EBERHARD KARLS UNIVERSITÄT TÜBINGEN

machine learning
new perspectives
for science

Seminar Summer Term 2024

Bayesian Optimization for HPO

Katharina Eggensperger

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

April 23rd, 2024

AutoML.org

Story Line Today

[?] Questions regarding the organization

[60min] **Bayesian Optimization for HPO**

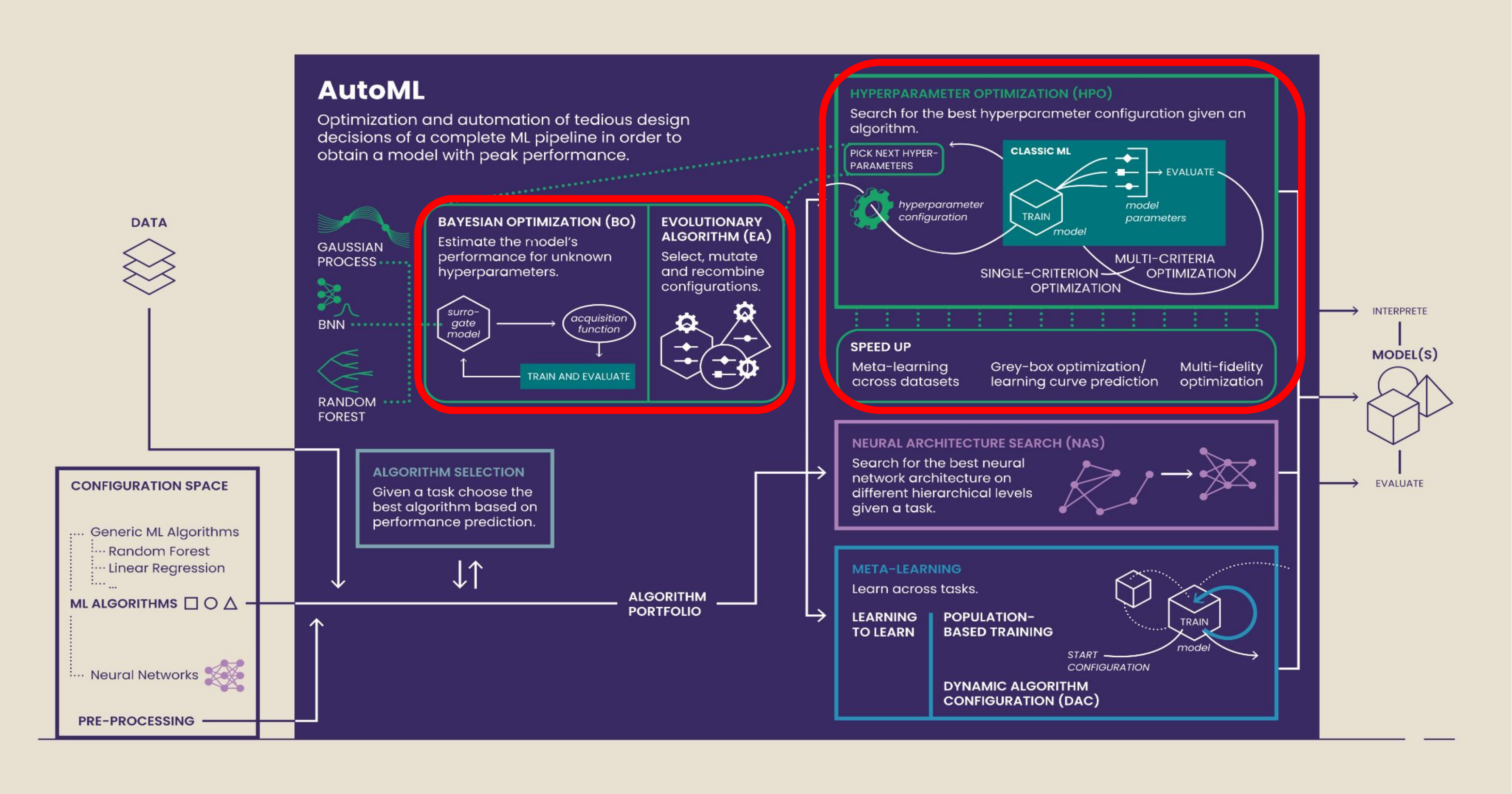
[?] Your Questions

[25min] **How to give a good presentation**

[?] 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
EVALUATE
model parameters
MULTI-CRITERIA OPTIMIZATION
SINGLE-CRITERION 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.
ALGORITHM SELECTION
Given a task choose the best algorithm based on performance prediction.
CONFIGURATION SPACE
Generic ML Algorithms
Random Forest
Linear Regression
...
ML ALGORITHMS
Neural Networks
PRE-PROCESSING
ALGORITHM PORTFOLIO
META-LEARNING
Learn across tasks.
LEARNING TO LEARN
POPULATION-BASED TRAINING
TRAIN
model
START CONFIGURATION
DYNAMIC ALGORITHM CONFIGURATION (DAC)
INTERPRETE
MODEL(S)
EVALUATE

Some motivating examples for/from the seminar

In particular, Bayesian optimization was a significant factor in the strength of AlphaGo in the highly publicized match against Lee Sedol.

HPO for tuning game playing hyperparameters for AlphaGo [Chen et al. arXiv'18]

Table 2. Results using 'best' GPT-3.5 model (among code-davinci-002, text-davinci-002, and text-davinci-003) according to HELM.

Method	APPS	HumanEval	MATH	XSum
HELM	0.03	0.465	0.378	0.140
EcoOptiGen (HELM budget)	0.05	0.521	0.414	0.144

HPO for tuning LLM inference hyperparameters [Wang et al. AutoML'2023]

We need HPO methods

Large models can improve HPO

	Normalized Regret			Rank		
	Micro	Mini	Extended	Micro	Mini	Extended
BEiT+Default HP	$0.229_{\pm 0.081}$	$0.281_{\pm 0.108}$	$0.225_{\pm 0.059}$	$2.583_{\pm 0.829}$	$2.611_{\pm 0.465}$	$3.136_{\pm 0.215}$
XCiT+Default HP	$0.223_{\pm 0.075}$	$0.290_{\pm 0.107}$	$0.199_{\pm 0.057}$	$2.500_{\pm 0.751}$	$2.694_{\pm 0.264}$	$2.522_{\pm 0.344}$
DLA+Default HP	$0.261_{\pm 0.074}$	$0.325_{\pm 0.111}$	$0.219_{\pm 0.076}$	$3.062_{\pm 0.770}$	$3.138_{\pm 0.248}$	$2.977_{\pm 0.284}$
Quick-Tune	$\mathbf{0.153}_{\pm 0.054}$	$\mathbf{0.139}_{\pm 0.112}$	$\mathbf{0.052}_{\pm 0.031}$	$\mathbf{1.854}_{\pm 1.281}$	$\mathbf{1.555}_{\pm 0.531}$	$\mathbf{1.363}_{\pm 0.376}$

HPO for fine-tuning image classification [Pineda et al. arXiv'18]

Session #5: AutoML4LargeModels

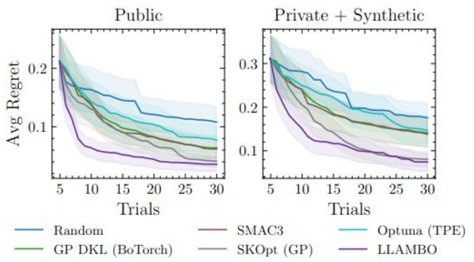


Figure 7: **End-to-end performance of `LLAMBO`. (Left)** average regret on public

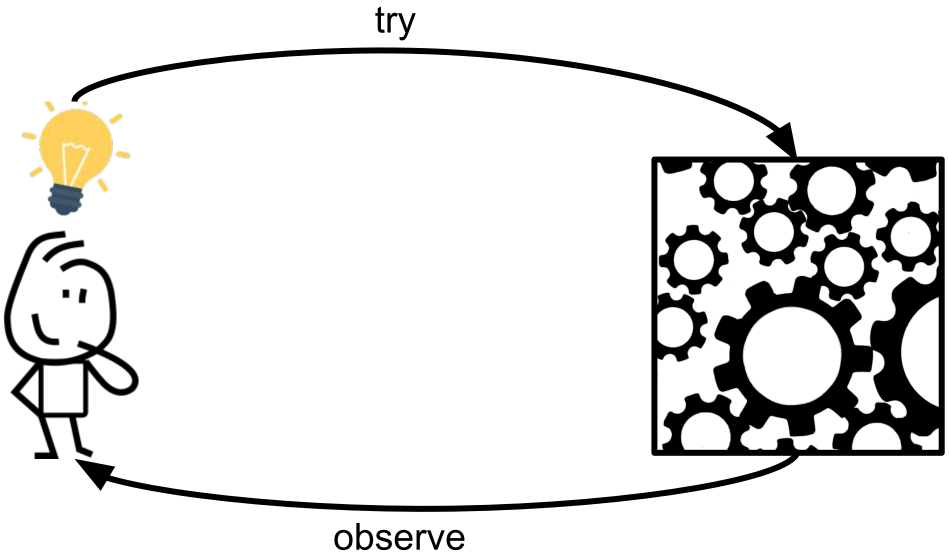
LLMs for improving Bayesian Optimization [Liu et al. ICLR'24]

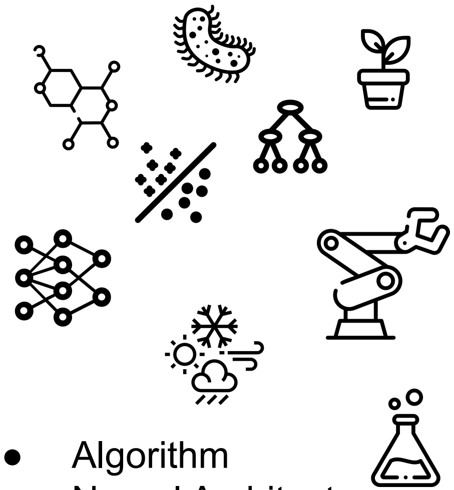
Session #1: LLMs4BO

Sequential experimentation as part of the workflow

Why?

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



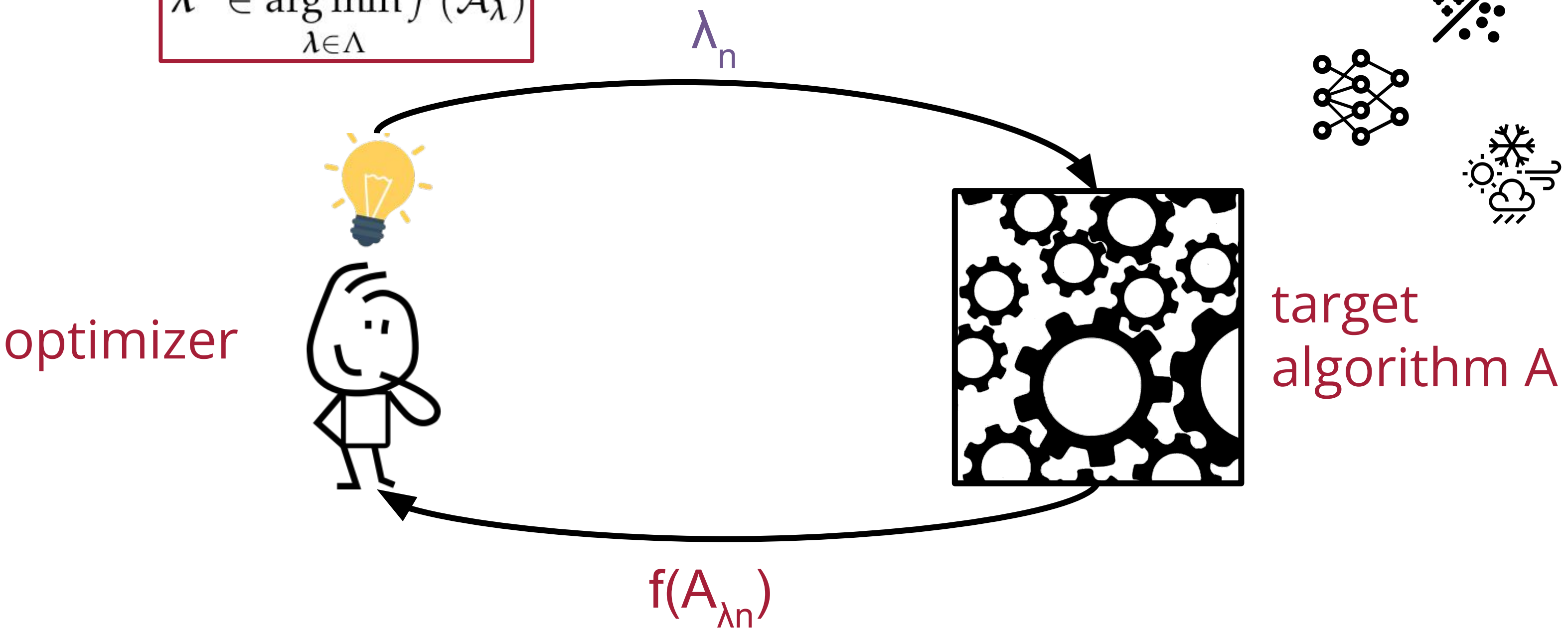


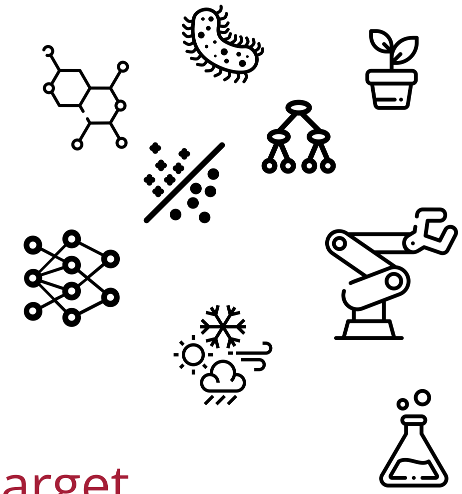
- Algorithm
- Neural Architecture
- Network Training
- Data Science Pipeline
- Simulator
- Robot

Sequential experimentation to find the optimal solution

Goal: Find the best performing configuration:

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





(Hyper-) Parameters

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

Model- vs Hyperparameters

Model parameters

optimized during training; output of the training.

Examples:

- **Splits** of a Tree
- **Weights** of a NN
- **Coefficients** of a linear model

Hyperparameters

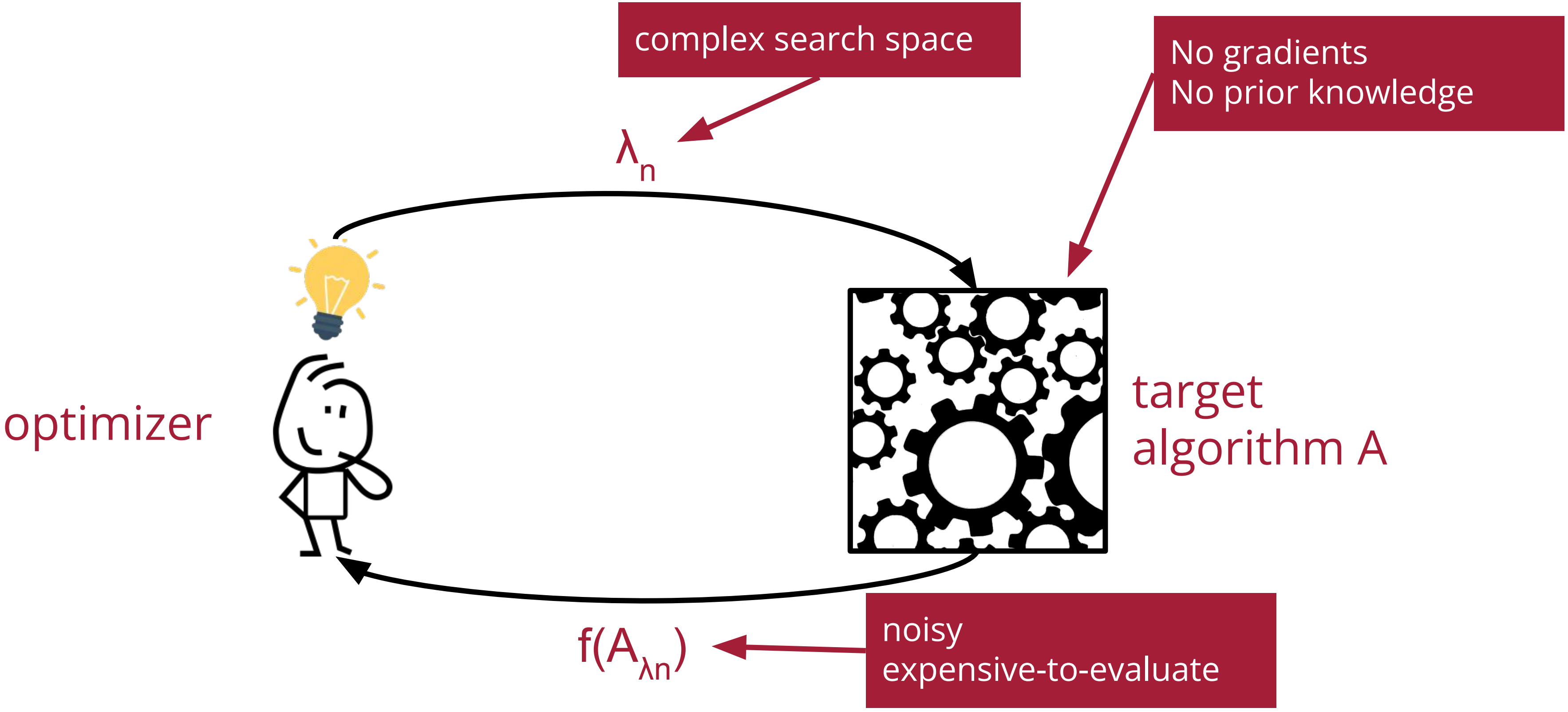
set before training; control flexibility, structure and complexity of the model/ training procedure and performance

Examples:

- **Learning Rate** for XGBoost
- **Kernel Type** for GPs
- **#Layers** for ANNs

Can be **real-valued, integer, categorical** and **hierarchically dependent** on each other

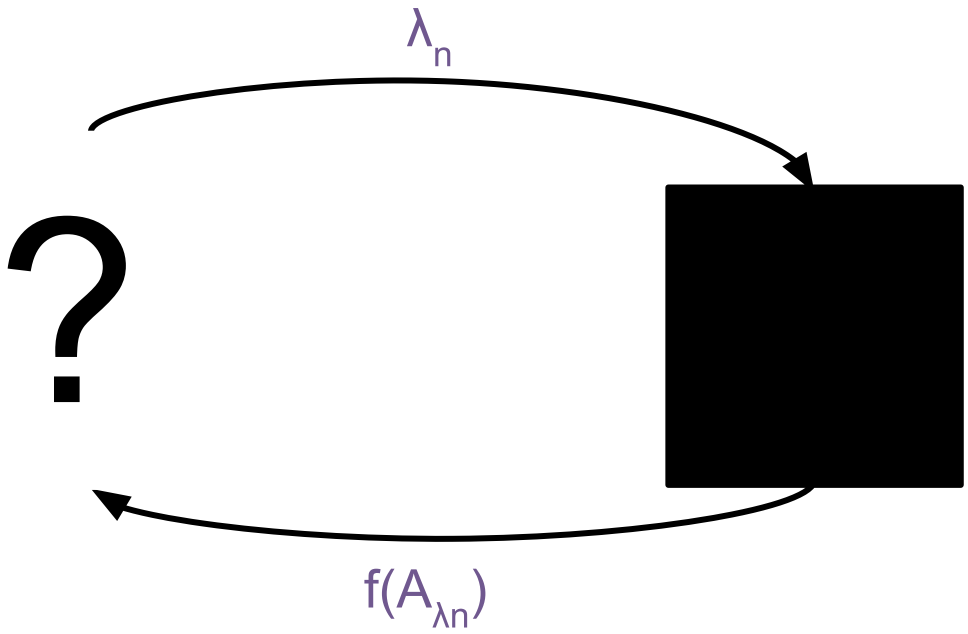
Hyperparameter optimization is challenging



How do we optimize it?

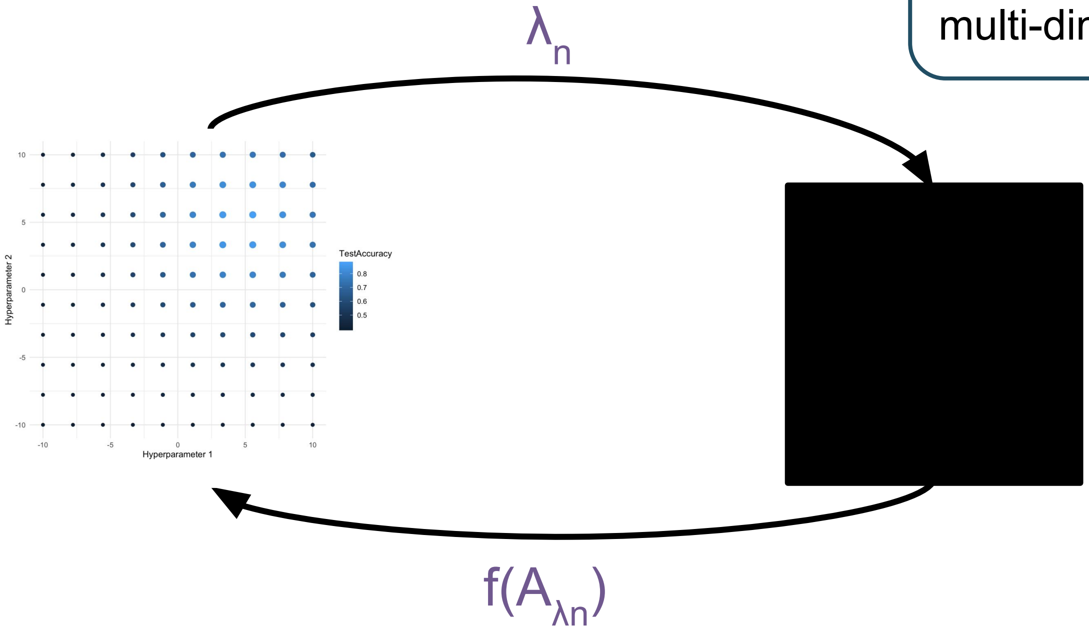
`>> Here’s my algorithm, data, metric and search space, what should I do?`

HPO is a black-box optimization problem

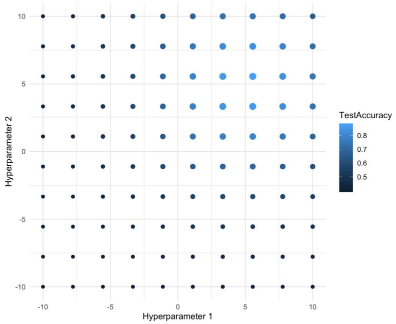


Grid-Search as a popular and simple approach

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



Advantages and disadvantages of grid search



Advantages

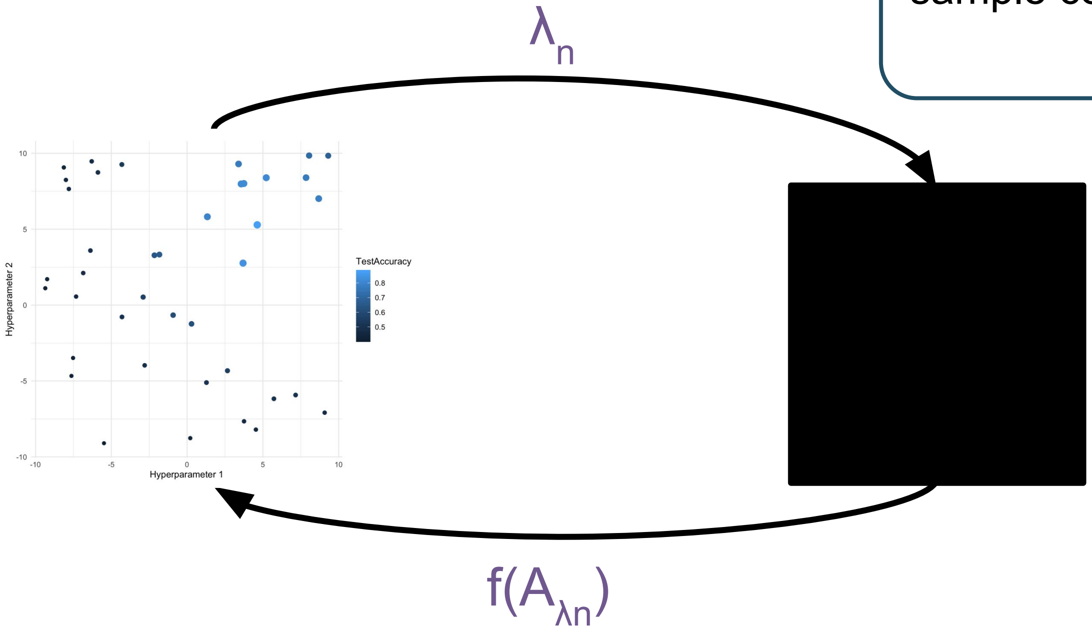
- Very easy to implement
- Very easy to parallelize
- Can handle all types of hyperparameters

Disadvantages

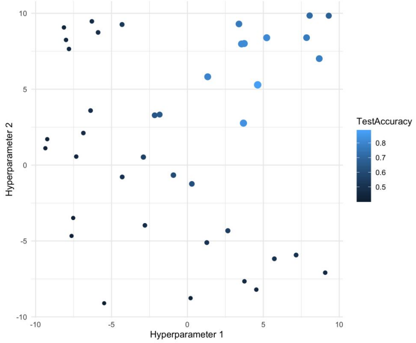
- Scales badly with #dimensions
- Inefficient: Searches irrelevant areas
- Requires to manual define discretization
- All grid points need to be evaluated

Random search as an alternative

Variation of Grid Search: Uniformly sample configurations at random



Advantages and disadvantages as grid search



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

Random search is more effective

With a **budget** of T iterations:

- **Grid Search** evaluates $T^{1/d}$
- **Random Search** evaluates (most likely) **T**

unique values per hyperparameter dimension

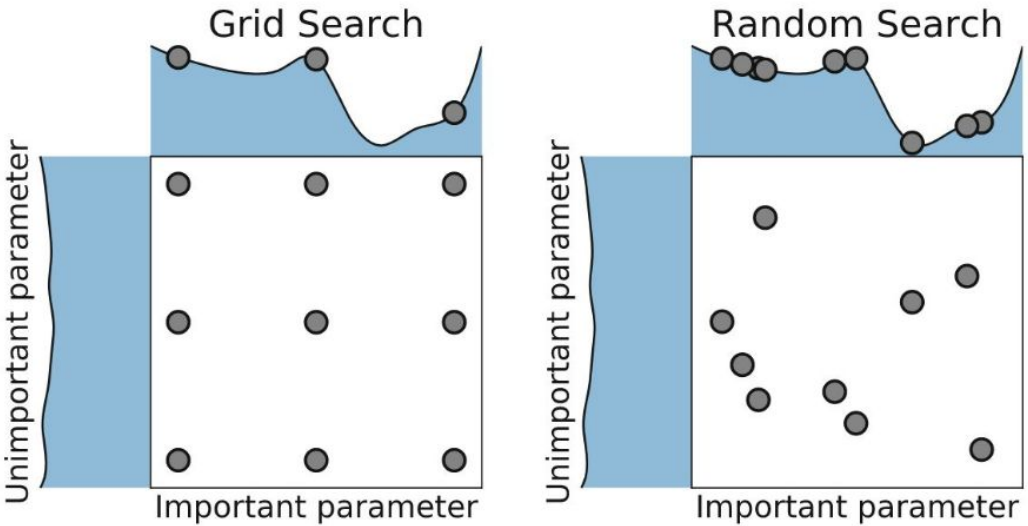


Image source: [Hutter et al. 2019]

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

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

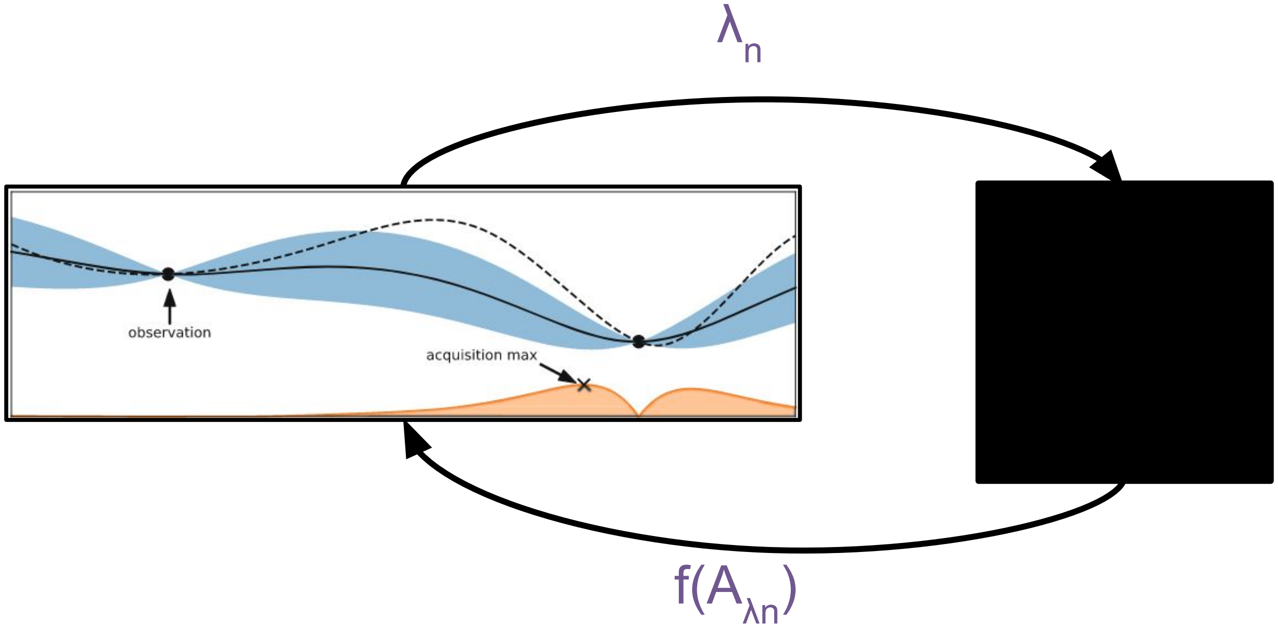
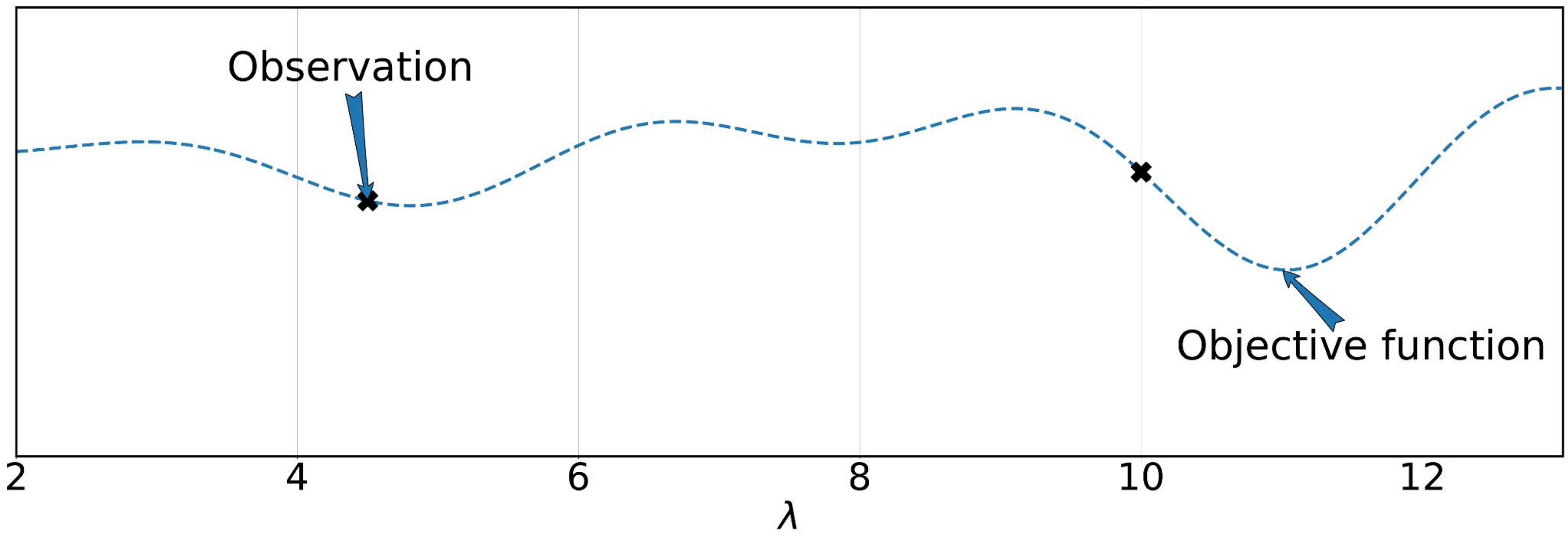
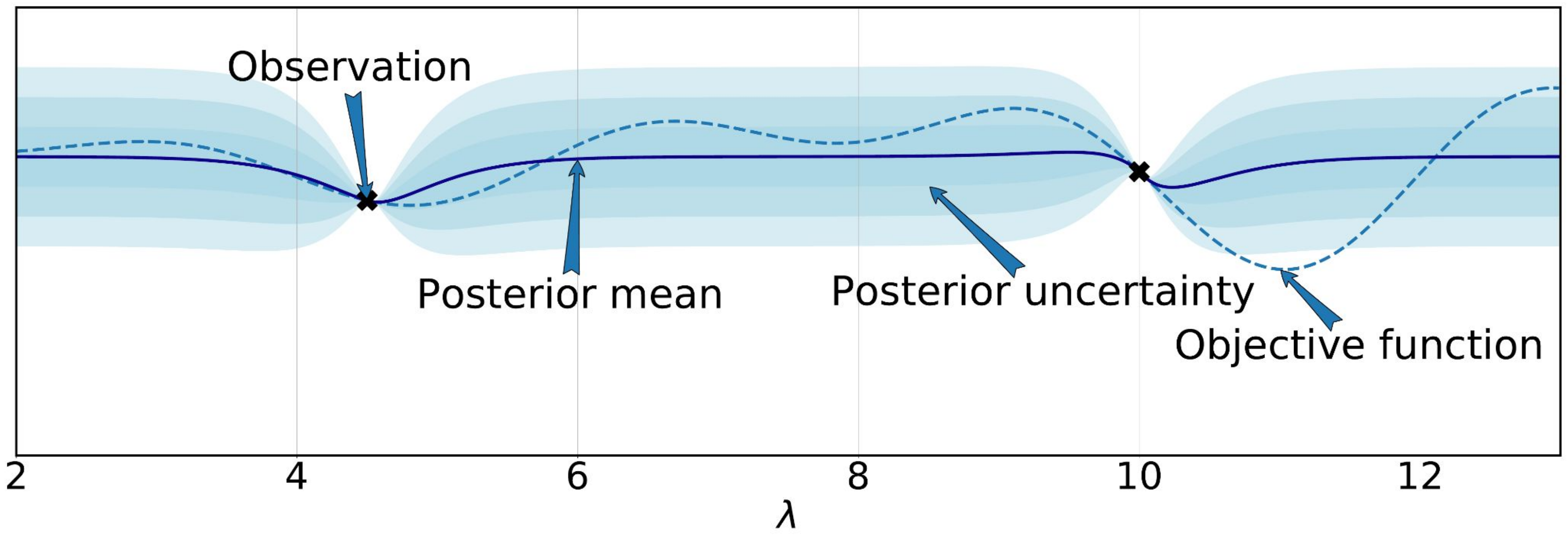


Photo by Wilhelm Gunkel on Unsplash
Image by Feurer, Hutter: Hyperparameter Optimization. In: Automated Machine Learning

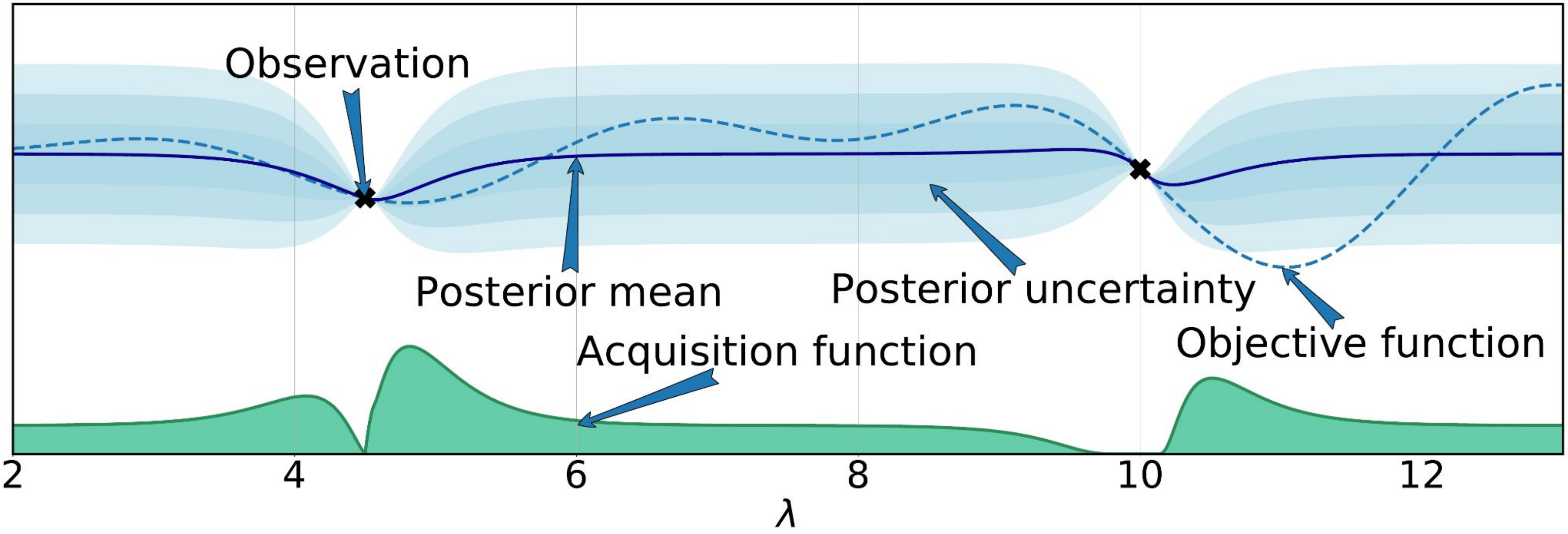
Bayesian Optimization in a Nutshell



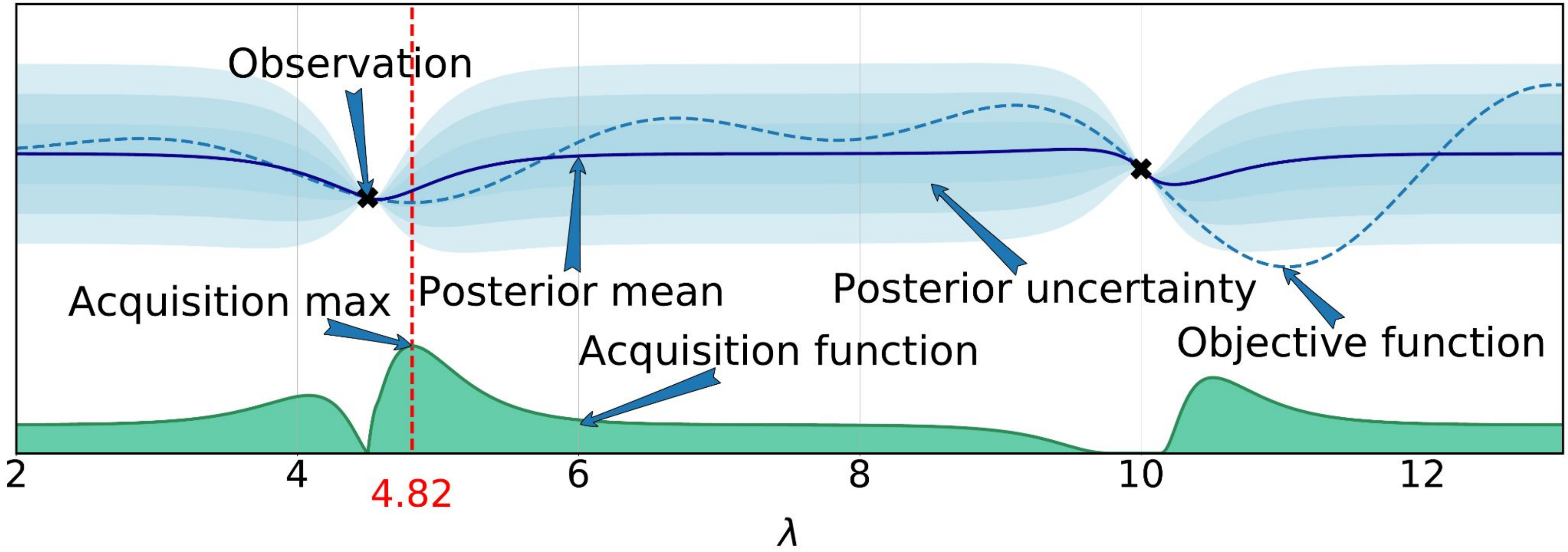
Bayesian Optimization in a Nutshell



Bayesian Optimization in a Nutshell



Bayesian Optimization in a Nutshell



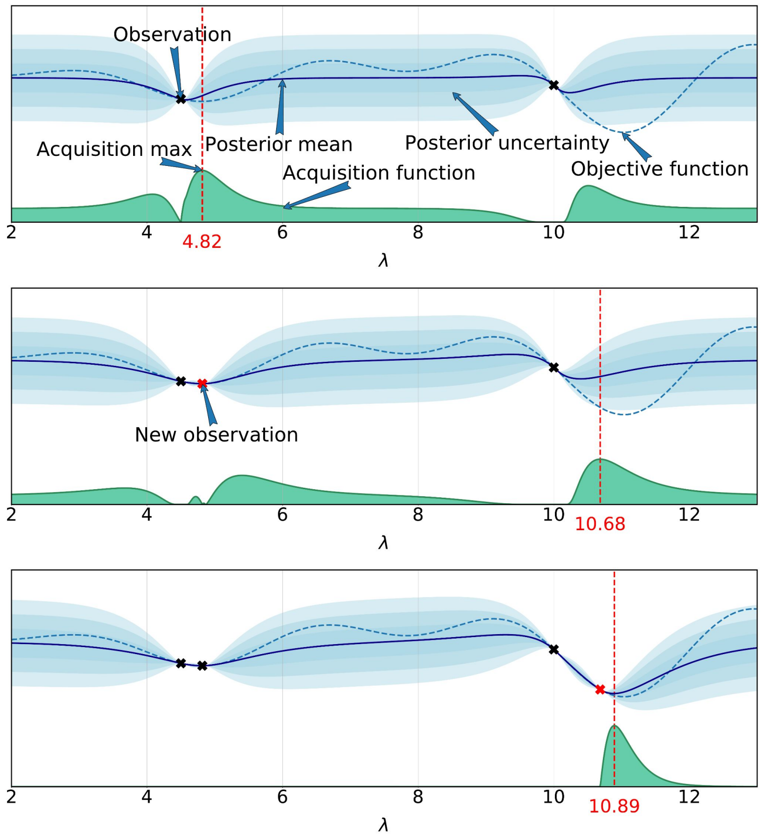
Bayesian Optimization in a Nutshell

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 $\hat{c}$)

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

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

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

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

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

  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

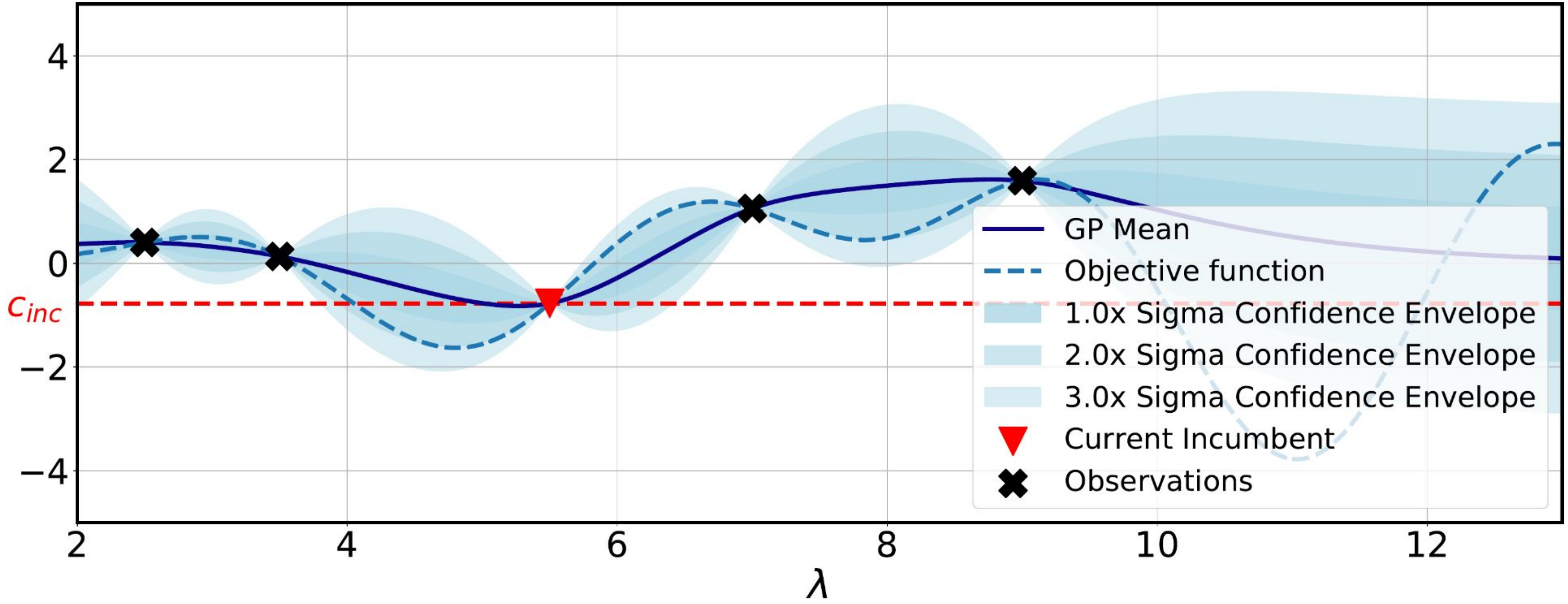
The acquisition function:

- 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**

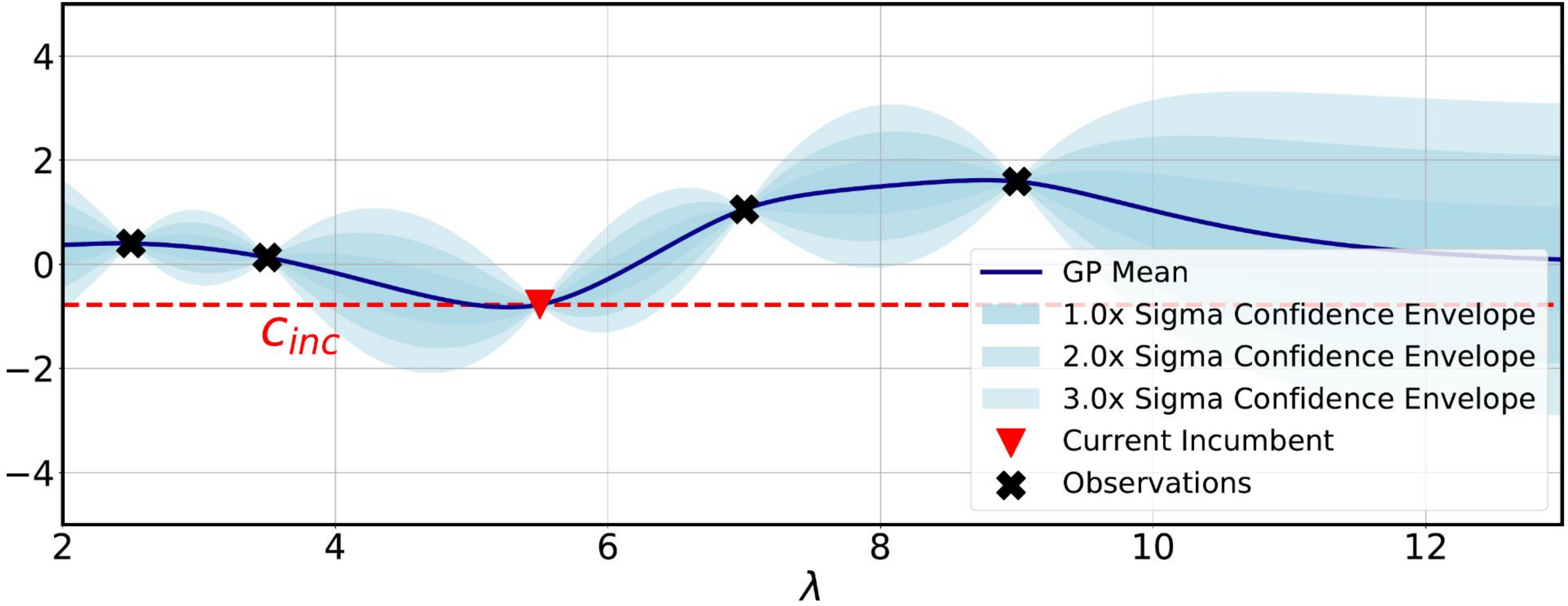
- Just picking the configuration with the lowest prediction would be too greedy
- It needs to consider the uncertainty of the surrogate model

Expected Improvement (EI)



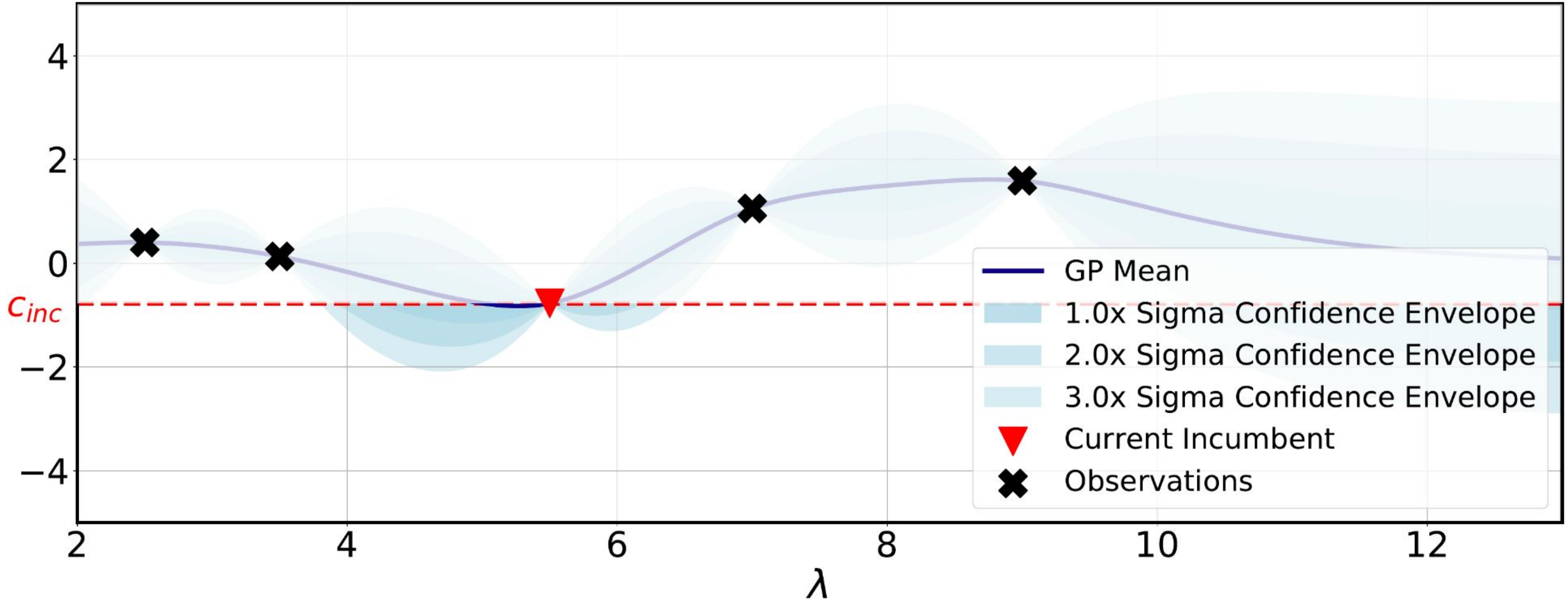
Given some observations and a fitted surrogate,

Expected Improvement (EI)



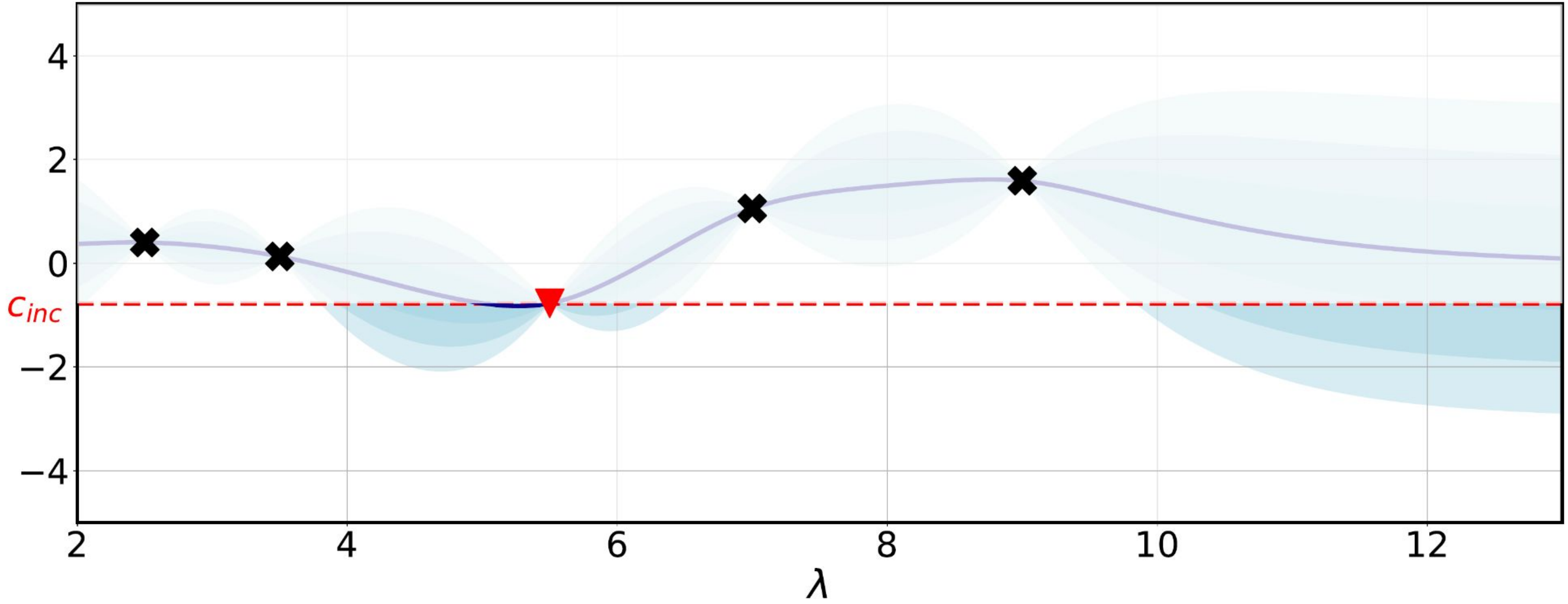
Given some observations and a fitted surrogate,

Expected Improvement (EI)



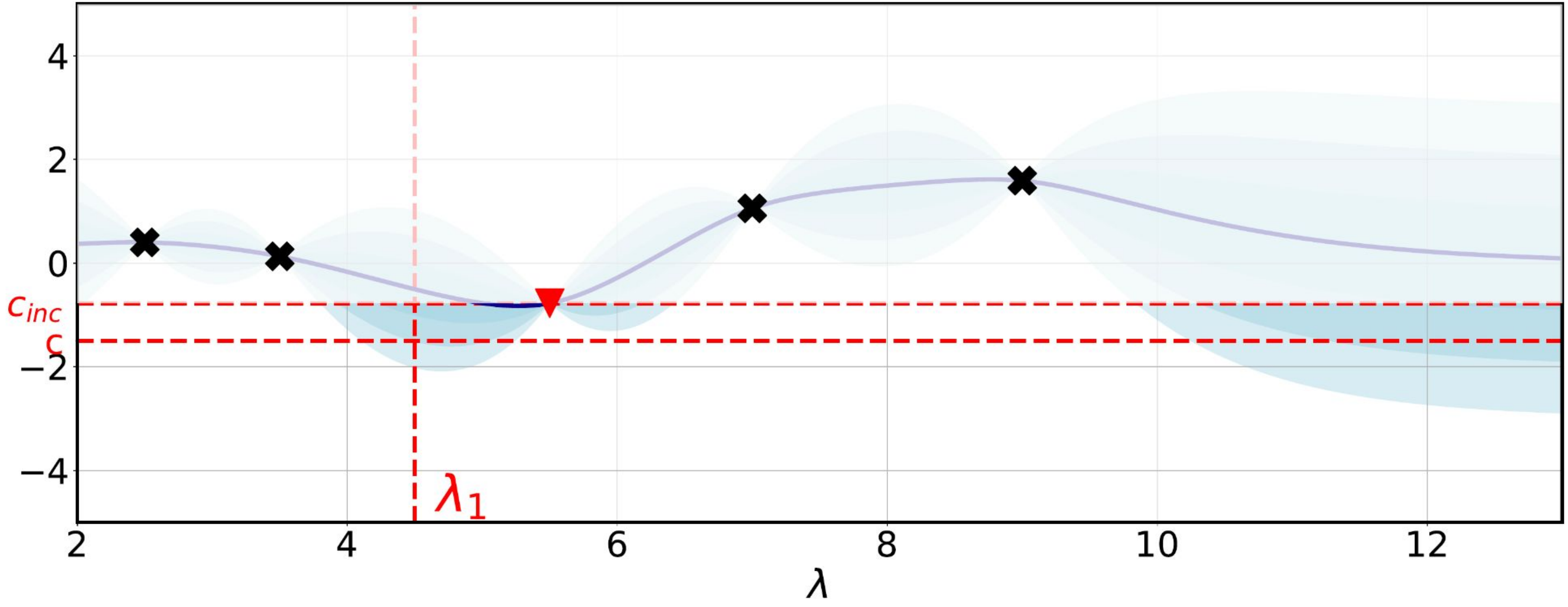
We care about *improving* over the c_{inc}.

Expected Improvement (EI)



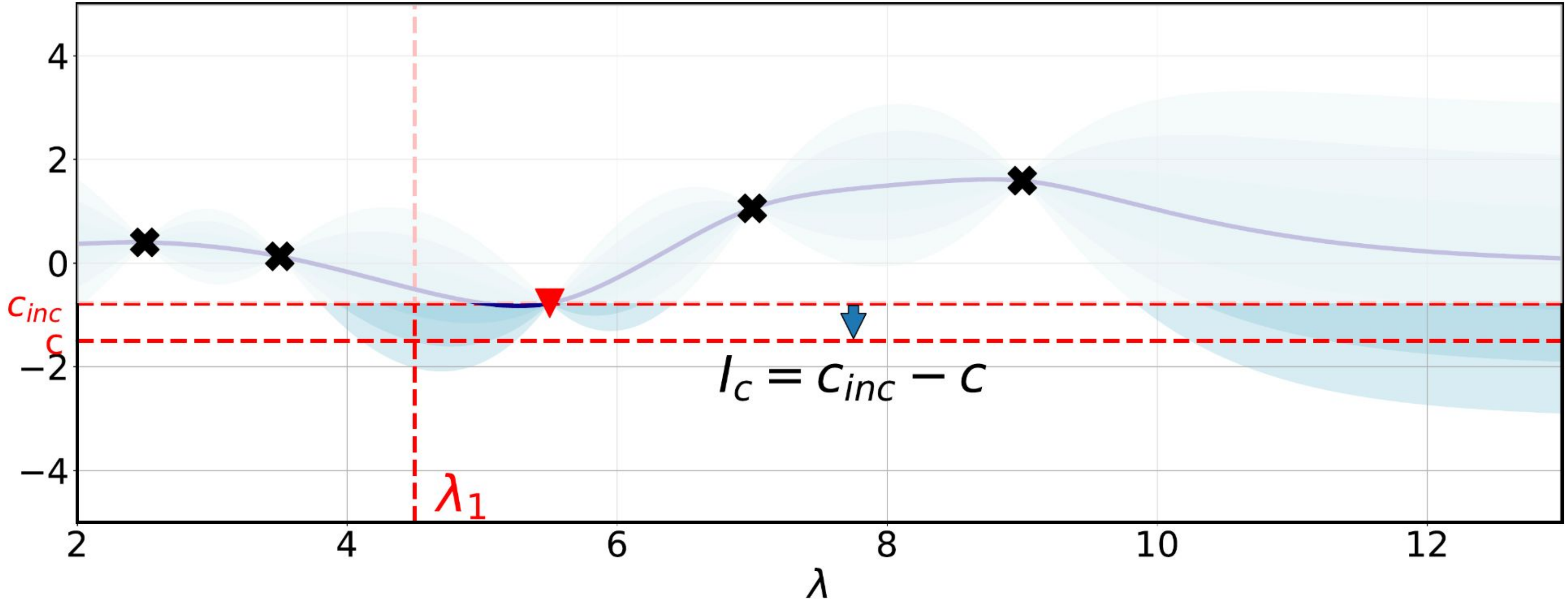
We care about *improving* over the c_{inc}.

Expected Improvement (EI)



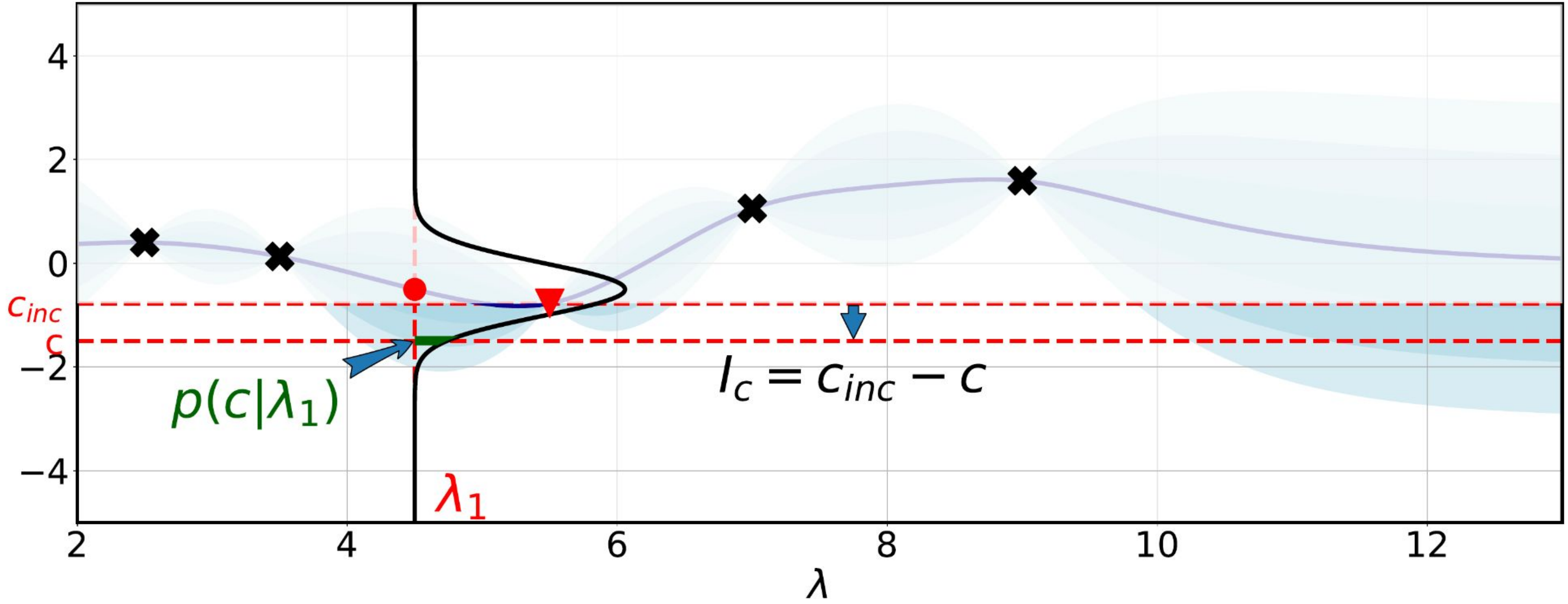
Let's look at a candidate configuration λ_1 and its hypothetical cost c.

Expected Improvement (EI)



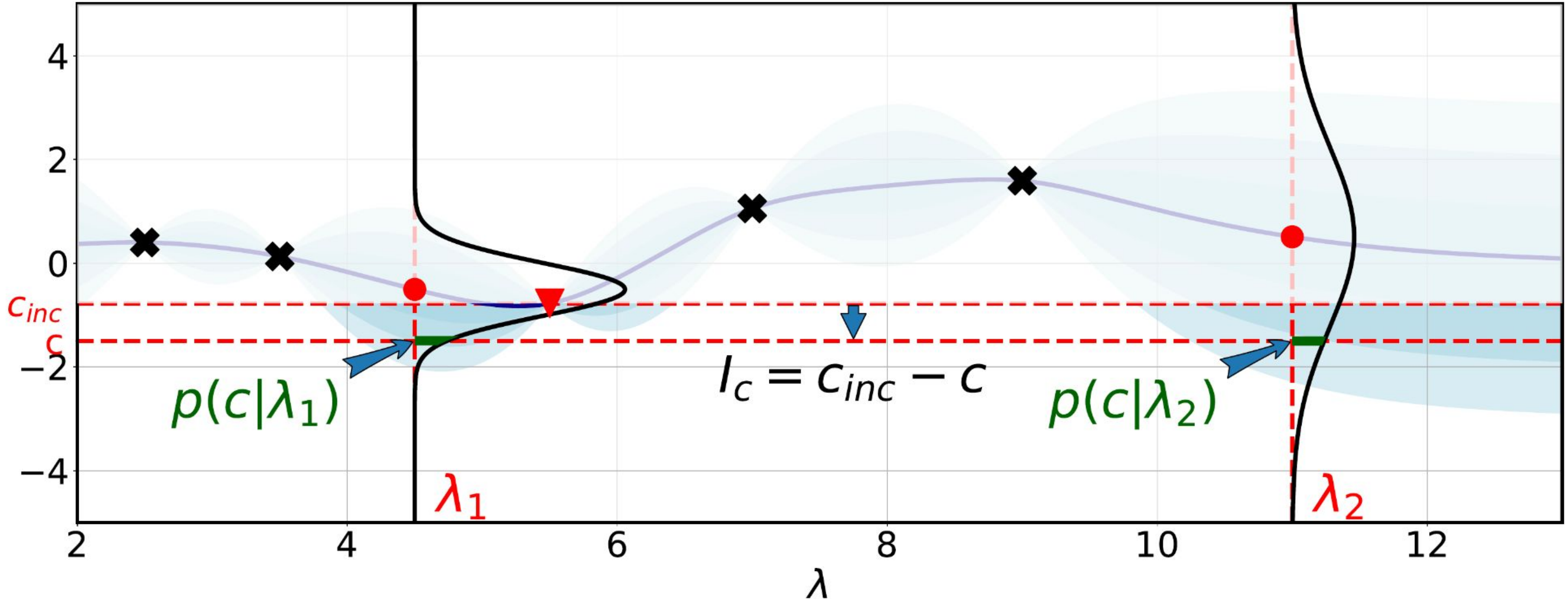
We can compute the improvement $I_c(\lambda_1)$. But how likely is it?

Expected Improvement (EI)



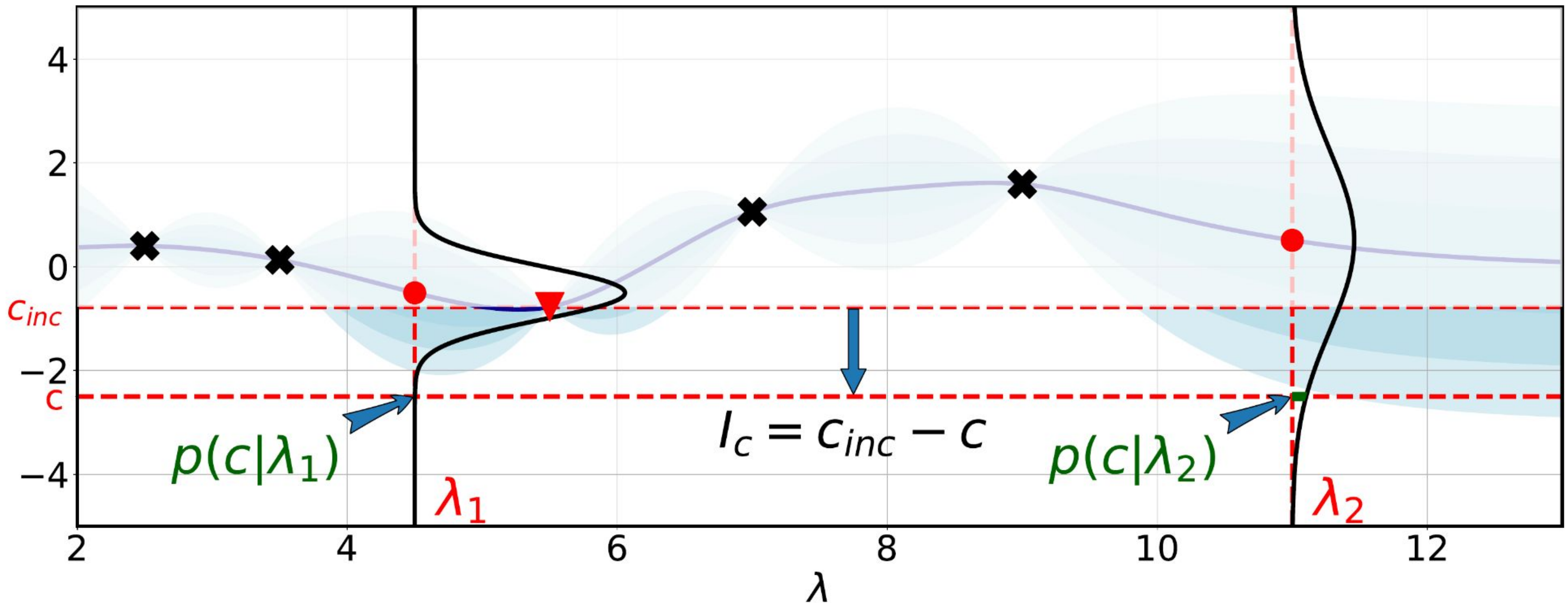
Knowing that $\hat{c}(\boldsymbol{\lambda}) = \mathcal{N}(\mu(\boldsymbol{\lambda}), \sigma^2(\boldsymbol{\lambda}))$, we can compute $p(c|\boldsymbol{\lambda})$

Expected Improvement (EI)



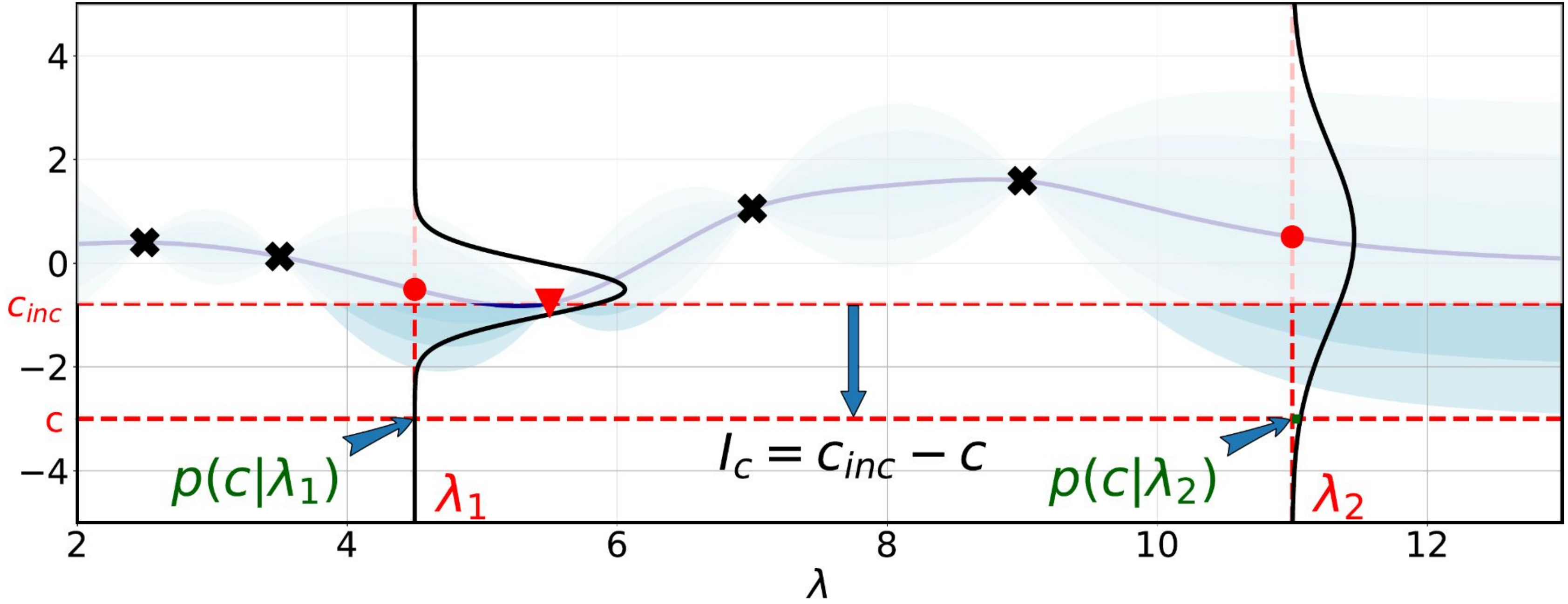
Comparing this for different configurations

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

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

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

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?

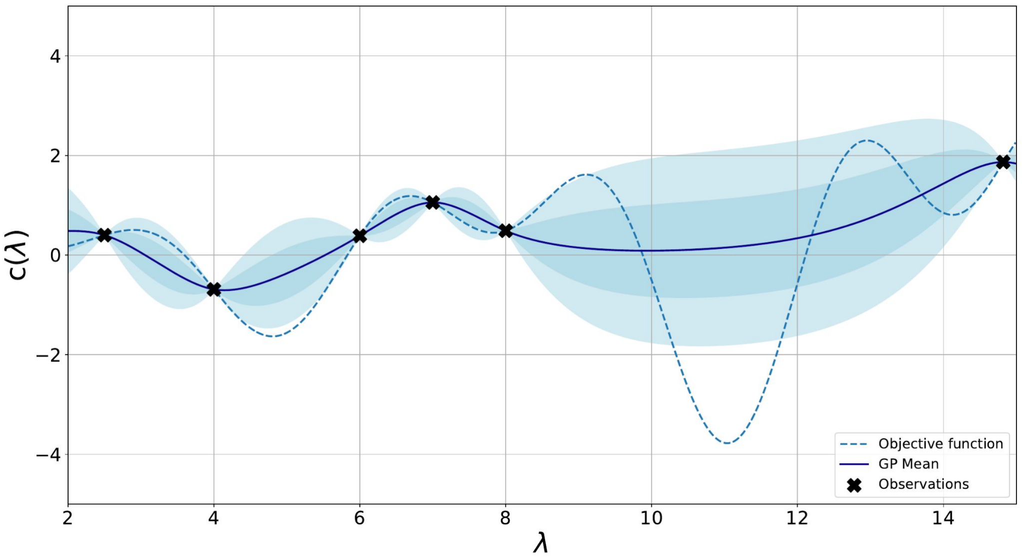
Main Ingredient II: The Surrogate Model

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

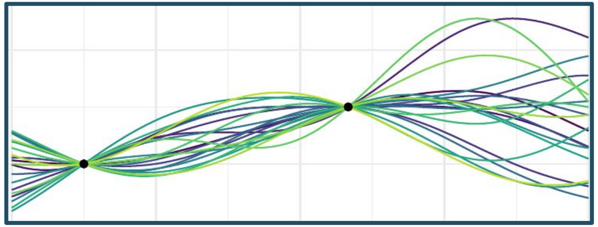


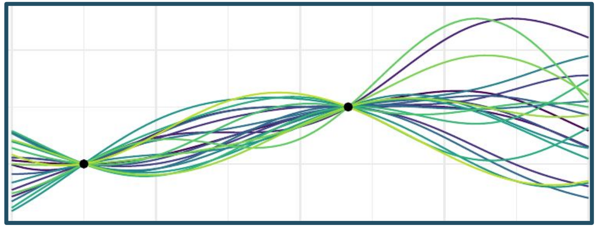
Photo by Filip Zrnzević on Unsplash
Photo by Alina Grubnyak on Unsplash

Gaussian Processes

$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]$$

$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}\left[(f(\mathbf{x}) - \mathbb{E}[f(\mathbf{x})])\,(f(\mathbf{x}') - \mathbb{E}[f(\mathbf{x}')])\right]$$

$$f(\mathbf{x}) \sim \mathcal{G}\left(m(\mathbf{x}), k\left(\mathbf{x}, \mathbf{x}'\right)\right)$$



Advantages

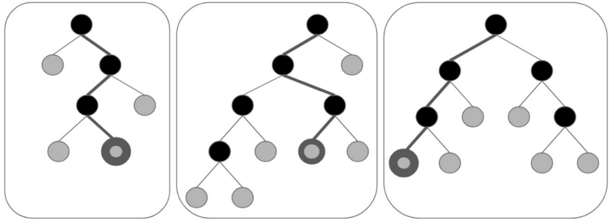
- Smooth uncertainty estimates
- Strong sample efficiency
- Expert knowledge can be encoded in the kernel
- Accurate predictions

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

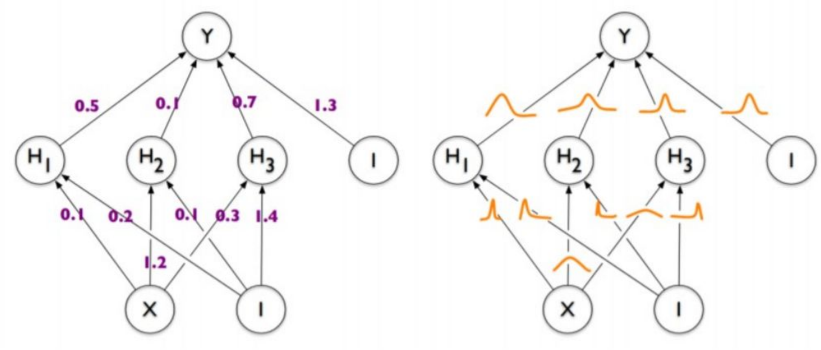


Image source: [Blundell et al. 2015]

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

Questions?

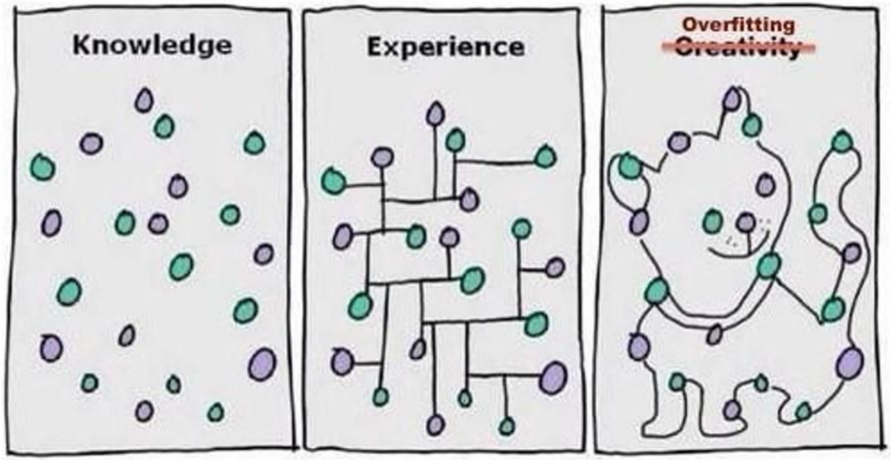
How to evaluate ML models when using HPO?

We want solutions that **generalize to new data!**

→ “reasonable” predictions on **new data**

This might include:

- ignoring outliers
- smooth
- capturing general trend



Source: Kaushik, 2016

Splitting data for basic ML evaluation

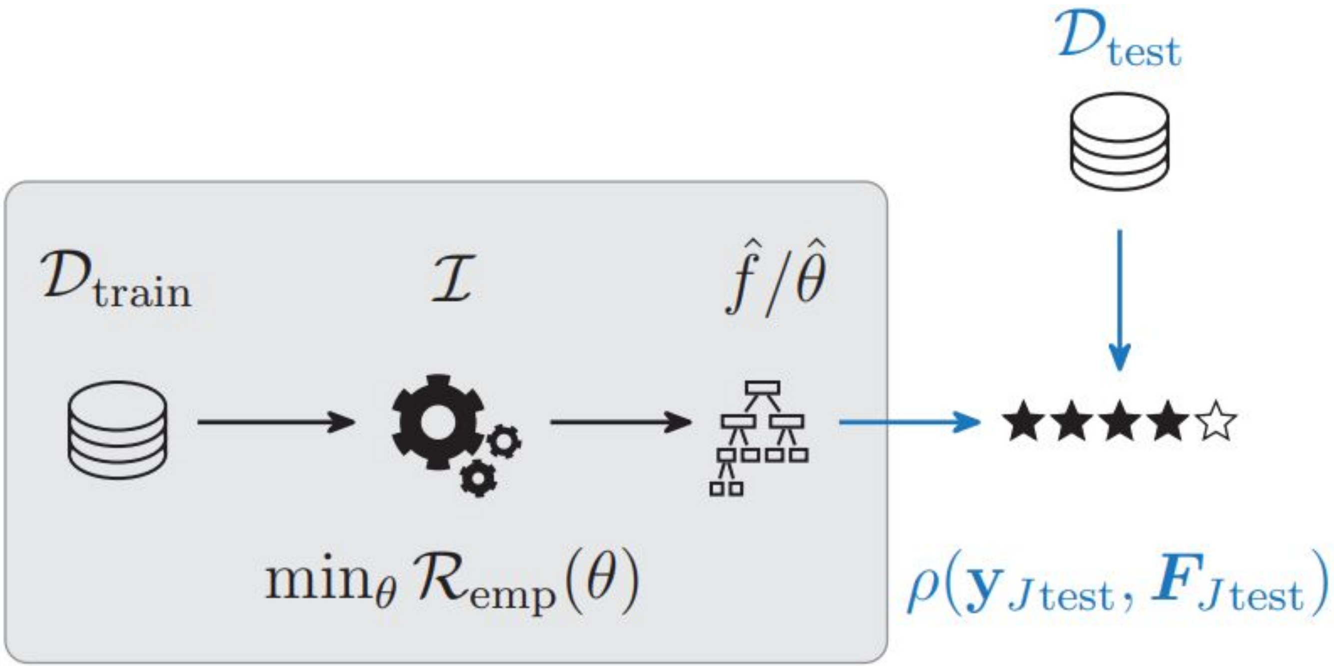


Image: Bischl et al. 2023

Nested crossvalidation for AutoML

For ML users

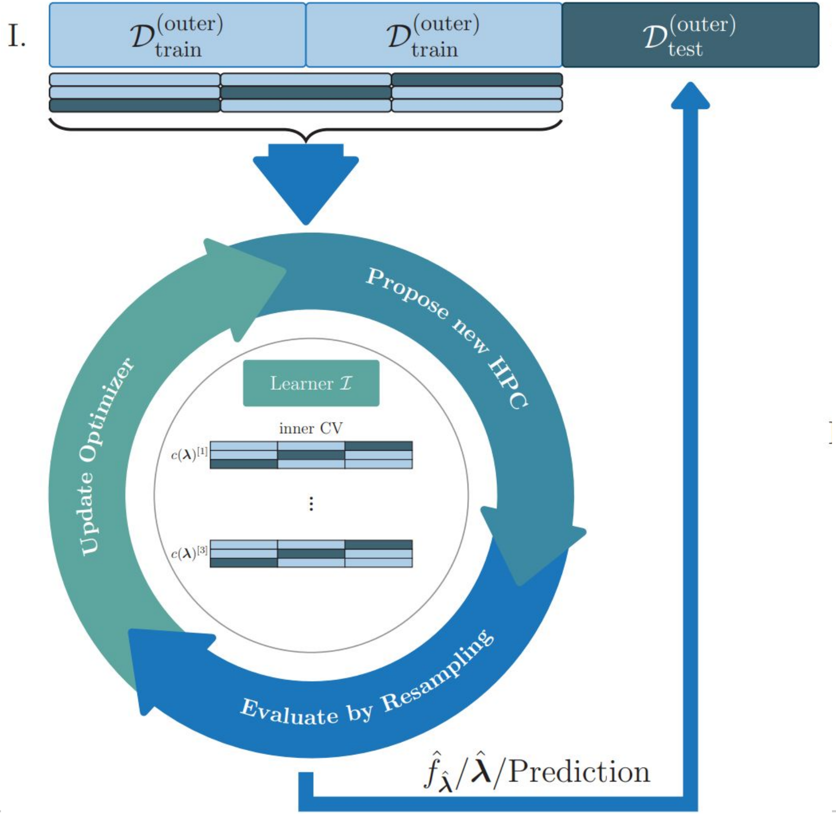


Image: Bischl et al. 2023

Nested CV for AutoML

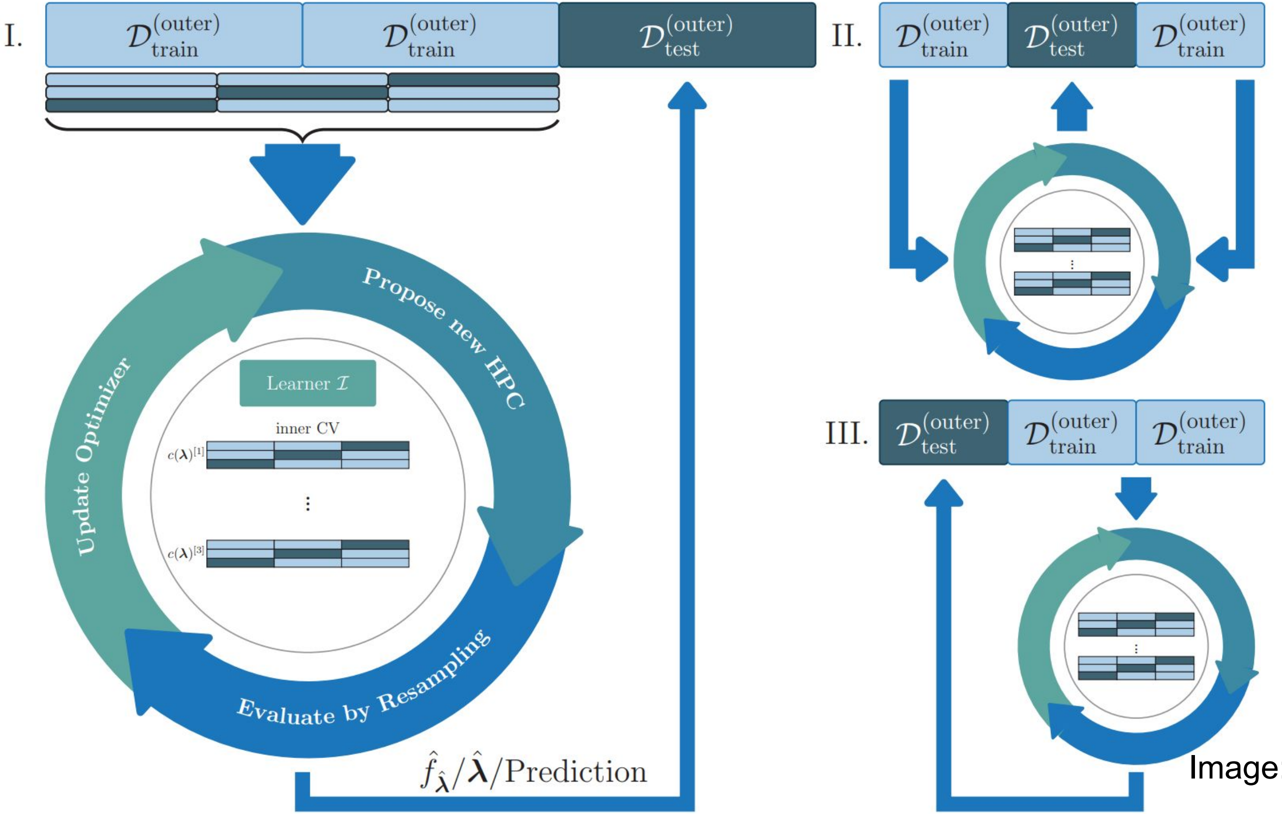


Image: Bischl et al. 2023

Illustration: Resampling vs. Nested Resampling

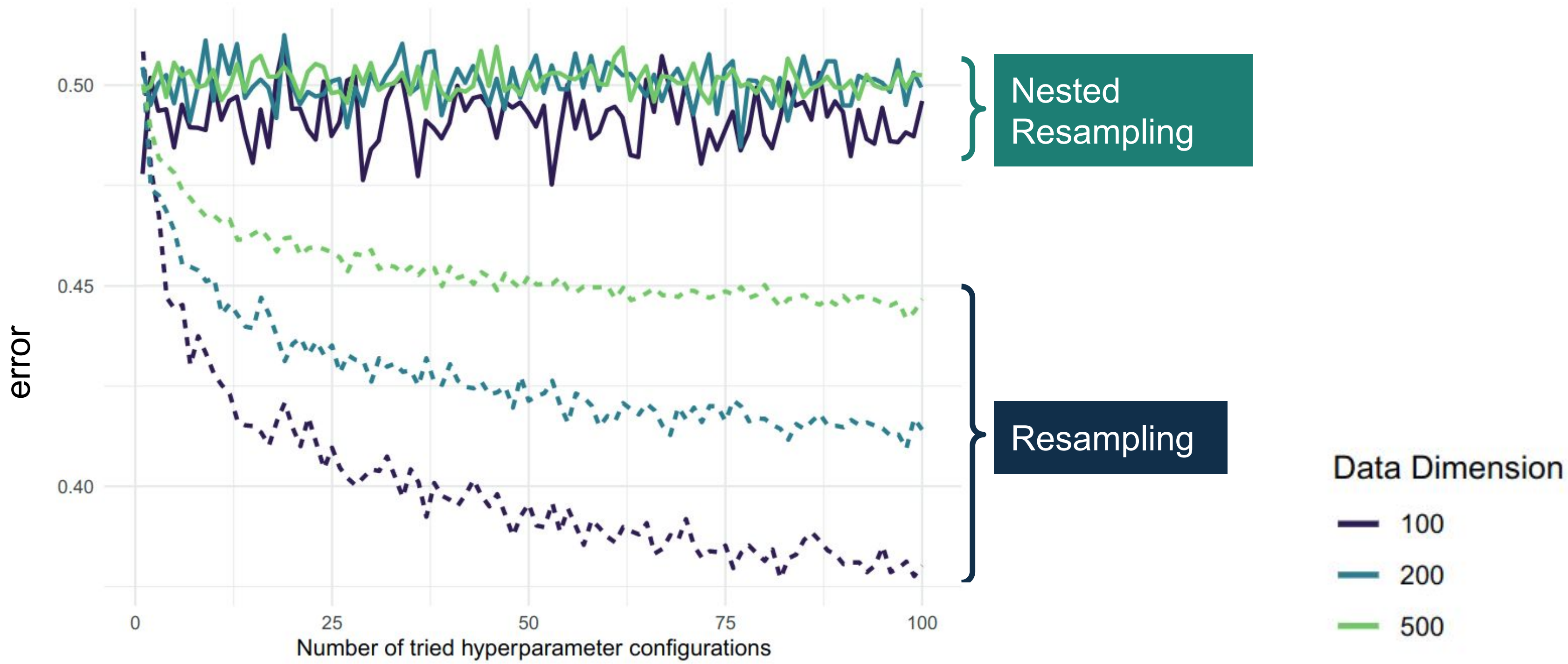


Image: Bischl et al. 2023

Questions?

Further Reading

Introduction to Bayesian Optimization

- R. Garnett “Bayesian Optimization” (2023) - https://bayesoptbook.com/
- P. Frazier “A tutorial on Bayesian Optimization” (2018) - https://arxiv.org/abs/1807.02811
- B. Bischl et al. “Hyperparameter Optimization: Foundation, Algorithms, Best Practuces and Open Challenges” - https://arxiv.org/pdf/2107.05847.pdf

What's happening next week?

Some homework:

1. (recommended) Watch/Read intro to LLMs and transformers
   - A. Karpathy - https://www.youtube.com/watch?v=zjkBMFhNj_g
   - L. Beyer - https://www.youtube.com/watch?v=EixI6t5oif0
   - Vaswani et al. "Attention is all you need" (2017) - https://proceedings.neurips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html
2. Send me any question you might have
3. Think of at least one use case for pre-trained models for AutoML and vice versa (!)

Note: You can look into "AutoML in the Age of LLMs - https://arxiv.org/abs/2306.08107", but try to think about it first. Only read "Callenges", "Opportunities" and "Risks"; no need to understand details about methods yet, this will be handled in the presentations.

Next week:

- Clarification/finalization of orga / topics / dates
- Answering your questions
- Discuss the applications you came up with