EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

machine learning
new perspectives
for science

Seminar WS23/24

Bayesian Optimization for HPO

Katharina Eggensperger

/KEggensperger
katharina.eggensperger@uni-tuebingen.de
AutoML for Science

October 25th, 2023

AutoML.org

Story Line Today

[?] Questions regarding the organization

[25min] **How to give a good presentation (not only in this seminar)**

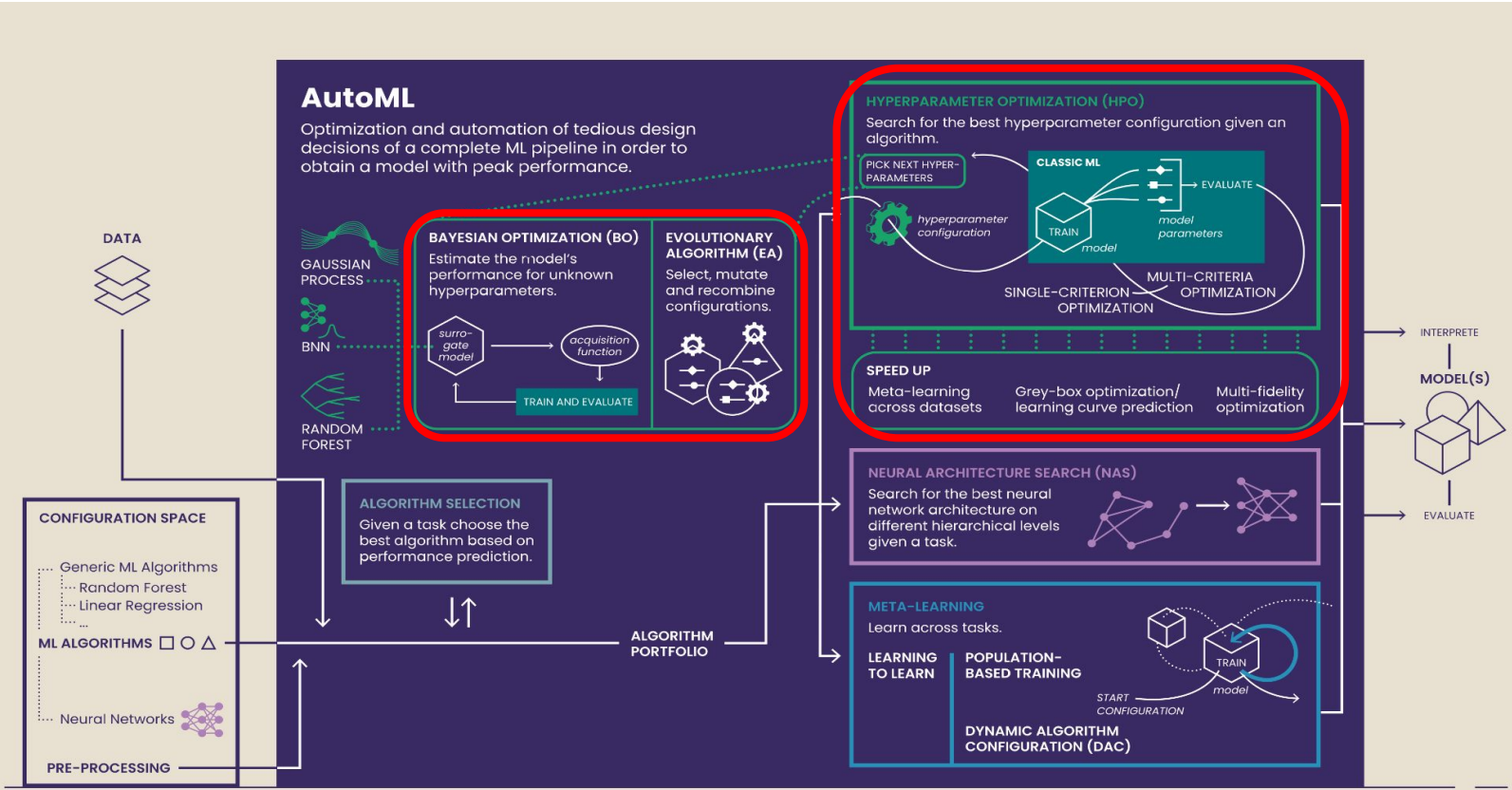
[?] Your Questions

[40min] **Bayesian Optimization for HPO**

[?] Your Questions

The Big Picture

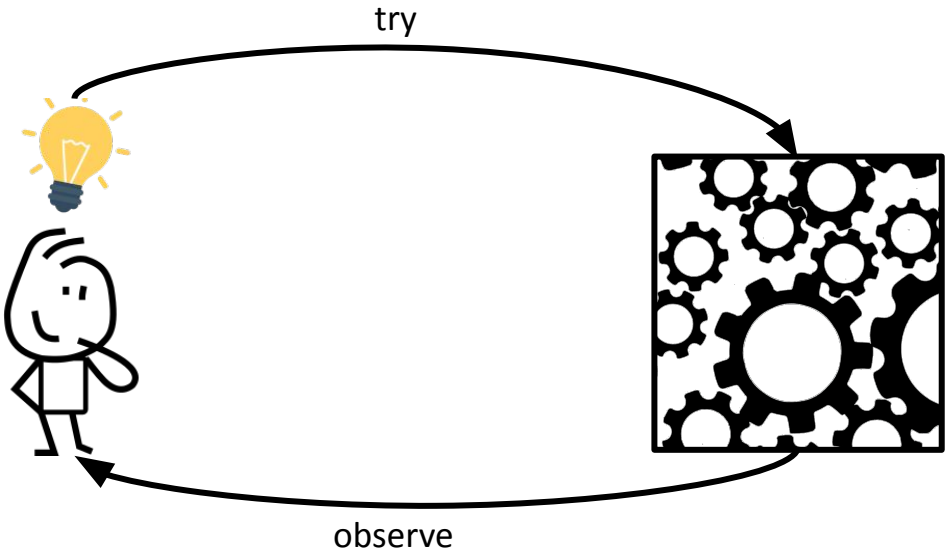
>> What is this about?

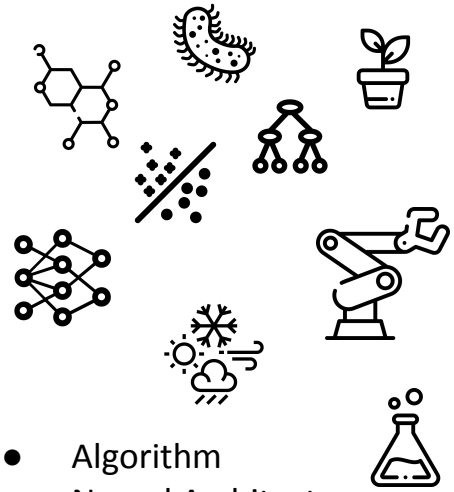

AutoML
Optimization and automation of tedious design decisions of a complete ML pipeline in order to obtain a model with peak performance.
DATA
GAUSSIAN PROCESS
BNN
RANDOM FOREST
BAYESIAN OPTIMIZATION (BO)
Estimate the model's performance for unknown hyperparameters.
surro-gate model
acquisition function
TRAIN AND EVALUATE
EVOLUTIONARY ALGORITHM (EA)
Select, mutate and recombine configurations.
HYPERPARAMETER OPTIMIZATION (HPO)
Search for the best hyperparameter configuration given an algorithm.
PICK NEXT HYPER-PARAMETERS
CLASSIC ML
hyperparameter configuration
TRAIN
model
model parameters
EVALUATE
SINGLE-CRITERION OPTIMIZATION
MULTI-CRITERIA OPTIMIZATION
SPEED UP
Meta-learning across datasets
Grey-box optimization/ learning curve prediction
Multi-fidelity optimization
NEURAL ARCHITECTURE SEARCH (NAS)
Search for the best neural network architecture on different hierarchical levels given a task.
META-LEARNING
Learn across tasks.
LEARNING TO LEARN
POPULATION-BASED TRAINING
TRAIN
model
START CONFIGURATION
DYNAMIC ALGORITHM CONFIGURATION (DAC)
INTERPRETE
MODEL(S)
EVALUATE
CONFIGURATION SPACE
Generic ML Algorithms
Random Forest
Linear Regression
...
ML ALGORITHMS
Neural Networks
PRE-PROCESSING
ALGORITHM SELECTION
Given a task choose the best algorithm based on performance prediction.
ALGORITHM PORTFOLIO

Sequential Experimentation

Why?

- Learn about the problem → *Active Learning*
- Find best setting → *Black-Box optimization*
- Minimize average regret → *Multi-armed Bandits*



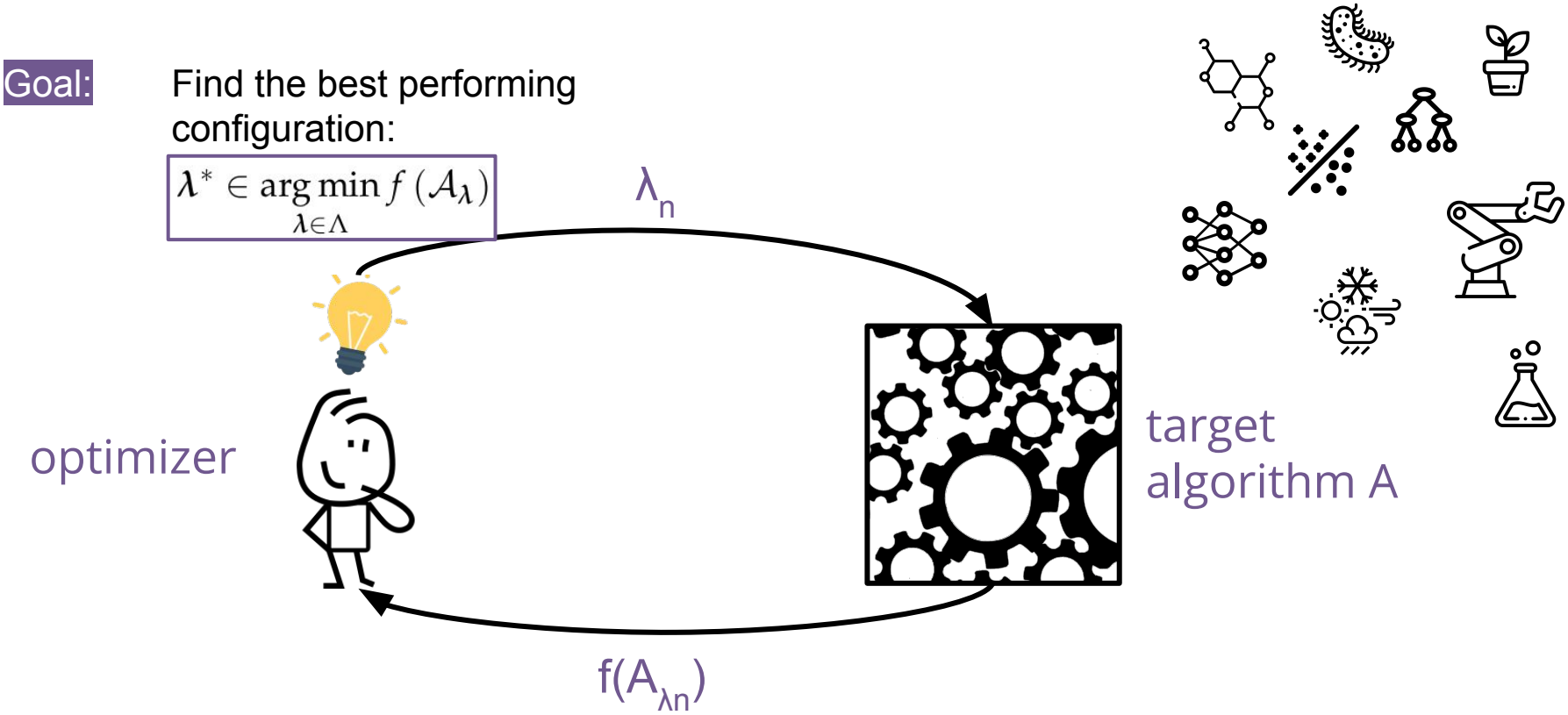


- Algorithm
- Neural Architecture
- Network Training
- Data Science Pipeline
- Simulator
- Robot

Sequential Experimentation

Goal: Find the best performing configuration:

$$\lambda^* \in \arg\min_{\lambda \in \Lambda} f\left(\mathcal{A}_\lambda\right)$$



Topics for Today

- What do we optimize?
  - Parameters vs. Hyperparameters
  - Challenges for AutoML

- How do we optimize it?
  - Grid Search
  - Random Search

- How do we optimize it efficiently?
  - Bayesian Optimization

(Hyper-) Parameters

>> What can we tune? What should we tune?

Model- vs Hyperparameters

Model parameters can be optimized during training and are the output of the training.

Examples:

- Splits of a Tree
- Weights of a Neural Network
- Coefficients of a linear model

Hyperparameters need to be set before training and control the flexibility, structure and complexity of the model and training procedure.

Examples:

- Learning Rate for Gradient Boosting
- Optimizer for Neural Network Training
- K for K-Nearest Neighbours

They can be:

- **real-valued, integer** and **categorical**
- **hierarchically dependent** on each other
- be on a **log-scale**

Why is Hyperparameter Optimization Challenging?

Goal: Find the best performing configuration:

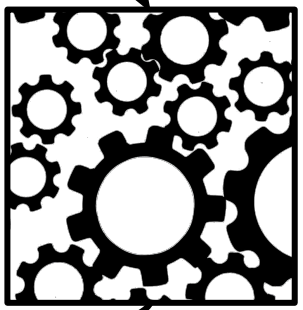
$$\lambda^* \in \arg\min_{\lambda \in \Lambda} f(\mathcal{A}_\lambda)$$

complex search space

No gradients
No prior knowledge

λ_n

optimizer



target algorithm A

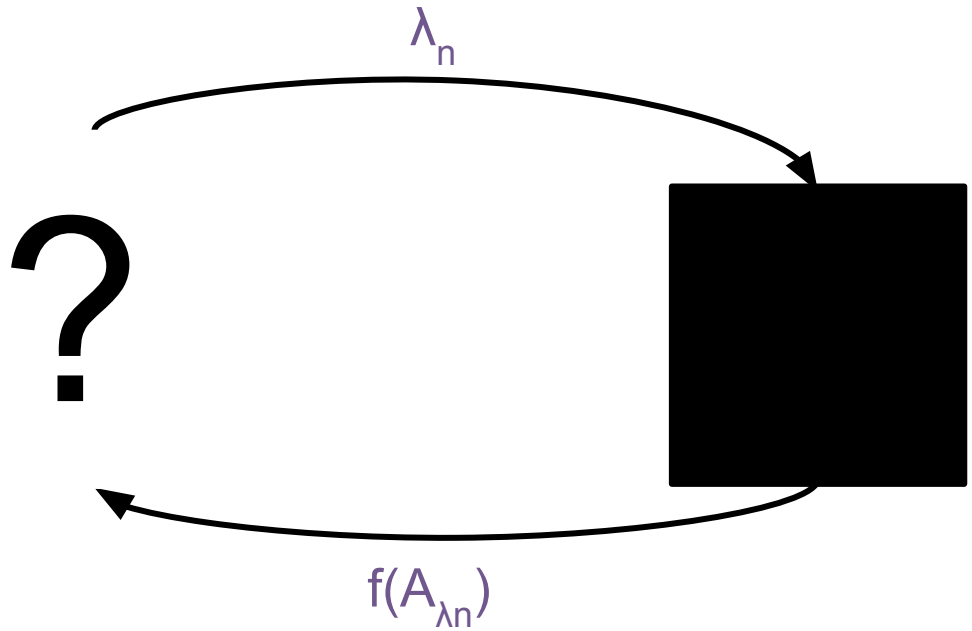
$f(A_{\lambda n})$

noisy
expensive-to-evaluate

How do we optimize it?

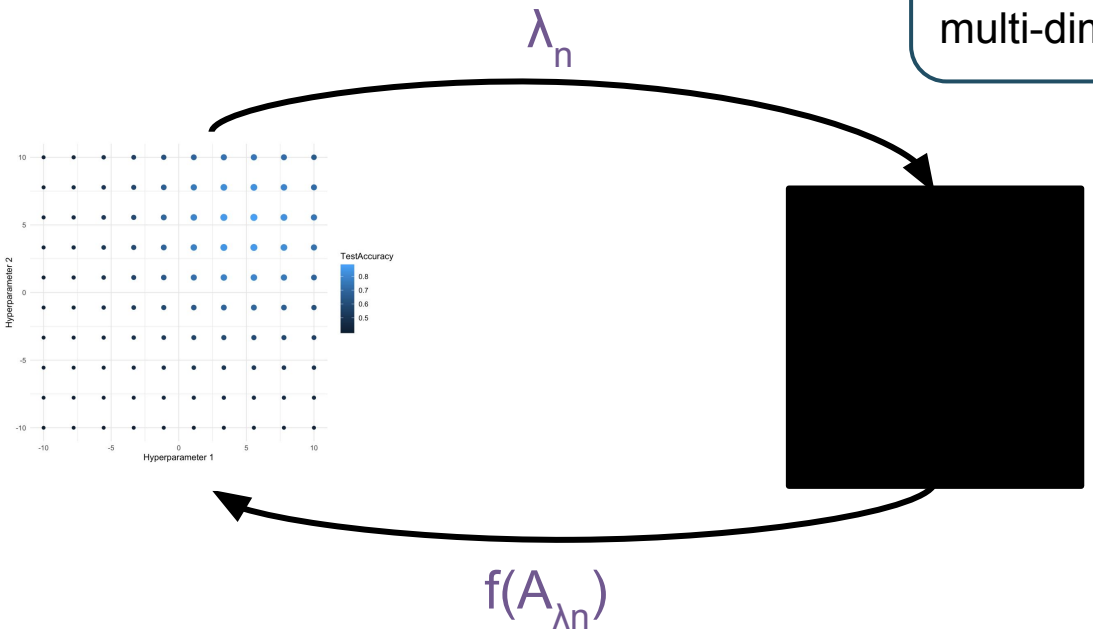
>> Here's my algorithm, data, metric and search space, what should I do?

Black-Box Optimization Problem

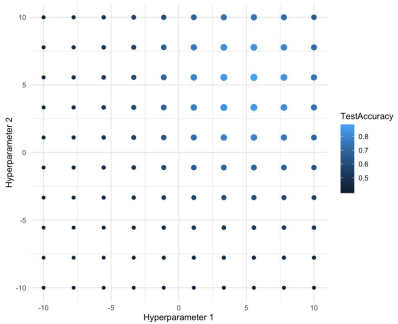


Option 1: Grid Search

Popular technique: Evaluates all combinations on a pre-defined multi-dimensional grid



Option 1: Grid Search II



Advantages

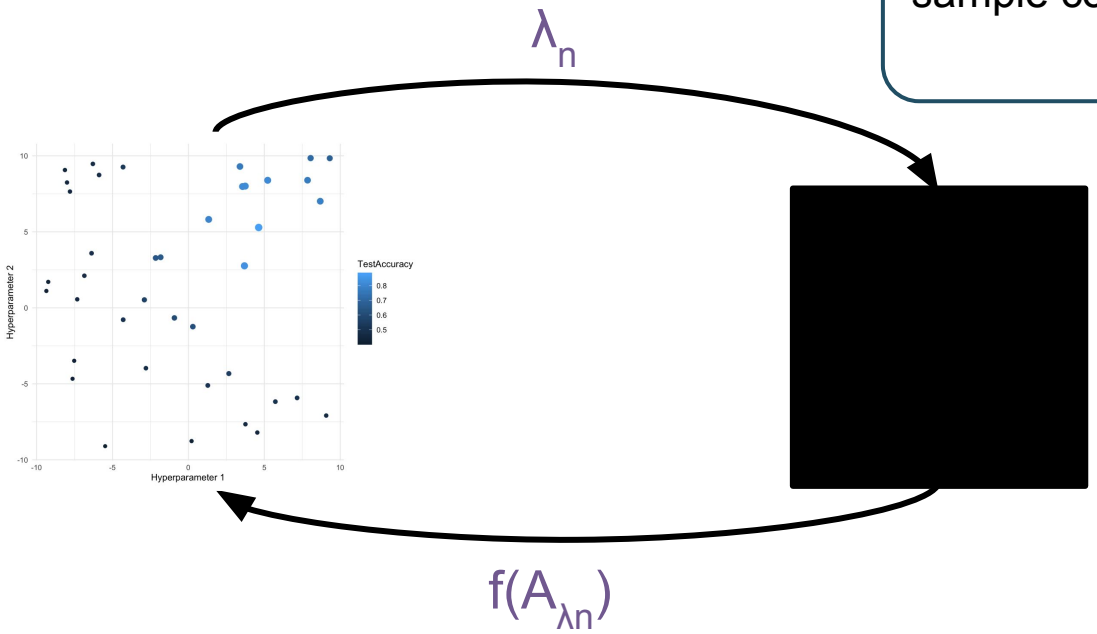
- Very easy to implement
- Very easy to parallelize
- Can handle all types of hyperparameters

Disadvantages

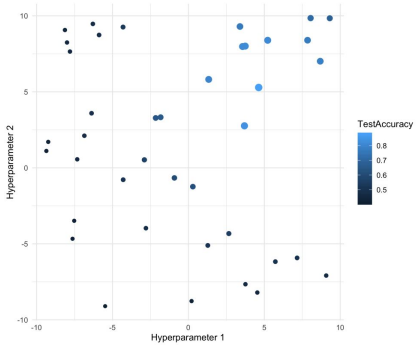
- Scales badly with #dimensions
- Inefficient: Searches irrelevant areas
- Requires to manual define discretization
- All grid points need to be evaluated

Option 2: Random Search

Variation of Grid Search: Uniformly sample configurations at random



Option 2: Random Search II



Advantages

- Very easy to implement
- Very easy to parallelize
- Can handle all types of hyperparameters
- No discretization required
- Anytime algorithm: Can be stopped and continued based on the available budget and performance goal.

Disadvantages

- Scales badly with #dimensions
- Inefficient: Searches irrelevant areas

Grid Search vs. Random Search

With a **budget** of T iterations:

Grid Search evaluates only $T^{\frac{1}{d}}$ unique values per dimension

Random Search evaluates (most likely) T different values per dimension

→ Grid search can be disadvantageous if some hyperparameters have little of no impact on the performance [Bergstra et al. 2012]

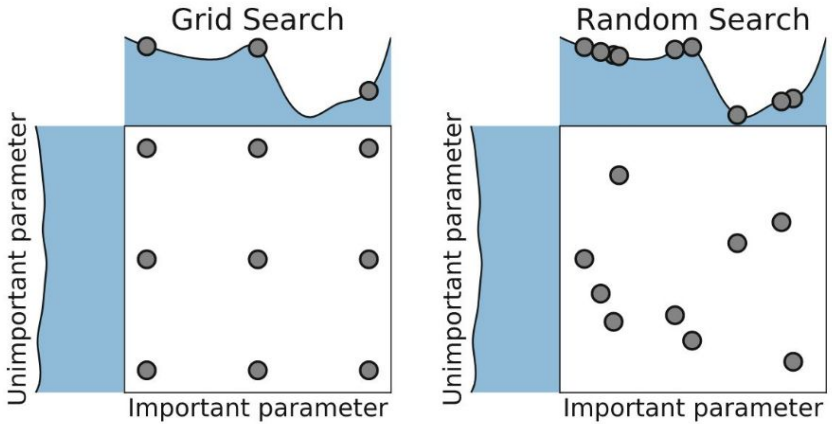


Image source: [Hutter et al. 2019]

Questions?

How do we optimize it efficiently?

>> Here's my algorithm, data and design space and I have only limited time, what should I do?

Model-based Optimization

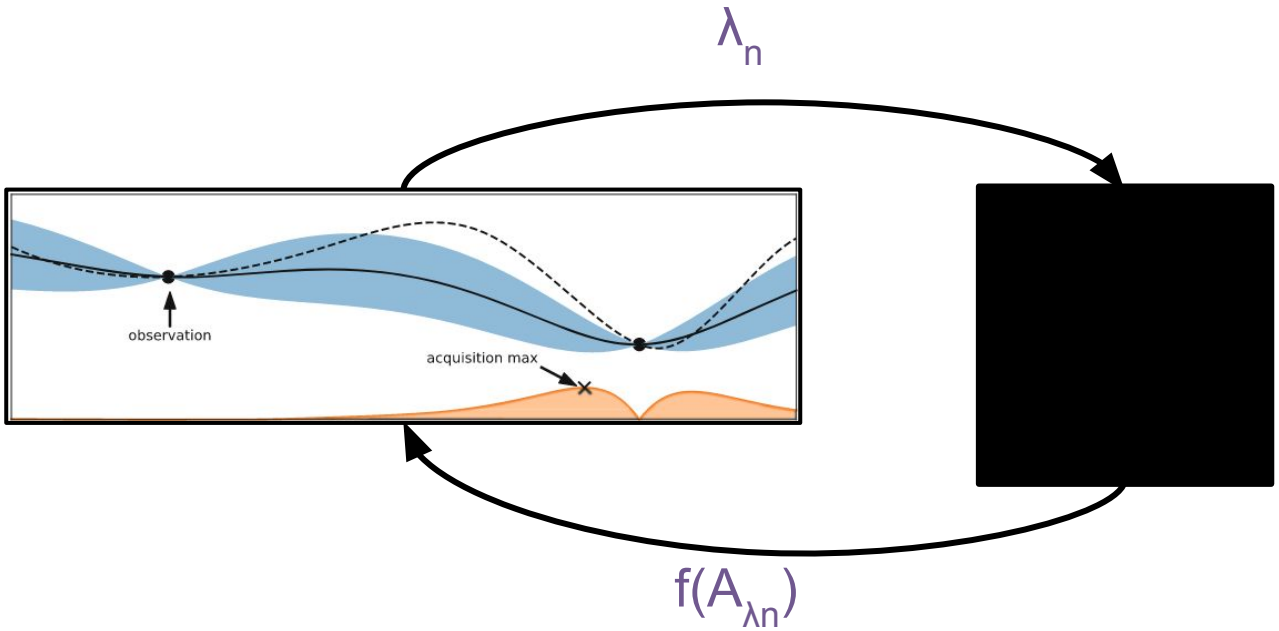
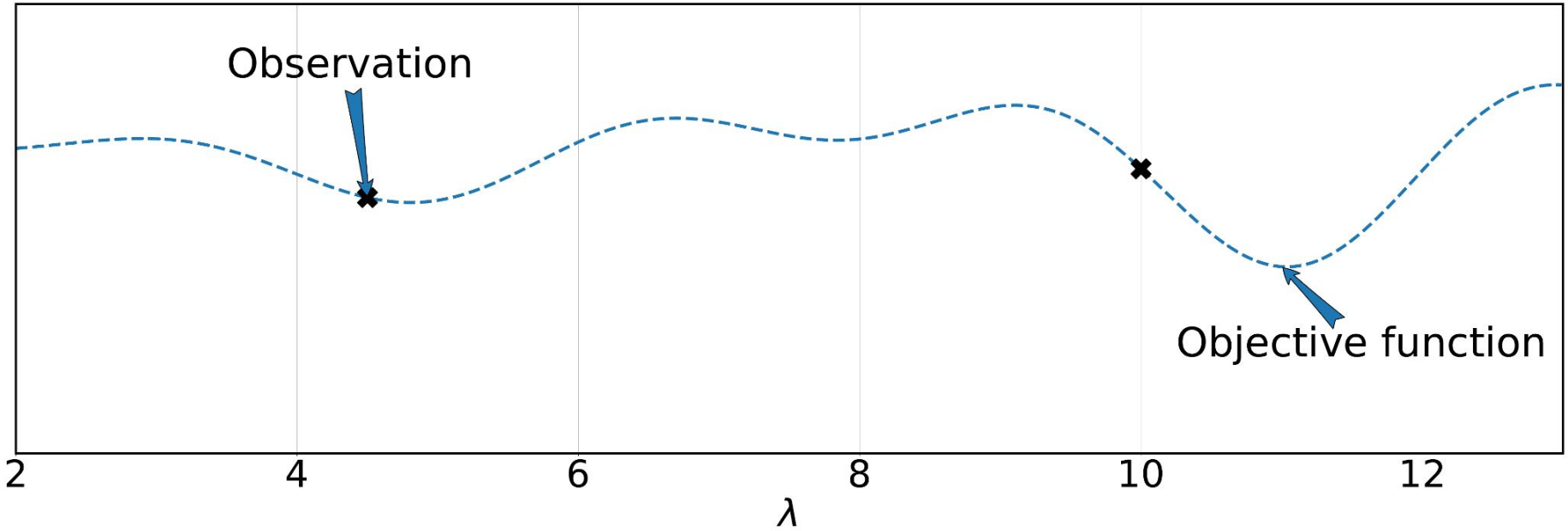
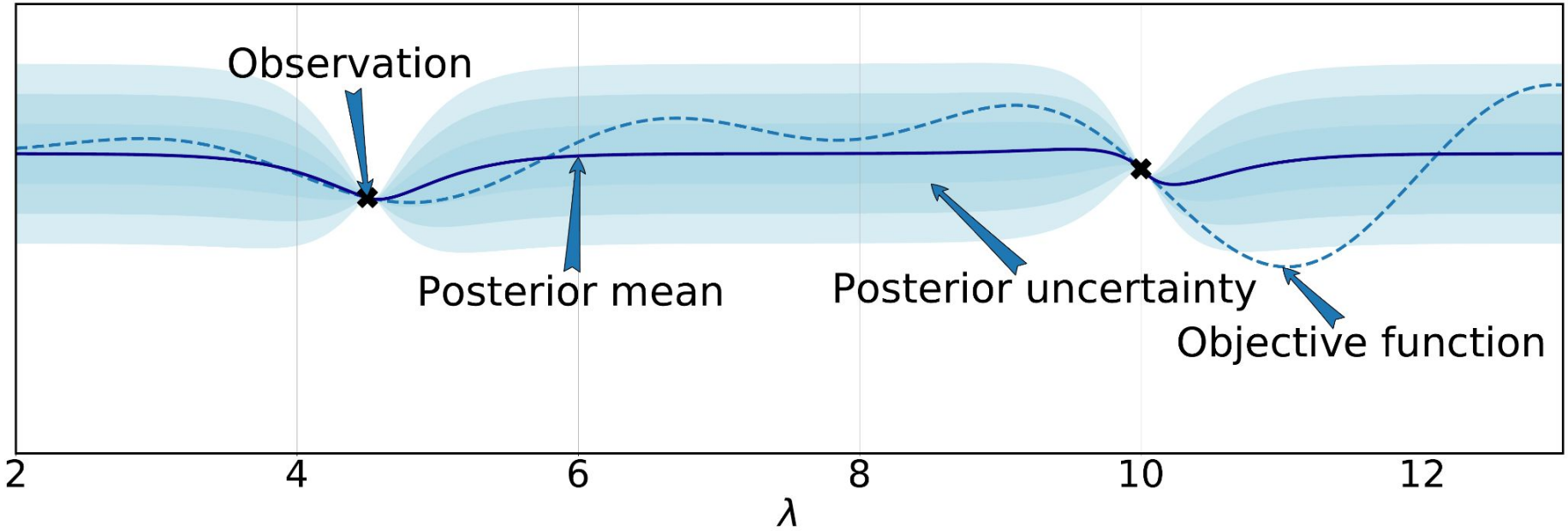


Photo by Wilhelm Gunkel on Unsplash
Image by Feurer, Hutter: Hyperparameter Optimization.
In: Automated Machine Learning

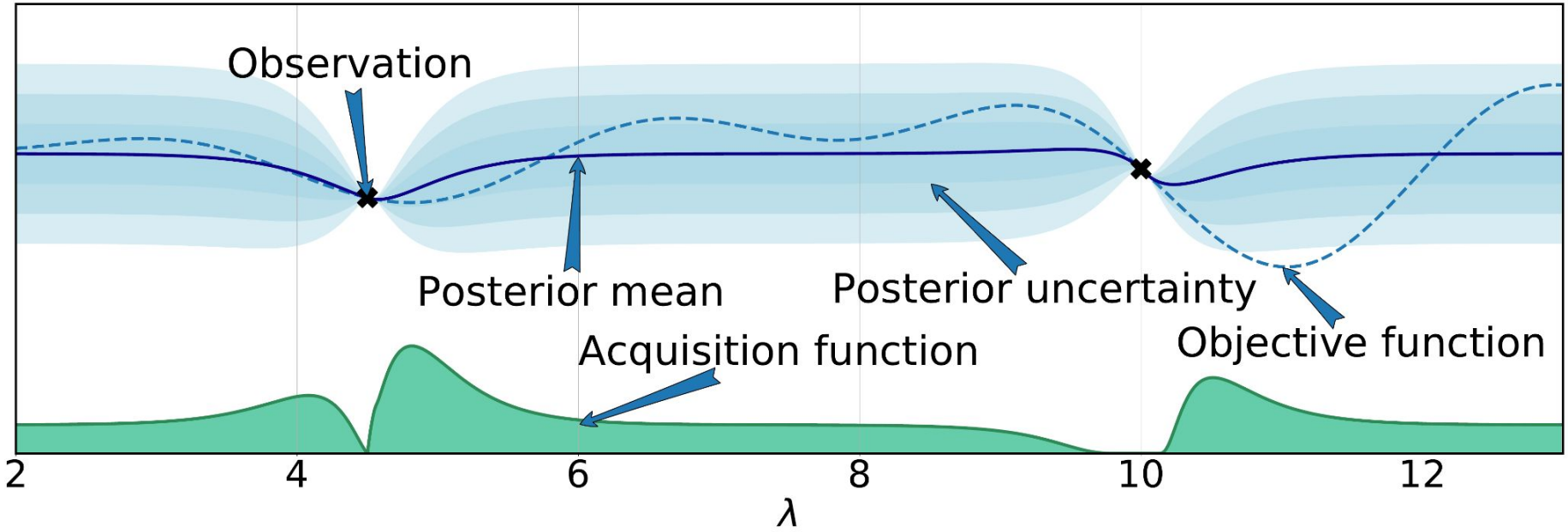
Bayesian Optimization in a Nutshell



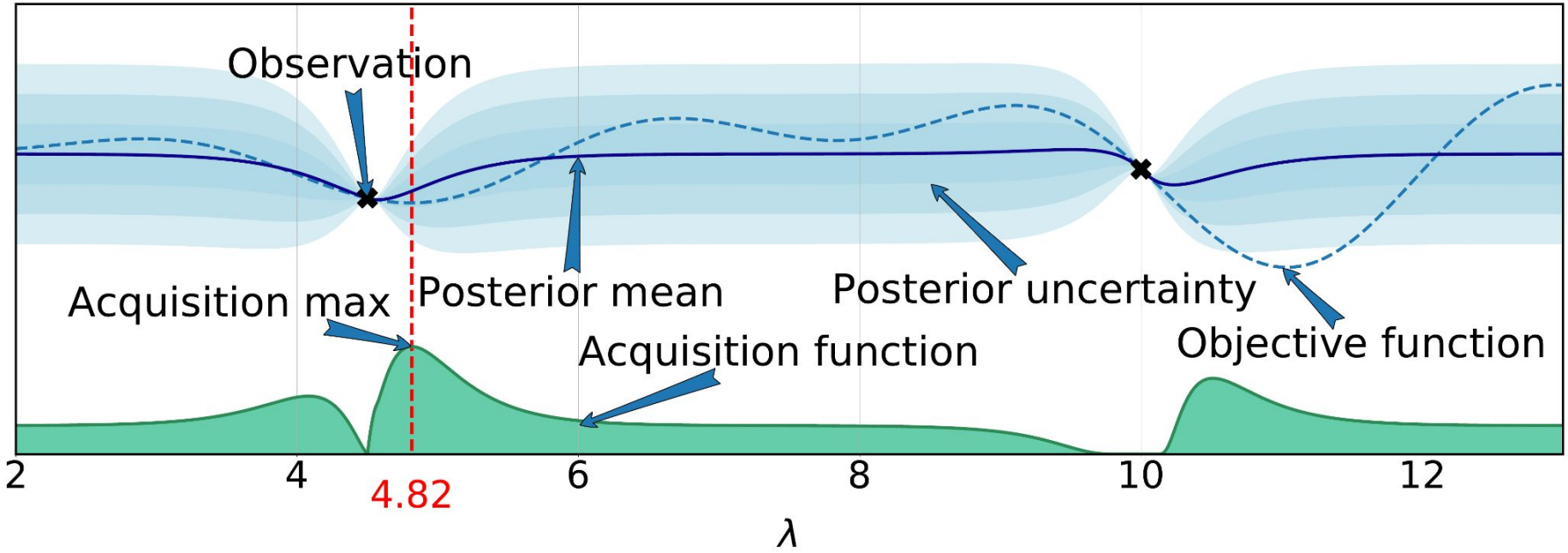
Bayesian Optimization in a Nutshell



Bayesian Optimization in a Nutshell



Bayesian Optimization in a Nutshell



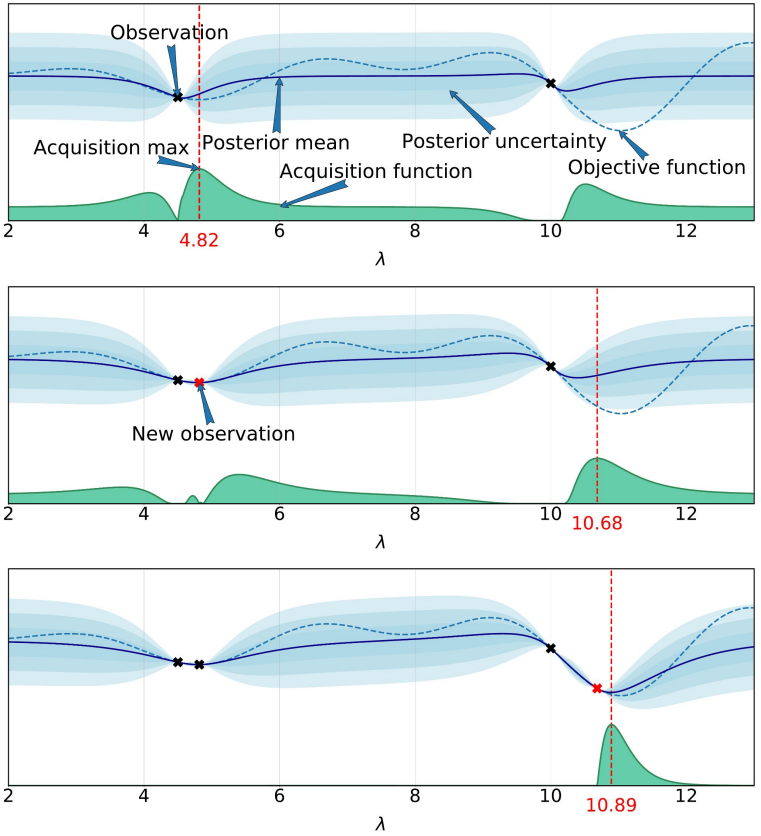
Bayesian Optimization in a Nutshell

General approach

- Fit a probabilistic model to the collected function samples $\langle \boldsymbol{\lambda}, c(\boldsymbol{\lambda}) \rangle$
- Use the model to guide optimization, trading off exploration *vs* exploitation

Popular approach in the statistics literature since Mockus et al. [1978]

- Efficient in #function evaluations
- Works when objective is nonconvex, noisy, has unknown derivatives, etc.
- Recent convergence results
  [Srinivas et al. 2009; Bull et al. 2011; de Freitas et al. 2012; Kawaguchi et al. 2015]



Bayesian Optimization: Pseudocode

BO loop

Require: Search space $\boldsymbol{\Lambda}$, cost function c, acquisition function u, predictive model $\hat{c}$, maximal number of function evaluations T

Result : Best configuration $\hat{\boldsymbol{\lambda}}$ (according to $\mathcal{D}$ or $\hat{c}$)

1 Initialize data $\mathcal{D}^{(0)}$ with initial observations

2 **for** $t = 1$ **to** T **do**

3 Fit predictive model $\hat{c}^{(t)}$ on $\mathcal{D}^{(t-1)}$

4 Select next query point: $\boldsymbol{\lambda}^{(t)} \in \arg\max_{\boldsymbol{\lambda} \in \boldsymbol{\Lambda}} u(\boldsymbol{\lambda}; \mathcal{D}^{(t-1)}, \hat{c}^{(t)})$

5 Query $c(\boldsymbol{\lambda}^{(t)})$

6 Update data: $\mathcal{D}^{(t)} \leftarrow \mathcal{D}^{(t-1)} \cup \{\langle \boldsymbol{\lambda}^{(t)}, c(\boldsymbol{\lambda}^{(t)}) \rangle\}$

Why is it called Bayesian Optimization?

- Bayesian optimization uses Bayes' theorem:

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)} \propto P(B|A) \times P(A)$$

- Bayesian optimization uses this to compute a posterior over functions:

$$P(f|\mathcal{D}_{1:t}) \propto P(\mathcal{D}_{1:t}|f) \times P(f), \qquad \text{where } \mathcal{D}_{1:t} = \{\boldsymbol{\lambda}_{1:t}, c(\boldsymbol{\lambda}_{1:t})\}$$

Meaning of the individual terms:

- $P(f)$ is the prior over functions, which represents our belief about the space of possible objective functions before we see any data
- $\mathcal{D}_{1:t}$ is the data (or observations, evidence)
- $P(\mathcal{D}_{1:t}|f)$ is the likelihood of the data given a function
- $P(f|\mathcal{D}_{1:t})$ is the posterior probability over functions given the data

Bayesian Optimization: Pros and Cons

Advantages

- Sample efficient
- Can handle noise
- Priors can be incorporated
- Does not require gradients
- Theoretical guarantees

Many extensions available:
Multi-Objective | Multi-Fidelity | Parallelization | Warmstarting | etc.

Disadvantages

- Overhead because of model training
- Crucially relies on robust surrogate model
- Has quite a few design decisions

Main Ingredient I: The Acquisition Function

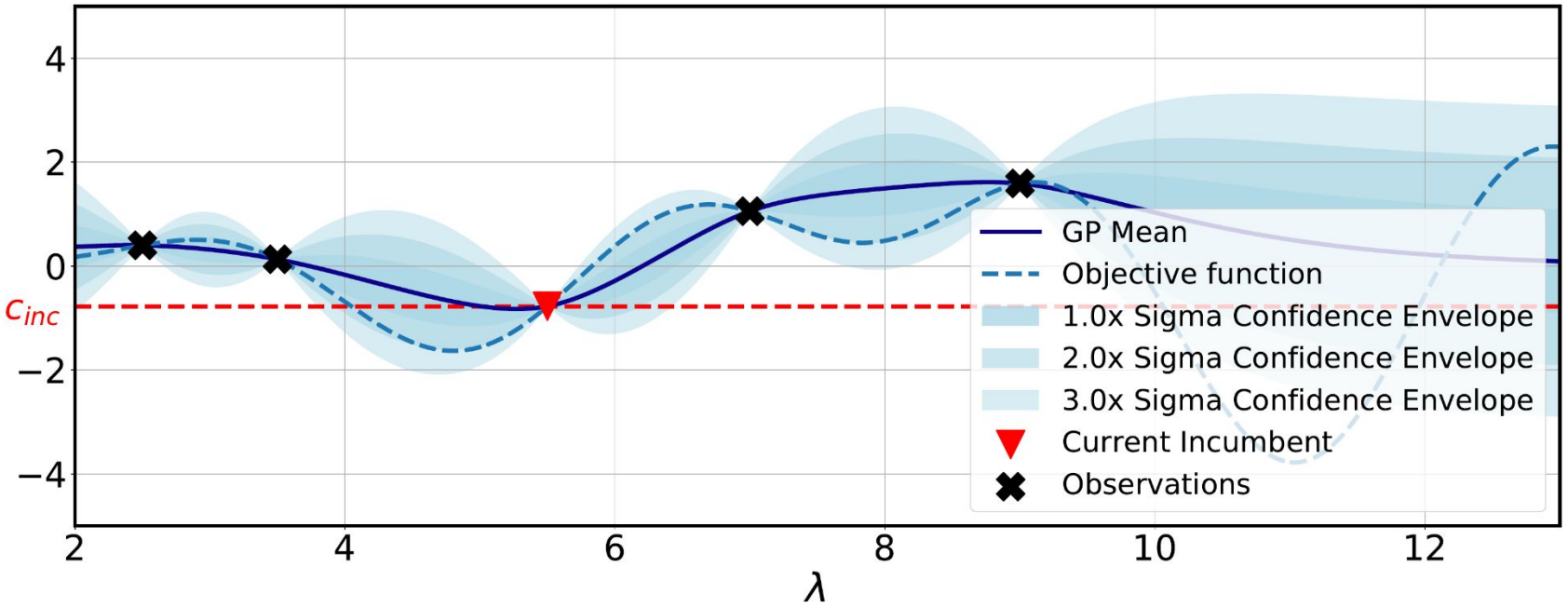
The acquisition function:

- decides which configuration to evaluate next
- judges the **utility** (or **usefulness**) of evaluating a configuration (based on the surrogate model)

→ It needs to trade-off **exploration and exploitation**

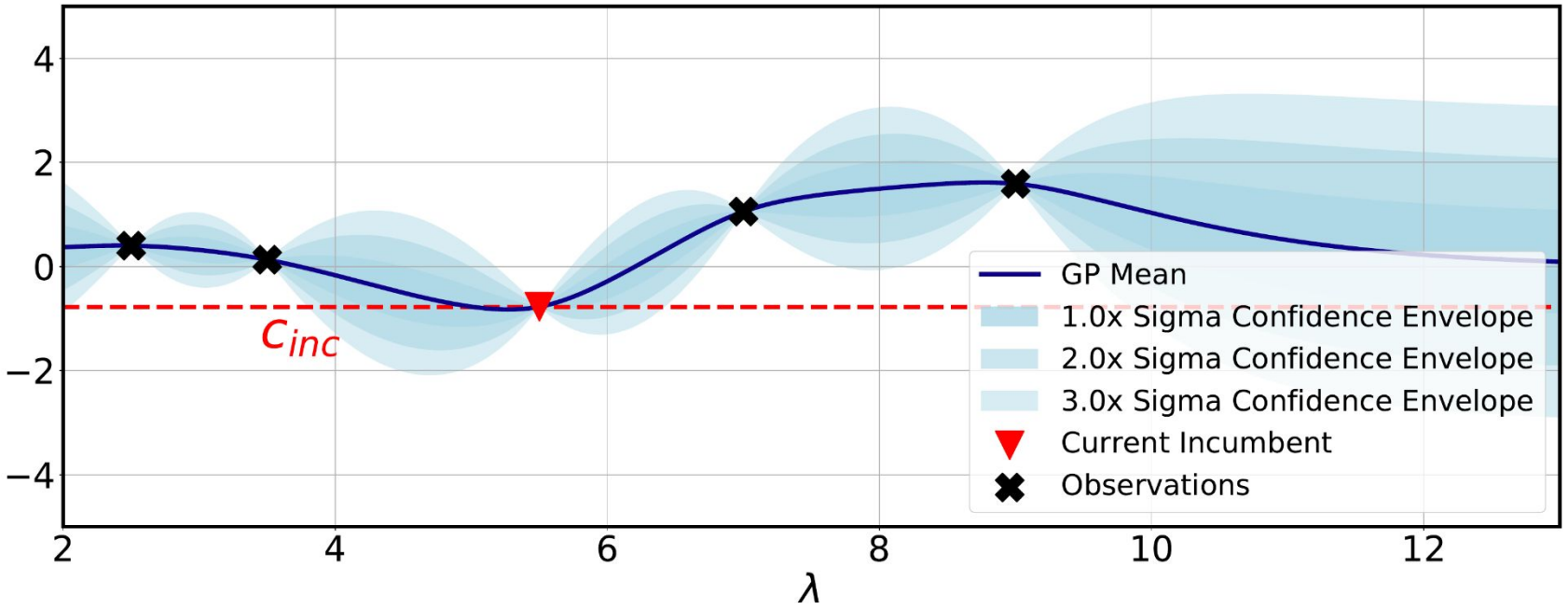
- Just picking the configuration with the lowest prediction would be too greedy
- It needs to consider the uncertainty of the surrogate model

Expected Improvement (EI)



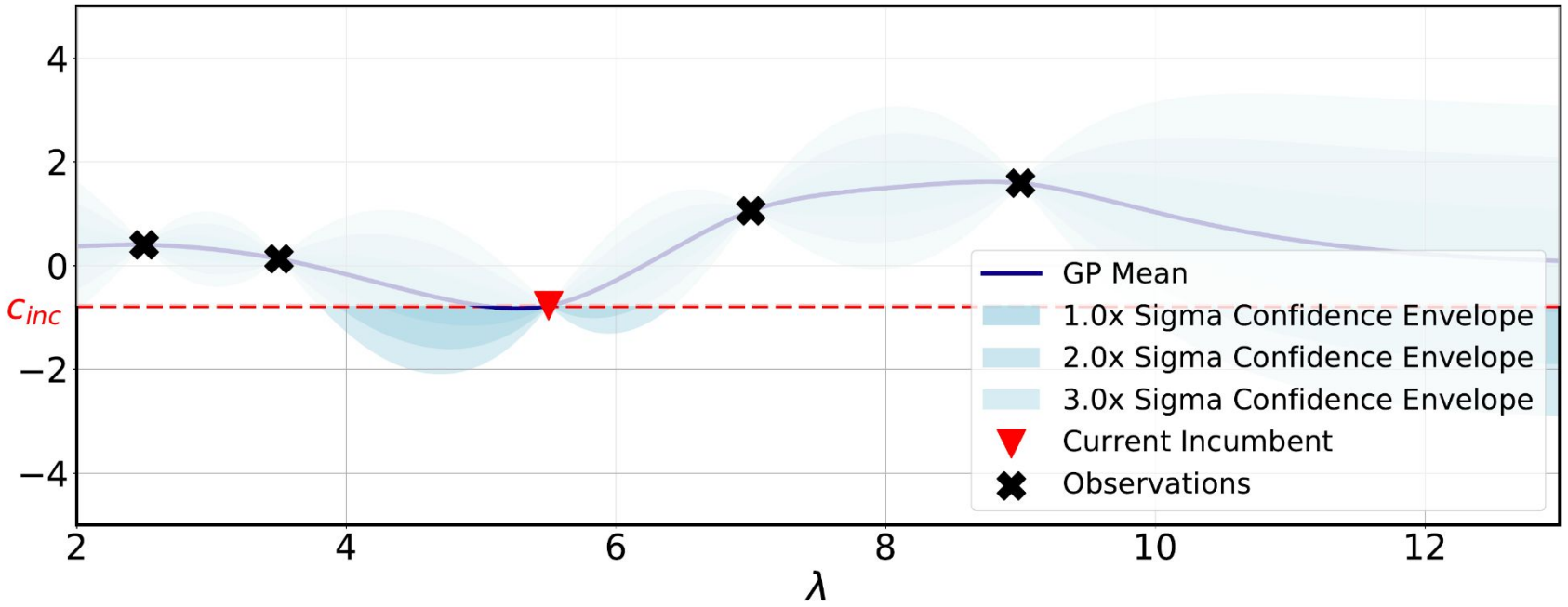
Given some observations and a fitted surrogate,

Expected Improvement (EI)



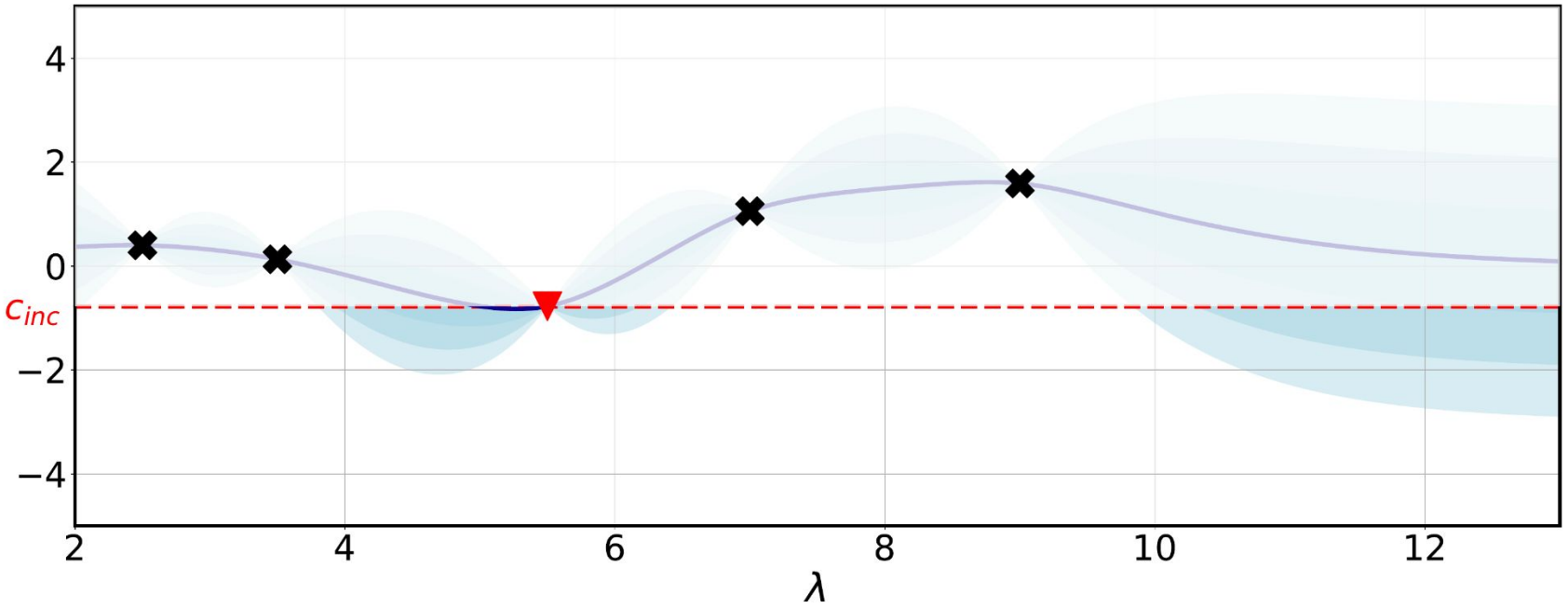
Given some observations and a fitted surrogate,

Expected Improvement (EI)



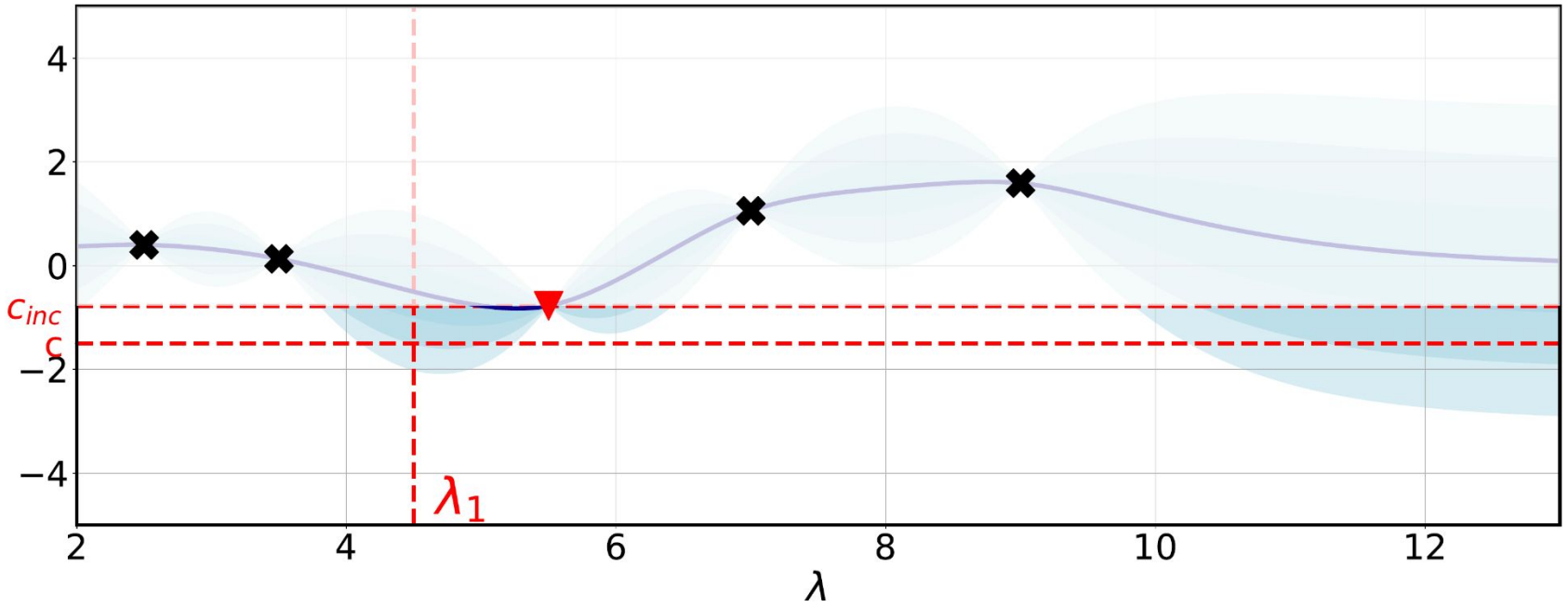
We care about *improving* over the c_{inc}.

Expected Improvement (EI)



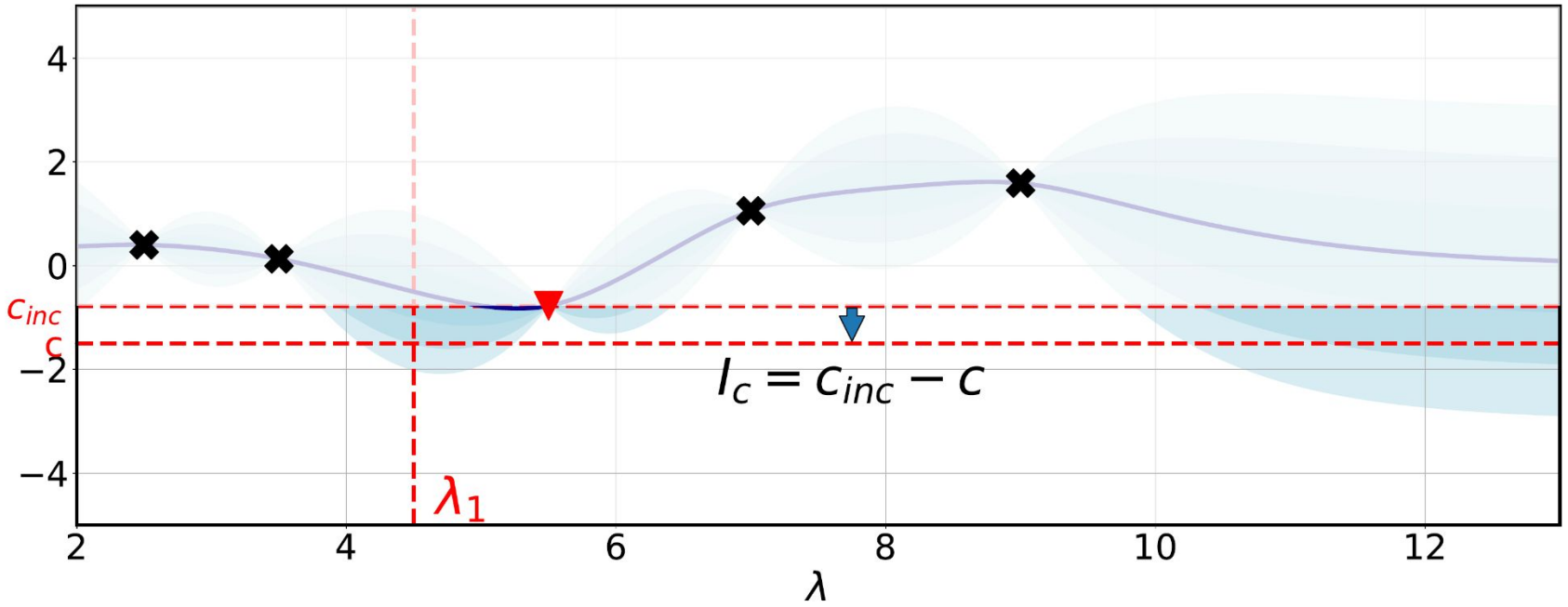
We care about *improving* over the c_{inc}.

Expected Improvement (EI)



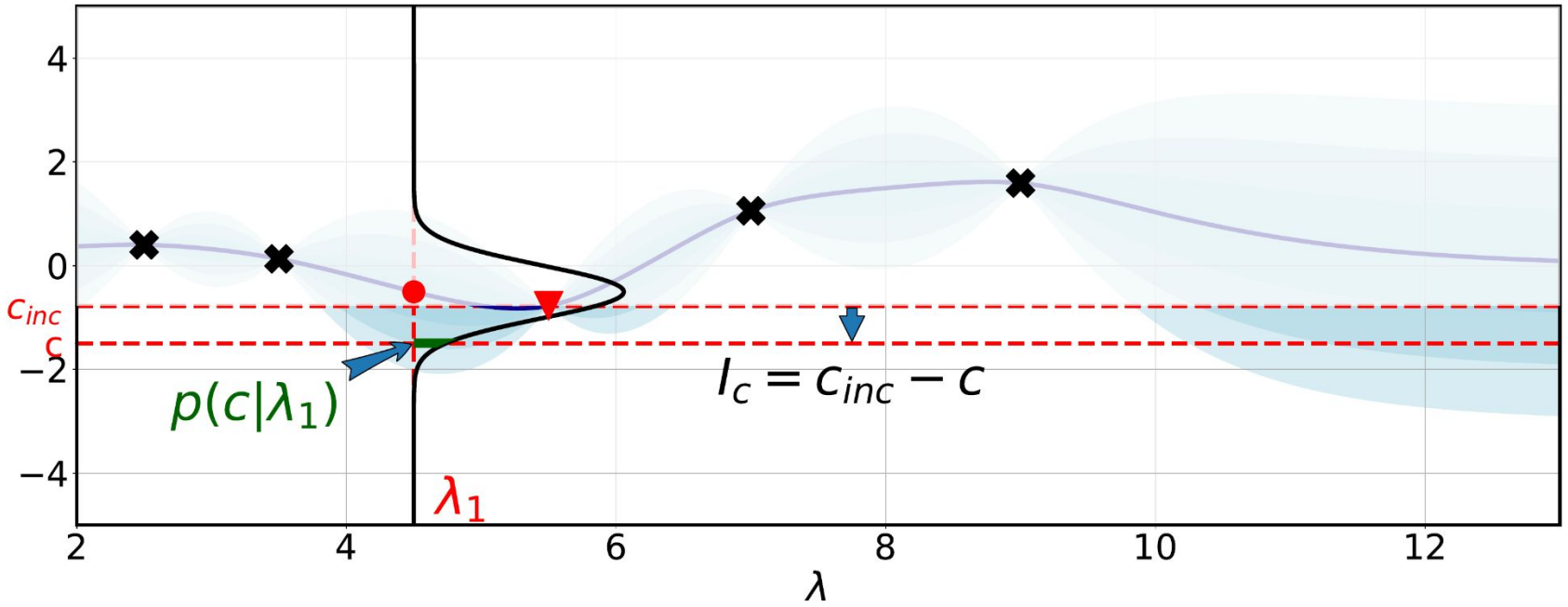
Let's look at a candidate configuration λ_1 and its hypothetical cost c.

Expected Improvement (EI)



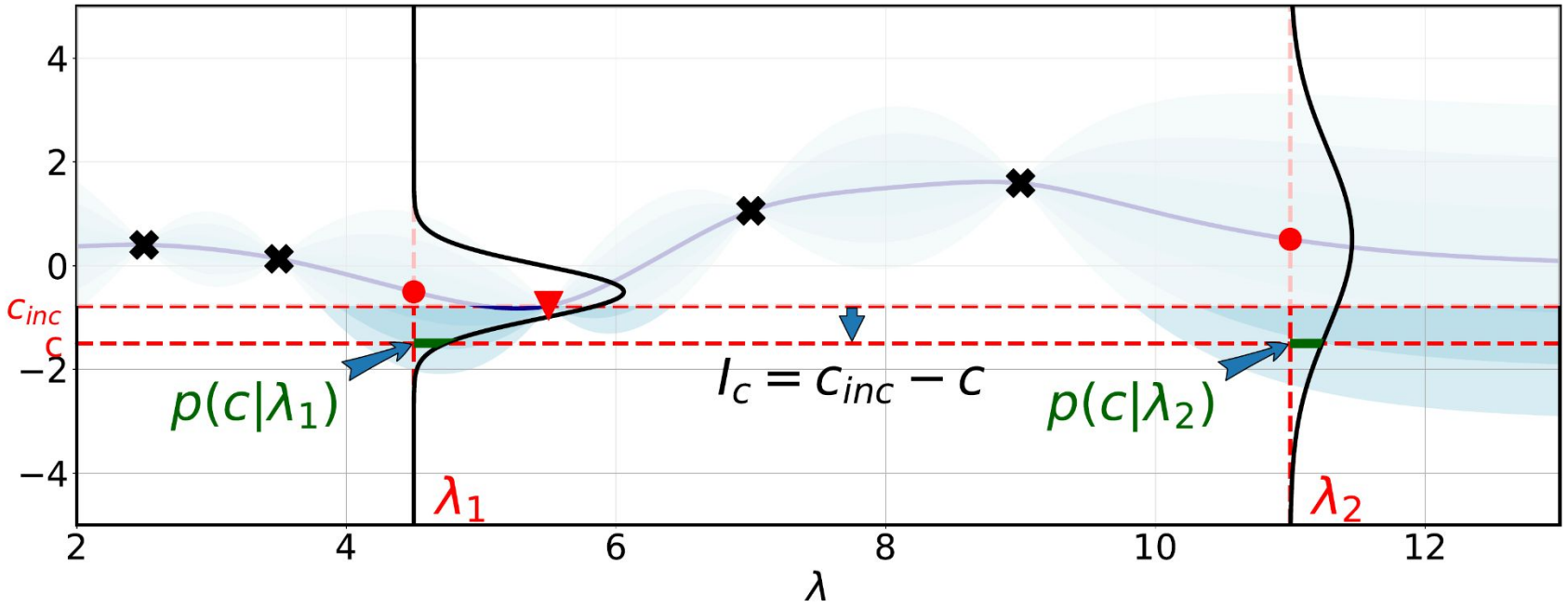
We can compute the improvement $I_c(\lambda_1)$. But how likely is it?

Expected Improvement (EI)



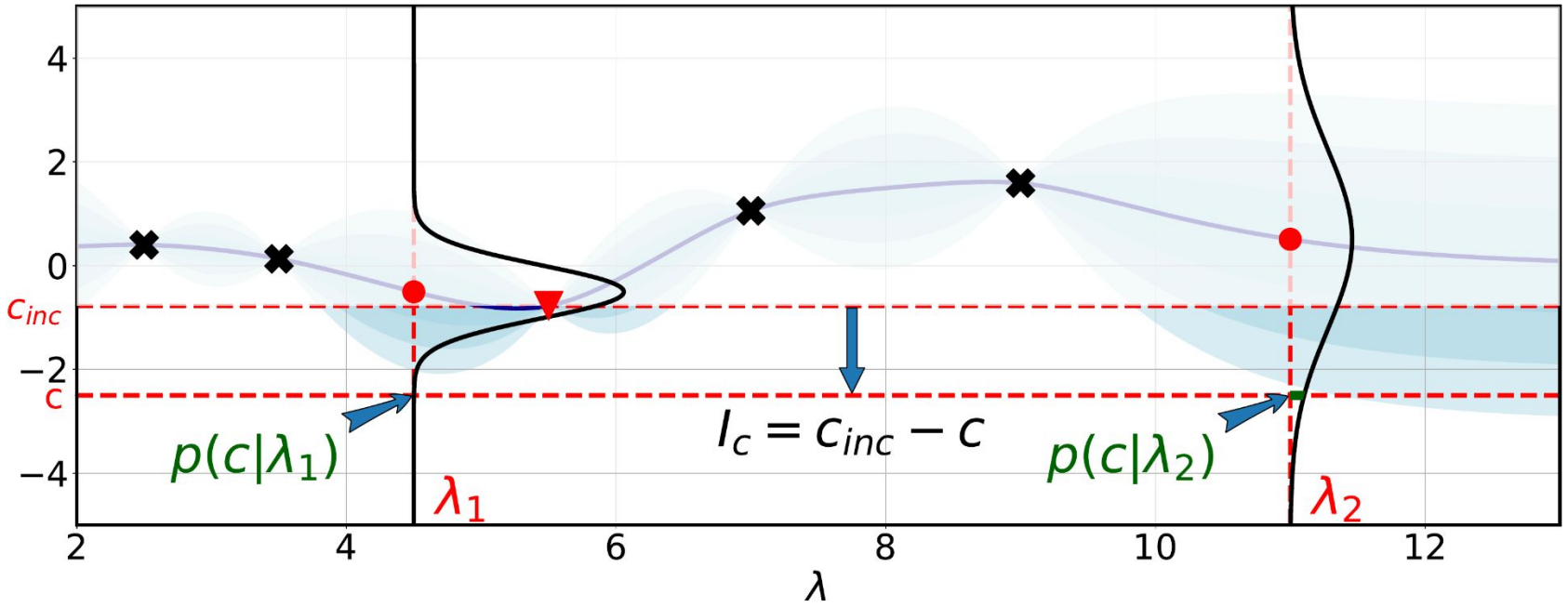
Knowing that $\hat{c}(\boldsymbol{\lambda}) = \mathcal{N}(\mu(\boldsymbol{\lambda}), \sigma^2(\boldsymbol{\lambda}))$, we can compute $p(c|\boldsymbol{\lambda})$

Expected Improvement (EI)



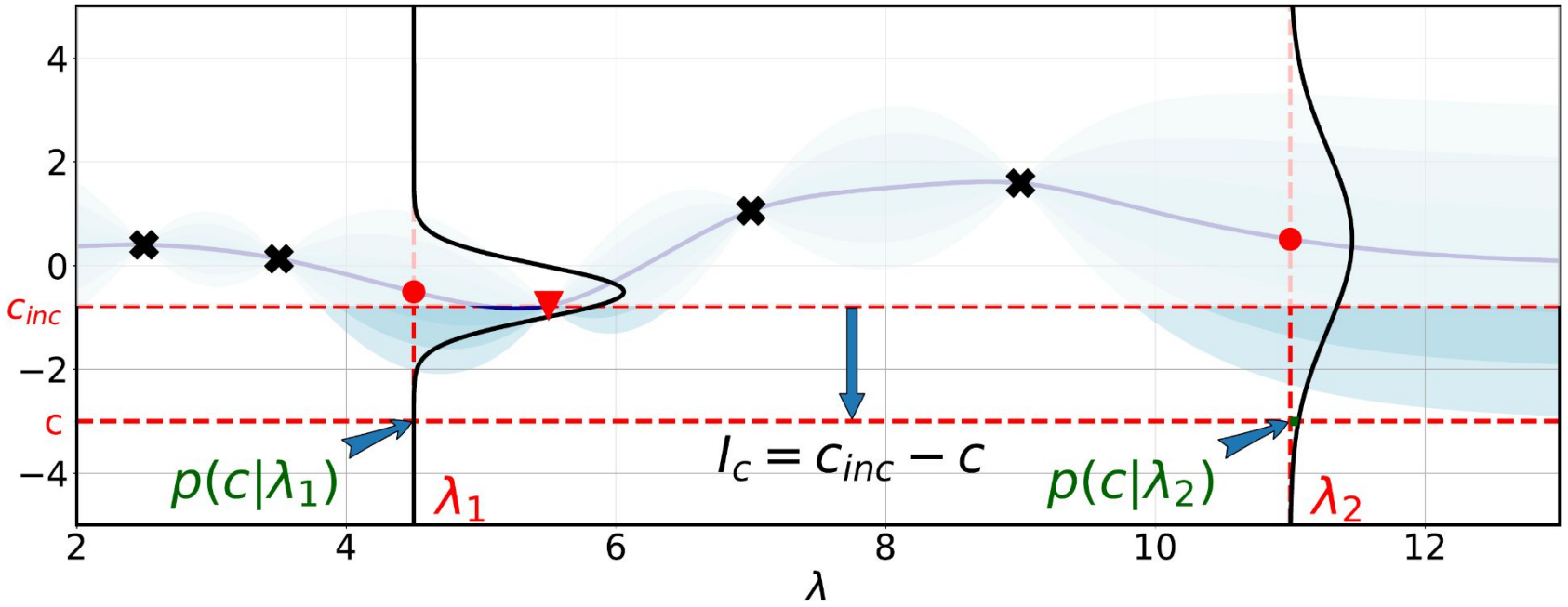
Comparing this for different configurations

Expected Improvement (EI)



and costs.

Expected Improvement (EI)



To compute EI, we sum all $p(c \mid \boldsymbol{\lambda}) \times I_c$ over all possible cost values.

Expected Improvement (EI) - Formal Definition

We define the one-step positive improvement over the current incumbent as

$$I^{(t)}(\boldsymbol{\lambda}) = \max(0, c_{inc} - c(\boldsymbol{\lambda}))$$

Expected Improvement is then defined as

$$u_{EI}^{(t)}(\boldsymbol{\lambda}) = \mathbb{E}[I^{(t)}(\boldsymbol{\lambda})] = \int_{-\infty}^{\infty} p^{(t)}(c \mid \boldsymbol{\lambda}) \times I^{(t)}(\boldsymbol{\lambda}) \; dc.$$

$$\text{Choose } \boldsymbol{\lambda}^{(t)} \in \arg\max_{\boldsymbol{\lambda} \in \boldsymbol{\Lambda}} (u_{EI}^{(t)}(\boldsymbol{\lambda}))$$

Other Acquisition Functions

- **Improvement-based** policies [Expected Improvement (EI), Probability of Improvement (PI), and Knowledge Gradient]
- **Optimistic** policies [Upper/Lower Confidence Bound (UCB/LCB)]
- **Information-based** policies [Entropy Search (ES)]
  - aim to increase certainty about the location of the minimizer
  - not necessarily evaluate promising configurations
- Methods **combining/mixing/switching** these

Questions?

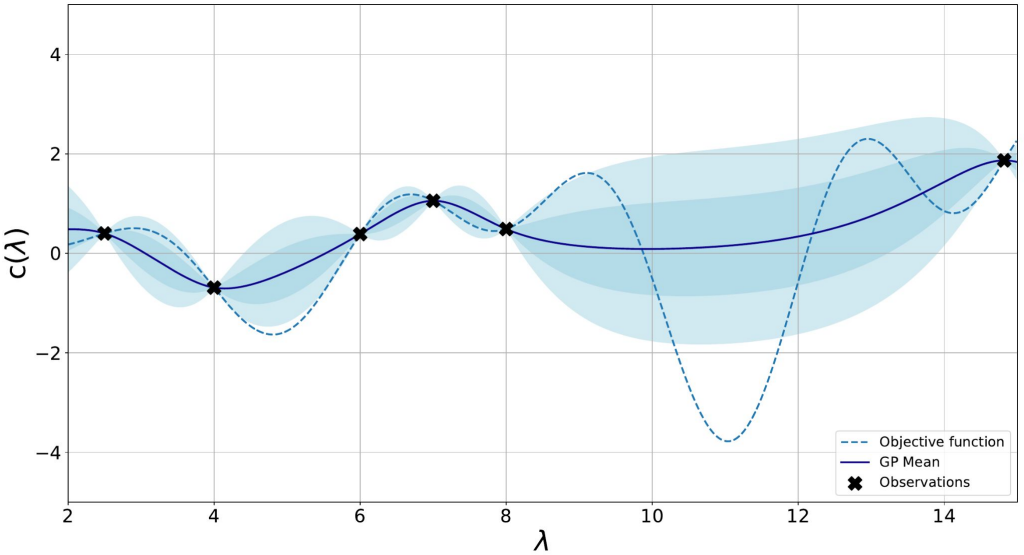
Main Ingredient II: The Surrogate Model

Required in all cases

- Regression model with uncertainty estimates
- Accurate predictions

Depending on the application

- Cheap to train
- Scales well with #observations and #dimensions
- Can handle different types of hyperparameters



Types of Surrogates Models

- Gaussian Processes
- Random Forests
- Bayesian Neural Networks

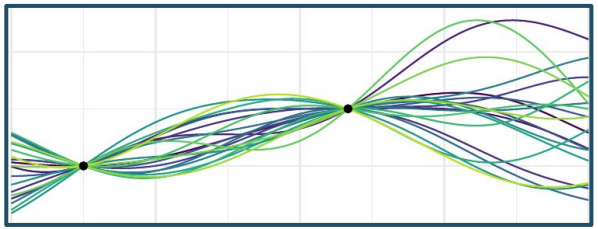
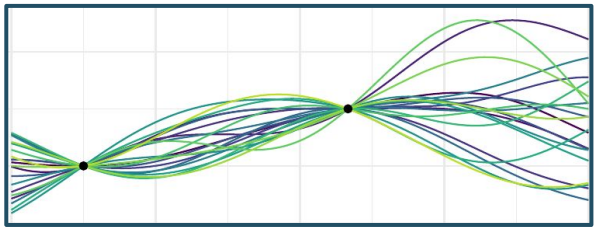


Photo by Filip Zrnzević on Unsplash
Photo by Alina Grubnyak on Unsplash

Gaussian Processes

$$
\begin{aligned}
m(\mathbf{x}) &= \mathbb{E}[f(\mathbf{x})] \\
k(\mathbf{x}, \mathbf{x}') &= \mathbb{E}\left[(f(\mathbf{x}) - \mathbb{E}[f(\mathbf{x})])\,(f(\mathbf{x}') - \mathbb{E}[f(\mathbf{x}')])\right] \\
f(\mathbf{x}) &\sim \mathcal{G}\left(m(\mathbf{x}), k\left(\mathbf{x}, \mathbf{x}'\right)\right)
\end{aligned}
$$



Advantages

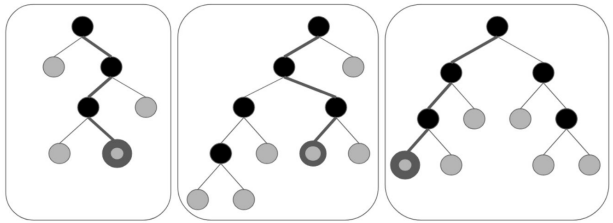
- Smooth uncertainty estimates
- Strong sample efficiency
- Expert knowledge can be encoded in the kernel
- Accurate predictions

Disadvantages

- Cost scales cubically with #observations
- Weak performance for high dimensionality
- Not easily applicable in discrete, categorical or conditional spaces
- Sensitive wrt its own hyperparameters

→ These make GPs the most commonly used model for Bayesian optimization

Tree-Based Methods



Advantages

- Scales well with #dimensions and #observations
- Training can be parallelized and is fast
- Can easily handle discrete, categorical and conditional spaces
- Robust wrt. its own hyperparameters

Disadvantages

- Poor uncertainty estimates
- Poor extrapolation (constant)
- Expert knowledge can not be easily incorporated

→ These make RFs a robust option in high dimensions, a high number of evaluations and for mixed spaces

Photo by Filip Zrnzević on Unsplash
Photo by Alina Grubnyak on Unsplash

Bayesian Neural Networks

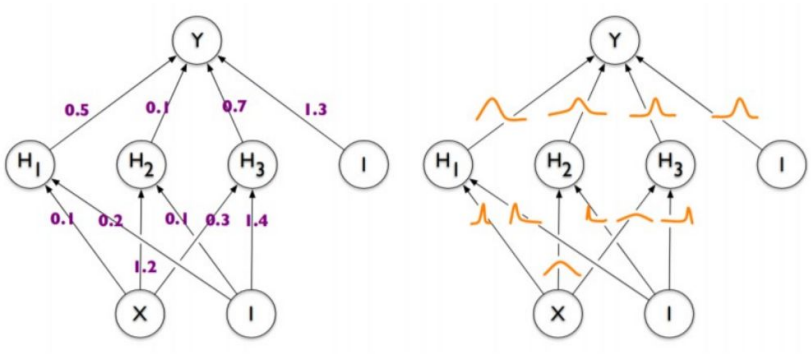


Image source: [Blundell et al. 2015]

Advantages

- Scales linear #observations
- (Can yield) smooth uncertainty estimates
- Flexibility wrt. discrete and categorical spaces

Disadvantages

- Needs many #observations
- Uncertainty estimates often worse than for GPs
- Many hyperparameters
- No robust off-the-shelf model

→ These make BNNs a promising alternative. [Li et al. 2023]

Photo by Filip Zrnzević on Unsplash
Photo by Alina Grubnyak on Unsplash

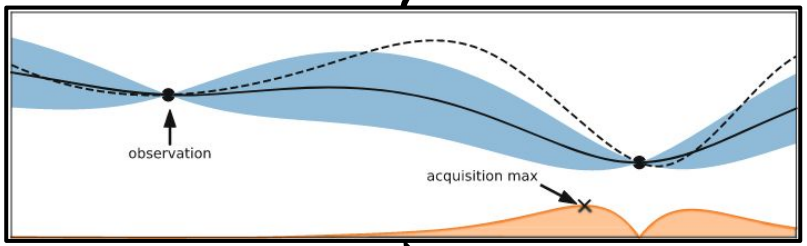
Bayesian Optimization: Extensions

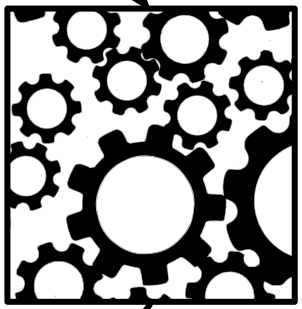
Replace everything with a LLM

Different models to handle new design spaces

Increase efficiency by using lower fidelities

λ_n





Multi-objective

Constraints?

User priors?

Leverage observations across experiments

$f(A_{\lambda n})$

Questions?