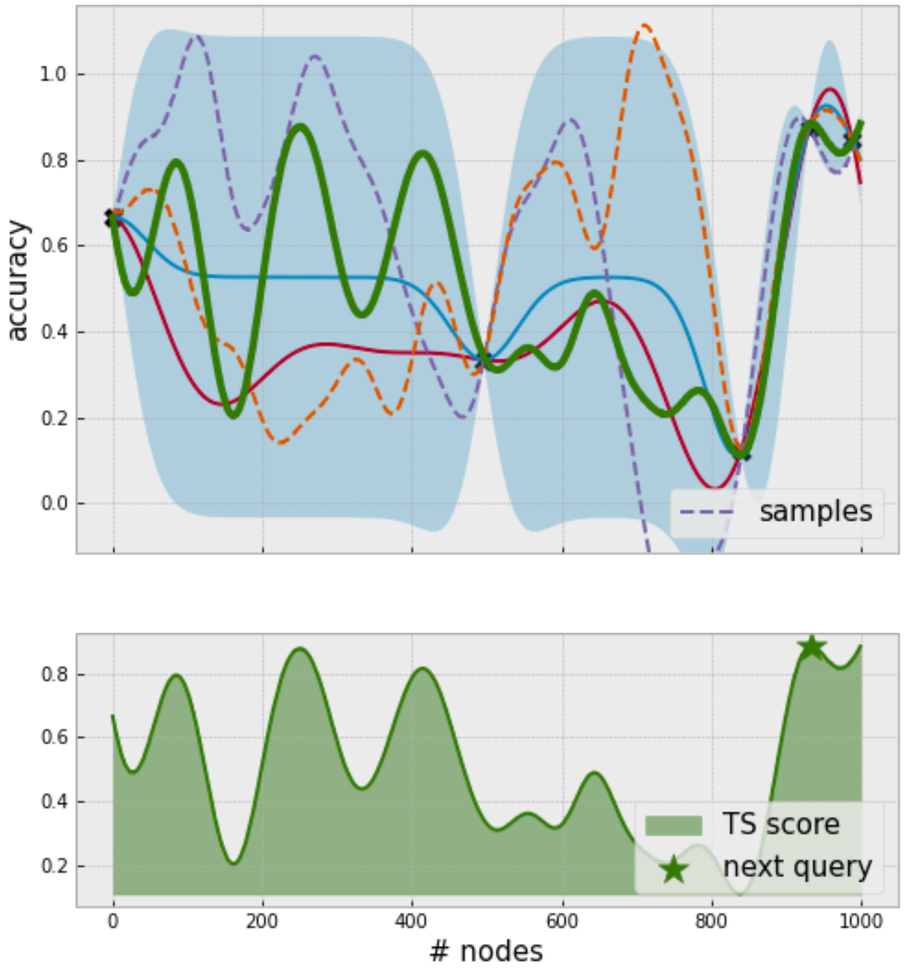
- From reinforcement learning's multi-armed bandit:
  - *Upper Confidence Bound (UCB)*
  - *Thompson Sampling*



Other Bayesian optimization policies

different heuristics to decision-making

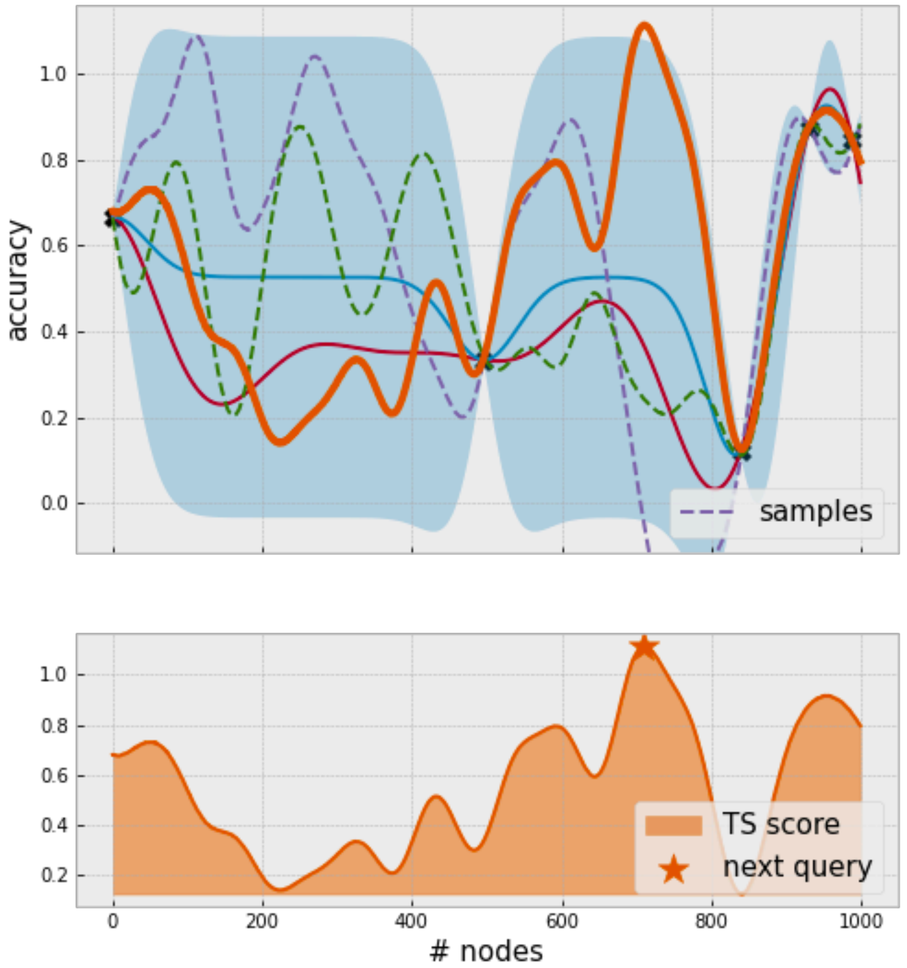
- From reinforcement learning's multi-armed bandit:
  - *Upper Confidence Bound (UCB)*
  - *Thompson Sampling*



Other Bayesian optimization policies

different heuristics to decision-making

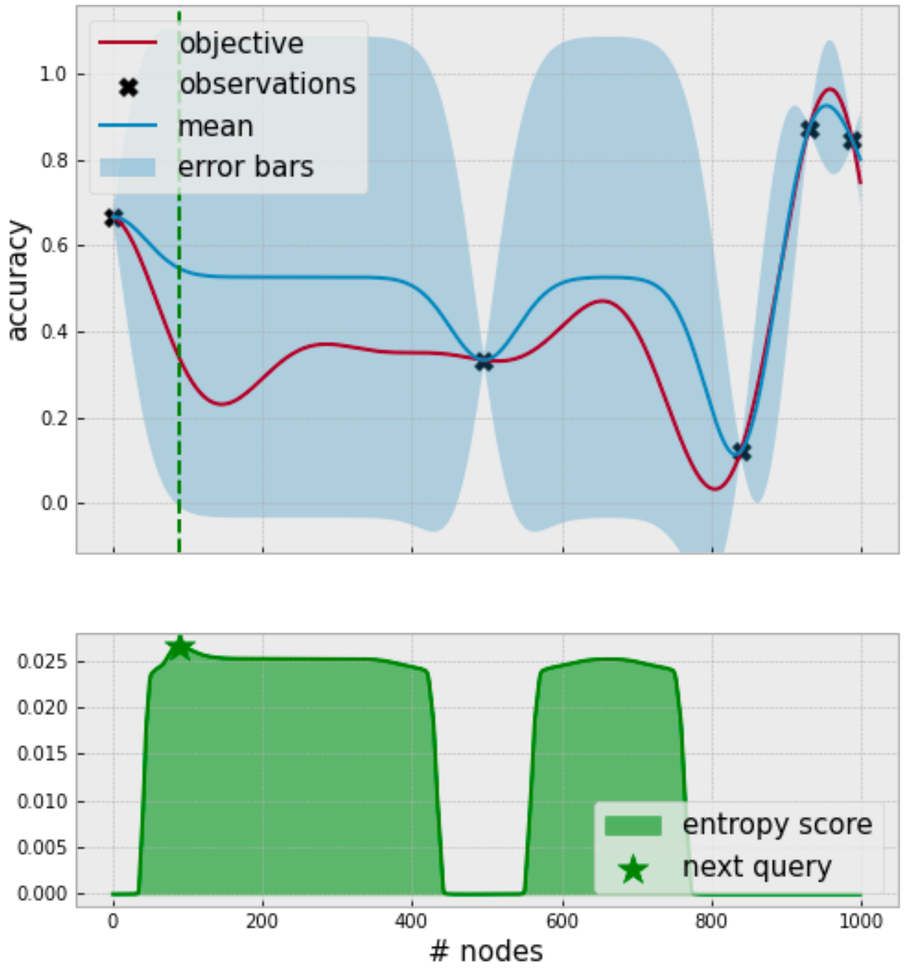
- From reinforcement learning's multi-armed bandit:
  - *Upper Confidence Bound (UCB)*
  - *Thompson Sampling*



Other Bayesian optimization policies

different heuristics to decision-making

- From reinforcement learning's multi-armed bandit:
  - *Upper Confidence Bound (UCB)*
  - *Thompson Sampling*
- From information theory (entropy): *Entropy Search*

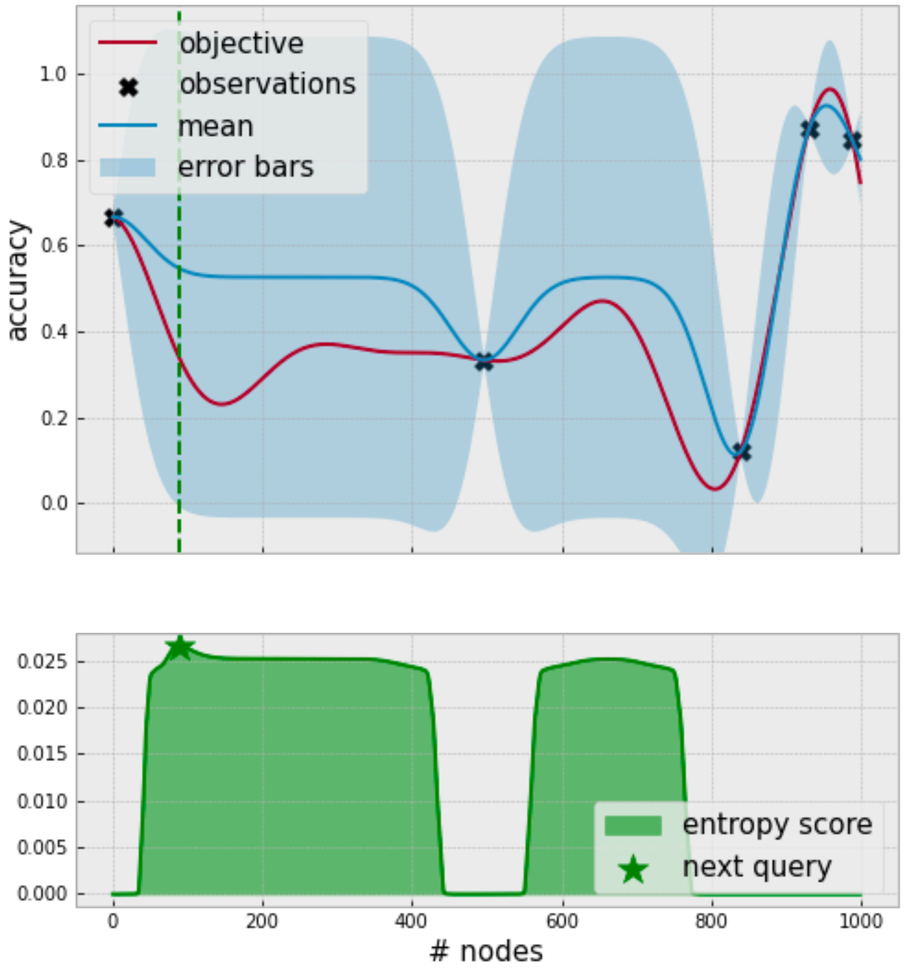


Other Bayesian optimization policies

different heuristics to decision-making

- From reinforcement learning's multi-armed bandit:
  - *Upper Confidence Bound (UCB)*
  - *Thompson Sampling*
- From information theory (entropy): *Entropy Search*
- Each policy balances between exploitation and exploration

no one-size-fits-all policy



Success stories

how others have used Bayesian optimization

Success stories

how others have used Bayesian optimization

- *Conclusive success* at

  hyperparameter tuning

Bayesian Optimization is Superior to Random Search for Machine Learning Hyperparameter Tuning: Analysis of the Black-Box Optimization Challenge 2020

Ryan Turner
rturner@twitter.com
Twitter

David Eriksson
deriksson@fb.com
Facebook

Michael McCourt
mccourt@sigopt.com
SigOpt, an Intel company

Juha Kiili
juha@valohai.com
Valohai

Eero Laaksonen
eero@valohai.com
Valohai

Zhen Xu
xuzhen@4paradigm.com
4Paradigm

Isabelle Guyon
guyon@chalearn.org
ChaLearn

Turner et al. https://arxiv.org/pdf/2104.10201.pdf

Success stories

how others have used Bayesian optimization

- *Conclusive success* at hyperparameter tuning
- *Nobel-winning* enzyme discovery

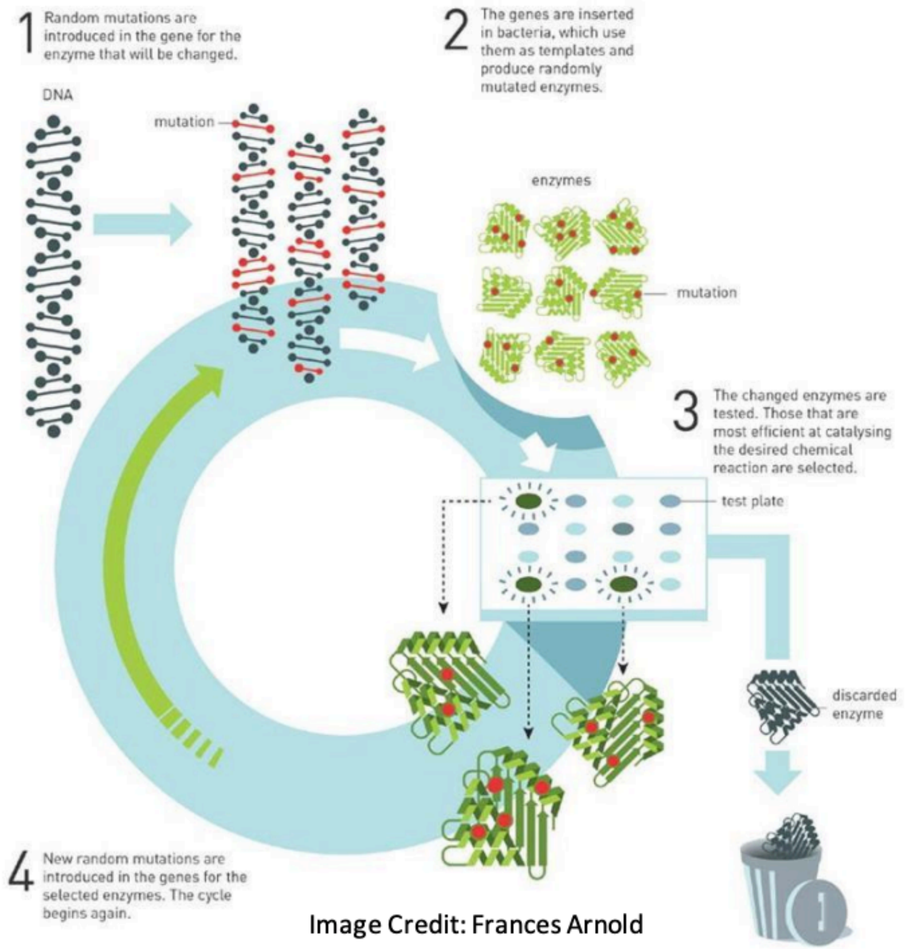


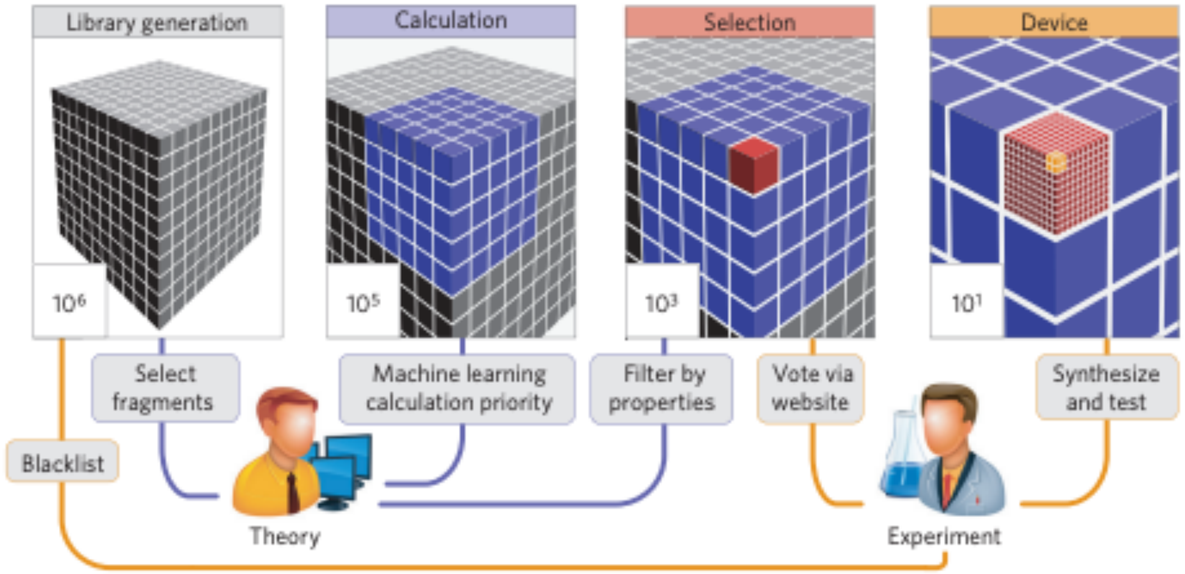
Image Credit: Frances Arnold

http://www.yisongyue.com/talks/real_world_bo.pdf

Success stories

how others have used Bayesian optimization

- *Conclusive success* at hyperparameter tuning
- *Nobel-winning* enzyme discovery
- Molecule discovery on *Nature*



Gómez-Bombarelli et al. https://www.nature.com/articles/nmat4717

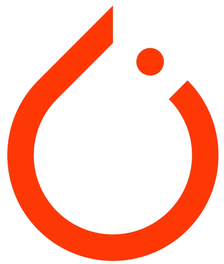
Implementing Bayesian optimization in Python

using a cohesive software ecosystem

The Torch ecosystem of Bayesian optimization

The Torch ecosystem of Bayesian optimization

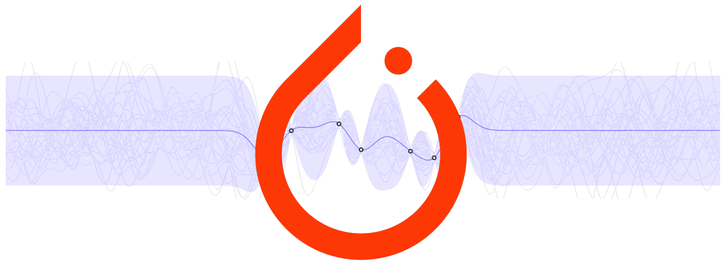
PyTorch for
tensor manipulation



The Torch ecosystem of Bayesian optimization

PyTorch for
tensor manipulation

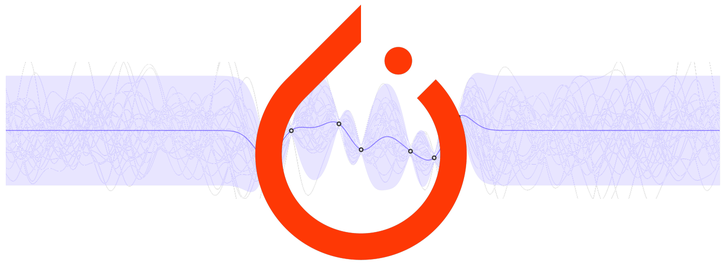
GPyTorch for
Gaussian process modeling



The Torch ecosystem of Bayesian optimization

PyTorch for
tensor manipulation

GPyTorch for
Gaussian process modeling



BoTorch for
BayesOpt policies

GPyTorch for Gaussian process modeling

modular implementation of Gaussian processes

Need to declare two components:

- **Mean function** models the *expected trend*
- **Kernel** models *smoothness*

GPyTorch for Gaussian process modeling

modular implementation of Gaussian processes

Need to declare two components:

- **Mean function** models the *expected trend*
- **Kernel** models *smoothness*

```
class GPModel(ExactGP):
    def __init__(self, train_x, train_y, likelihood):
        super().__init__(train_x, train_y, likelihood)
        self.mean_module = ConstantMean()
        self.covar_module = RBFKernel()

    def forward(self, x):
        mean_x = self.mean_module(x)
        covar_x = self.covar_module(x)
        return MultivariateNormal(mean_x, covar_x)
```

GPyTorch for Gaussian process modeling

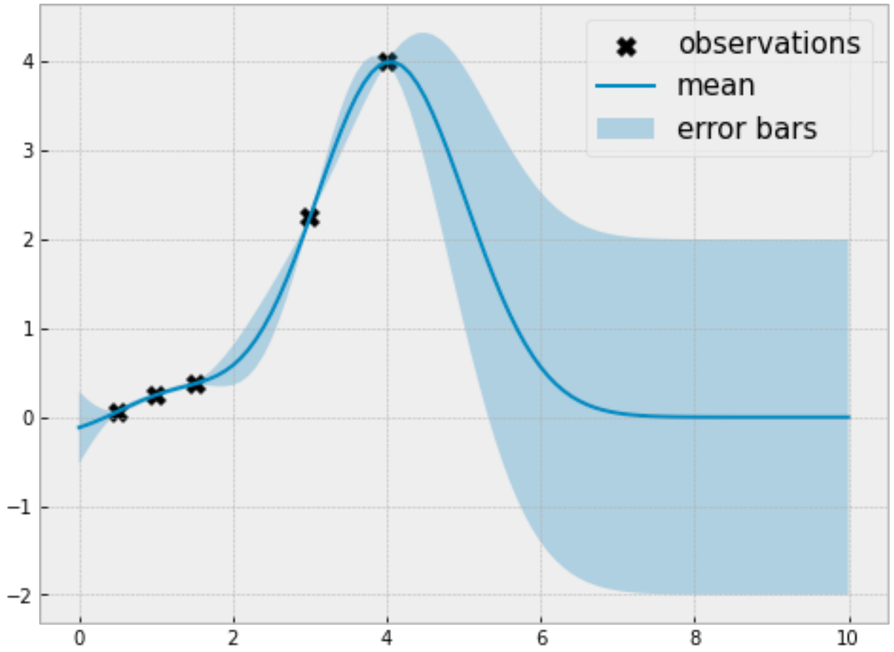
modular implementation of Gaussian processes

Need to declare two components:

- **Mean function** models the *expected trend*
- **Kernel** models *smoothness*

```
class GPModel(ExactGP):
    def __init__(self, train_x, train_y, likelihood):
        super().__init__(train_x, train_y, likelihood)
        self.mean_module = ConstantMean()
        self.covar_module = RBFKernel()

    def forward(self, x):
        mean_x = self.mean_module(x)
        covar_x = self.covar_module(x)
        return MultivariateNormal(mean_x, covar_x)
```



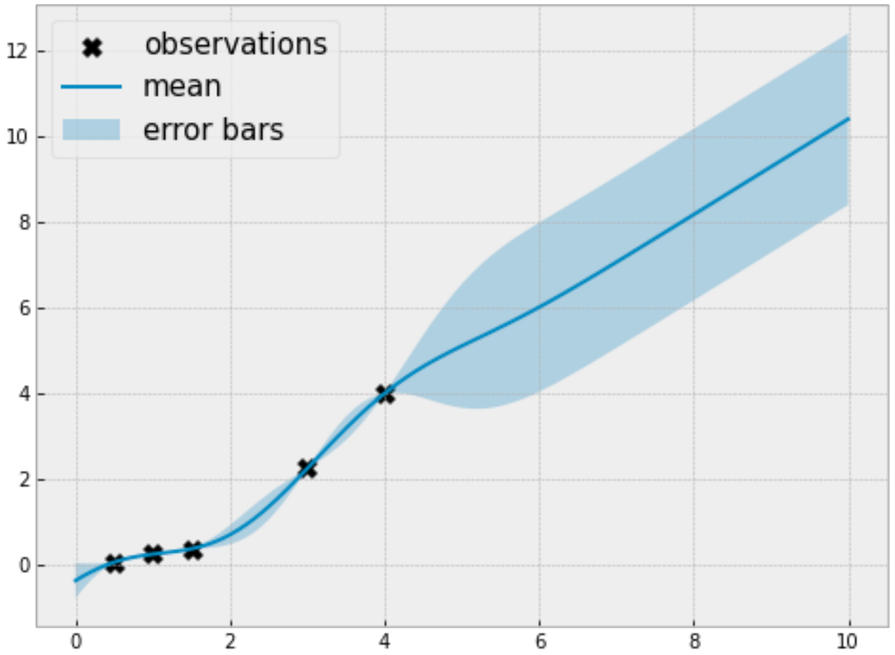
GPyTorch for Gaussian process modeling

modular implementation of Gaussian processes

Need to declare two components:

- **Mean function** models the *expected trend*
- **Kernel** models *smoothness*

```
class GPModel(ExactGP):
    def __init__(self, train_x, train_y, likelihood):
        super().__init__(train_x, train_y, likelihood)
        self.mean_module = LinearMean(1)
        self.covar_module = RBFKernel()
```



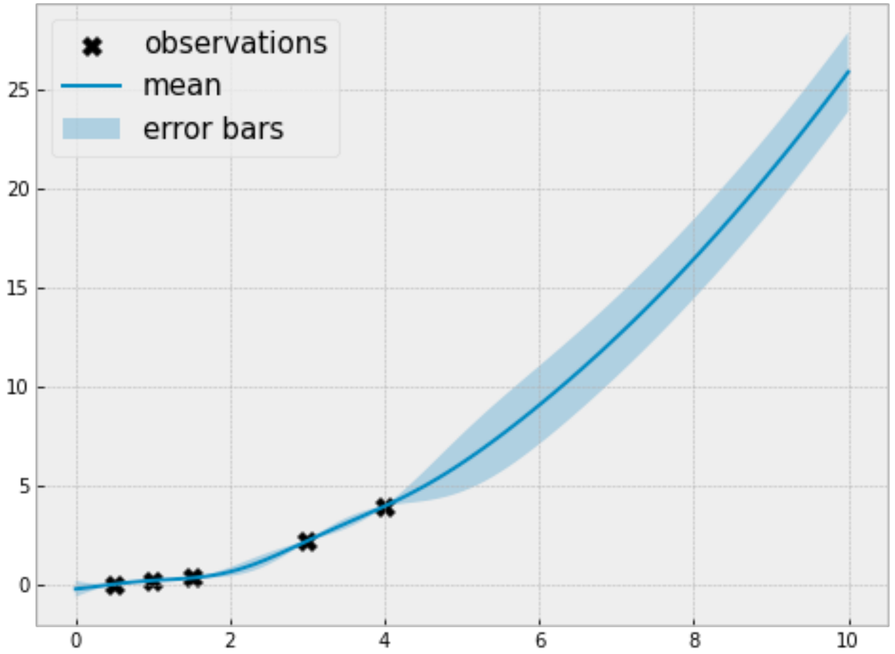
GPyTorch for Gaussian process modeling

modular implementation of Gaussian processes

Need to declare two components:

- **Mean function** models the *expected trend*
- **Kernel** models *smoothness*

```
class QuadraticMean(Mean):
    def forward(self, x):
        res = (
            x.pow(2).matmul(self.second)
            + x.matmul(self.first)
            + self.bias
        )
        return res
```



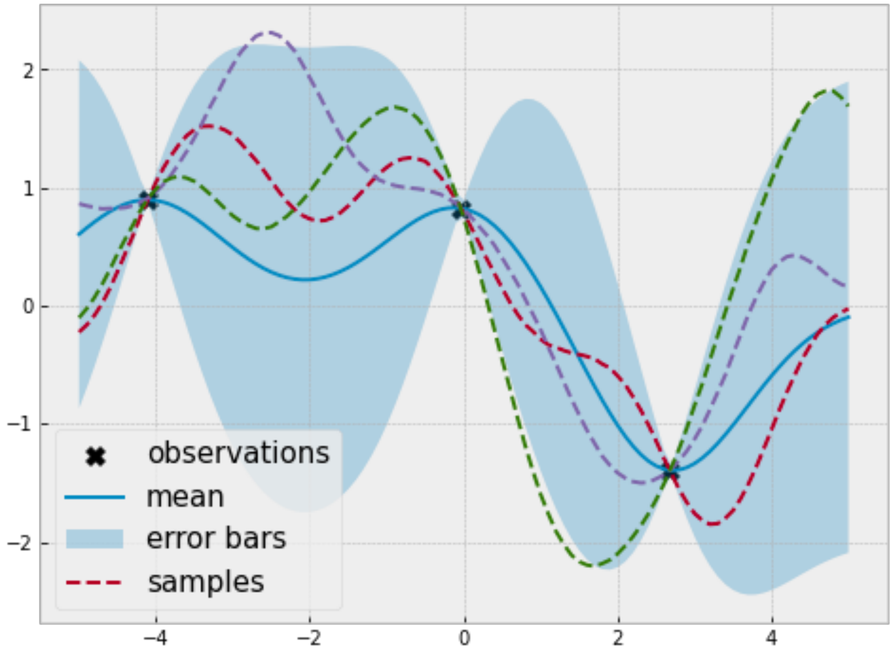
GPyTorch for Gaussian process modeling

modular implementation of Gaussian processes

Need to declare two components:

- **Mean function** models the *expected trend*
- **Kernel** models *smoothness*

```
class GPModel(ExactGP):
    def __init__(self, train_x, train_y, likelihood):
        super().__init__(train_x, train_y, likelihood)
        self.mean_module = ConstantMean()
        self.covar_module = RBFKernel()
```



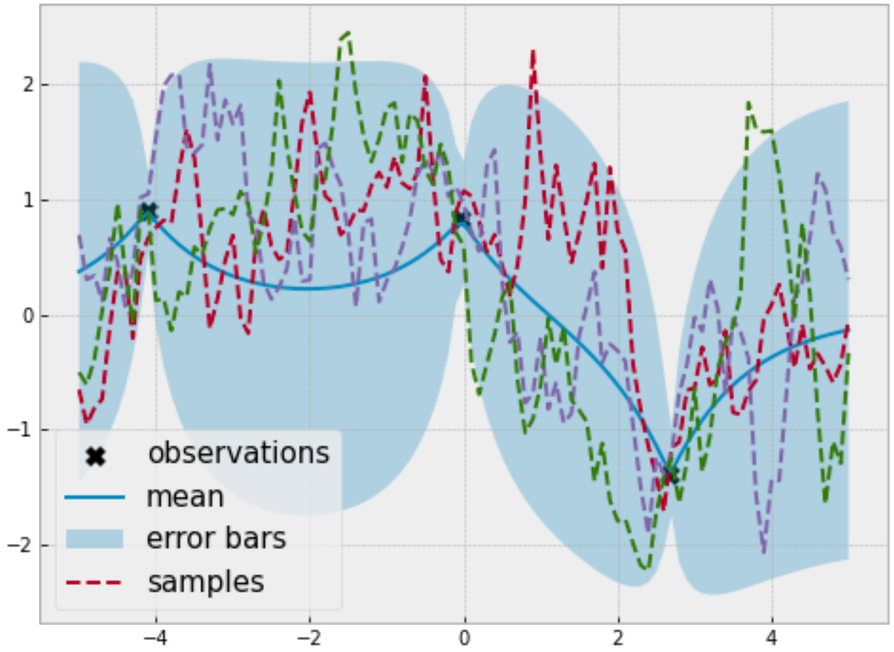
GPyTorch for Gaussian process modeling

modular implementation of Gaussian processes

Need to declare two components:

- **Mean function** models the *expected trend*
- **Kernel** models *smoothness*

```
class GPModel(ExactGP):
    def __init__(self, train_x, train_y, likelihood):
        super().__init__(train_x, train_y, likelihood)
        self.mean_module = ConstantMean()
        self.covar_module = MaternKernel(nu=0.5)
```



BoTorch for Bayesian optimization policies

modular decision making

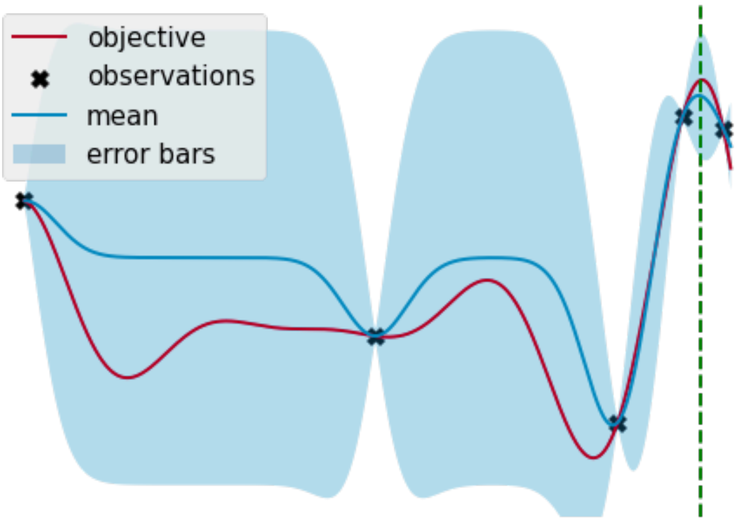
changing the policy == replacing one line of code (most of the time)

BoTorch for Bayesian optimization policies

modular decision making

changing the policy == replacing one line of code (most of the time)

```
policy = ExpectedImprovement(
    model, best_f=train_y.max()
)
```

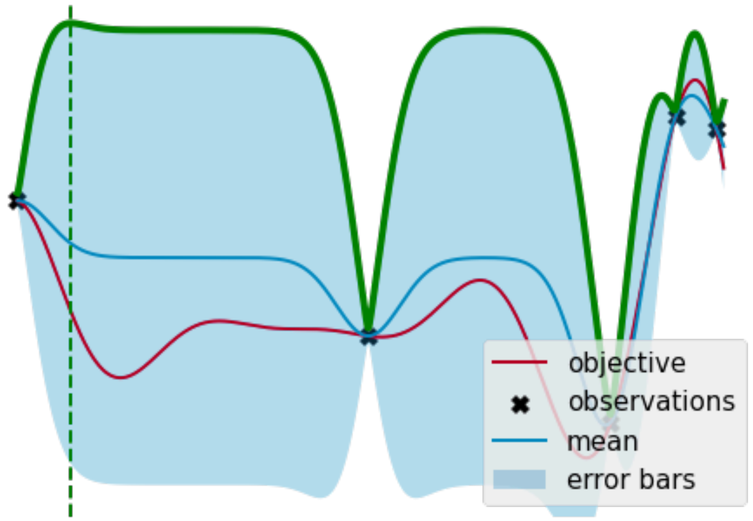


BoTorch for Bayesian optimization policies

modular decision making

changing the policy == replacing one line of code (most of the time)

```
policy = UpperConfidenceBound(
    model, beta=2
)
```

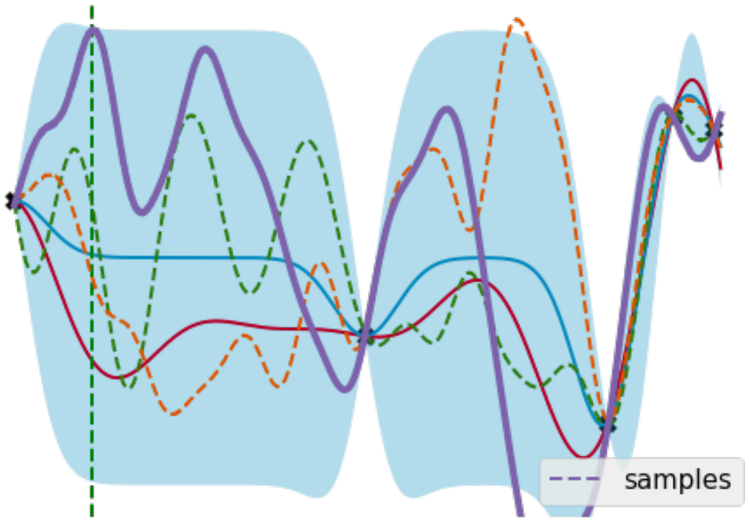


BoTorch for Bayesian optimization policies

modular decision making

changing the policy == replacing one line of code (most of the time)

```
sobol = SobolEngine(1, scramble=True)
candidate_x = sobol.draw(1000)
ts = MaxPosteriorSampling(
    model, replacement=False
)
```



Bayesian optimization in practice

using Bayesian optimization in the real world

Bayesian optimization in practice

using Bayesian optimization in the real world

- **Batch optimization** to maximize throughput

Bayesian optimization in practice

using Bayesian optimization in the real world

- **Batch optimization** to maximize throughput
  - Making multiple evaluations *at the same time*

Bayesian optimization in practice

using Bayesian optimization in the real world

- **Batch optimization** to maximize throughput
  - Making multiple evaluations *at the same time*
- **Constrained optimization** for safe solutions

Bayesian optimization in practice

using Bayesian optimization in the real world

- **Batch optimization** to maximize throughput
  - Making multiple evaluations *at the same time*
- **Constrained optimization** for safe solutions
  - *Satisfying constraints* during optimization

Bayesian optimization in practice

using Bayesian optimization in the real world

- **Batch optimization** to maximize throughput
  - Making multiple evaluations *at the same time*
- **Constrained optimization** for safe solutions
  - *Satisfying constraints* during optimization
- **Multi-objective optimization** for balanced solutions

Bayesian optimization in practice

using Bayesian optimization in the real world

- **Batch optimization** to maximize throughput
  - Making multiple evaluations *at the same time*
- **Constrained optimization** for safe solutions
  - *Satisfying constraints* during optimization
- **Multi-objective optimization** for balanced solutions
  - Trading off *competing objectives*

Bayesian optimization in practice

using Bayesian optimization in the real world

- **Batch optimization** to maximize throughput
  - Making multiple evaluations *at the same time*
- **Constrained optimization** for safe solutions
  - *Satisfying constraints* during optimization
- **Multi-objective optimization** for balanced solutions
  - Trading off *competing objectives*

All handled by **BayesOpt**!

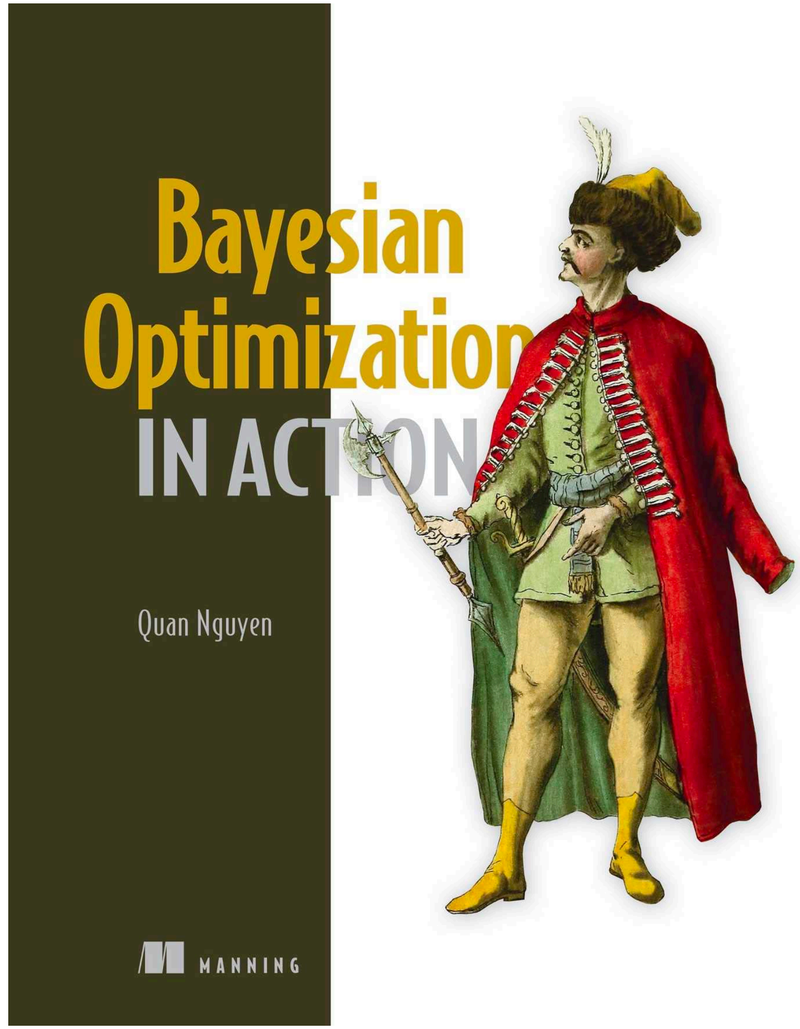
Manning's Bayesian Optimization in Action

a hands-on guide to Python implementation

Manning's Bayesian Optimization in Action

a hands-on guide to Python implementation

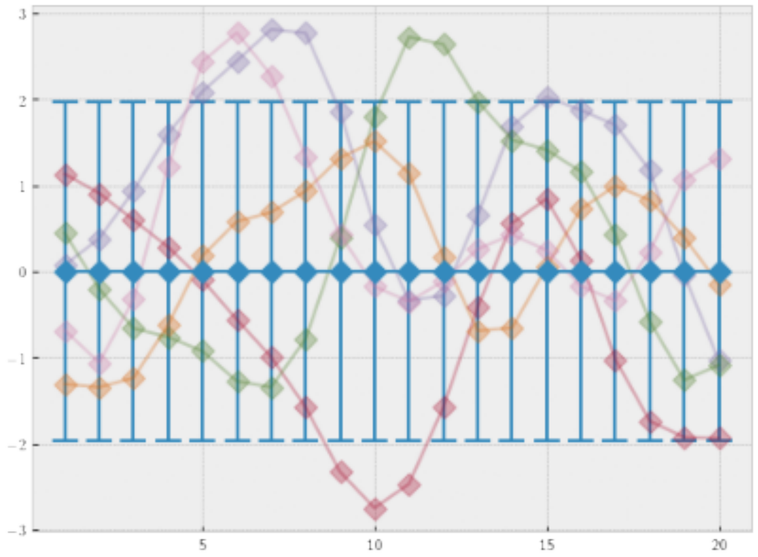
- https://www.manning.com/books/bayesian-optimization-in-action

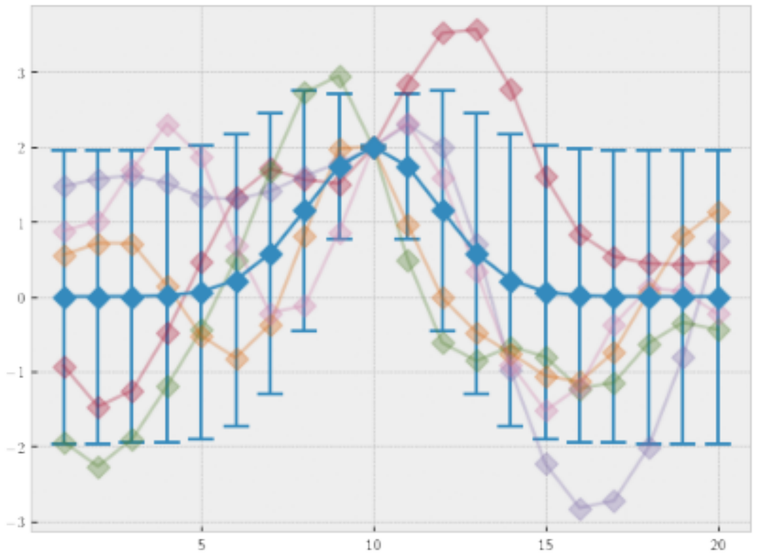


Manning's Bayesian Optimization in Action

a hands-on guide to Python implementation

- https://www.manning.com/books/bayesian-optimization-in-action

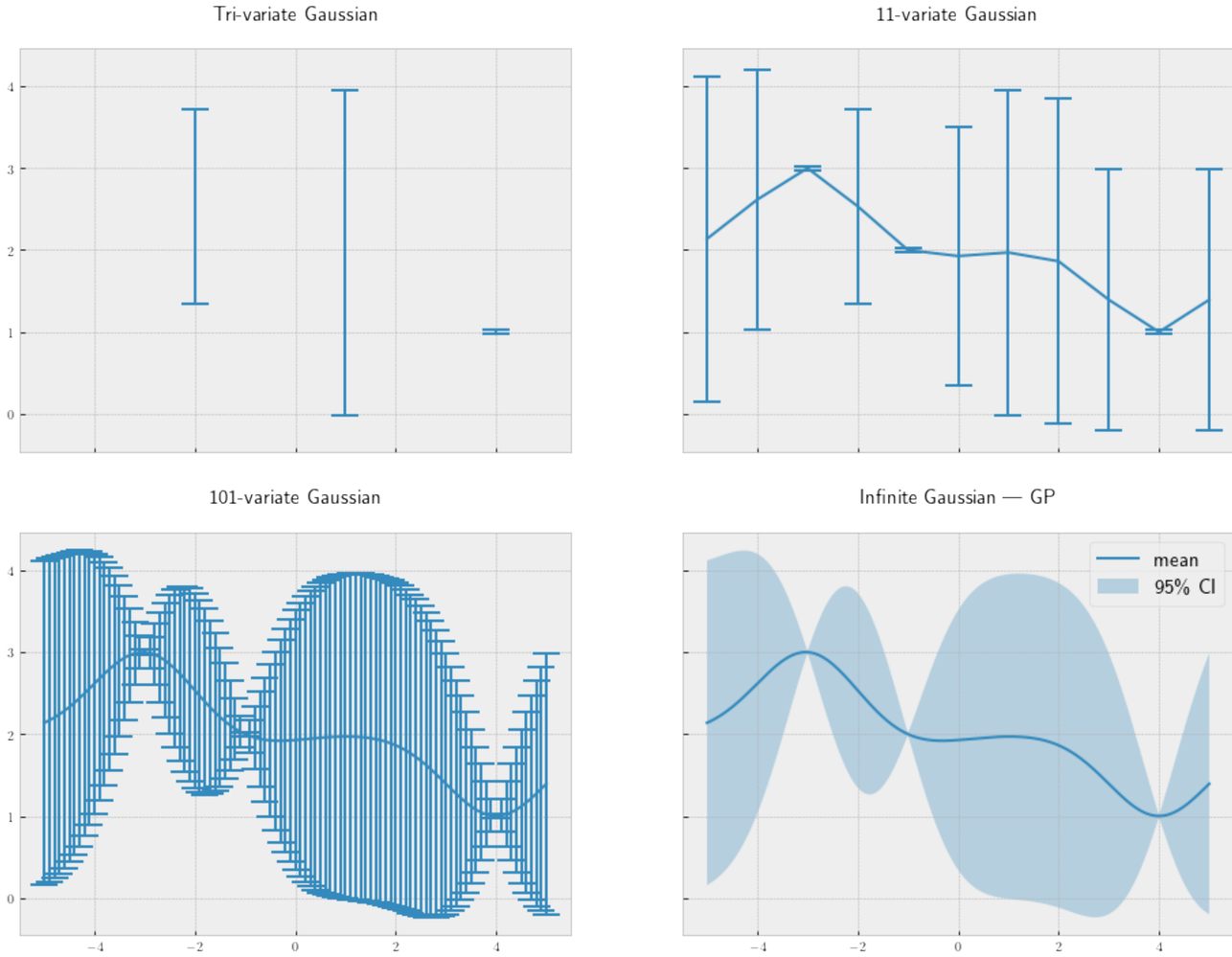




Manning's Bayesian Optimization in Action

a hands-on guide to Python implementation

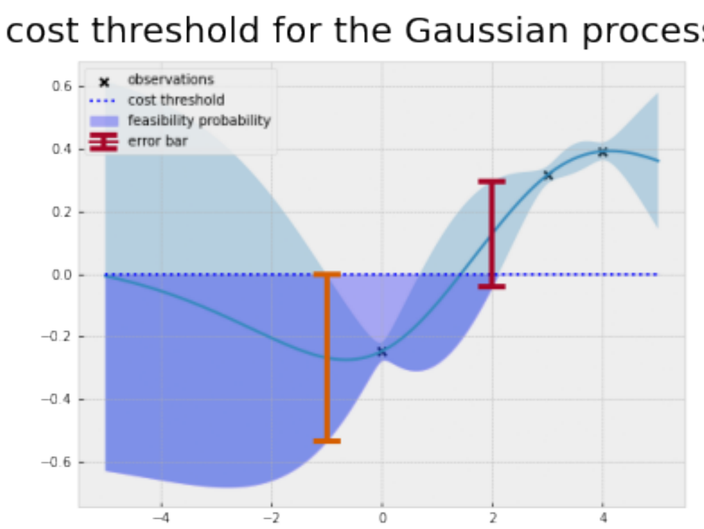
- https://www.manning.com/books/bayesian-optimization-in-action

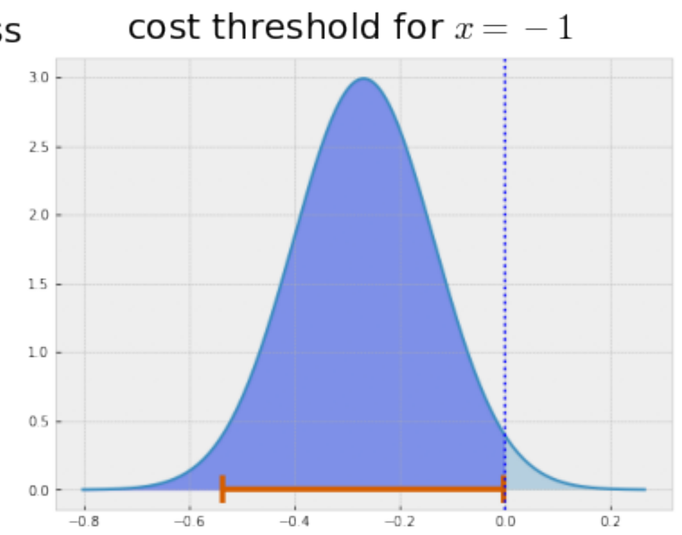


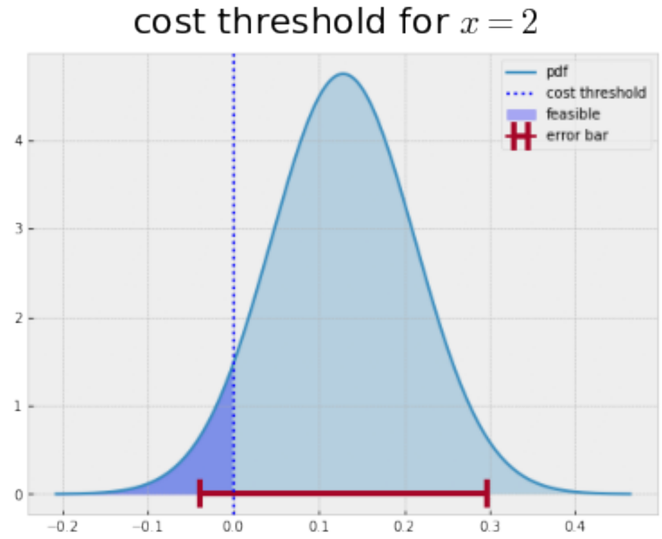
Manning's Bayesian Optimization in Action

a hands-on guide to Python implementation

- https://www.manning.com/books/bayesian-optimization-in-action



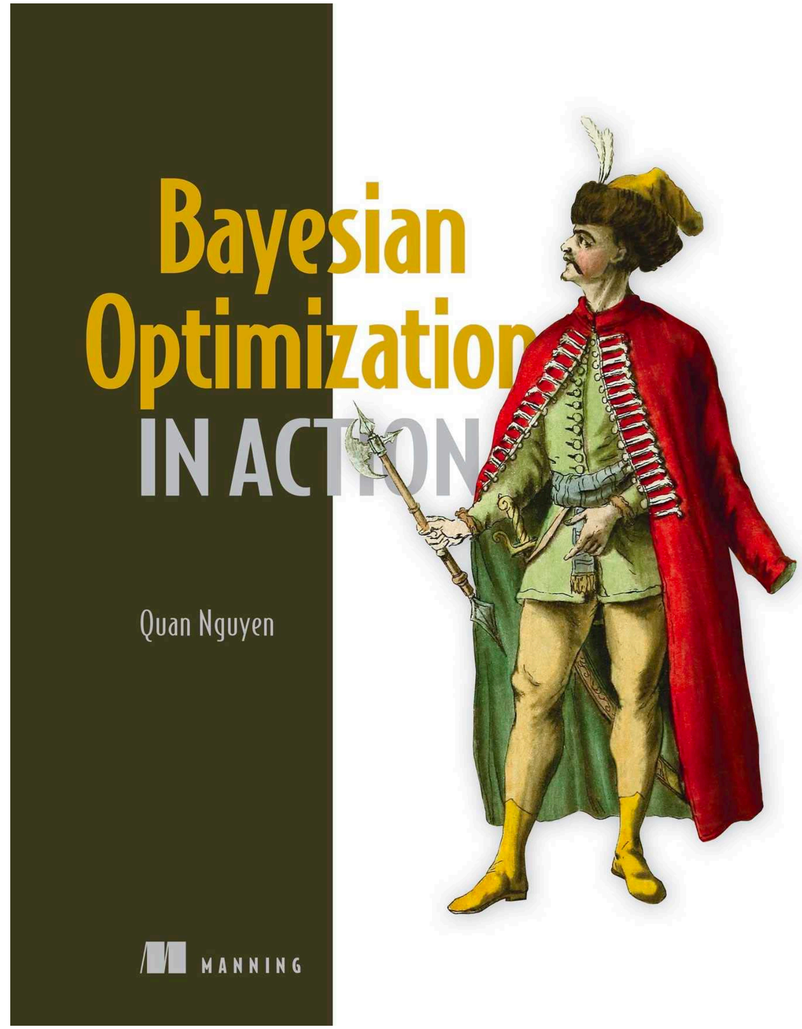




Manning's Bayesian Optimization in Action

a hands-on guide to Python implementation

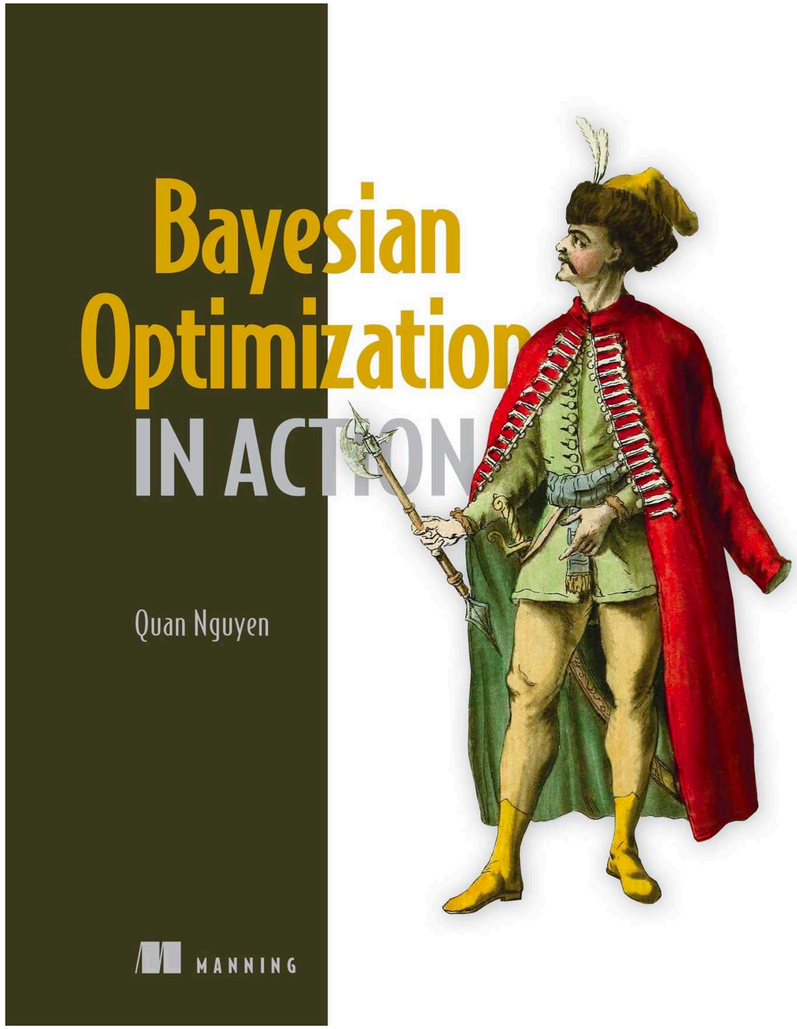
- https://www.manning.com/books/bayesian-optimization-in-action
- 35% discount code: **ctwpydatagl22** via http://mng.bz/7Zqv (QR code below)



Manning's Bayesian Optimization in Action

a hands-on guide to Python implementation

- https://www.manning.com/books/bayesian-optimization-in-action
- 35% discount code: **ctwpydatagl22** via http://mng.bz/7Zqv (QR code below)
- Contact me for a free eBook copy (3 uses) nguyenminhquan135@gmail.com



Bayesian optimization in practice

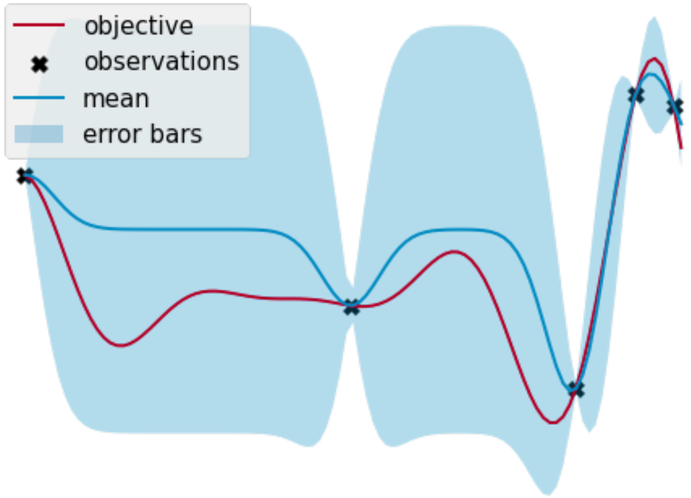
using Bayesian optimization in the real world

Optimization policy for decision making

choosing the best course of action, the Bayesian way

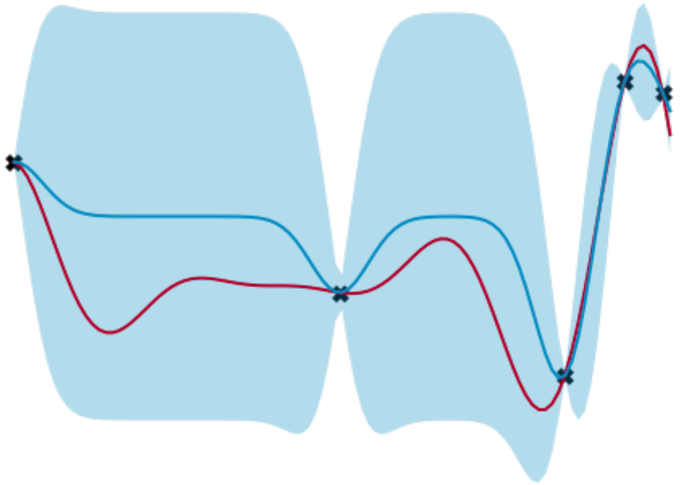
Optimization policy for decision making

choosing the best course of action, the Bayesian way



Optimization policy for decision making

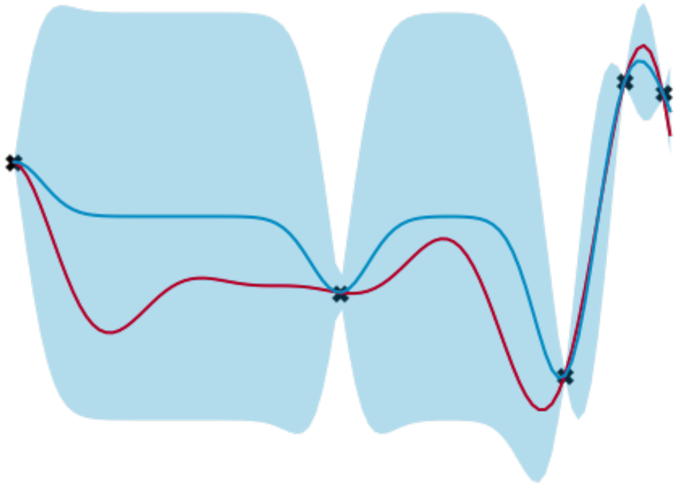
choosing the best course of action, the Bayesian way



Optimization policy for decision making

choosing the best course of action, the Bayesian way

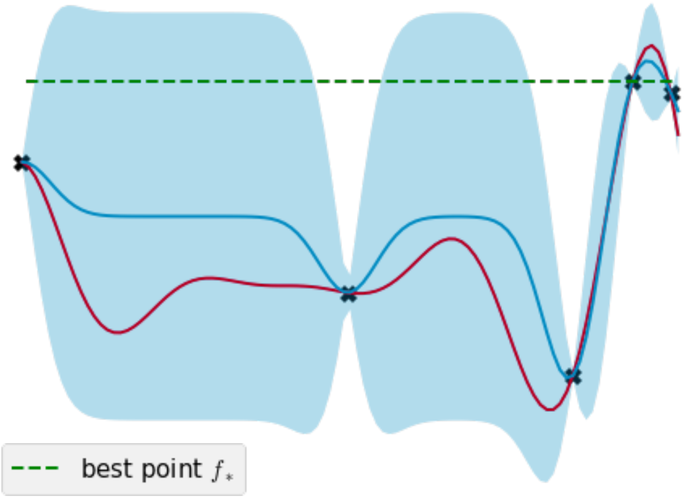
- **Goal**: pick x to improve from the best point f_*



Optimization policy for decision making

choosing the best course of action, the Bayesian way

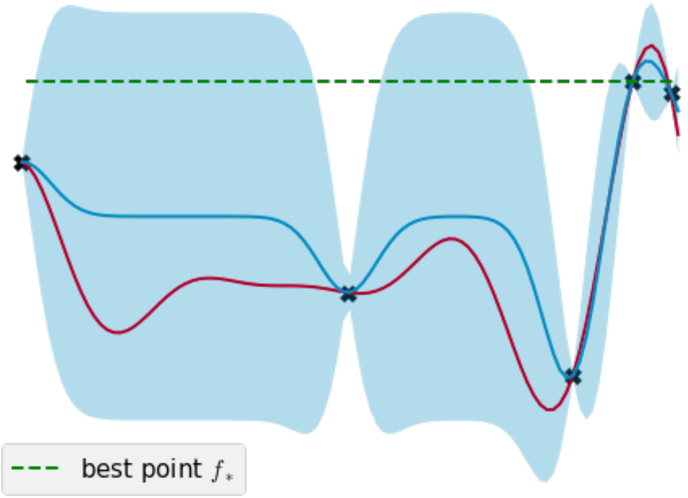
- **Goal**: pick x to improve from the best point f_*



Optimization policy for decision making

choosing the best course of action, the Bayesian way

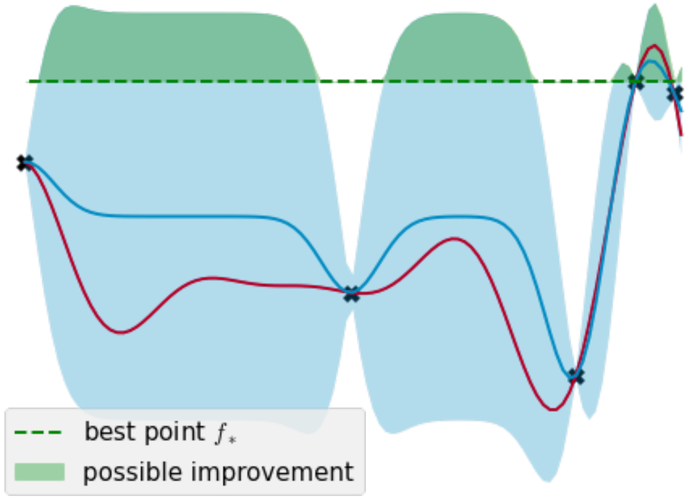
- **Goal**: pick x to improve from the best point f_*
  - Want to maximize $\max\{0, f(x) - f_*\}$



Optimization policy for decision making

choosing the best course of action, the Bayesian way

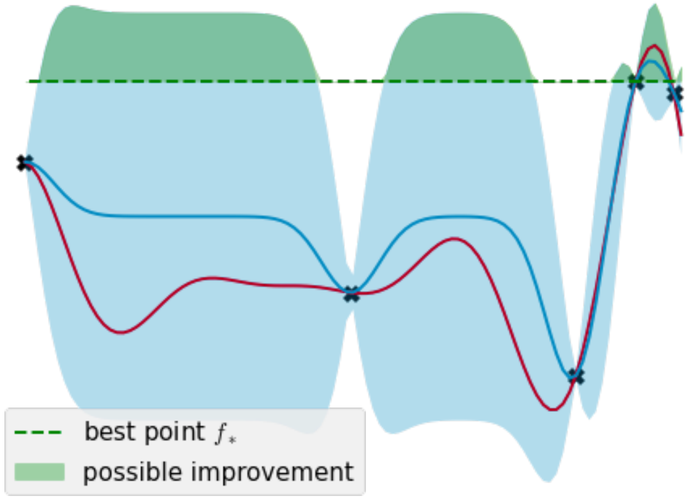
- **Goal**: pick x to improve from the best point f_*
  - Want to maximize $\max\{0, f(x) - f_*\}$



Optimization policy for decision making

choosing the best course of action, the Bayesian way

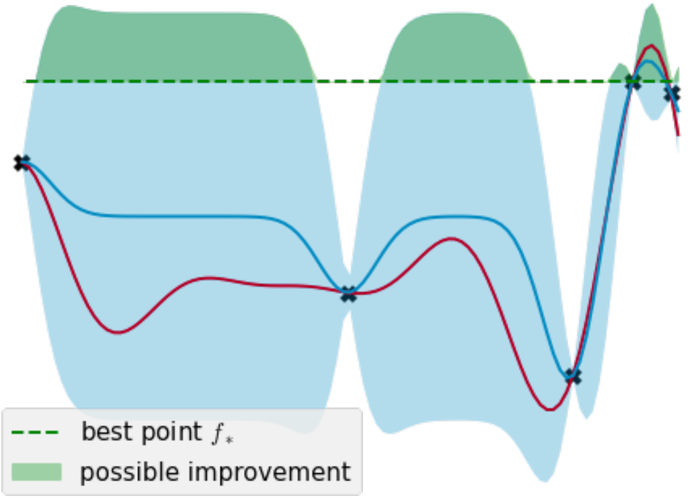
- **Goal**: pick x to improve from the best point f_*
  - Want to maximize $\max\{0, f(x) - f_*\}$
- **Problem**: $f(x)$ is unknown



Optimization policy for decision making

choosing the best course of action, the Bayesian way

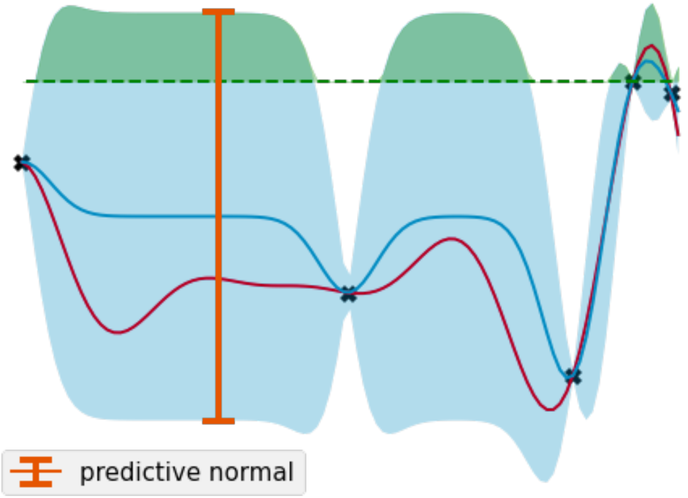
- **Goal**: pick x to improve from the best point f_*
  - Want to maximize $\max\{0, f(x) - f_*\}$
- **Problem**: $f(x)$ is unknown
- **Solution**: $f(x)$ follows a normal distribution



Optimization policy for decision making

choosing the best course of action, the Bayesian way

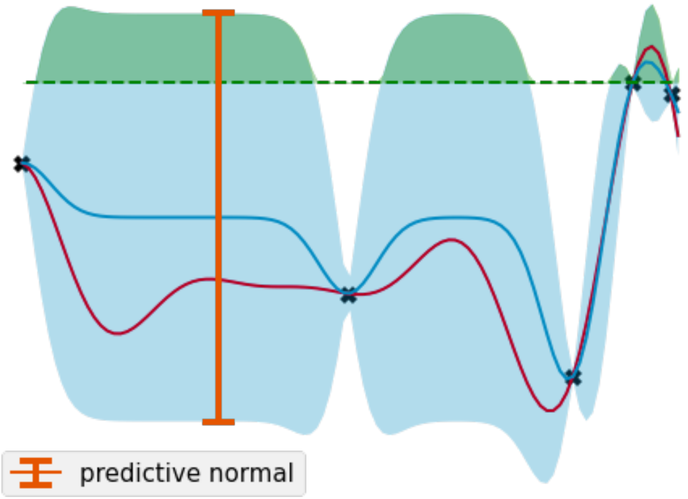
- **Goal**: pick x to improve from the best point f_*
  - Want to maximize $\max\{0, f(x) - f_*\}$
- **Problem**: $f(x)$ is unknown
- **Solution**: $f(x)$ follows a normal distribution



Optimization policy for decision making

choosing the best course of action, the Bayesian way

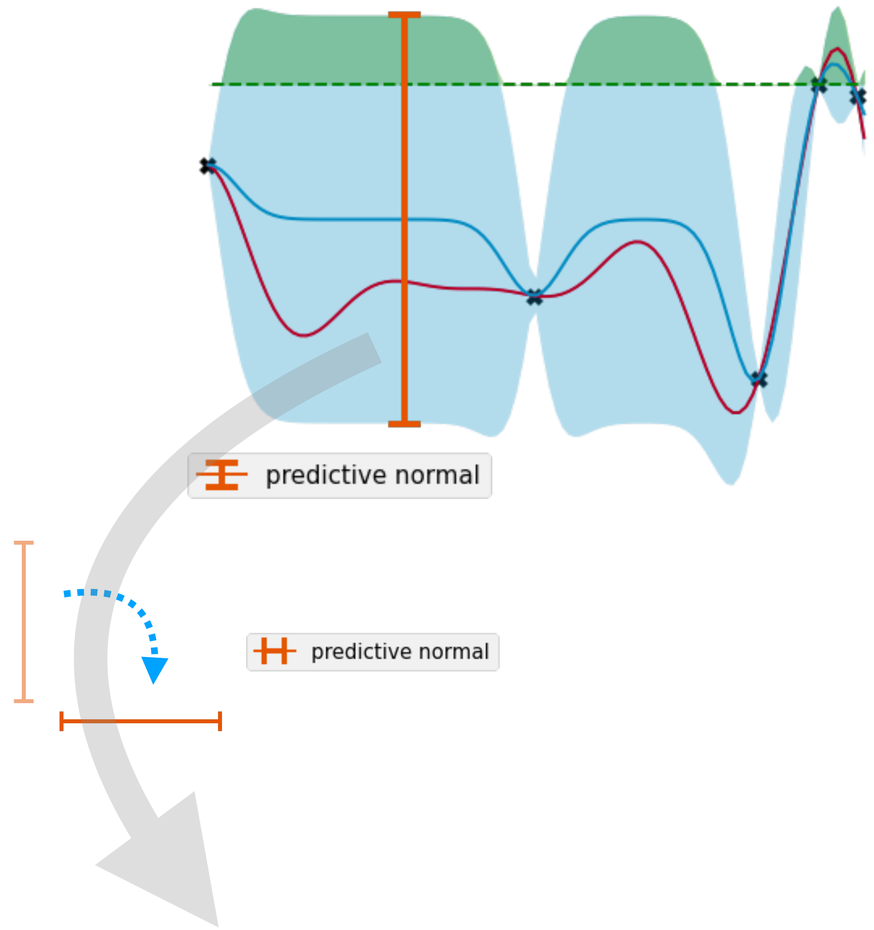
- **Goal**: pick x to improve from the best point f_*
  - Want to maximize $\max\{0, f(x) - f_*\}$
- **Problem**: $f(x)$ is unknown
- **Solution**: $f(x)$ follows a normal distribution
  - Can compute the expected improvement (**EI**) value $\mathbb{E}\left[\max\{0, f(x) - f_*\}\right]$



Optimization policy for decision making

choosing the best course of action, the Bayesian way

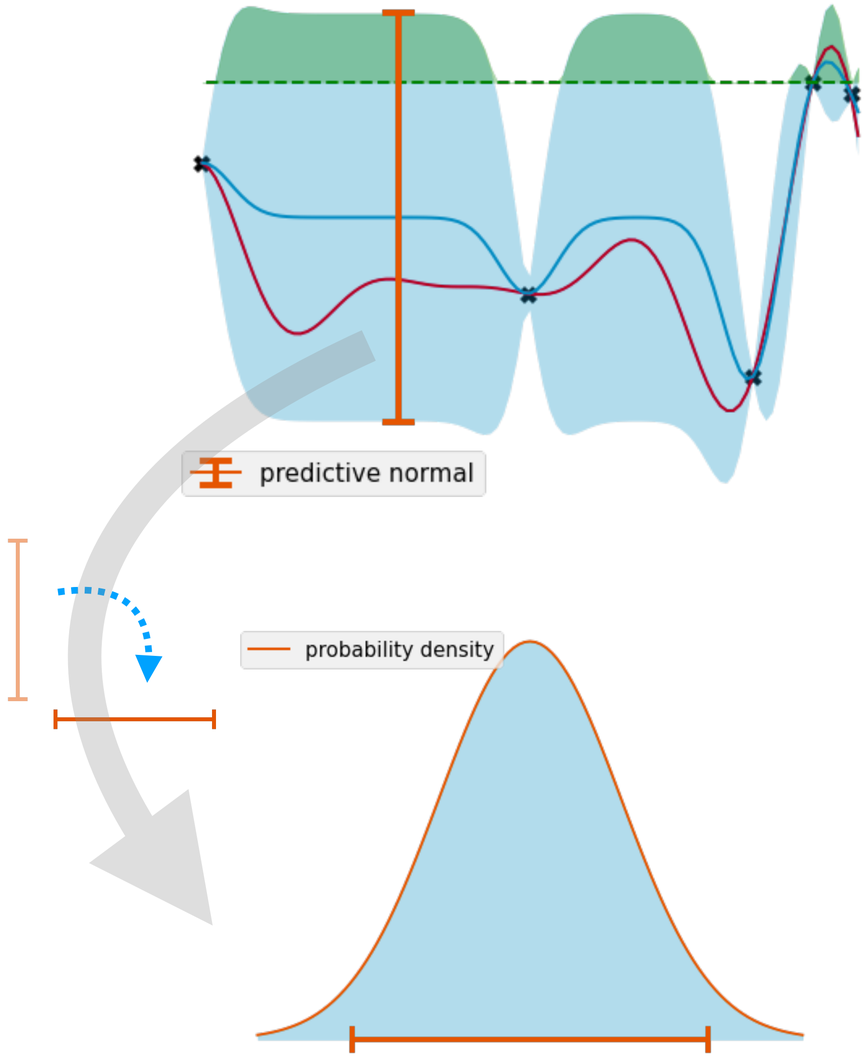
- **Goal**: pick x to improve from the best point f_*
  - Want to maximize $\max\{0, f(x) - f_*\}$
- **Problem**: $f(x)$ is unknown
- **Solution**: $f(x)$ follows a normal distribution
  - Can compute the expected improvement (**EI**) value $\mathbb{E}\left[\max\{0, f(x) - f_*\}\right]$



Optimization policy for decision making

choosing the best course of action, the Bayesian way

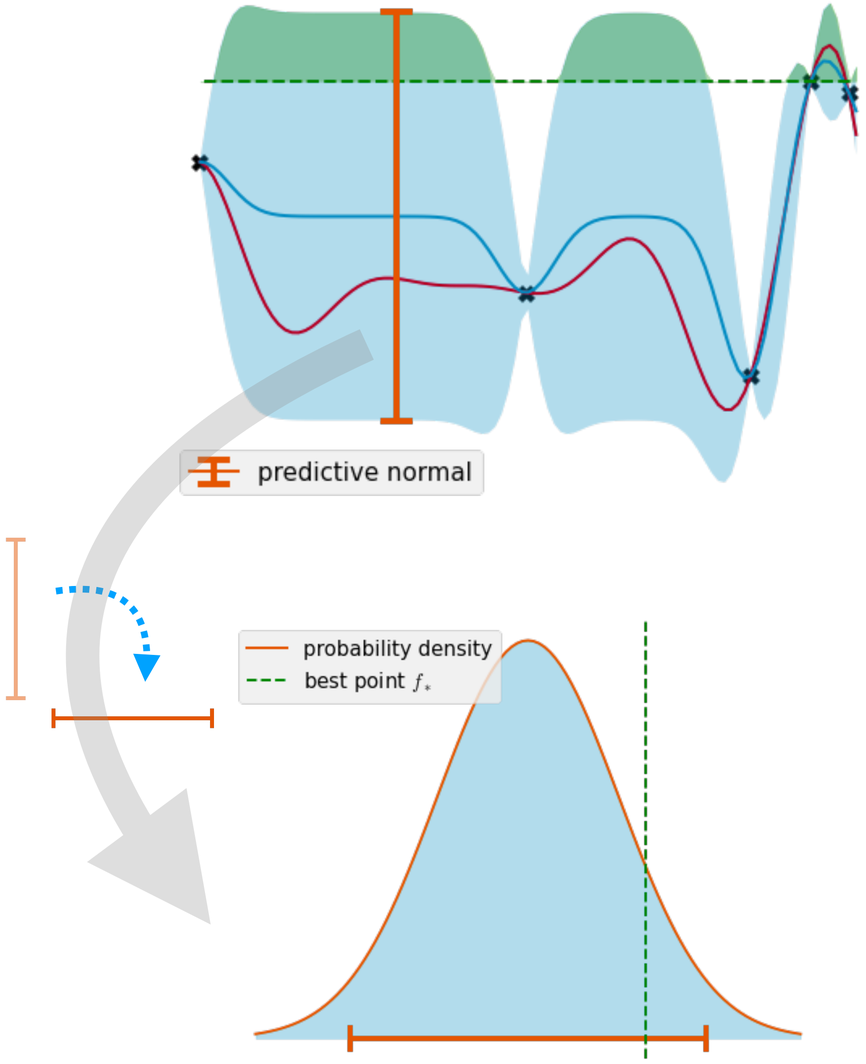
- **Goal**: pick x to improve from the best point f_*
  - Want to maximize $\max\{0, f(x) - f_*\}$
- **Problem**: $f(x)$ is unknown
- **Solution**: $f(x)$ follows a normal distribution
  - Can compute the expected improvement (**EI**) value $\mathbb{E}\left[\max\{0, f(x) - f_*\}\right]$



Optimization policy for decision making

choosing the best course of action, the Bayesian way

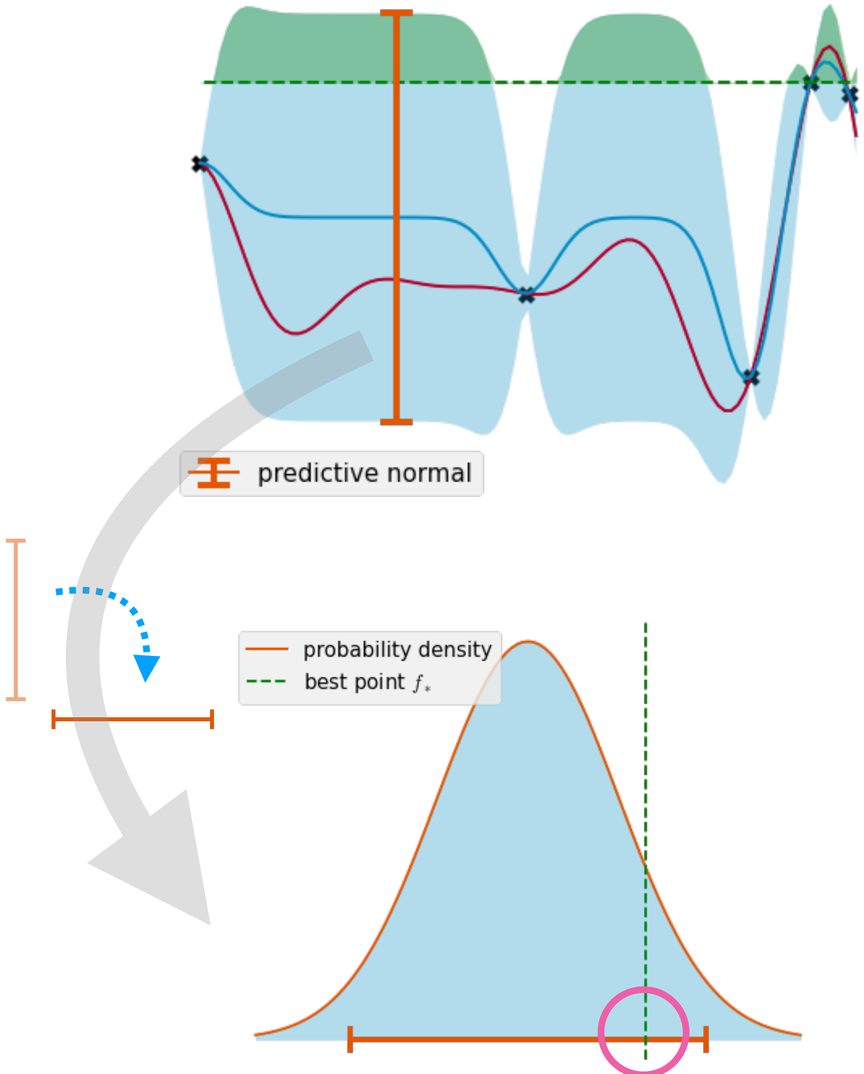
- **Goal**: pick x to improve from the best point f_*
  - Want to maximize $\max\{0, f(x) - f_*\}$
- **Problem**: $f(x)$ is unknown
- **Solution**: $f(x)$ follows a normal distribution
  - Can compute the expected improvement (**EI**) value $\mathbb{E}\left[\max\{0, f(x) - f_*\}\right]$



Optimization policy for decision making

choosing the best course of action, the Bayesian way

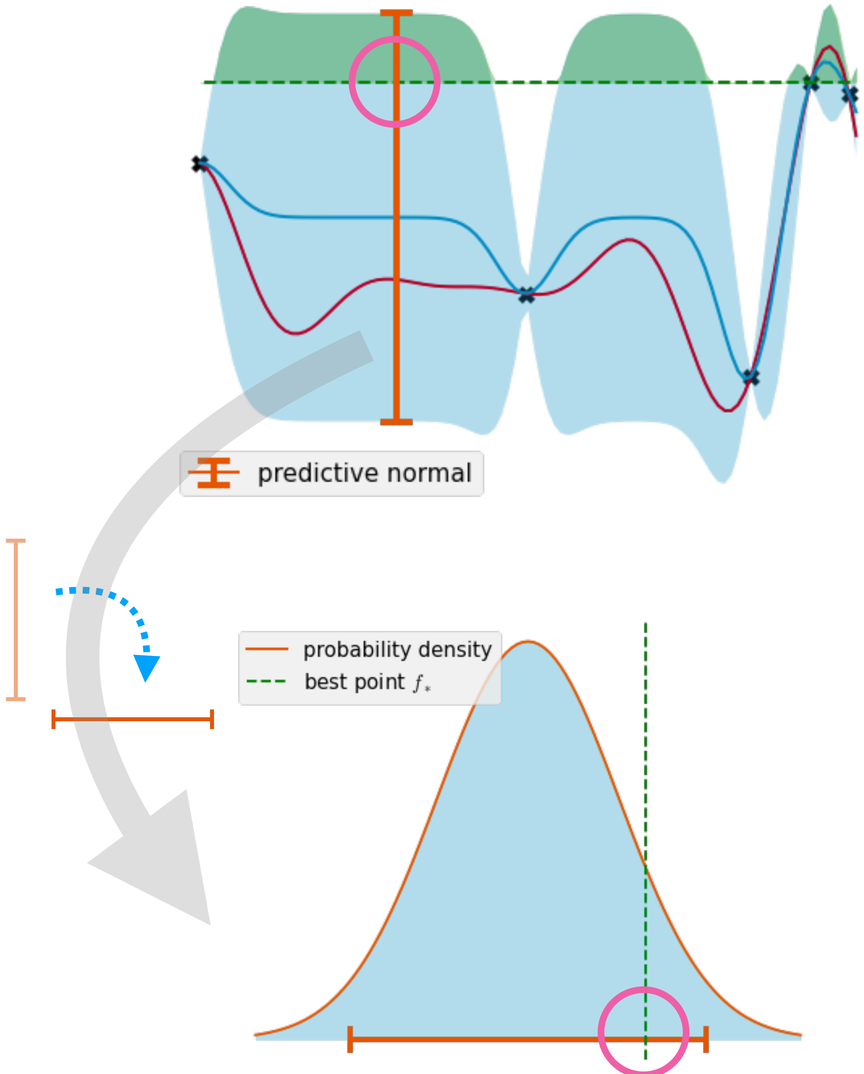
- **Goal**: pick x to improve from the best point f_*
  - Want to maximize $\max\{0, f(x) - f_*\}$
- **Problem**: $f(x)$ is unknown
- **Solution**: $f(x)$ follows a normal distribution
  - Can compute the expected improvement (**EI**) value $\mathbb{E}\left[\max\{0, f(x) - f_*\}\right]$



Optimization policy for decision making

choosing the best course of action, the Bayesian way

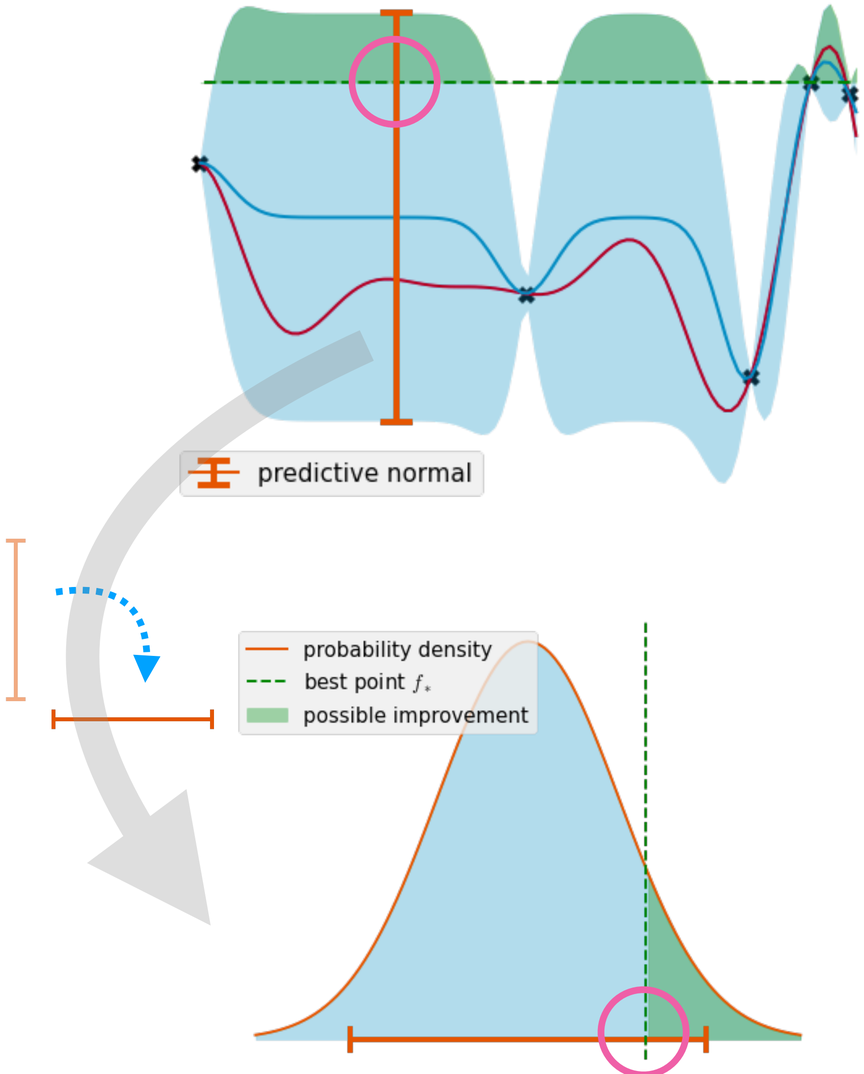
- **Goal**: pick x to improve from the best point f_*
  - Want to maximize $\max\{0, f(x) - f_*\}$
- **Problem**: $f(x)$ is unknown
- **Solution**: $f(x)$ follows a normal distribution
  - Can compute the expected improvement (**EI**) value $\mathbb{E}\left[\max\{0, f(x) - f_*\}\right]$



Optimization policy for decision making

choosing the best course of action, the Bayesian way

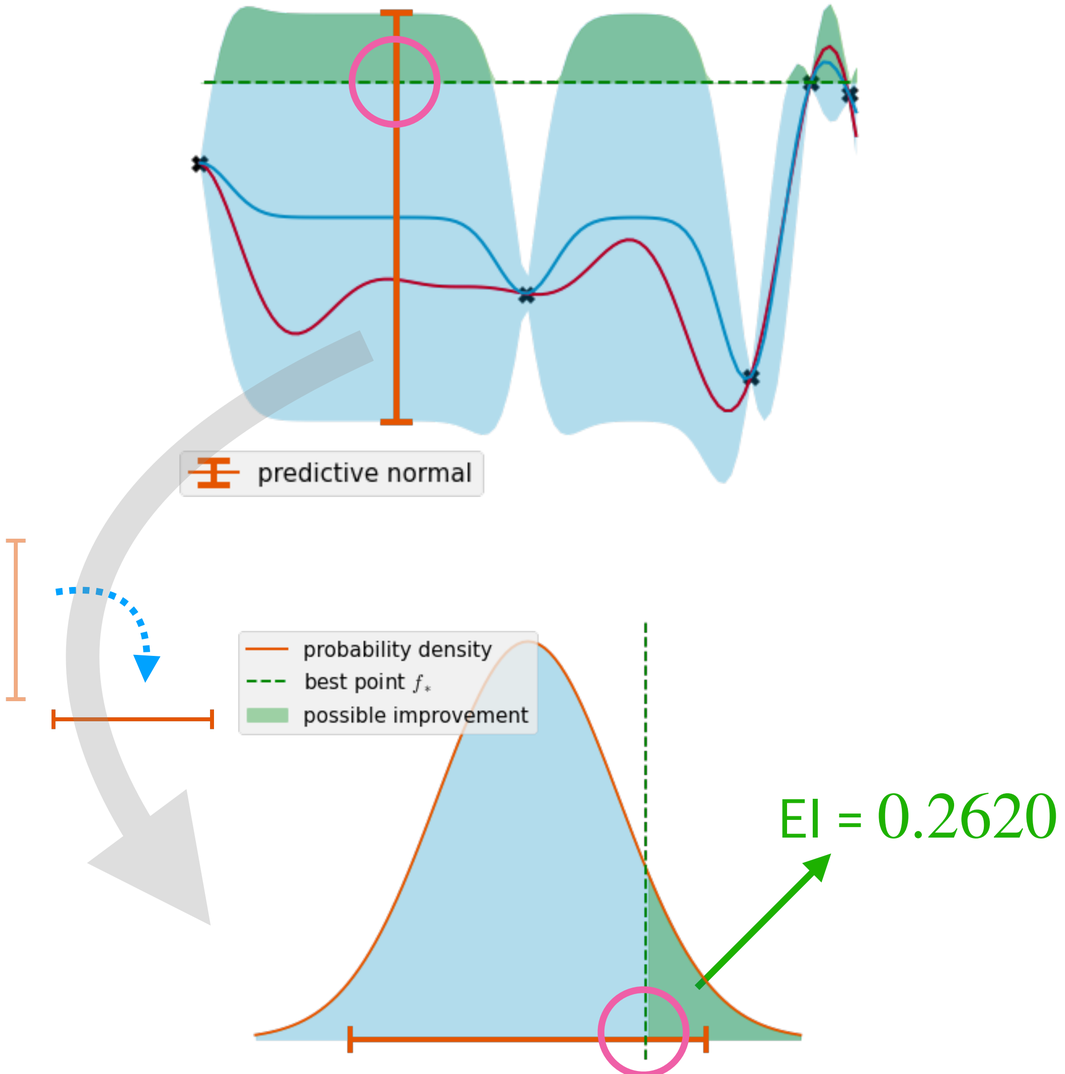
- **Goal**: pick x to improve from the best point f_*
  - Want to maximize $\max\{0, f(x) - f_*\}$
- **Problem**: $f(x)$ is unknown
- **Solution**: $f(x)$ follows a normal distribution
  - Can compute the expected improvement (**EI**) value $\mathbb{E}\left[\max\{0, f(x) - f_*\}\right]$



Optimization policy for decision making

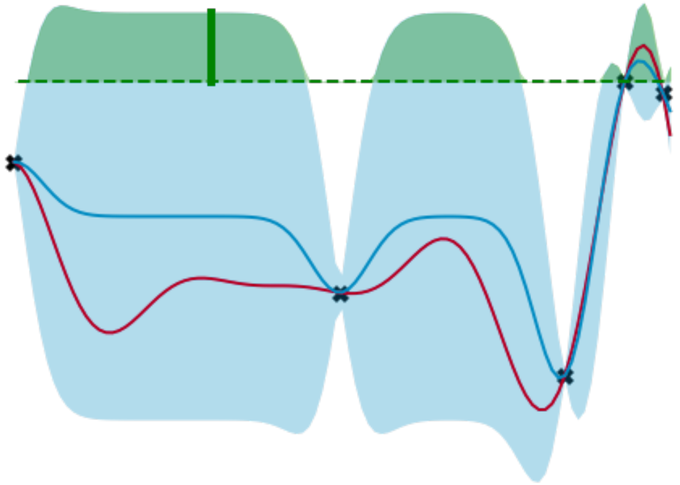
choosing the best course of action, the Bayesian way

- **Goal**: pick x to improve from the best point f_*
  - Want to maximize $\max\{0, f(x) - f_*\}$
- **Problem**: $f(x)$ is unknown
- **Solution**: $f(x)$ follows a normal distribution
  - Can compute the expected improvement (**EI**) value $\mathbb{E}\left[\max\{0, f(x) - f_*\}\right]$



Optimization policy for decision making

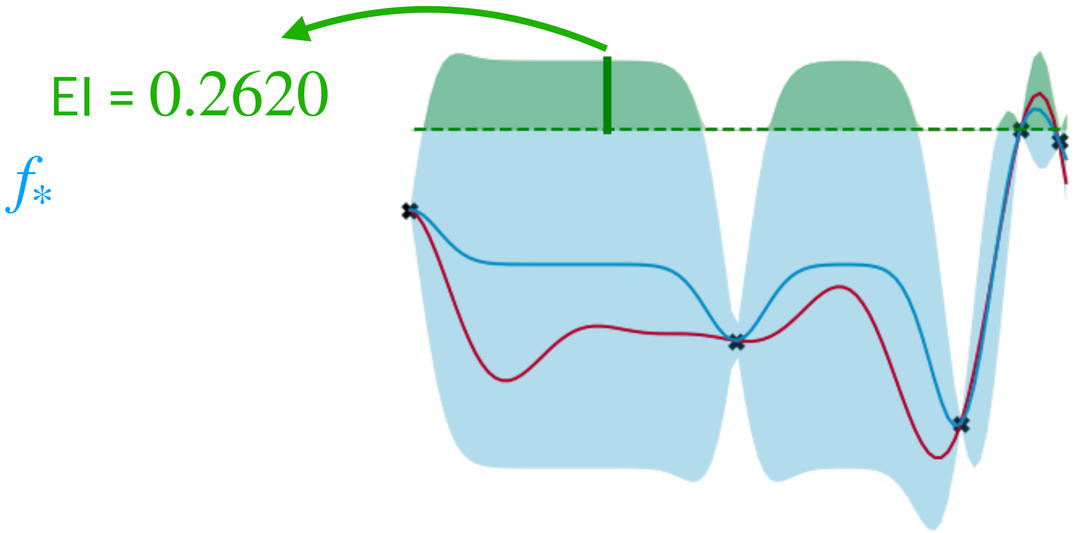
choosing the best course of action, the Bayesian way



- **Goal**: pick x to improve from the best point f_*
  - Want to maximize $\max\{0, f(x) - f_*\}$
- **Problem**: $f(x)$ is unknown
- **Solution**: $f(x)$ follows a normal distribution
  - Can compute the expected improvement (**EI**) value $\mathbb{E}\left[\max\{0, f(x) - f_*\}\right]$

Optimization policy for decision making

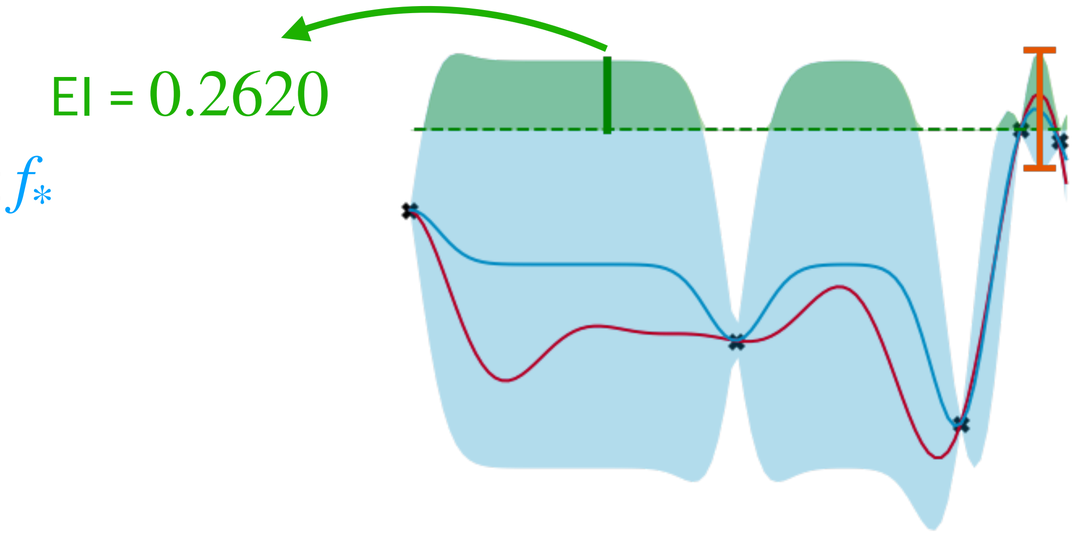
choosing the best course of action, the Bayesian way



- **Goal**: pick x to improve from the best point f_*
  - Want to maximize $\max\{0, f(x) - f_*\}$
- **Problem**: $f(x)$ is unknown
- **Solution**: $f(x)$ follows a normal distribution
  - Can compute the expected improvement (**EI**) value $\mathbb{E}\left[\max\{0, f(x) - f_*\}\right]$

Optimization policy for decision making

choosing the best course of action, the Bayesian way

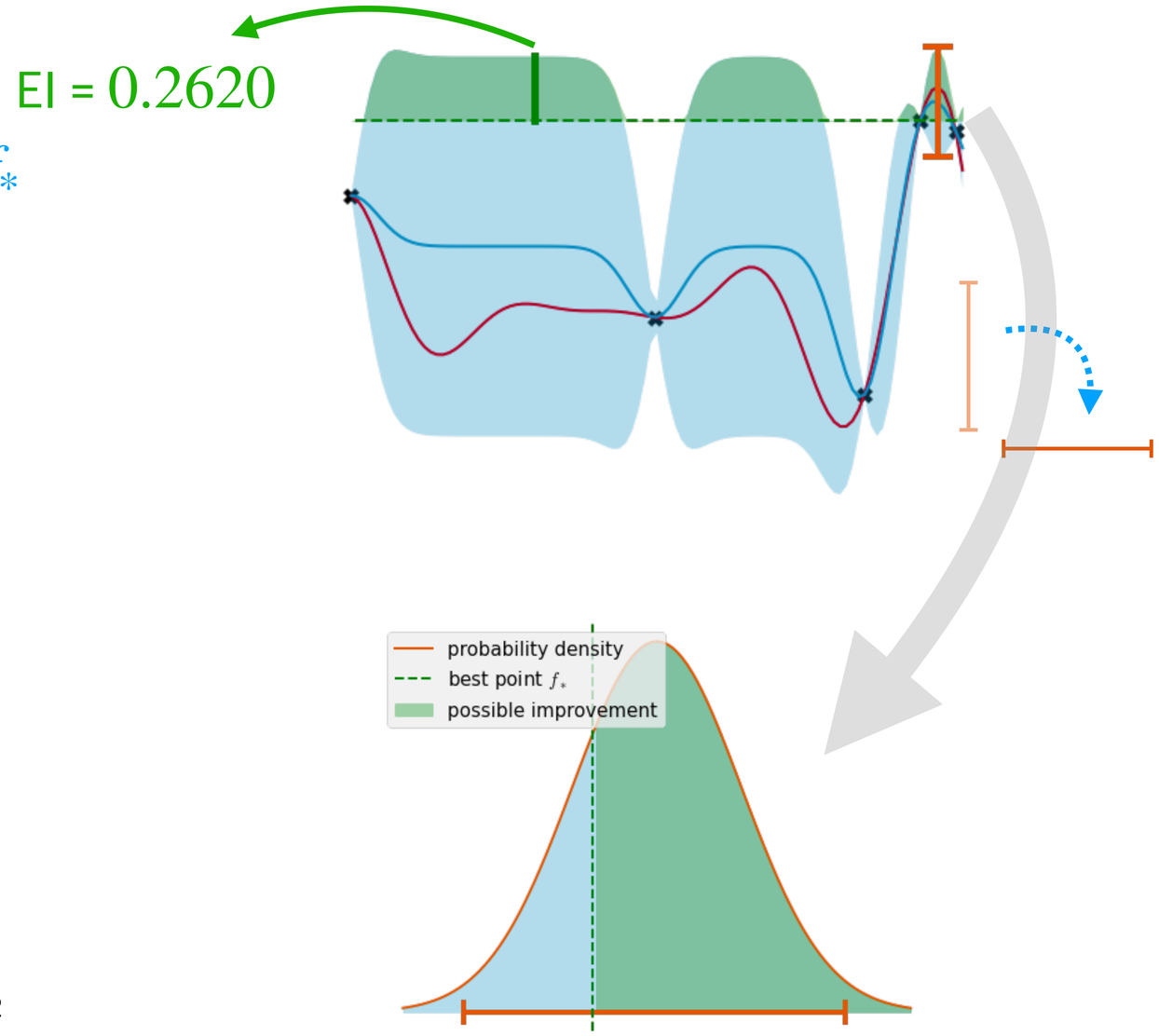


- **Goal**: pick x to improve from the best point f_*
  - Want to maximize $\max\{0, f(x) - f_*\}$
- **Problem**: $f(x)$ is unknown
- **Solution**: $f(x)$ follows a normal distribution
  - Can compute the expected improvement (**EI**) value $\mathbb{E}\left[\max\{0, f(x) - f_*\}\right]$

Optimization policy for decision making

choosing the best course of action, the Bayesian way

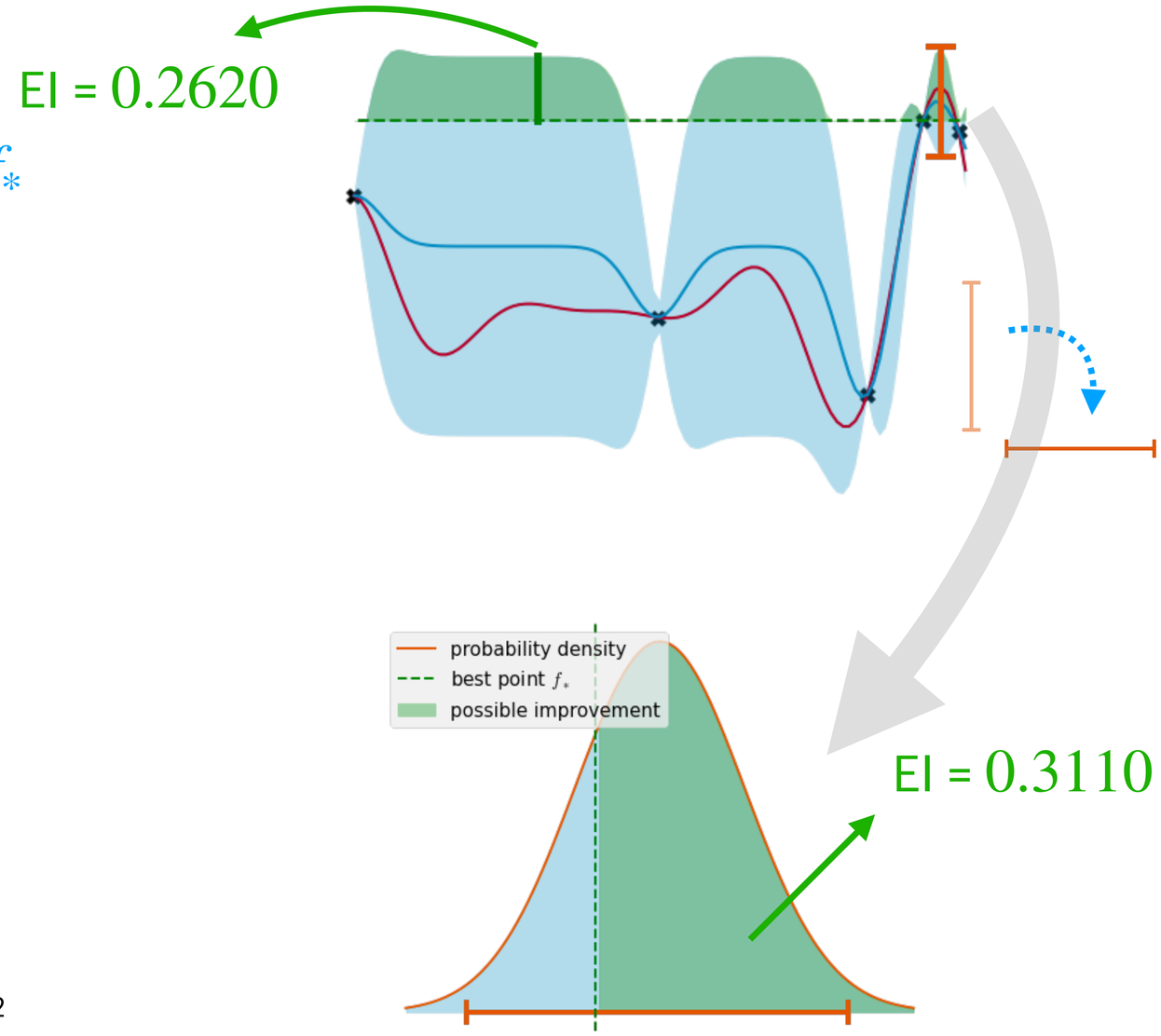
- **Goal**: pick x to improve from the best point f_*
  - Want to maximize $\max\{0, f(x) - f_*\}$
- **Problem**: $f(x)$ is unknown
- **Solution**: $f(x)$ follows a normal distribution
  - Can compute the expected improvement (**EI**) value $\mathbb{E}\left[\max\{0, f(x) - f_*\}\right]$



Optimization policy for decision making

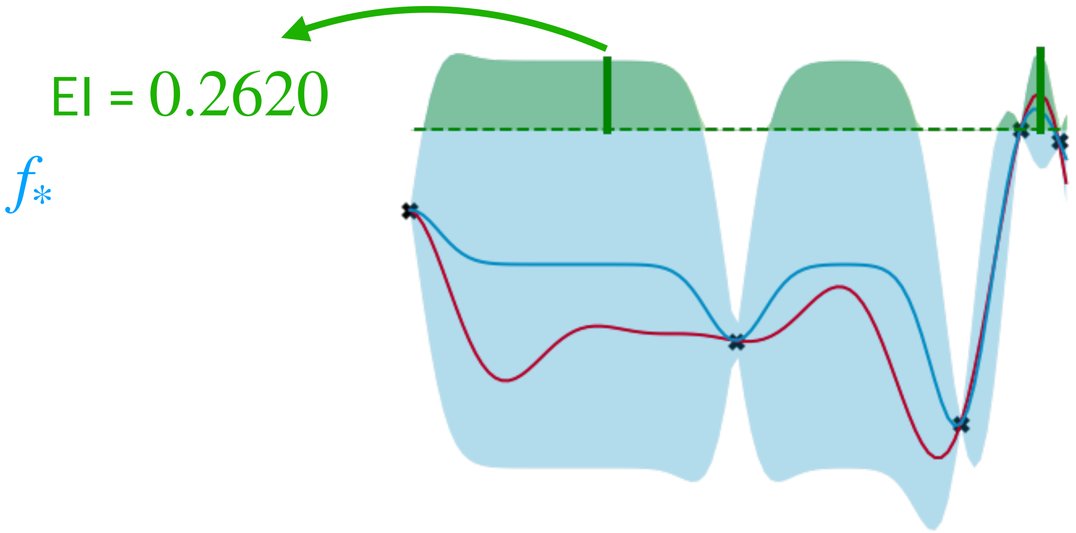
choosing the best course of action, the Bayesian way

- **Goal**: pick x to improve from the best point f_*
  - Want to maximize $\max\{0, f(x) - f_*\}$
- **Problem**: $f(x)$ is unknown
- **Solution**: $f(x)$ follows a normal distribution
  - Can compute the expected improvement (**EI**) value $\mathbb{E}\left[\max\{0, f(x) - f_*\}\right]$



Optimization policy for decision making

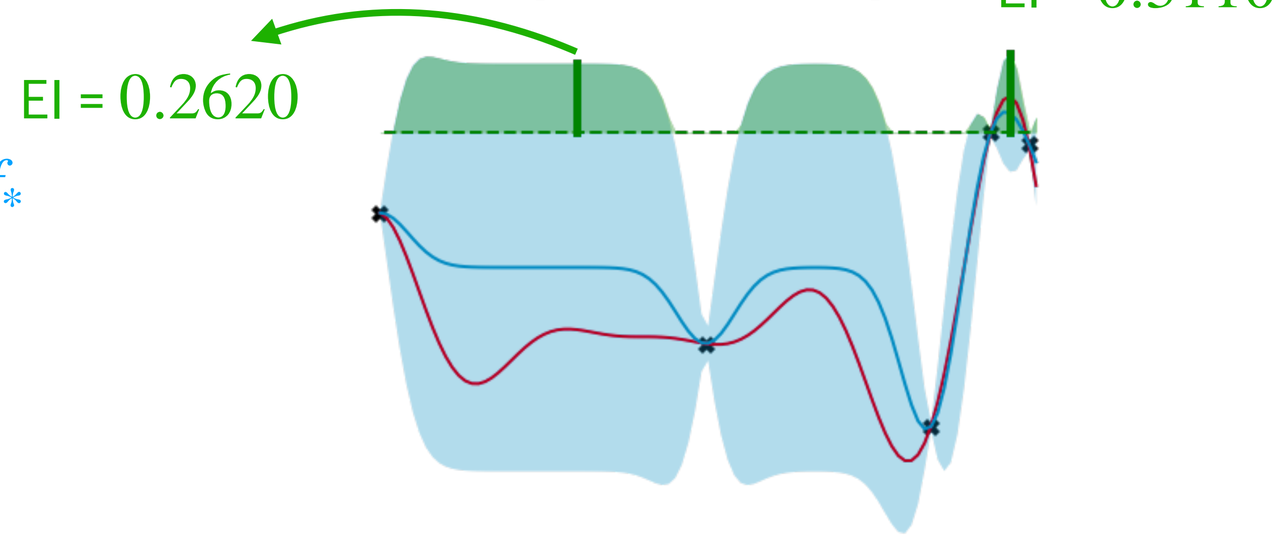
choosing the best course of action, the Bayesian way



- **Goal**: pick x to improve from the best point f_*
  - Want to maximize $\max\{0, f(x) - f_*\}$
- **Problem**: $f(x)$ is unknown
- **Solution**: $f(x)$ follows a normal distribution
  - Can compute the expected improvement (**EI**) value $\mathbb{E}\left[\max\{0, f(x) - f_*\}\right]$

Optimization policy for decision making

choosing the best course of action, the Bayesian way



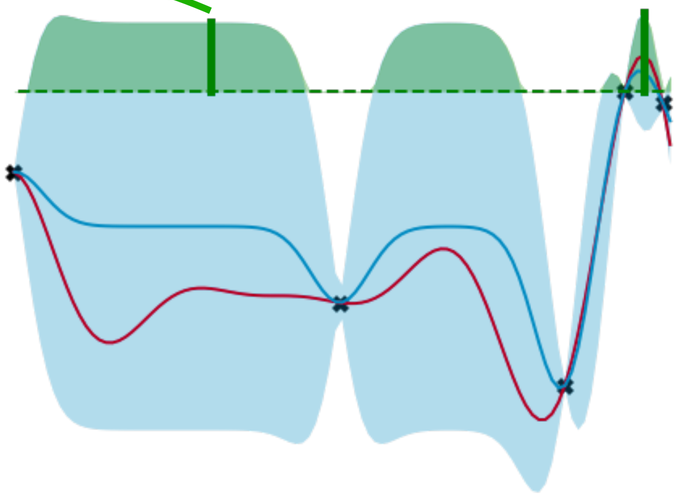
- **Goal**: pick x to improve from the best point f_*
  - Want to maximize $\max\{0, f(x) - f_*\}$
- **Problem**: $f(x)$ is unknown
- **Solution**: $f(x)$ follows a normal distribution
  - Can compute the expected improvement (**EI**) value $\mathbb{E}\left[\max\{0, f(x) - f_*\}\right]$

Optimization policy for decision making

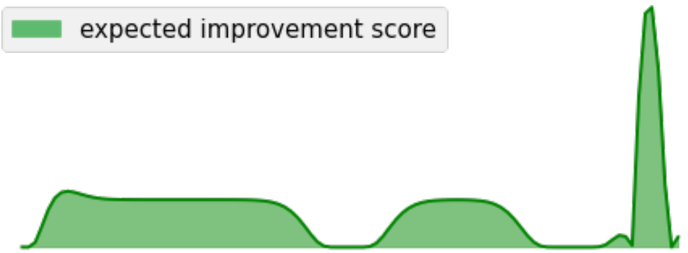
choosing the best course of action, the Bayesian way

EI = 0.3110

EI = 0.2620



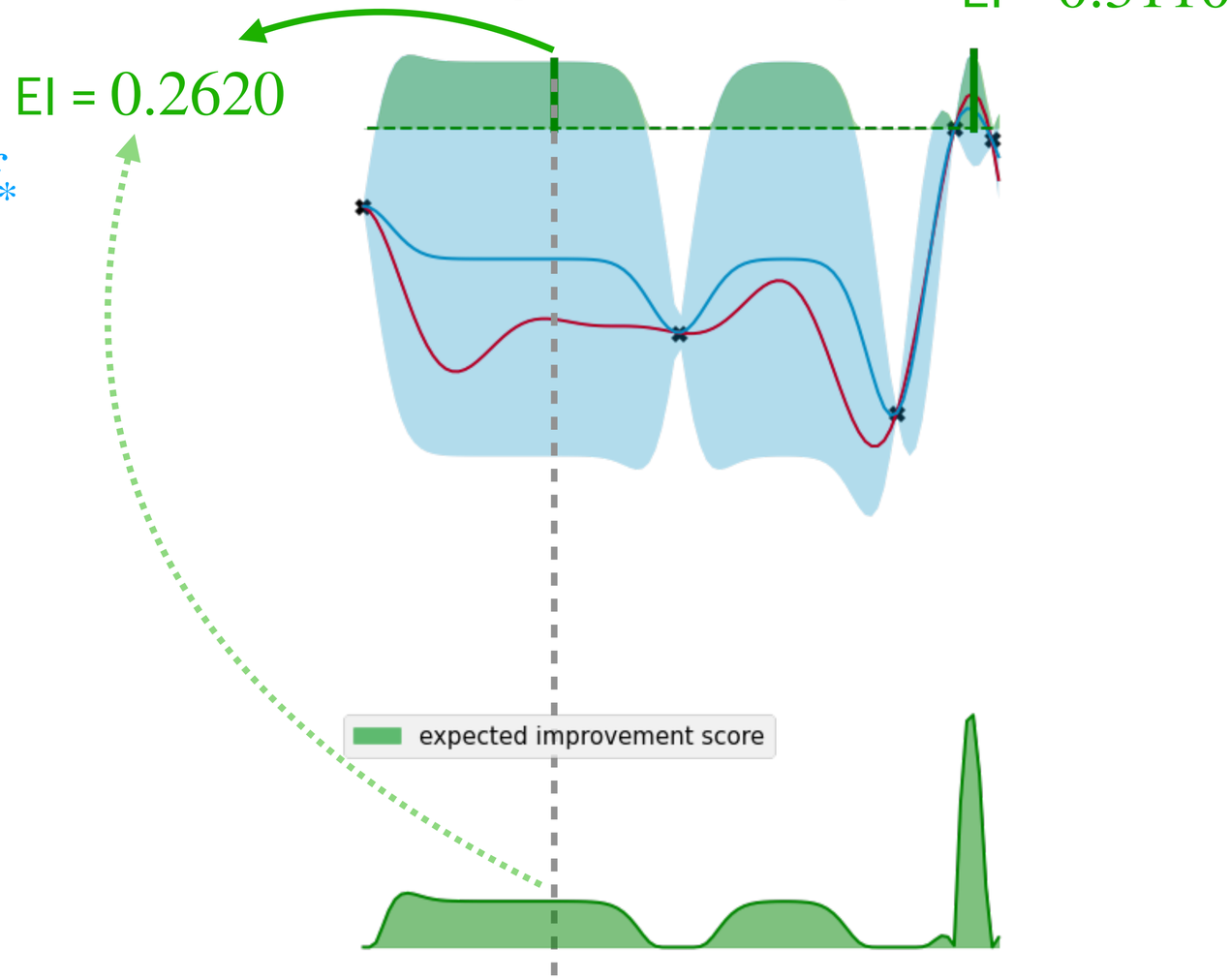
- **Goal**: pick x to improve from the best point f_*
  - Want to maximize $\max\{0, f(x) - f_*\}$
- **Problem**: $f(x)$ is unknown
- **Solution**: $f(x)$ follows a normal distribution
  - Can compute the expected improvement (**EI**) value $\mathbb{E}\left[\max\{0, f(x) - f_*\}\right]$



Optimization policy for decision making

choosing the best course of action, the Bayesian way

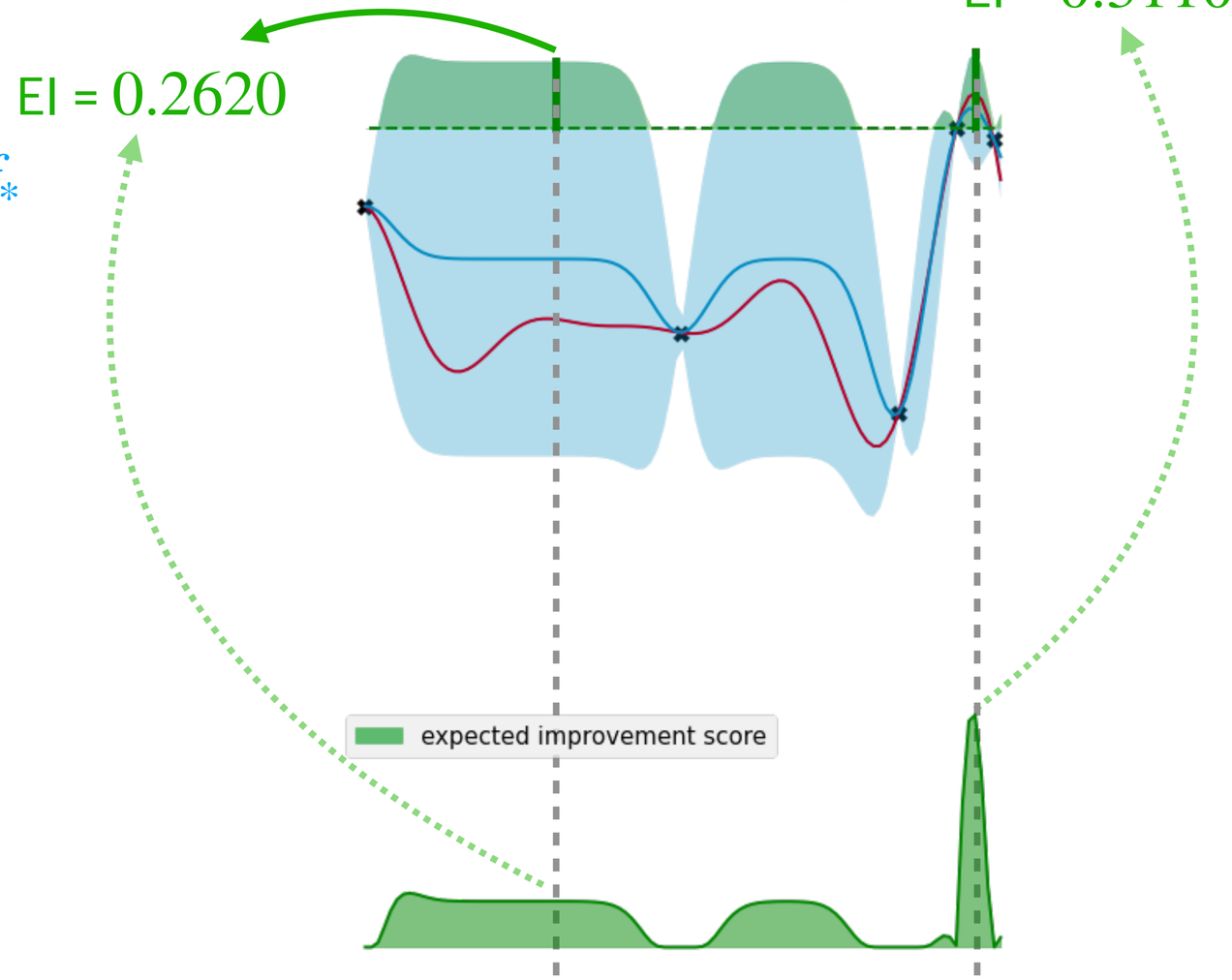
- **Goal**: pick x to improve from the best point f_*
  - Want to maximize $\max\{0, f(x) - f_*\}$
- **Problem**: $f(x)$ is unknown
- **Solution**: $f(x)$ follows a normal distribution
  - Can compute the expected improvement (**EI**) value $\mathbb{E}\left[\max\{0, f(x) - f_*\}\right]$



Optimization policy for decision making

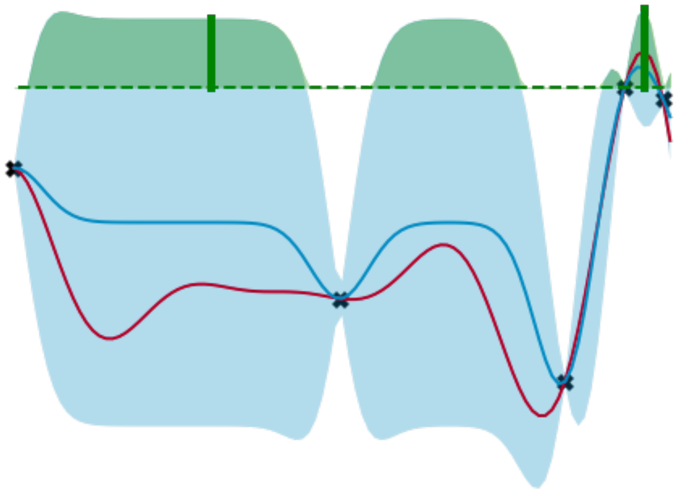
choosing the best course of action, the Bayesian way

- **Goal**: pick x to improve from the best point f_*
  - Want to maximize $\max\{0, f(x) - f_*\}$
- **Problem**: $f(x)$ is unknown
- **Solution**: $f(x)$ follows a normal distribution
  - Can compute the expected improvement (**EI**) value $\mathbb{E}\left[\max\{0, f(x) - f_*\}\right]$

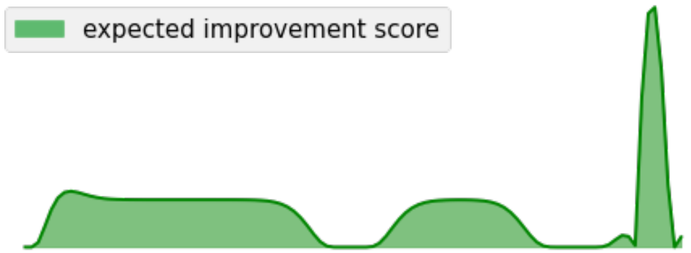


Optimization policy for decision making

choosing the best course of action, the Bayesian way

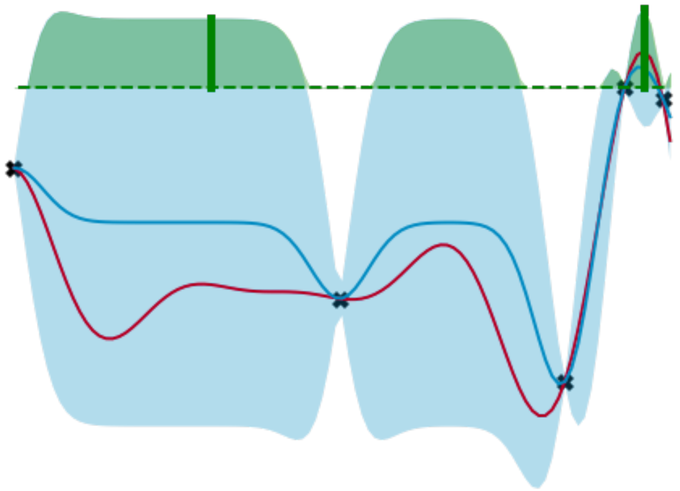


- **Goal**: pick x to improve from the best point f_*
  - Want to maximize $\max\{0, f(x) - f_*\}$
- **Problem**: $f(x)$ is unknown
- **Solution**: $f(x)$ follows a normal distribution
  - Can compute the expected improvement (**EI**) value $\mathbb{E}\left[\max\{0, f(x) - f_*\}\right]$

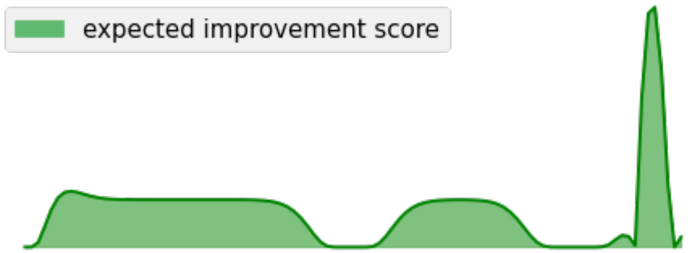


Optimization policy for decision making

choosing the best course of action, the Bayesian way

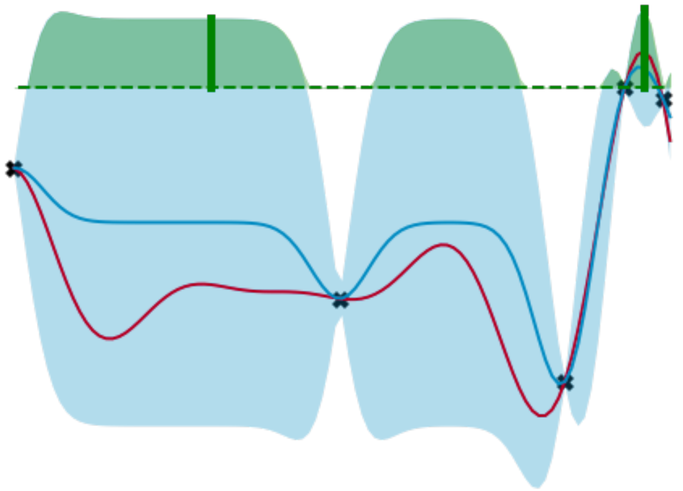


- **Goal**: pick x to improve from the best point f_*
  - Want to maximize $\max\{0, f(x) - f_*\}$
- **Problem**: $f(x)$ is unknown
- **Solution**: $f(x)$ follows a normal distribution
  - Can compute the expected improvement (**EI**) value $\mathbb{E}\left[\max\{0, f(x) - f_*\}\right]$
    - Balances large *means (exploit)* and *standard deviations (explore)*

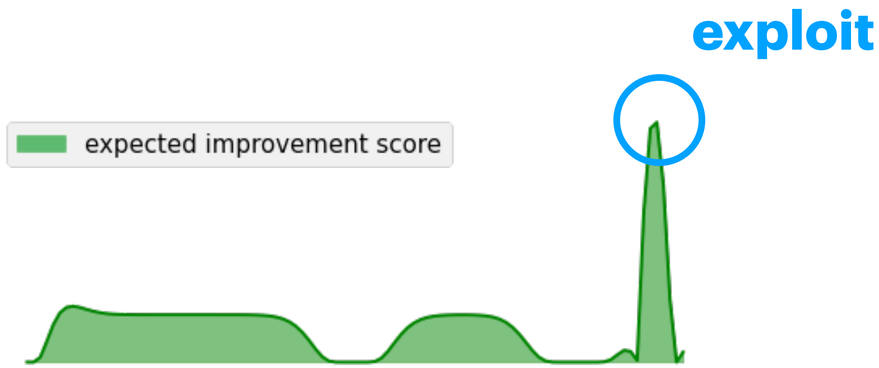


Optimization policy for decision making

choosing the best course of action, the Bayesian way

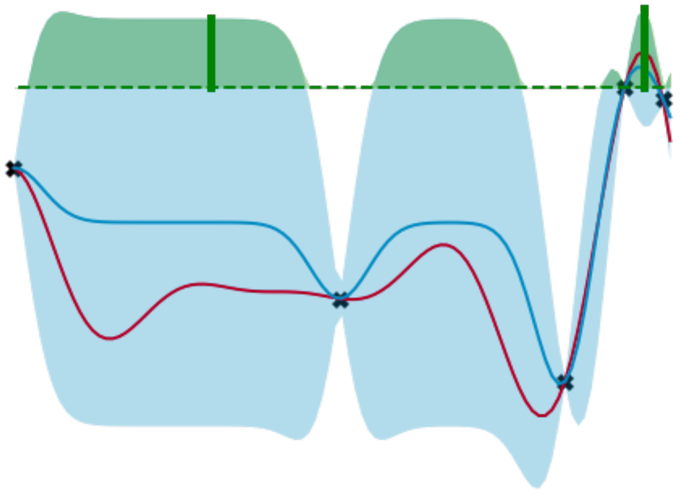


- **Goal**: pick x to improve from the best point f_*
  - Want to maximize $\max\{0, f(x) - f_*\}$
- **Problem**: $f(x)$ is unknown
- **Solution**: $f(x)$ follows a normal distribution
  - Can compute the expected improvement (**EI**) value $\mathbb{E}\left[\max\{0, f(x) - f_*\}\right]$
    - Balances large *means (exploit)* and *standard deviations (explore)*

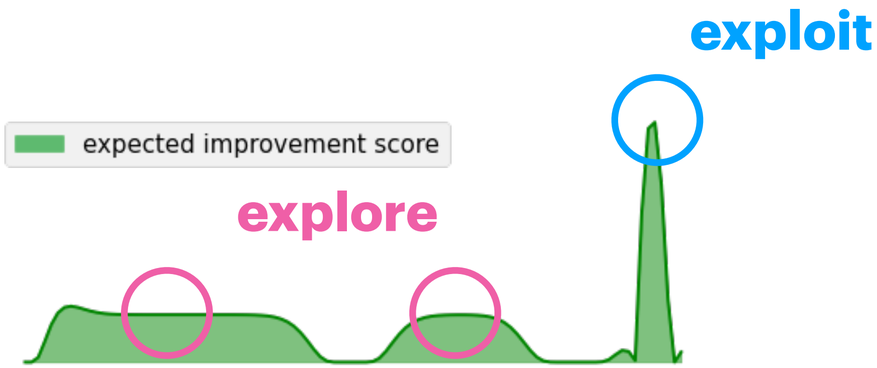


Optimization policy for decision making

choosing the best course of action, the Bayesian way

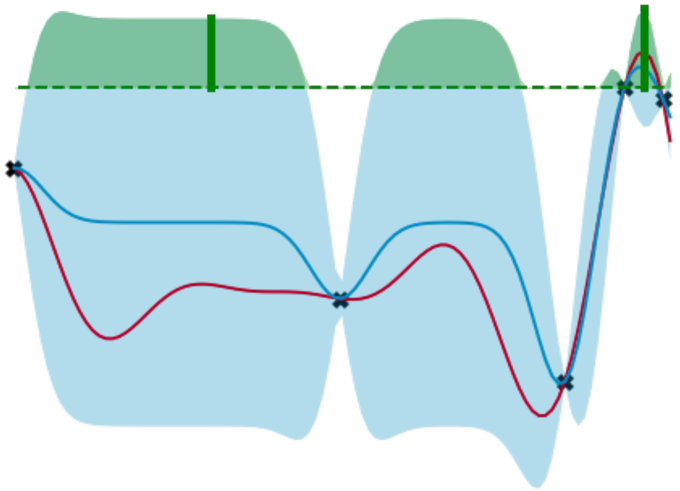


- **Goal**: pick x to improve from the best point f_*
  - Want to maximize $\max\{0, f(x) - f_*\}$
- **Problem**: $f(x)$ is unknown
- **Solution**: $f(x)$ follows a normal distribution
  - Can compute the expected improvement (**EI**) value $\mathbb{E}\left[\max\{0, f(x) - f_*\}\right]$
    - Balances large *means (exploit)* and *standard deviations (explore)*

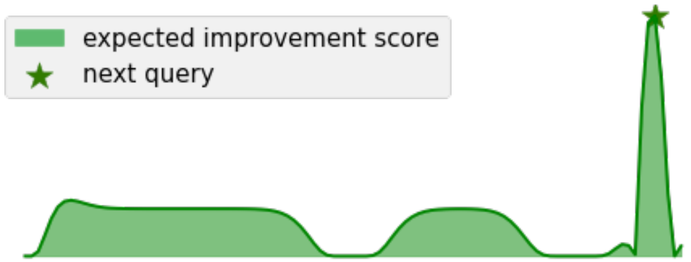


Optimization policy for decision making

choosing the best course of action, the Bayesian way



- **Goal**: pick x to improve from the best point f_*
  - Want to maximize $\max\{0, f(x) - f_*\}$
- **Problem**: $f(x)$ is unknown
- **Solution**: $f(x)$ follows a normal distribution
  - Can compute the expected improvement (**EI**) value $\mathbb{E}\left[\max\{0, f(x) - f_*\}\right]$
    - Balances large *means (exploit)* and *standard deviations (explore)*



Batch optimization to maximize throughput

making multiple queries at the same time

Batch optimization to maximize throughput

making multiple queries at the same time

- Setting allows evaluating many points simultaneously

Batch optimization to maximize throughput

making multiple queries at the same time

- Setting allows evaluating many points simultaneously
  - Increases throughput

Batch optimization to maximize throughput

making multiple queries at the same time

- Setting allows evaluating many points simultaneously
  - Increases throughput

Bayesian Optimization for a Better Dessert

Greg Kochanski, Daniel Golovin, John Karro, Benjamin Solnik, Subhodeep Moitra, and D. Sculley
{gpk, dgg, karro, bsolnik, smoitra, dsculley}@google.com; Google Brain Team

Abstract

We present a case study on applying Bayesian Optimization to a complex real-world system; our challenge was to optimize chocolate chip cookies. The process was a mixed-initiative system where both human chefs, human raters, and a machine optimizer participated in 144 experiments. This process resulted in highly rated cookies that deviated from expectations in some surprising ways – much less sugar in California, and cayenne in Pittsburgh. Our experience highlights the importance of incorporating domain expertise and the value of transfer learning approaches.

Kochanski et al. (2017) https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/46507.pdf

Batch optimization to maximize throughput

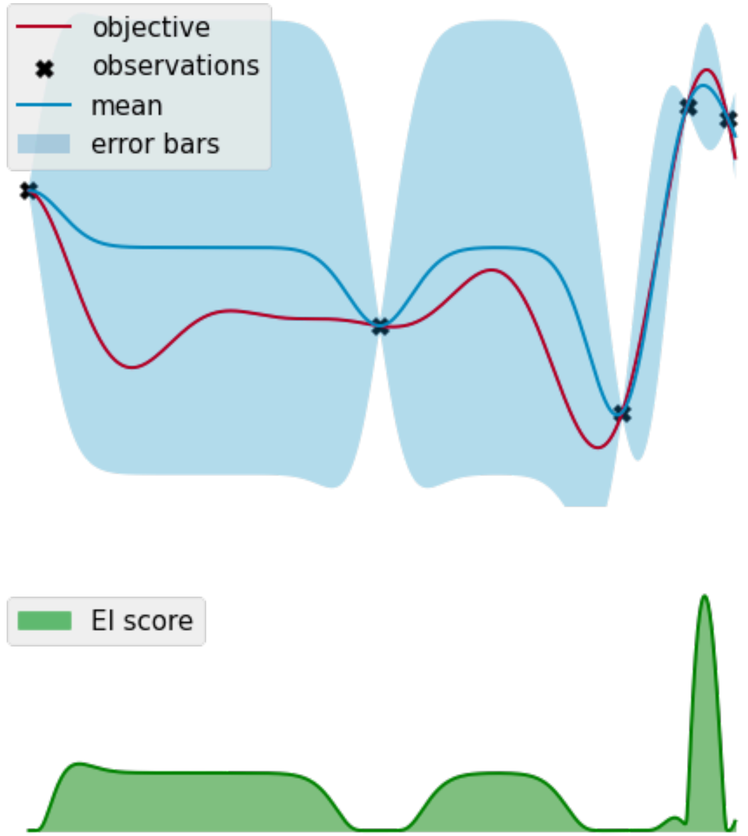
making multiple queries at the same time

- Setting allows evaluating many points simultaneously
  - Increases throughput
- **Challenge**: encourage diversity in a batch of queries

Batch optimization to maximize throughput

making multiple queries at the same time

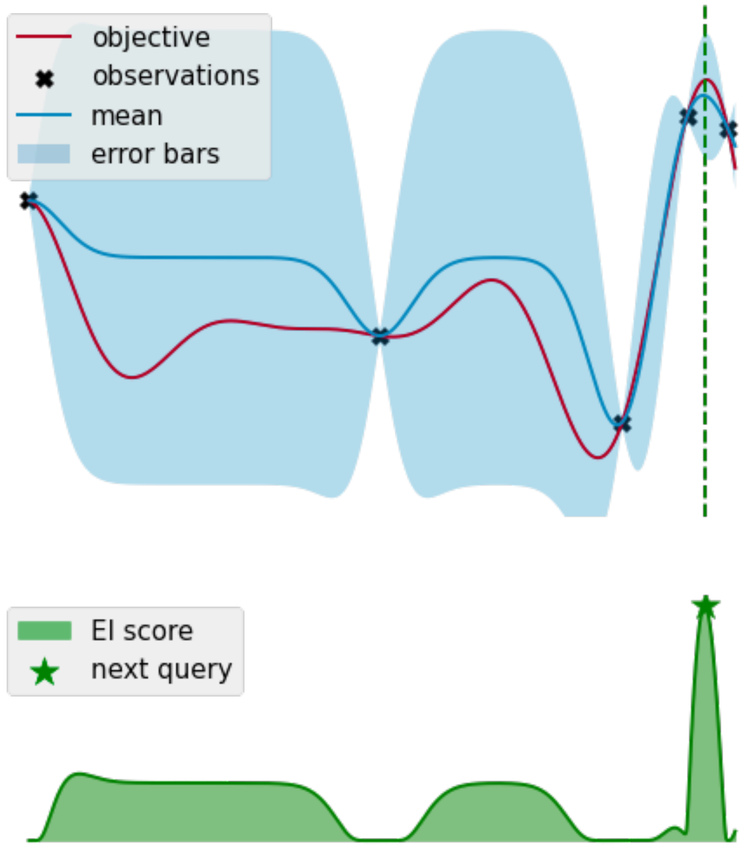
- Setting allows evaluating many points simultaneously
  - Increases throughput
- **Challenge**: encourage diversity in a batch of queries



Batch optimization to maximize throughput

making multiple queries at the same time

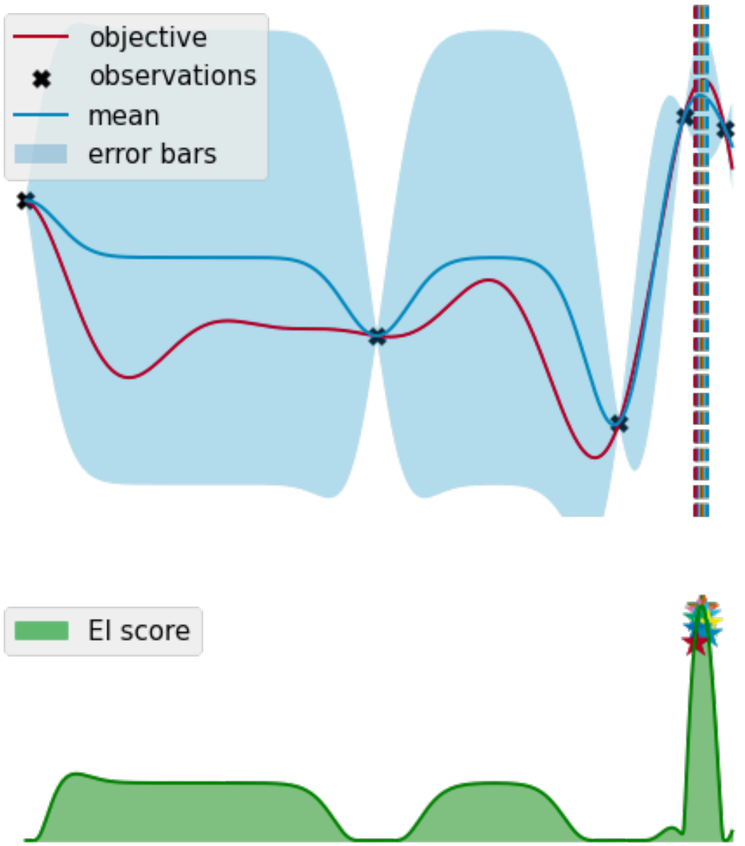
- Setting allows evaluating many points simultaneously
  - Increases throughput
- **Challenge**: encourage diversity in a batch of queries



Batch optimization to maximize throughput

making multiple queries at the same time

- Setting allows evaluating many points simultaneously
  - Increases throughput
- **Challenge**: encourage diversity in a batch of queries



Batch optimization to maximize throughput

making multiple queries at the same time

- Setting allows evaluating many points simultaneously
  - Increases throughput
- **Challenge**: encourage diversity in a batch of queries
  - **Motivation**: diversify our portfolio, avoid putting all our eggs in one basket

Batch optimization to maximize throughput

making multiple queries at the same time

- Setting allows evaluating many points simultaneously
  - Increases throughput
- **Challenge**: encourage diversity in a batch of queries
  - **Motivation**: diversify our portfolio, avoid putting all our eggs in one basket

```
policy = ExpectedImprovement(
    model, best_f=train_y.max()
)
```

Batch optimization to maximize throughput

making multiple queries at the same time

- Setting allows evaluating many points simultaneously
  - Increases throughput
- **Challenge**: encourage diversity in a batch of queries
  - **Motivation**: diversify our portfolio, avoid putting all our eggs in one basket

```
policy = ExpectedImprovement(
    model, best_f=train_y.max()
)
```

```
policy = qExpectedImprovement(
    model, best_f=train_y.max()
)
```

Constrained optimization for safe solutions

satisfying outcome constraints during exploration

Constrained optimization for safe solutions

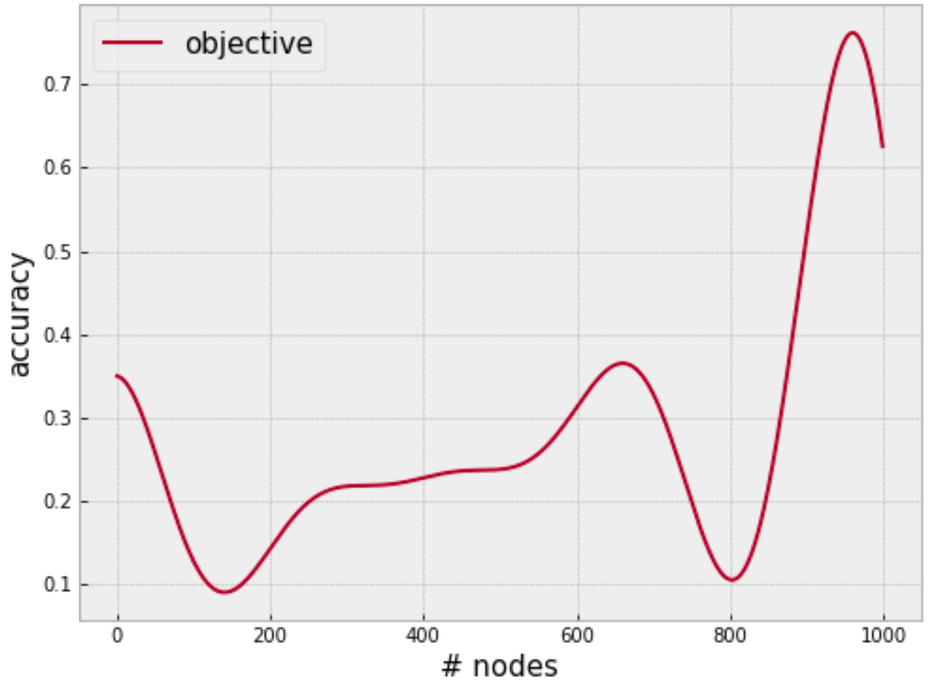
satisfying outcome constraints during exploration

- Unconstrained optimization leads to impractical/infeasible solutions

Constrained optimization for safe solutions

satisfying outcome constraints during exploration

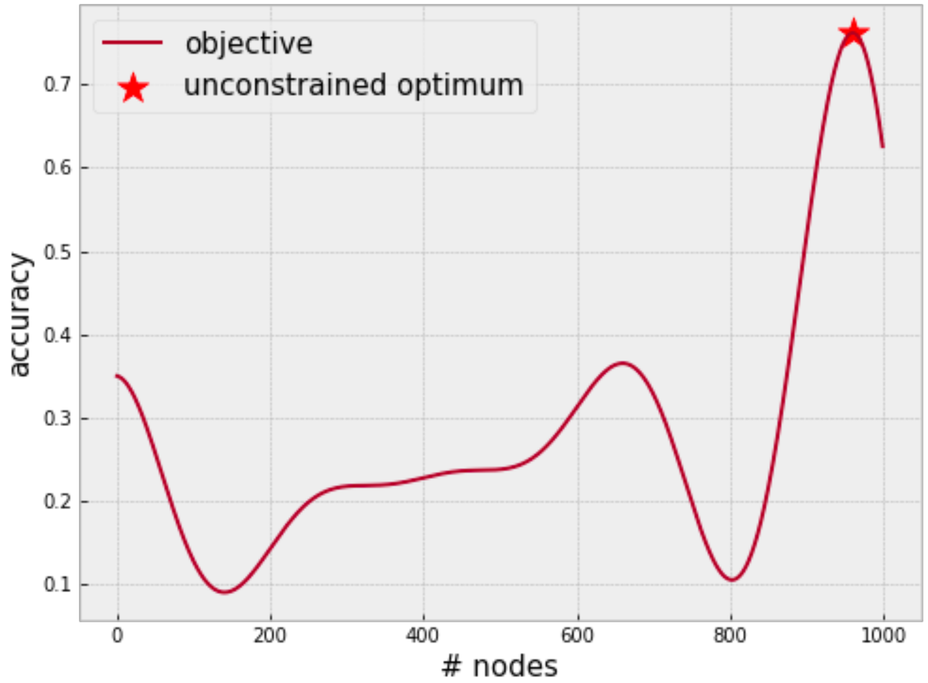
- Unconstrained optimization leads to impractical/infeasible solutions



Constrained optimization for safe solutions

satisfying outcome constraints during exploration

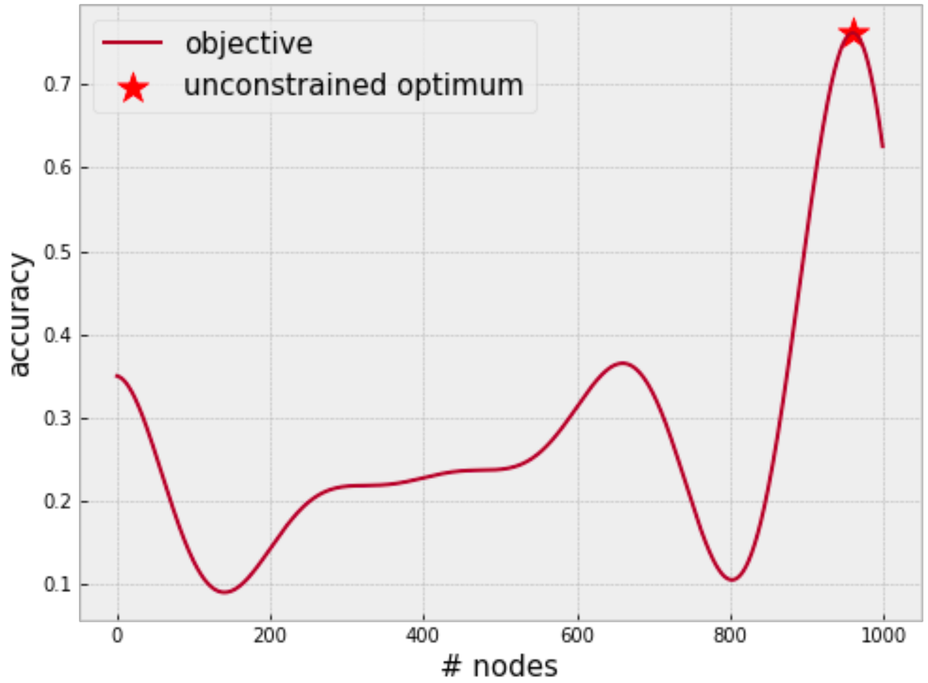
- Unconstrained optimization leads to impractical/infeasible solutions



Constrained optimization for safe solutions

satisfying outcome constraints during exploration

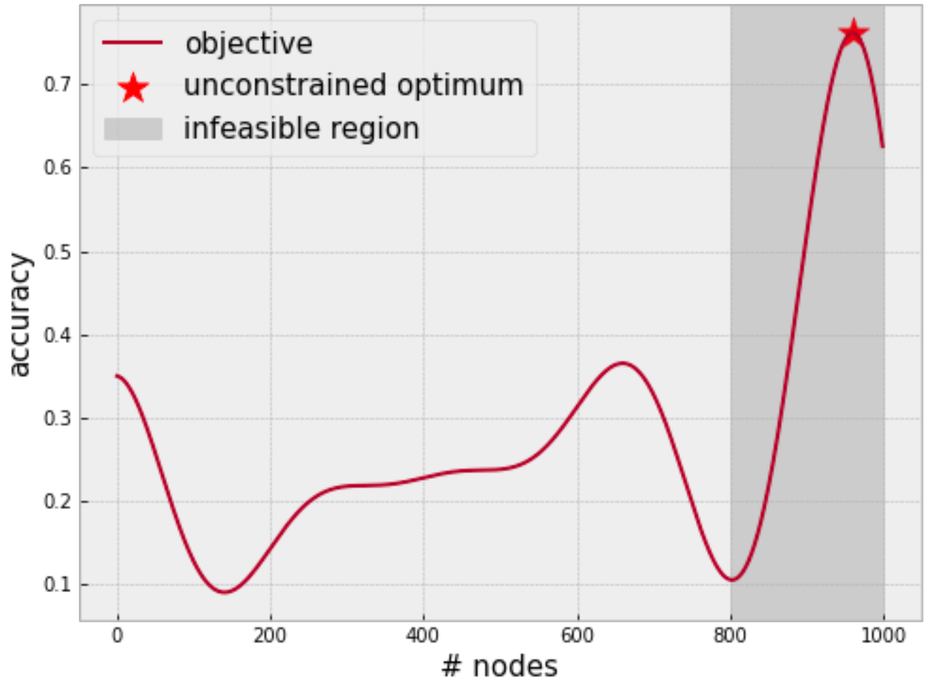
- Unconstrained optimization leads to impractical/infeasible solutions
  - Outcome constraints define *feasible solutions*



Constrained optimization for safe solutions

satisfying outcome constraints during exploration

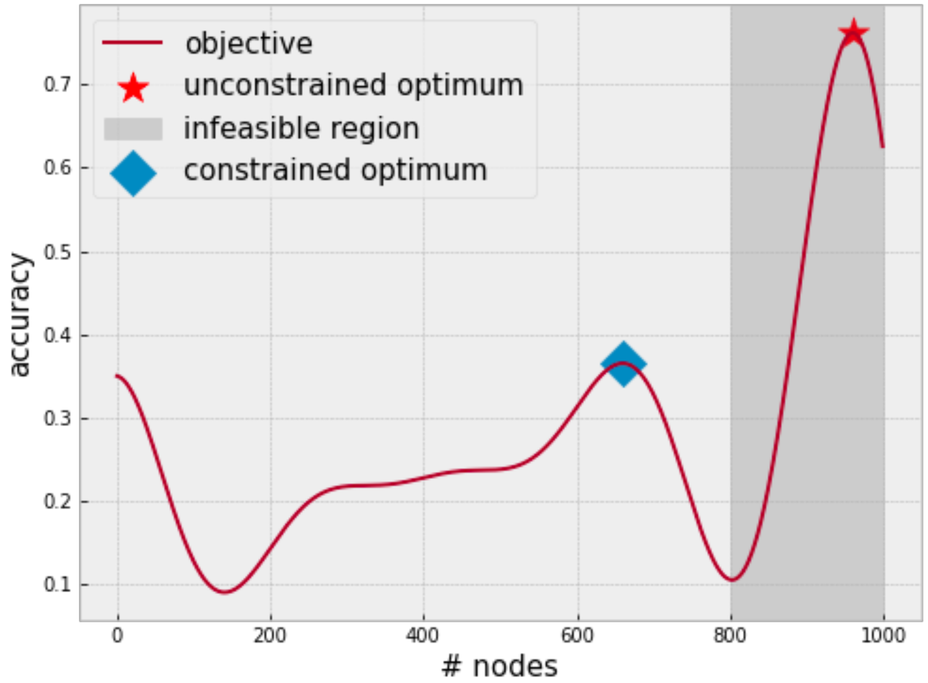
- Unconstrained optimization leads to impractical/infeasible solutions
  - Outcome constraints define *feasible solutions*



Constrained optimization for safe solutions

satisfying outcome constraints during exploration

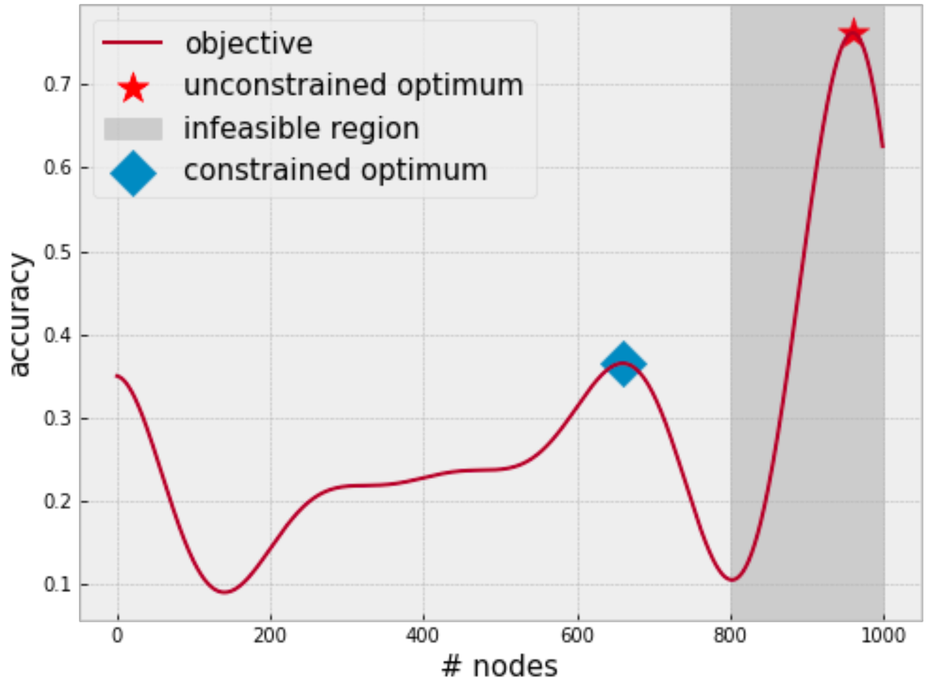
- Unconstrained optimization leads to impractical/infeasible solutions
  - Outcome constraints define *feasible solutions*



Constrained optimization for safe solutions

satisfying outcome constraints during exploration

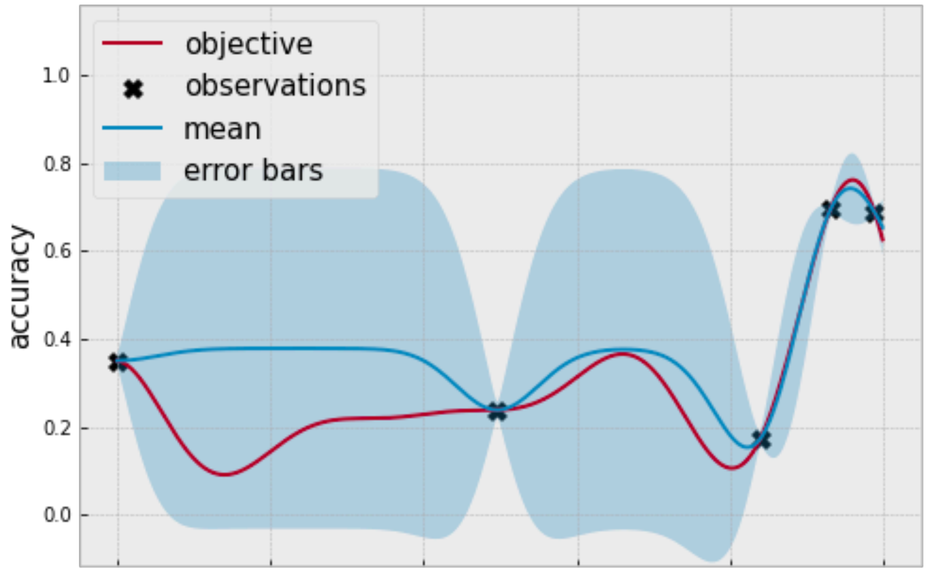
- Unconstrained optimization leads to impractical/infeasible solutions
  - Outcome constraints define *feasible solutions*
- **Challenge**: optimization while satisfying constraints



Constrained optimization for safe solutions

satisfying outcome constraints during exploration

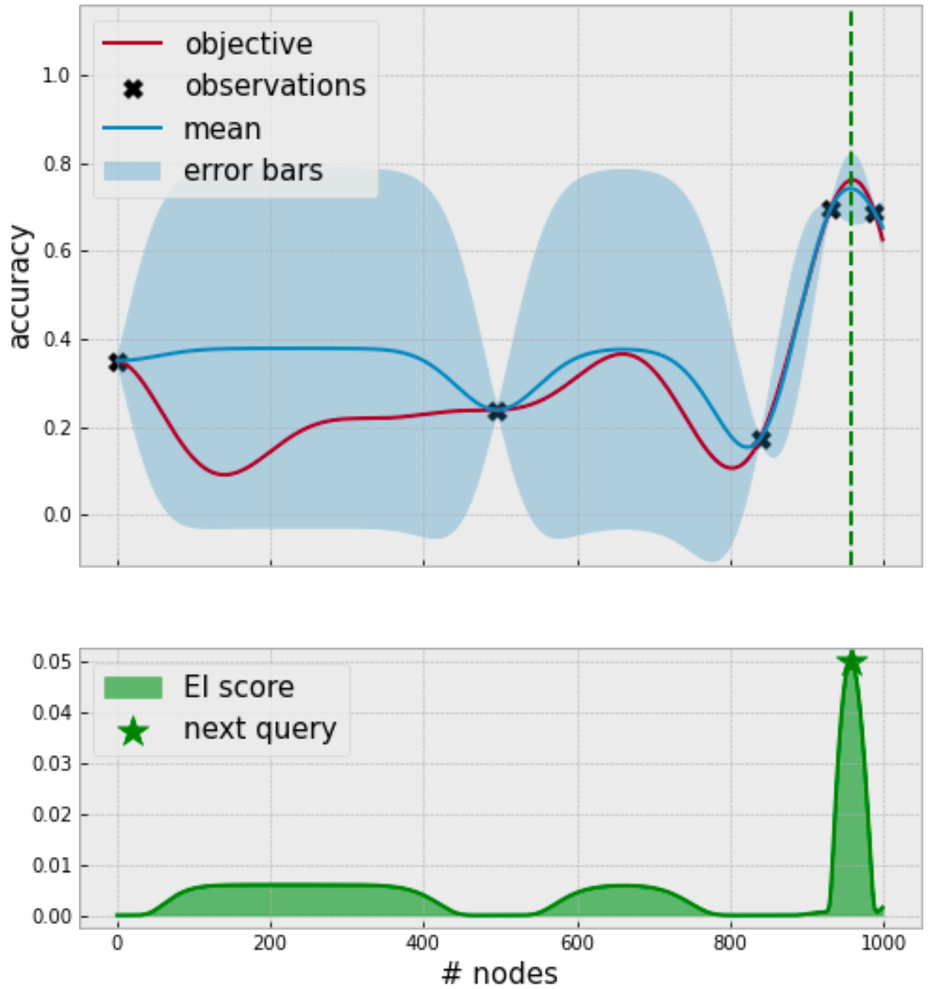
- Unconstrained optimization leads to impractical/infeasible solutions
  - Outcome constraints define *feasible solutions*
- **Challenge**: optimization while satisfying constraints



Constrained optimization for safe solutions

satisfying outcome constraints during exploration

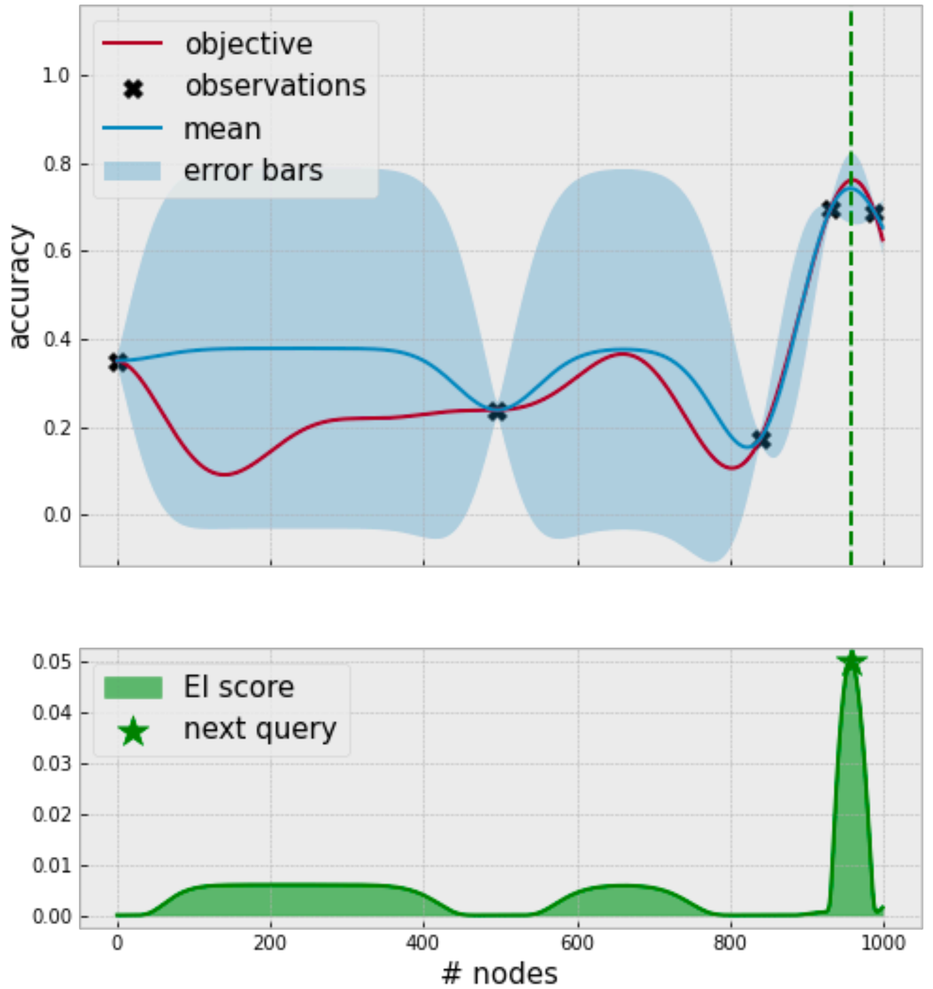
- Unconstrained optimization leads to impractical/infeasible solutions
  - Outcome constraints define *feasible solutions*
- **Challenge**: optimization while satisfying constraints



Constrained optimization for safe solutions

satisfying outcome constraints during exploration

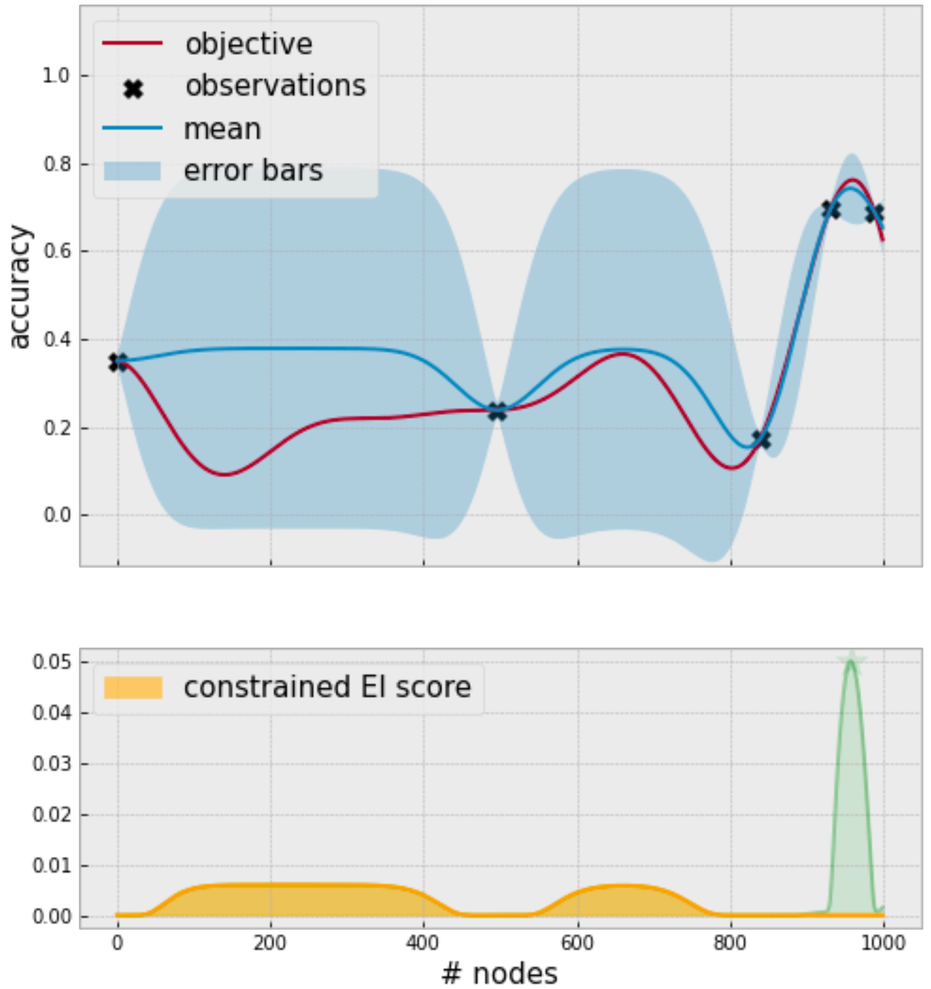
- Unconstrained optimization leads to impractical/infeasible solutions
  - Outcome constraints define *feasible solutions*
- **Challenge**: optimization while satisfying constraints
- **Solution**: multiple EI score with the *probability of feasibility*



Constrained optimization for safe solutions

satisfying outcome constraints during exploration

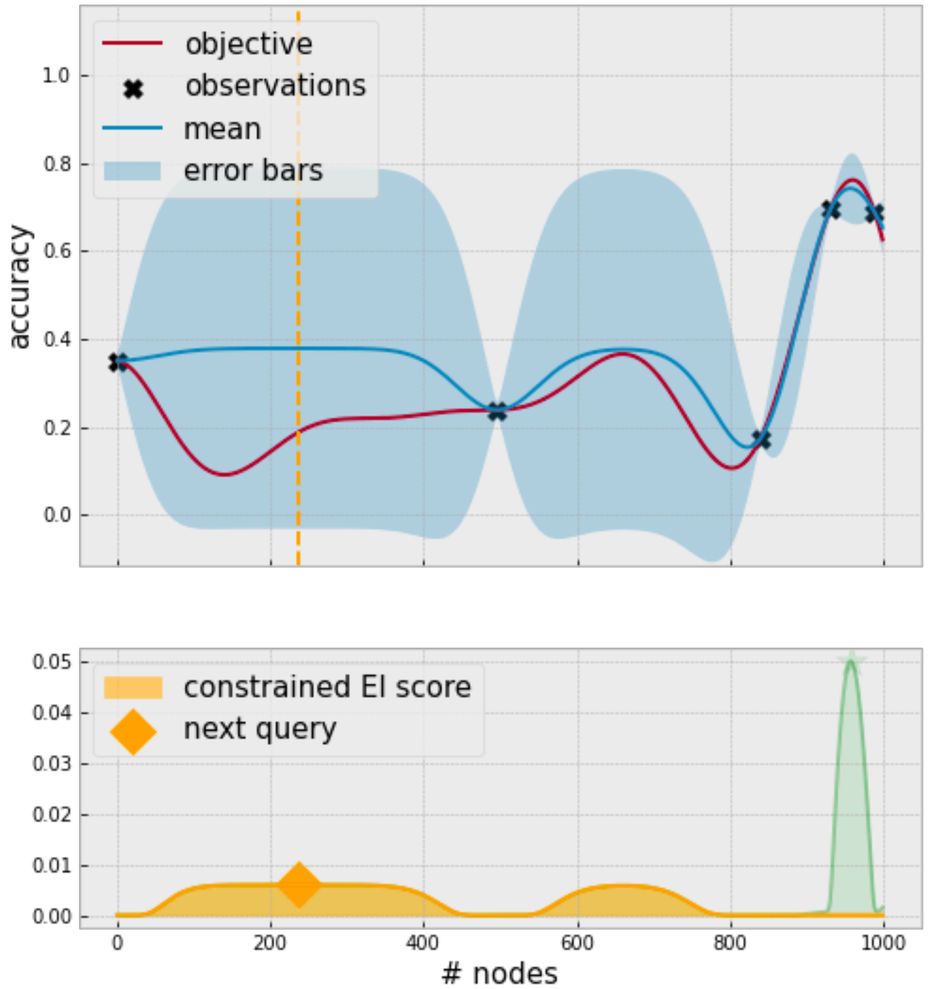
- Unconstrained optimization leads to impractical/infeasible solutions
  - Outcome constraints define *feasible solutions*
- **Challenge**: optimization while satisfying constraints
- **Solution**: multiple EI score with the *probability of feasibility*



Constrained optimization for safe solutions

satisfying outcome constraints during exploration

- Unconstrained optimization leads to impractical/infeasible solutions
  - Outcome constraints define *feasible solutions*
- **Challenge**: optimization while satisfying constraints
- **Solution**: multiple EI score with the *probability of feasibility*



Gardner et al. (2014) http://proceedings.mlr.press/v32/gardner14.pdf

Constrained optimization for safe solutions

satisfying outcome constraints during exploration

- Unconstrained optimization leads to impractical/infeasible solutions
  - Outcome constraints define *feasible solutions*
- **Challenge**: optimization while satisfying constraints
- **Solution**: multiple EI score with the *probability of feasibility*

```
policy = ConstrainedExpectedImprovement(
    model=ModelListGP(utility_model, cost_model),
    best_f=best_f,
    objective_index=0,
    constraints={1: [None, 800]}  # upper-bounded at 800
)
```

Gardner et al. (2014) http://proceedings.mlr.press/v32/gardner14.pdf

Multi-objective optimization for balanced solutions

trading off competing objectives

Multi-objective optimization for balanced solutions

trading off competing objectives

- *Multiple, competing objectives* to optimize simultaneously

Multi-objective optimization for balanced solutions

trading off competing objectives

- *Multiple, competing objectives* to optimize simultaneously

hyperparameter tuning

model's accuracy

Multi-objective optimization for balanced solutions

trading off competing objectives

- *Multiple, competing objectives* to optimize simultaneously

hyperparameter tuning

model's accuracy

vs.

model size, training time

Multi-objective optimization for balanced solutions

trading off competing objectives

- *Multiple, competing objectives* to optimize simultaneously

hyperparameter tuning

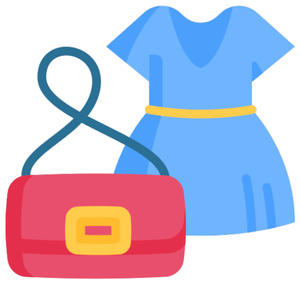
model's accuracy

vs.

model size, training time

product recommendation

style & looks



Multi-objective optimization for balanced solutions

trading off competing objectives

- *Multiple, competing objectives* to optimize simultaneously

hyperparameter tuning

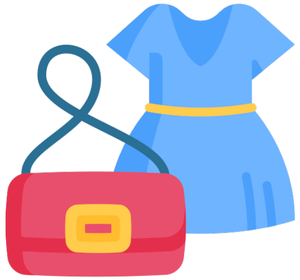
model's accuracy

vs.

model size, training time

product recommendation

style & looks



vs.

cost

Multi-objective optimization for balanced solutions

trading off competing objectives

- *Multiple, competing objectives* to optimize simultaneously
- **Challenge**: balance between competing objectives

hyperparameter tuning

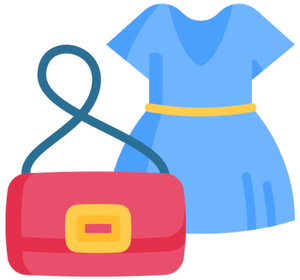
model's accuracy

vs.

model size, training time

product recommendation

style & looks



vs.

cost

Multi-objective optimization for balanced solutions

trading off competing objectives

- *Multiple, competing objectives* to optimize simultaneously
- **Challenge**: balance between competing objectives

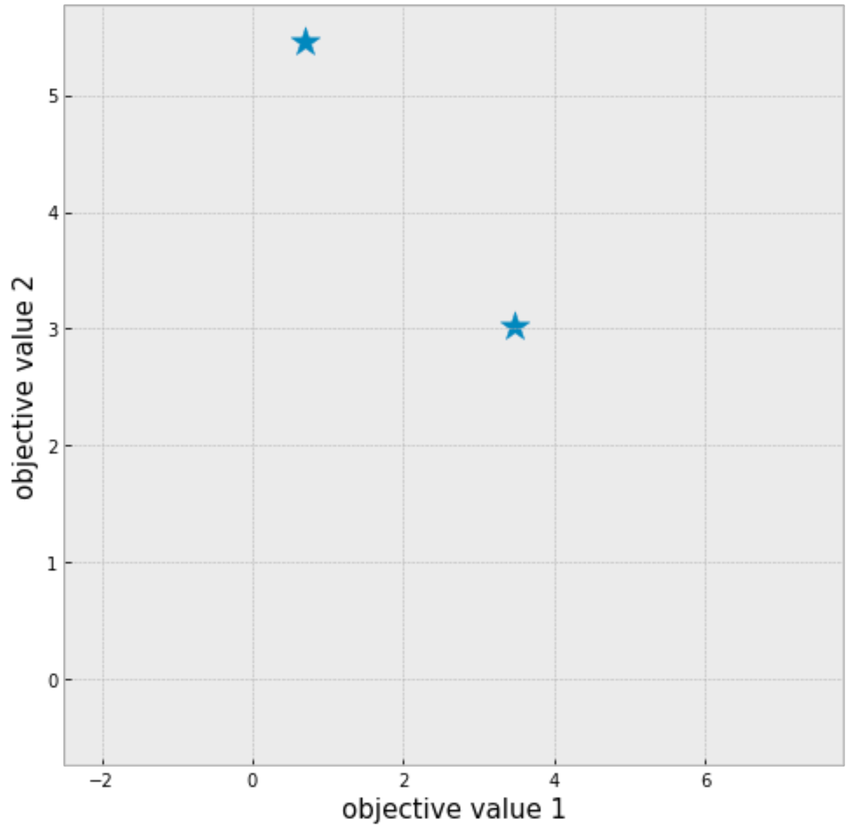
collected data	objective value 1	objective value 2
point A	0.6919	5.4533
point B	3.4861	3.0159

Multi-objective optimization for balanced solutions

trading off competing objectives

- *Multiple, competing objectives* to optimize simultaneously
- **Challenge**: balance between competing objectives

collected data	objective value 1	objective value 2
point A	0.6919	5.4533
point B	3.4861	3.0159

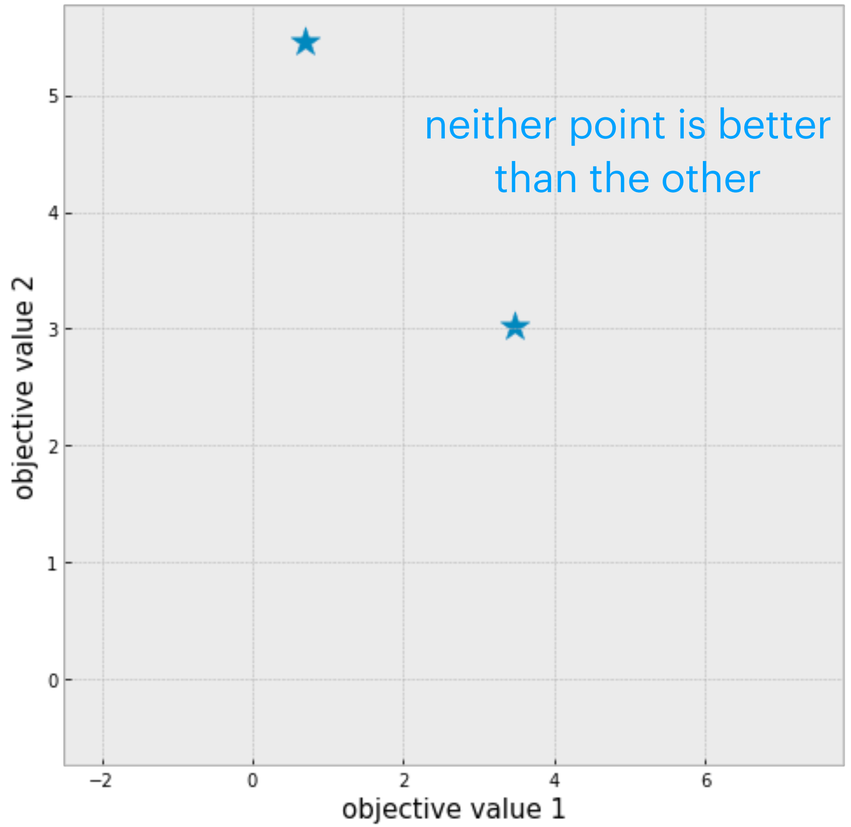


Multi-objective optimization for balanced solutions

trading off competing objectives

- *Multiple, competing objectives* to optimize simultaneously
- **Challenge**: balance between competing objectives

collected data	objective value 1	objective value 2
point A	0.6919	5.4533
point B	3.4861	3.0159

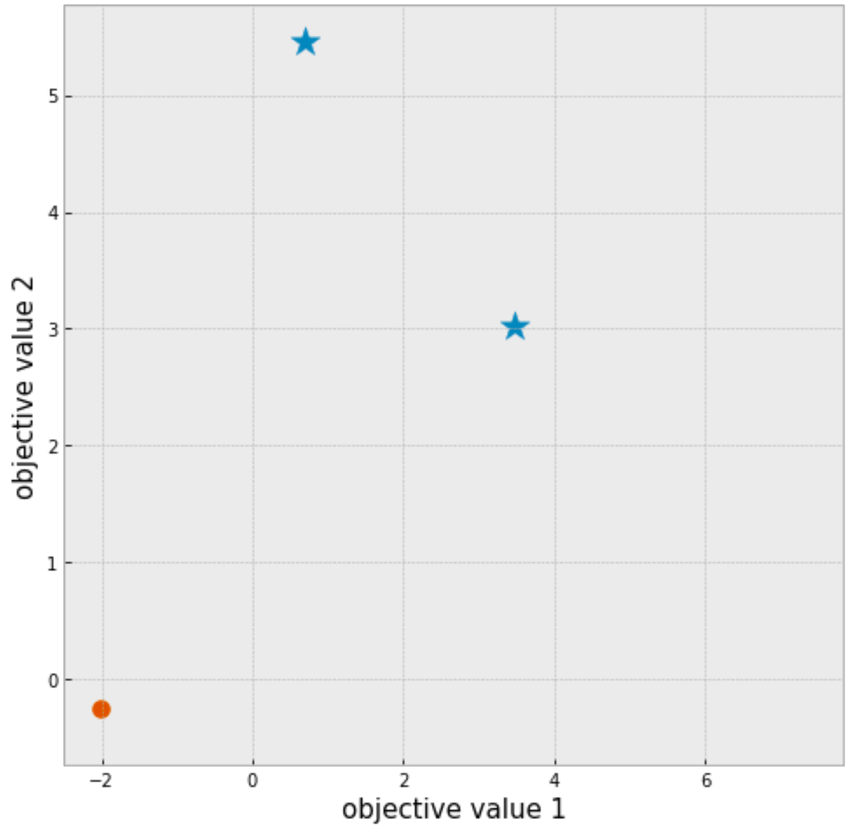


Multi-objective optimization for balanced solutions

trading off competing objectives

- *Multiple, competing objectives* to optimize simultaneously
- **Challenge**: balance between competing objectives

collected data	objective value 1	objective value 2
point A	0.6919	5.4533
point B	3.4861	3.0159
point C	-2.0160	-0.2506

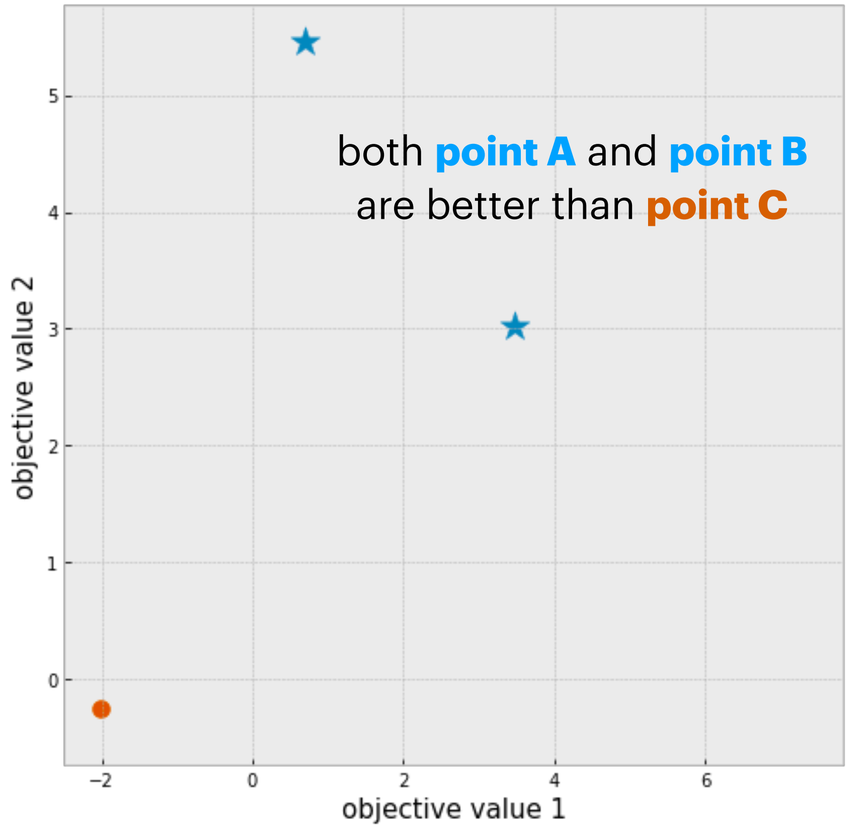


Multi-objective optimization for balanced solutions

trading off competing objectives

- *Multiple, competing objectives* to optimize simultaneously
- **Challenge**: balance between competing objectives

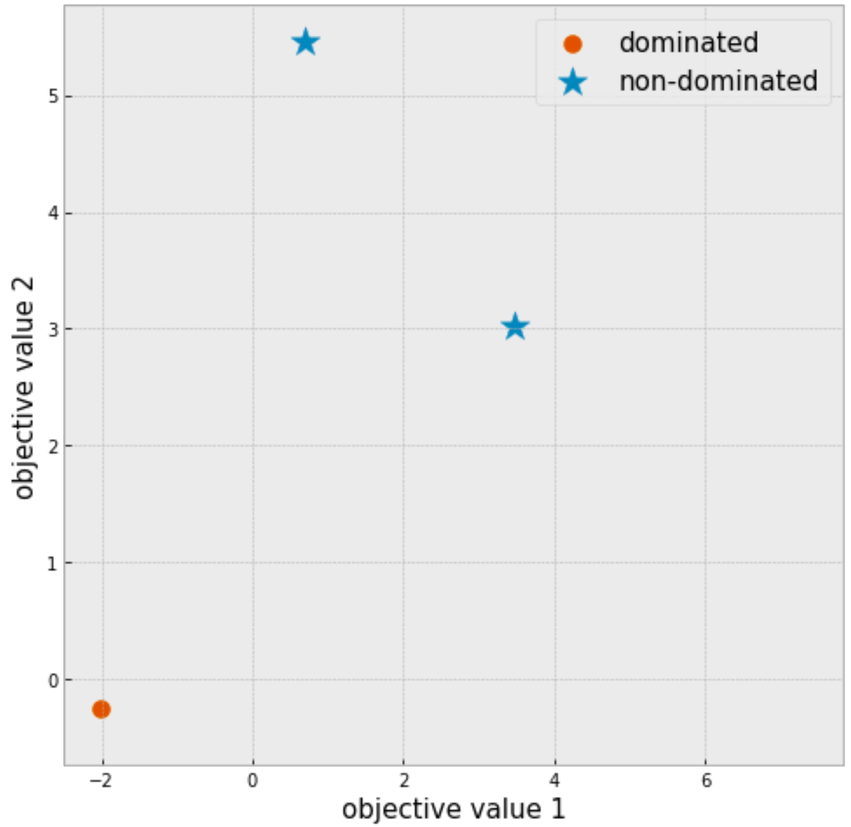
collected data	objective value 1	objective value 2
point A	0.6919	5.4533
point B	3.4861	3.0159
point C	-2.0160	-0.2506



Multi-objective optimization for balanced solutions

trading off competing objectives

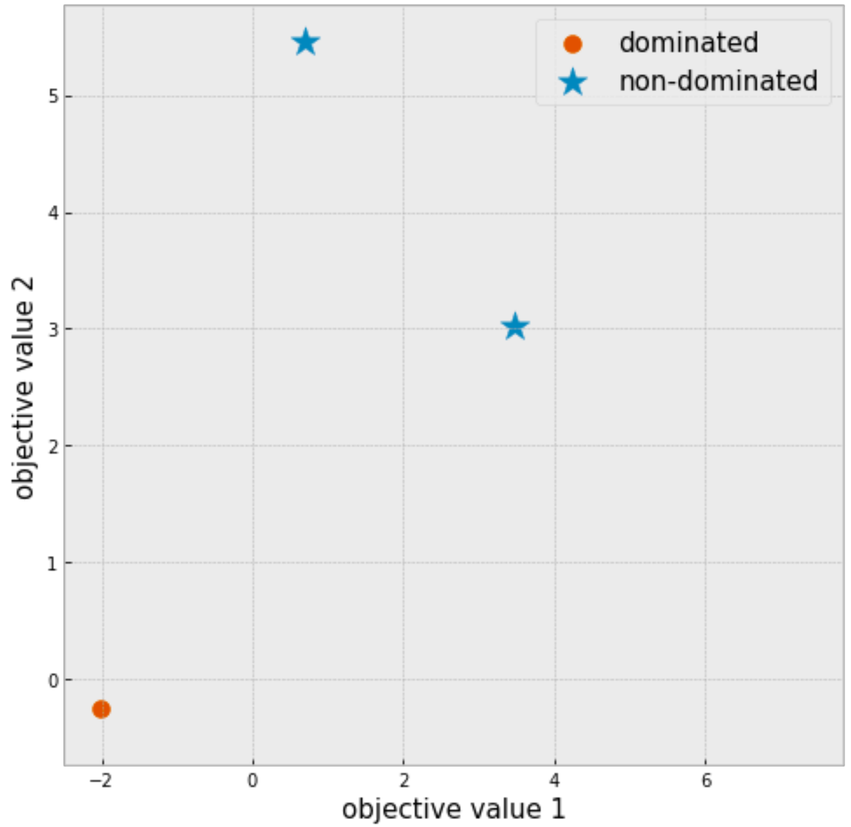
- *Multiple, competing objectives* to optimize simultaneously
- **Challenge**: balance between competing objectives



Multi-objective optimization for balanced solutions

trading off competing objectives

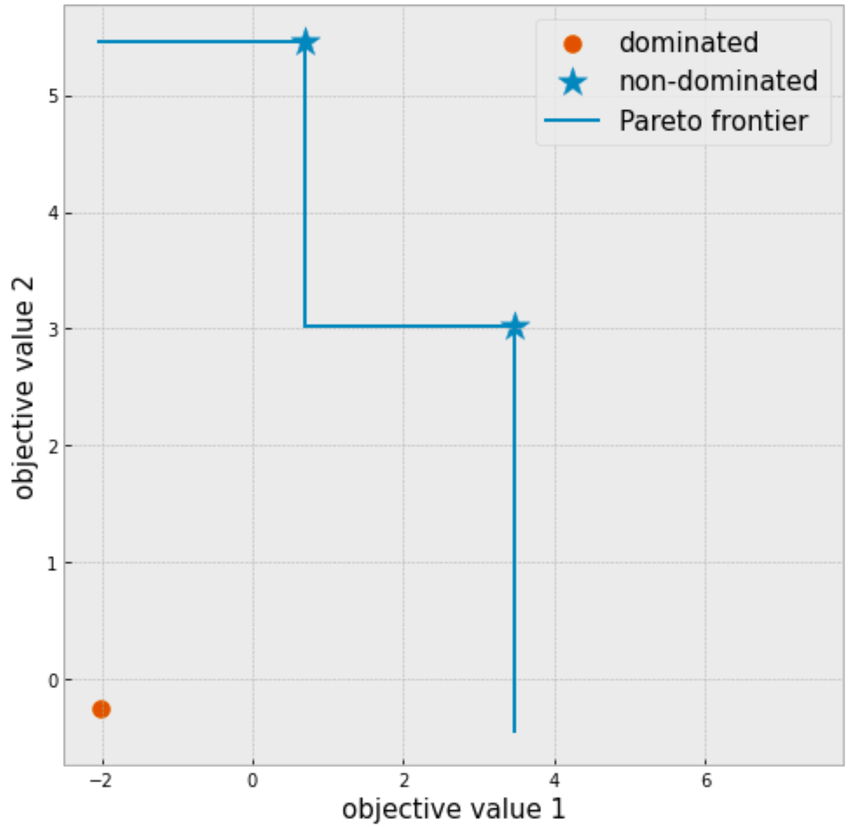
- *Multiple, competing objectives* to optimize simultaneously
- **Challenge**: balance between competing objectives
- **Concept**: the *Pareto frontier* contains *non-dominated points* and encodes multi-objective optimality



Multi-objective optimization for balanced solutions

trading off competing objectives

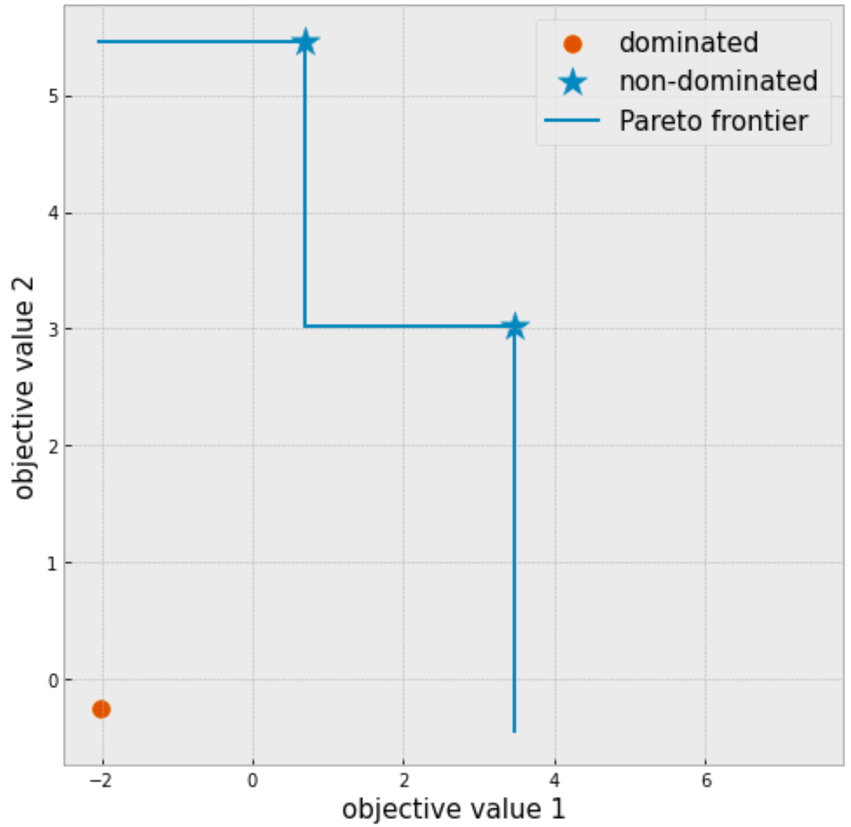
- *Multiple, competing objectives* to optimize simultaneously
- **Challenge**: balance between competing objectives
- **Concept**: the *Pareto frontier* contains *non-dominated points* and encodes multi-objective optimality



Multi-objective optimization for balanced solutions

trading off competing objectives

- *Multiple, competing objectives* to optimize simultaneously
- **Challenge**: balance between competing objectives
- **Concept**: the *Pareto frontier* contains *non-dominated points* and encodes multi-objective optimality
  - **Goal**: *improve* the Pareto frontier



Multi-objective optimization for balanced solutions

trading off competing objectives

- *Multiple, competing objectives* to optimize simultaneously
- **Challenge**: balance between competing objectives
- **Concept**: the *Pareto frontier* contains *non-dominated points* and encodes multi-objective optimality
  - **Goal**: *improve* the Pareto frontier

```
policy = ExpectedHypervolumeImprovement(
    model=ModelListGP(model1, model2),
    ref_point=ref_point,
    partitioning=FastNondominatedPartitioning(
        ref_point, train_y
    )  # computes the Pareto frontier
)
```

Multi-fidelity optimization to minimize cost

balancing between learning and cost

Multi-fidelity optimization to minimize cost

balancing between learning and cost

- The objective function is *expensive*

Multi-fidelity optimization to minimize cost

balancing between learning and cost

- The objective function is *expensive*
  - *Cheap surrogates* that approximate the objective

Multi-fidelity optimization to minimize cost

balancing between learning and cost

- The objective function is *expensive*
  - *Cheap surrogates* that approximate the objective

application	surrogate	ground truth

Multi-fidelity optimization to minimize cost

balancing between learning and cost

- The objective function is *expensive*
  - *Cheap surrogates* that approximate the objective

application	surrogate	ground truth
hyperparameter tuning ⚙️		

Multi-fidelity optimization to minimize cost

balancing between learning and cost

- The objective function is *expensive*
  - *Cheap surrogates* that approximate the objective

application	surrogate	ground truth
hyperparameter tuning ⚙️	training for 10 epochs: fast, inaccurate	

Multi-fidelity optimization to minimize cost

balancing between learning and cost

- The objective function is *expensive*
  - *Cheap surrogates* that approximate the objective

application	surrogate	ground truth
hyperparameter tuning ⚙️	training for 10 epochs: fast, inaccurate	training for 50 epochs: slow, accurate

Multi-fidelity optimization to minimize cost

balancing between learning and cost

- The objective function is *expensive*
  - *Cheap surrogates* that approximate the objective

application	surrogate	ground truth
hyperparameter tuning ⚙️	training for 10 epochs: fast, inaccurate	training for 50 epochs: slow, accurate
physics research & experiments 🧪		

Multi-fidelity optimization to minimize cost

balancing between learning and cost

- The objective function is *expensive*
  - *Cheap surrogates* that approximate the objective

application	surrogate	ground truth
hyperparameter tuning ⚙️	training for 10 epochs: fast, inaccurate	training for 50 epochs: slow, accurate
physics research & experiments 🧪	computer simulation: inexpensive, partial evidence	

Multi-fidelity optimization to minimize cost

balancing between learning and cost

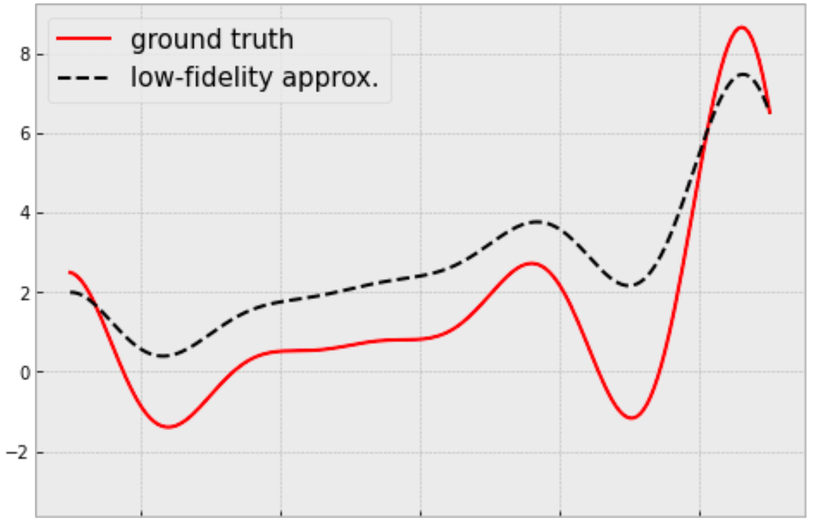
- The objective function is *expensive*
  - *Cheap surrogates* that approximate the objective

application	surrogate	ground truth
hyperparameter tuning ⚙️	training for 10 epochs: fast, inaccurate	training for 50 epochs: slow, accurate
physics research & experiments 🧪	computer simulation: inexpensive, partial evidence	actual experiment in a lab: effort + cost, conclusive

Multi-fidelity optimization to minimize cost

balancing between learning and cost

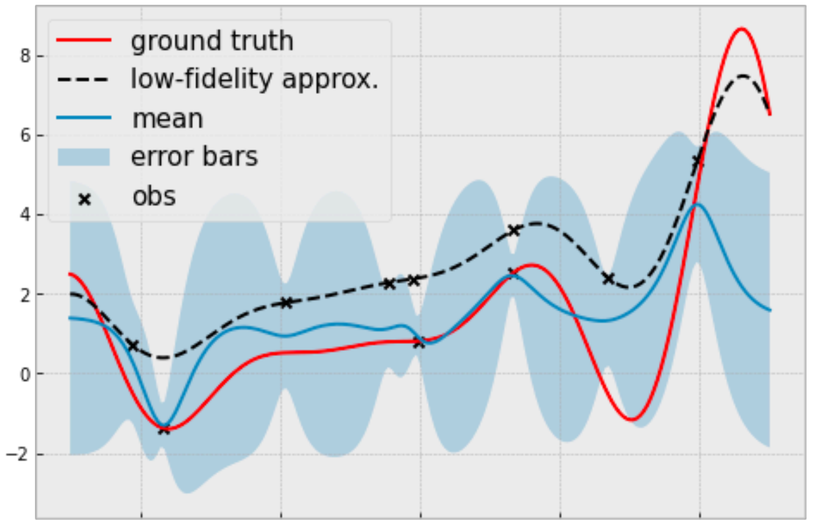
- The objective function is *expensive*
  - *Cheap surrogates* that approximate the objective



Multi-fidelity optimization to minimize cost

balancing between learning and cost

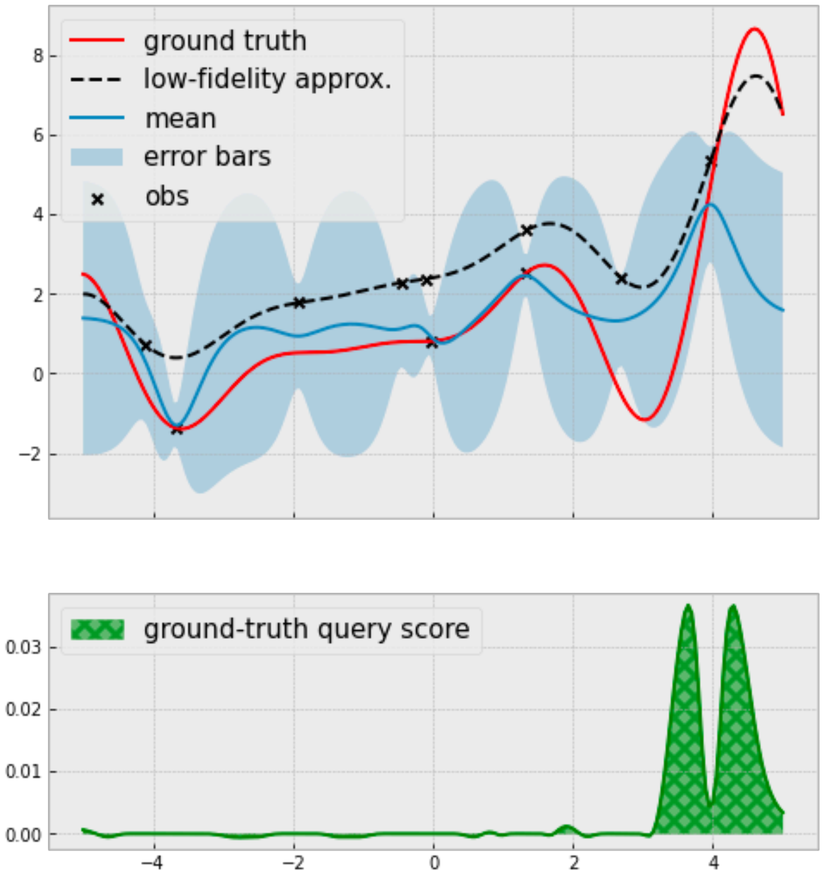
- The objective function is *expensive*
  - *Cheap surrogates* that approximate the objective



Multi-fidelity optimization to minimize cost

balancing between learning and cost

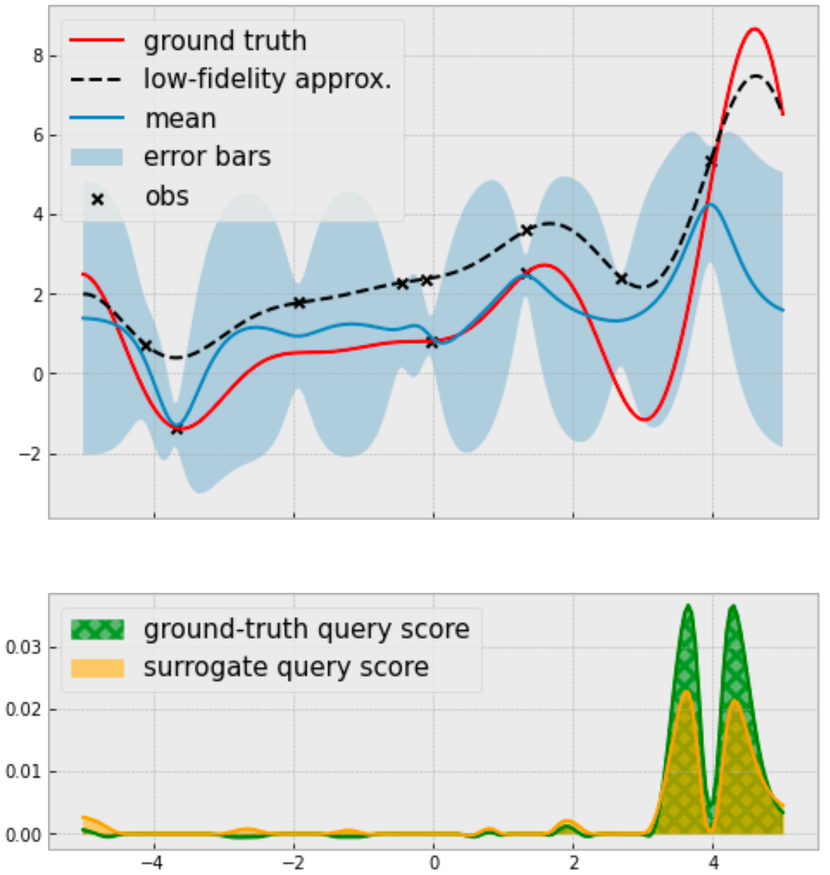
- The objective function is *expensive*
  - *Cheap surrogates* that approximate the objective



Multi-fidelity optimization to minimize cost

balancing between learning and cost

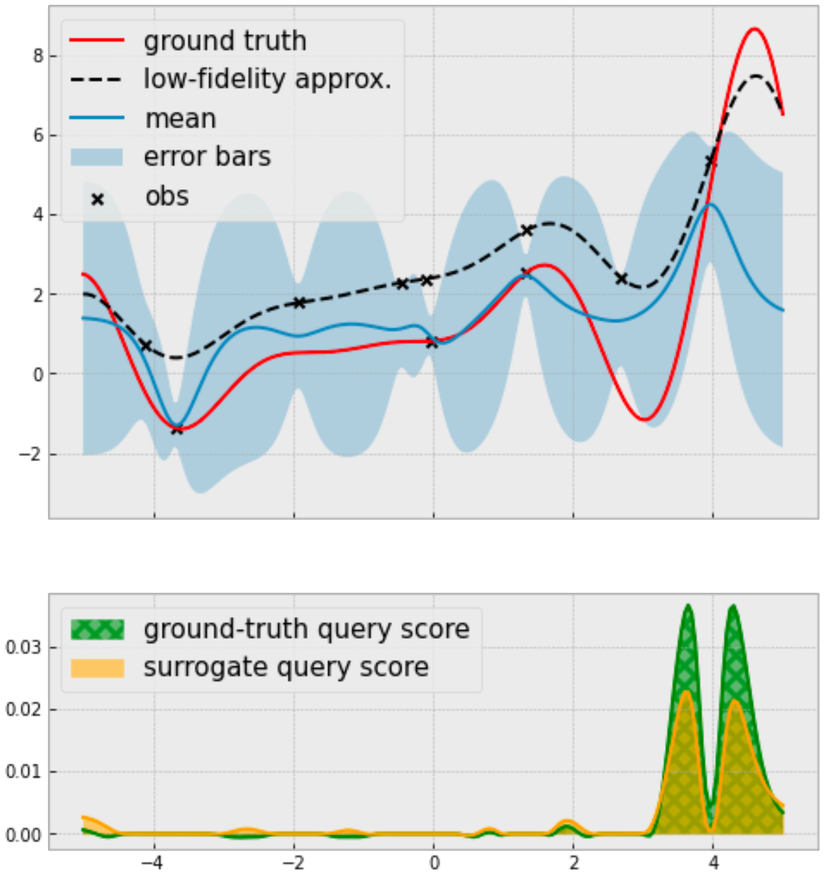
- The objective function is *expensive*
  - *Cheap surrogates* that approximate the objective



Multi-fidelity optimization to minimize cost

balancing between learning and cost

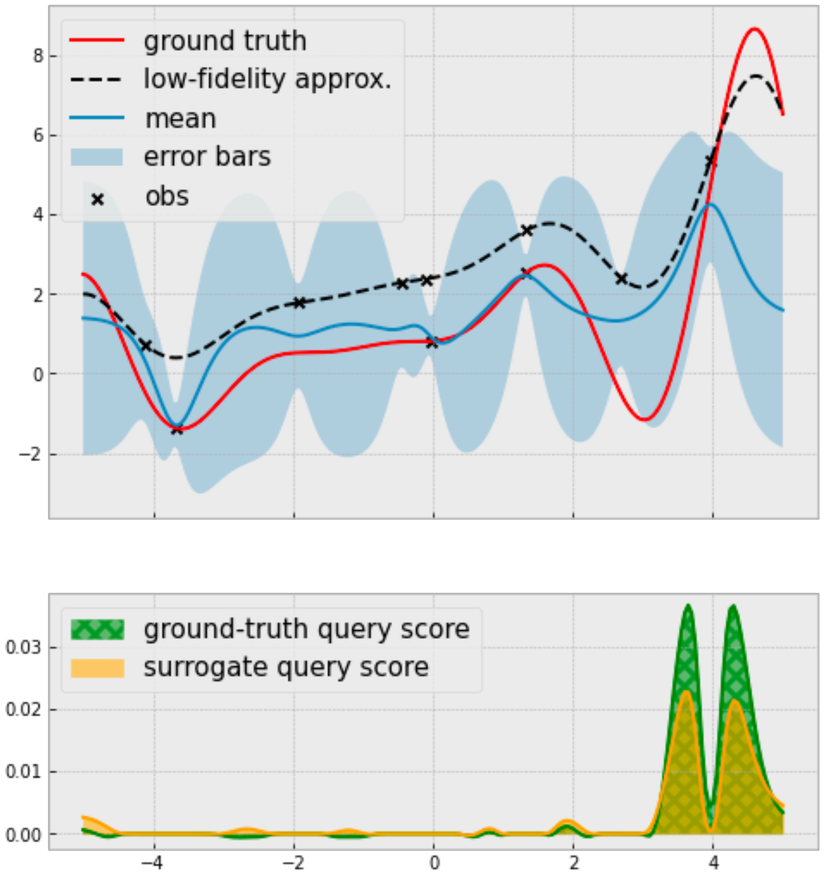
- The objective function is *expensive*
  - *Cheap surrogates* that approximate the objective
- **Challenge**: trade off *accurate information* from the ground truth vs. *cost-effectiveness* of the surrogate



Multi-fidelity optimization to minimize cost

balancing between learning and cost

- The objective function is *expensive*
  - *Cheap surrogates* that approximate the objective
- **Challenge**: trade off *accurate information* from the ground truth vs. *cost-effectiveness* of the surrogate
- **Solution**: return on investment (ROI)

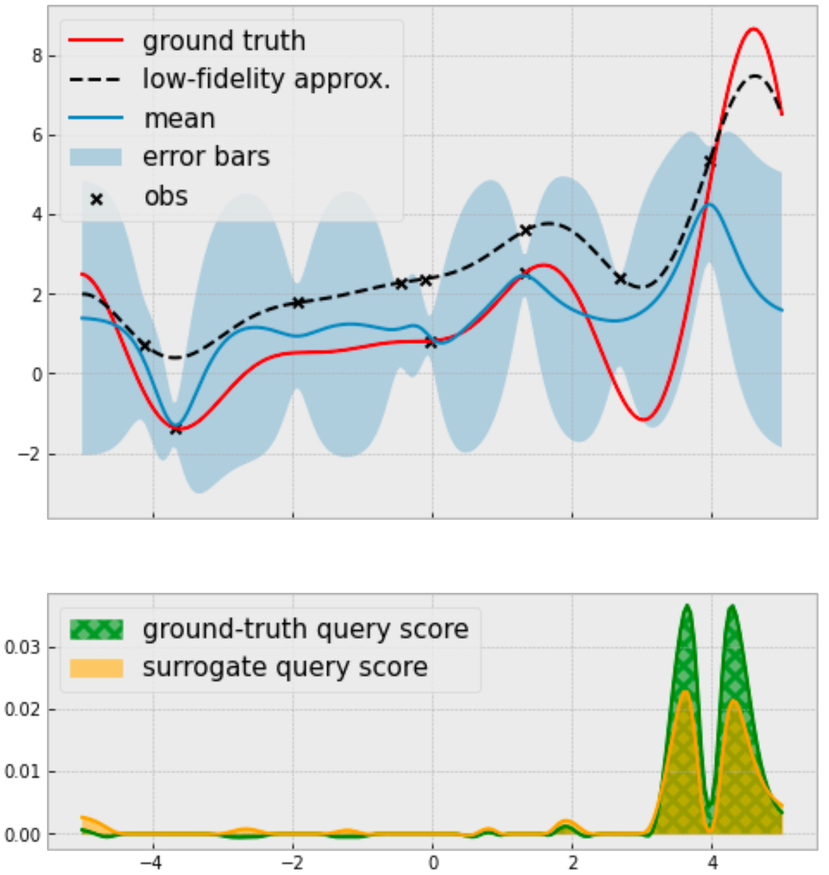


Multi-fidelity optimization to minimize cost

balancing between learning and cost

- The objective function is *expensive*
  - *Cheap surrogates* that approximate the objective
- **Challenge**: trade off *accurate information* from the ground truth vs. *cost-effectiveness* of the surrogate
- **Solution**: return on investment (ROI)

$$\text{score} = \frac{\text{utility}}{\text{cost}}$$

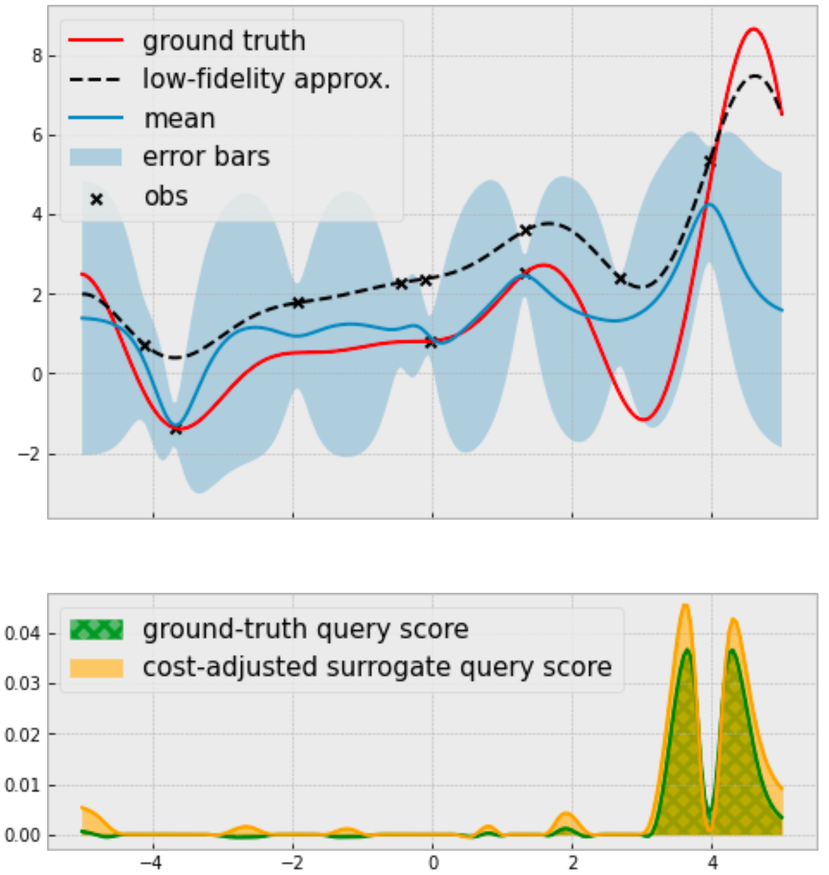


Multi-fidelity optimization to minimize cost

balancing between learning and cost

- The objective function is *expensive*
  - *Cheap surrogates* that approximate the objective
- **Challenge**: trade off *accurate information* from the ground truth vs. *cost-effectiveness* of the surrogate
- **Solution**: return on investment (ROI)

$$\text{score} = \frac{\text{utility}}{\text{cost}}$$



Multi-fidelity optimization to minimize cost

balancing between learning and cost

- The objective function is *expensive*
  - *Cheap surrogates* that approximate the objective
- **Challenge**: trade off *accurate information* from the ground truth vs. *cost-effectiveness* of the surrogate
- **Solution**: return on investment (ROI)

$$\text{score} = \frac{\text{utility}}{\text{cost}}$$

Multi-fidelity optimization to minimize cost

balancing between learning and cost

- The objective function is *expensive*
  - *Cheap surrogates* that approximate the objective
- **Challenge**: trade off *accurate information* from the ground truth vs. *cost-effectiveness* of the surrogate
- **Solution**: return on investment (ROI)

$$\text{score} = \frac{\text{utility}}{\text{cost}}$$

```
policy = qMultiFidelityMaxValueEntropy(
    model,
    candidate_x,
    cost_aware_utility=cost_aware_utility,  # ROI
    project=project_to_target_fidelity,
)
```