



Seminar Summer Term 2024

Bayesian Optimization for HPO

Katharina Eggensperger

 /KEggensperger

katharina.eggensperger@uni-tuebingen.de

AutoML for Science

April 23rd, 2024



[?] Questions regarding the organization

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

[?] Your Questions

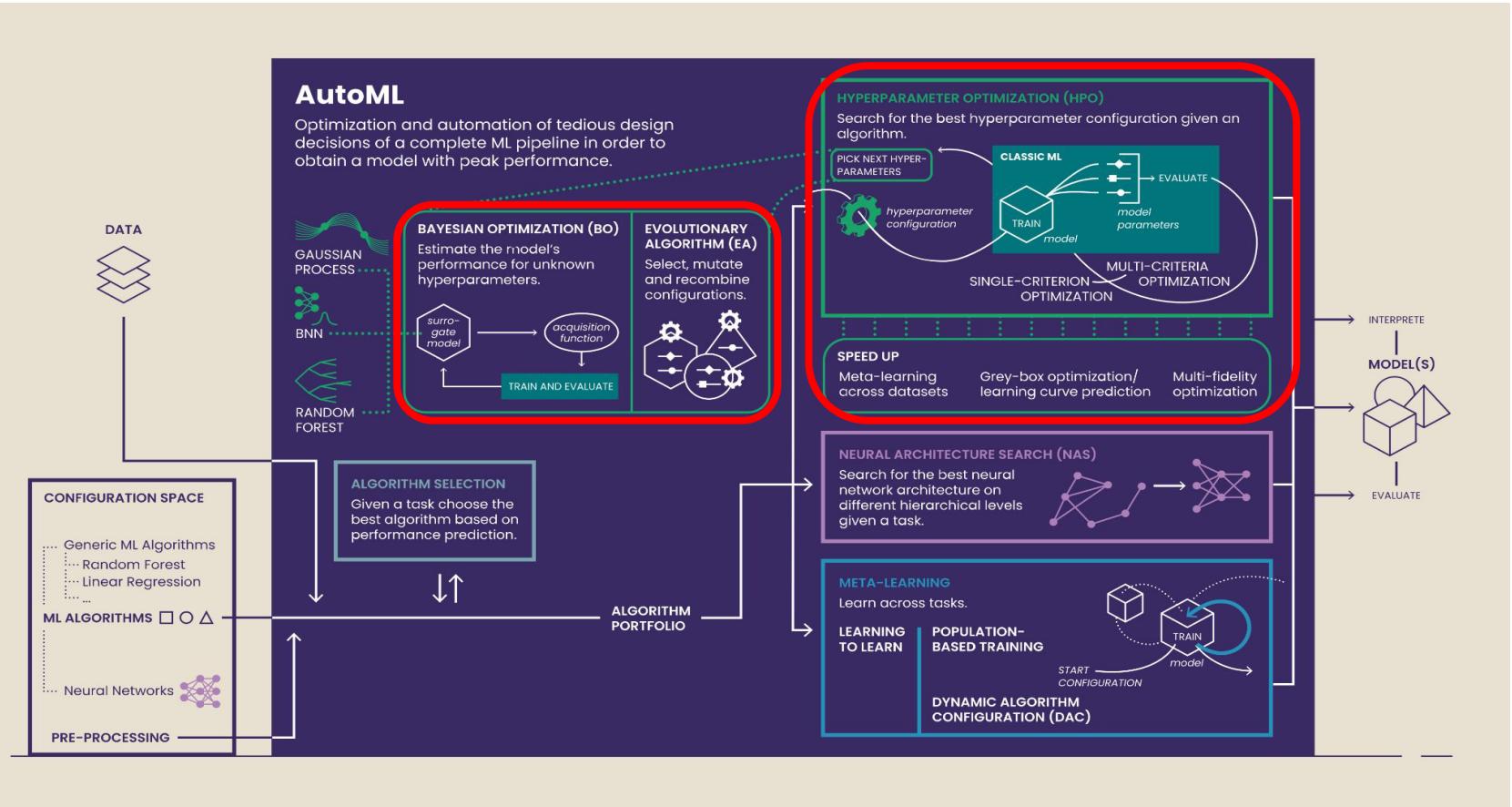
[25min] **How to give a good presentation**

[?] Your Questions



The Big Picture

>> What is this about?





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

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

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

Session #5: AutoML4LargeModels

Large models can improve HPO

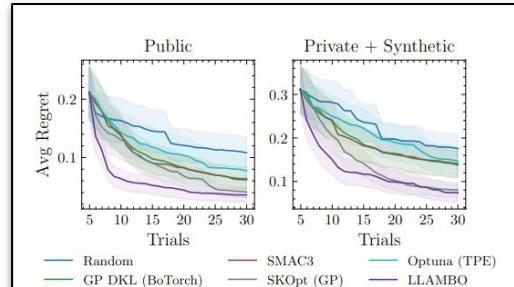


Figure 7: End-to-end performance of LLAMBO. (Left) average regret on public

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

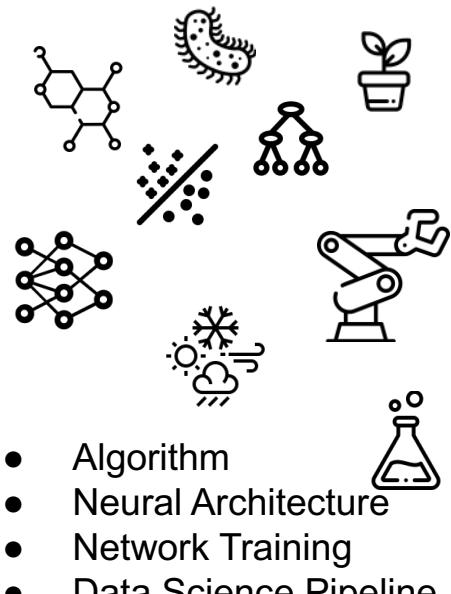
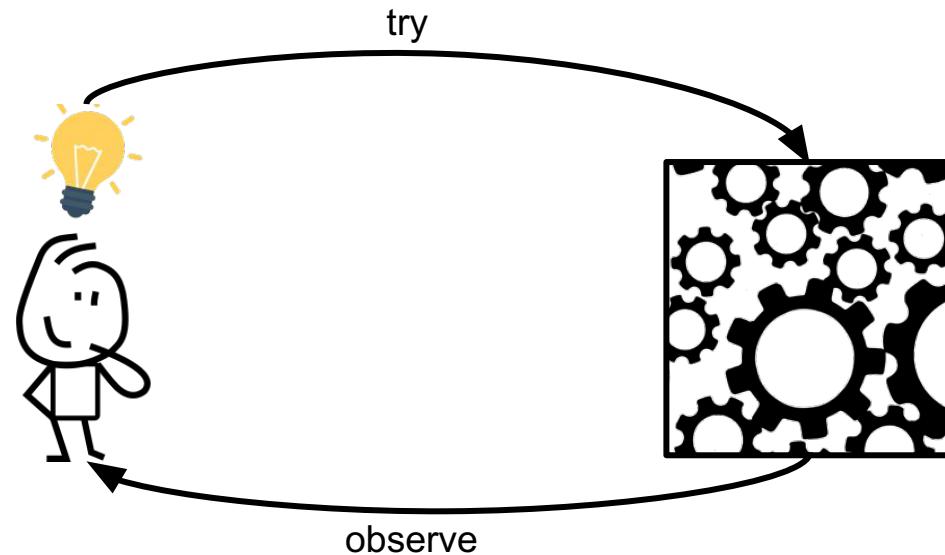
Session #1: LLMs4BO



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

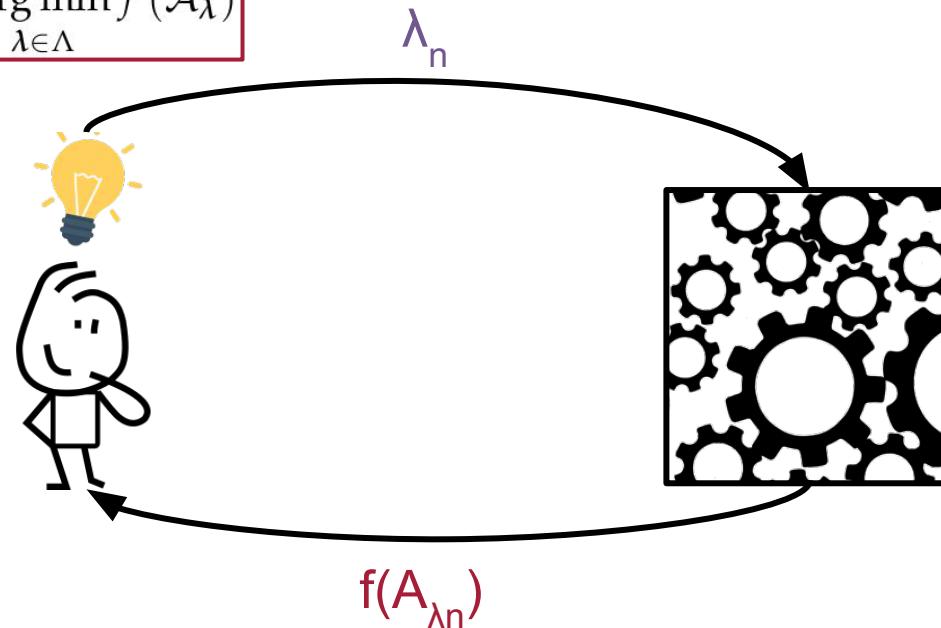


Sequential experimentation to find the optimal solution

Goal: Find the best performing configuration:

$$\lambda^* \in \arg \min_{\lambda \in \Lambda} f(A_\lambda)$$

optimizer



target
algorithm A



- 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 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

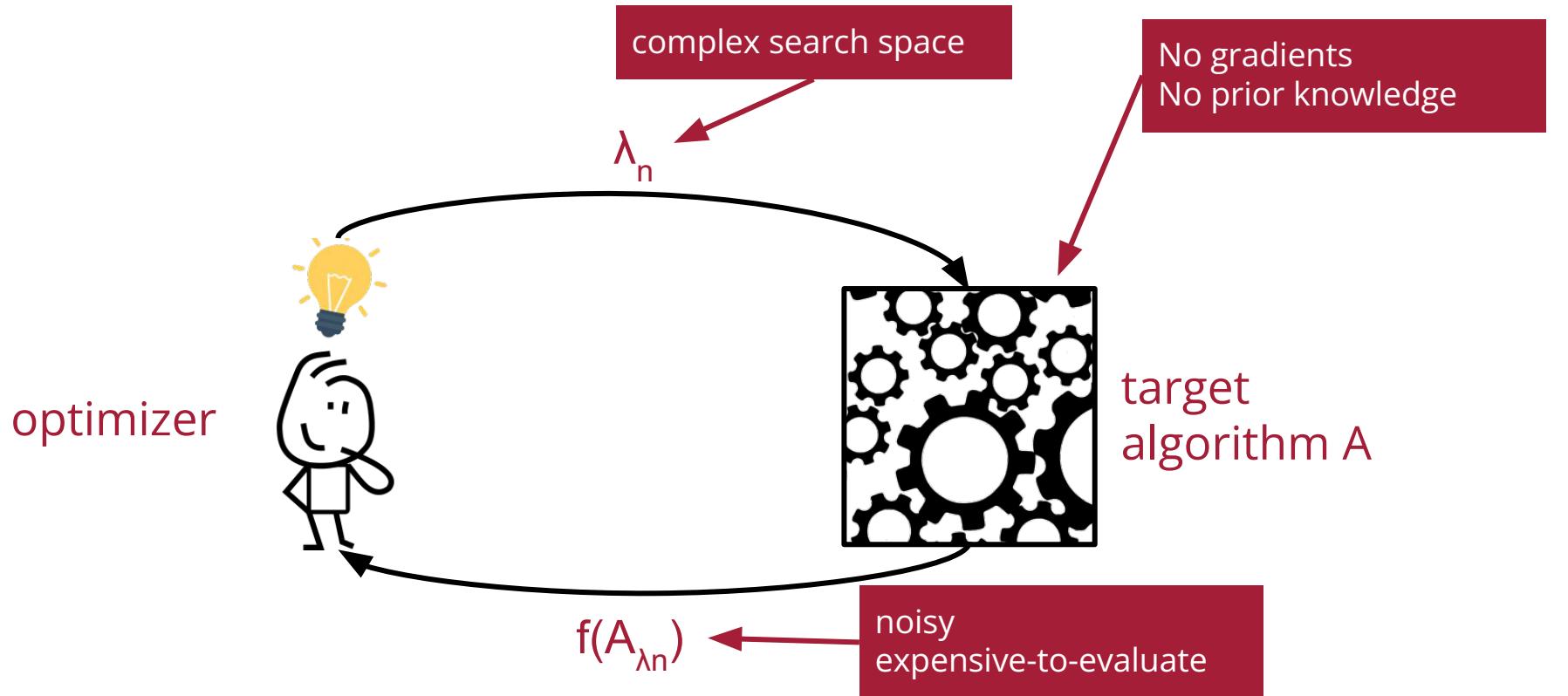
Examples:

- **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



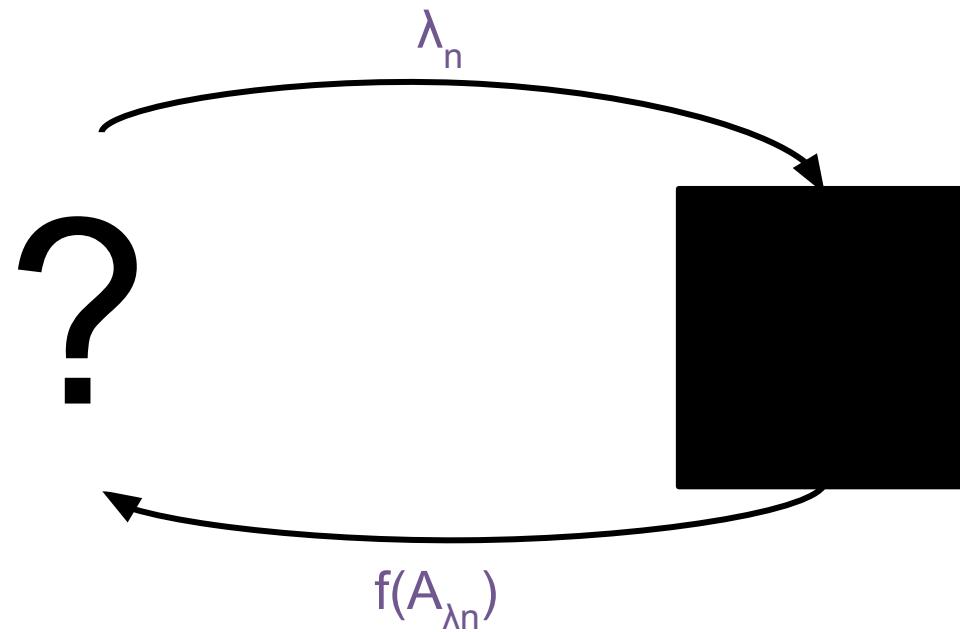


How do we optimize it?

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

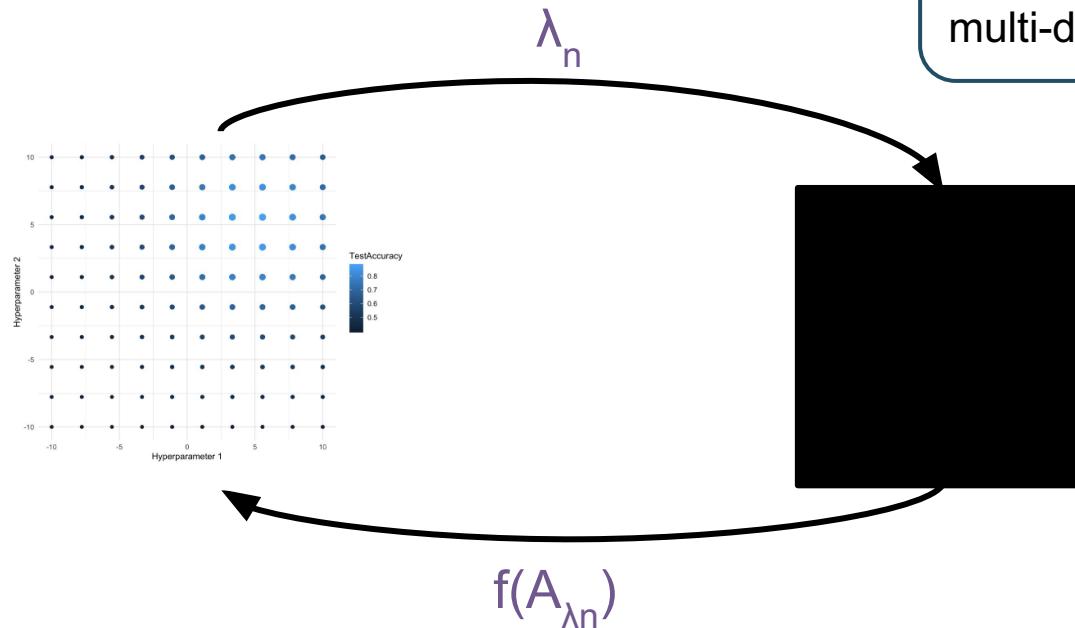


HPO is a black-box optimization problem





Grid-Search as a popular and simple approach

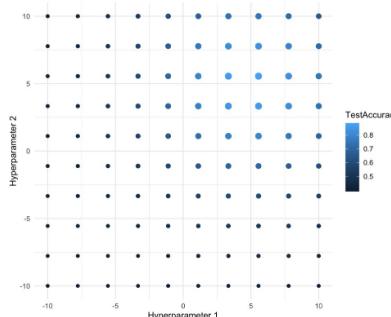


Popular technique: Evaluates all combinations on a pre-defined multi-dimensional grid



Advantages

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

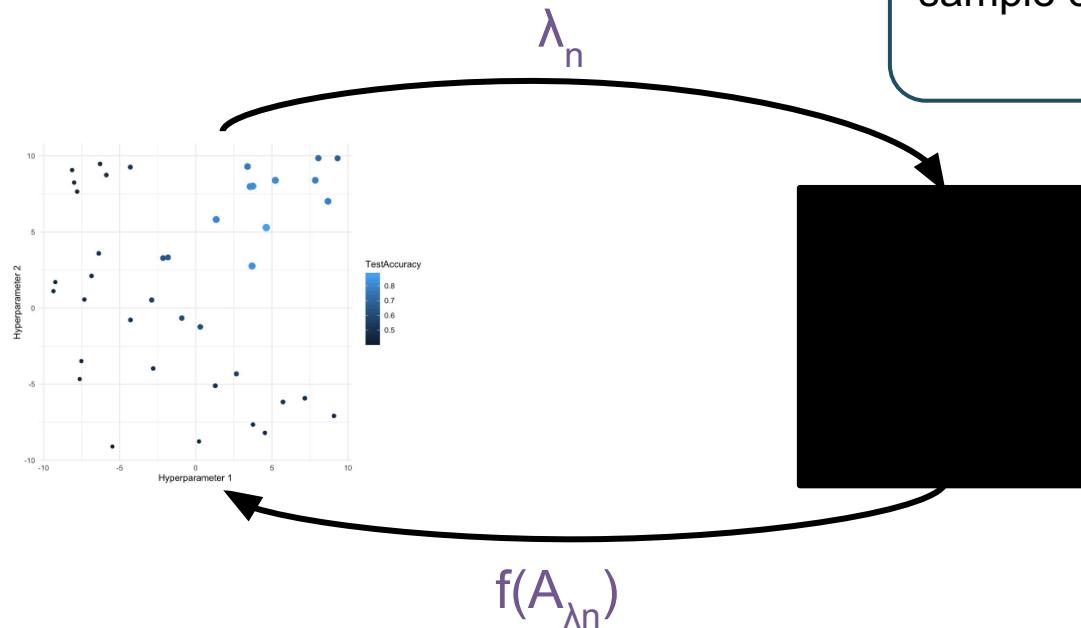


Disadvantages

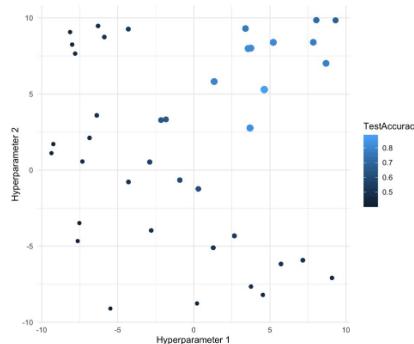
- Scales badly with #dimensions
- Inefficient: Searches irrelevant areas
- Requires to manually define discretization
- All grid points need to be evaluated



Random search as an alternative



Variation of Grid Search: Uniformly sample configurations at random



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

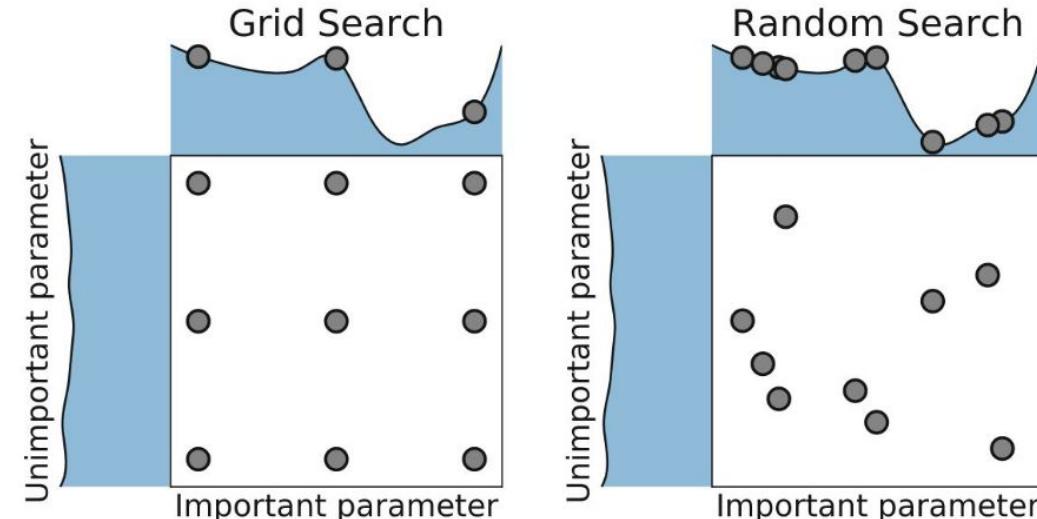


Image source: [\[Hutter et al. 2019\]](#)

→ 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?

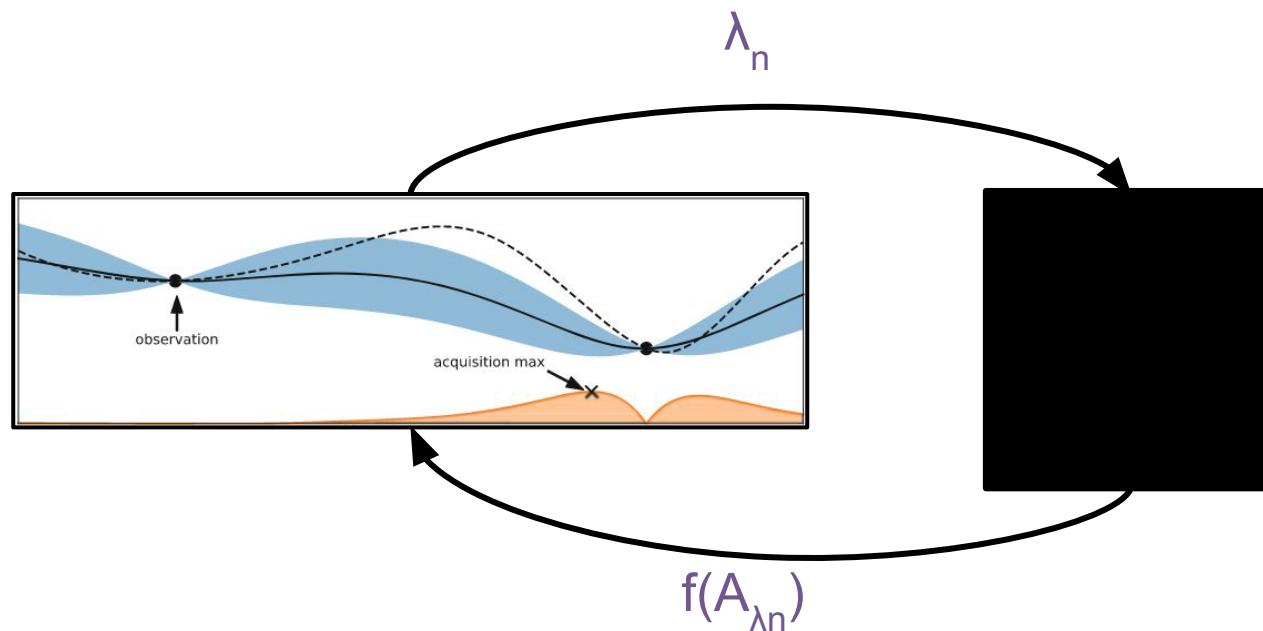
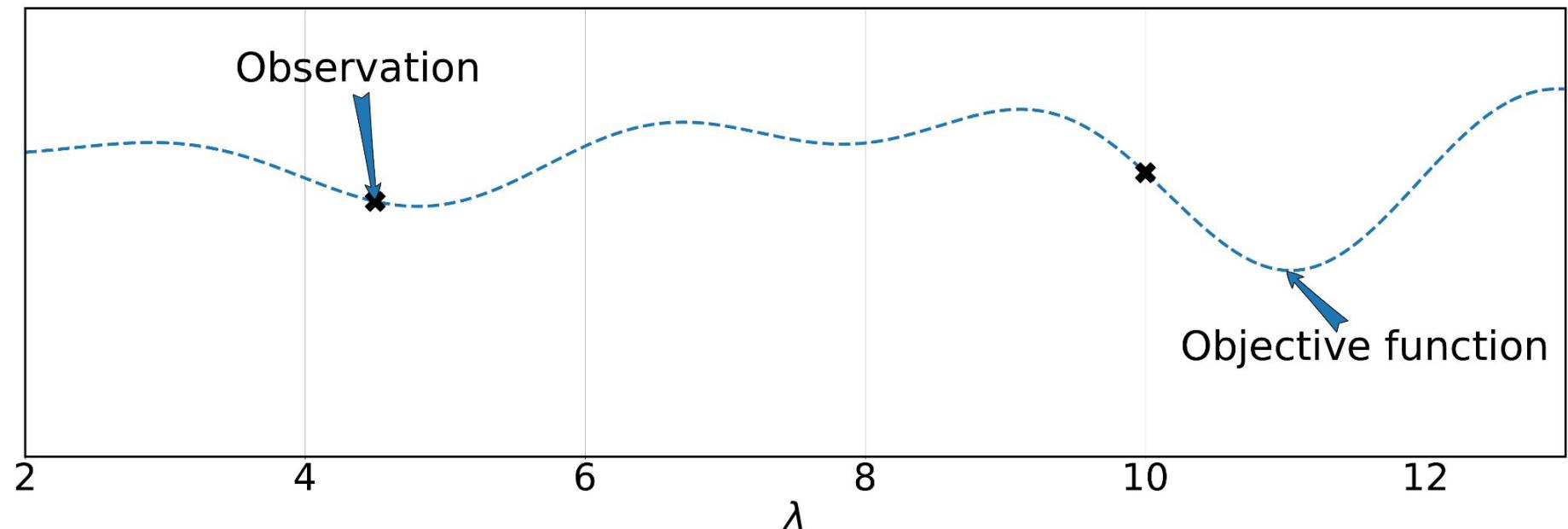
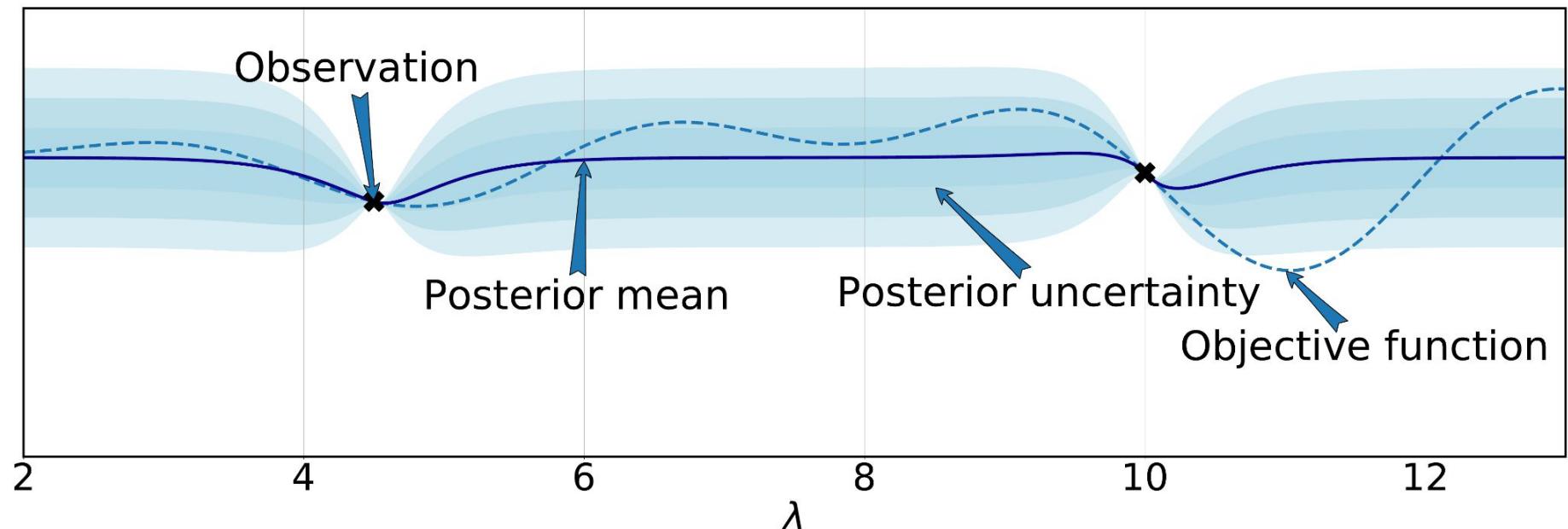
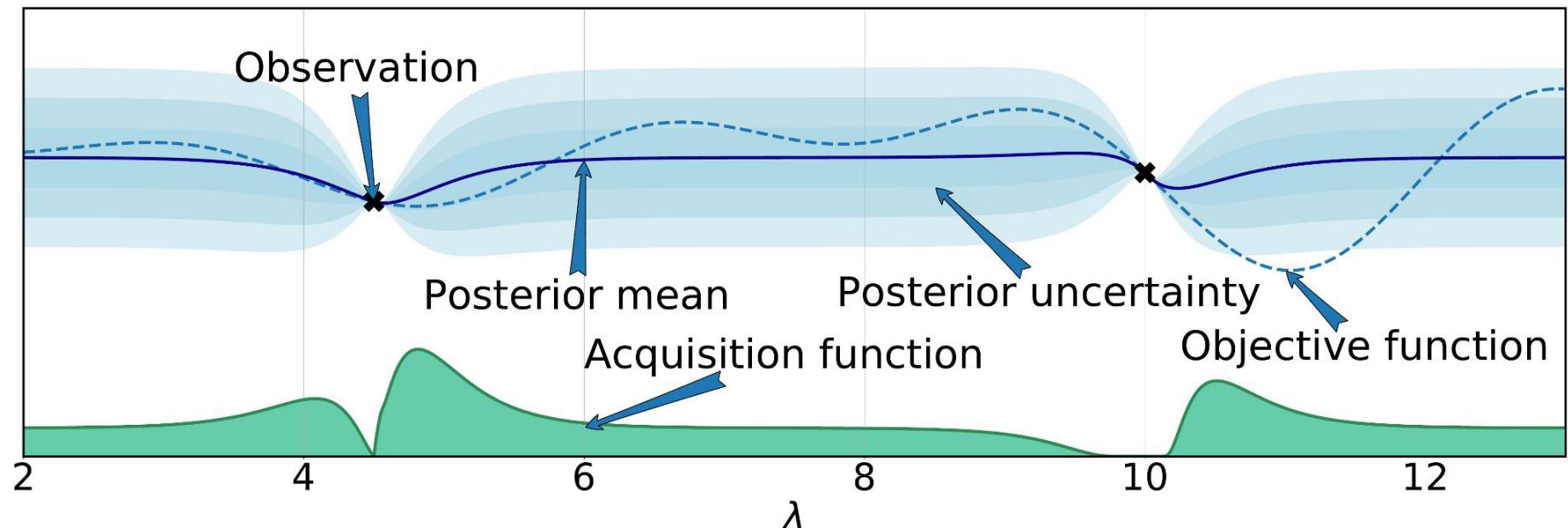
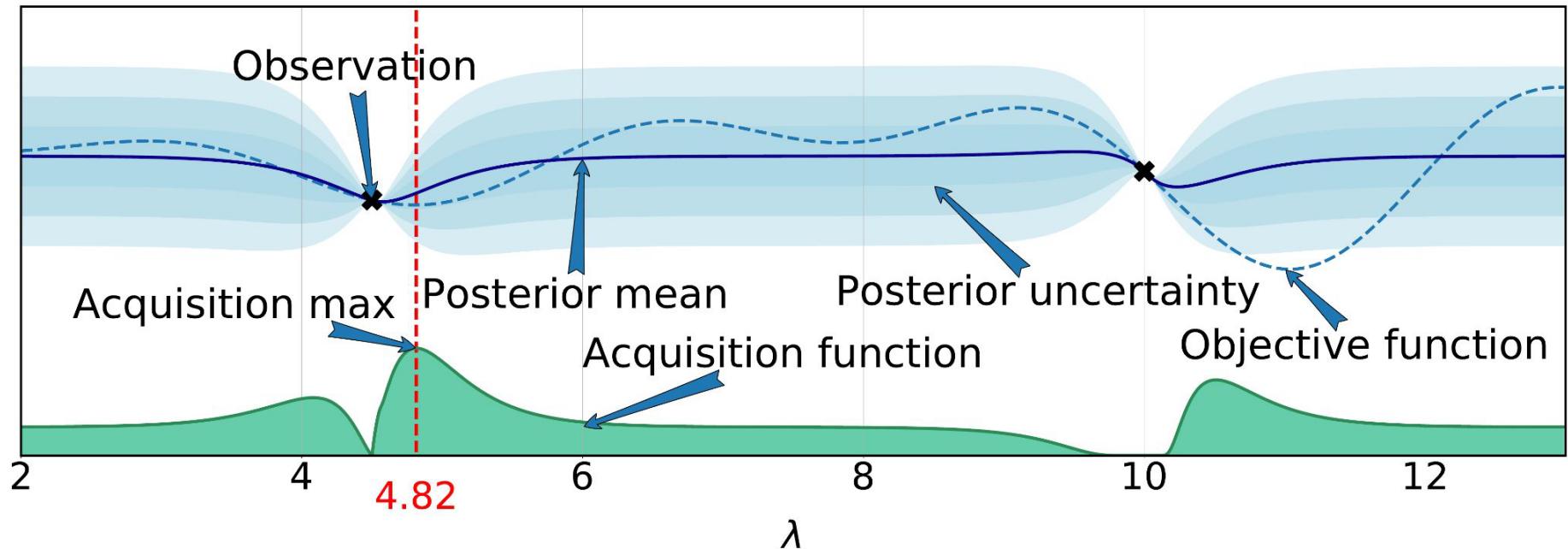


Photo by [Wilhelm Gunkel](#) on [Unsplash](#)
Image by Feurer, Hutter: Hyperparameter Optimization. In: Automated Machine Learning









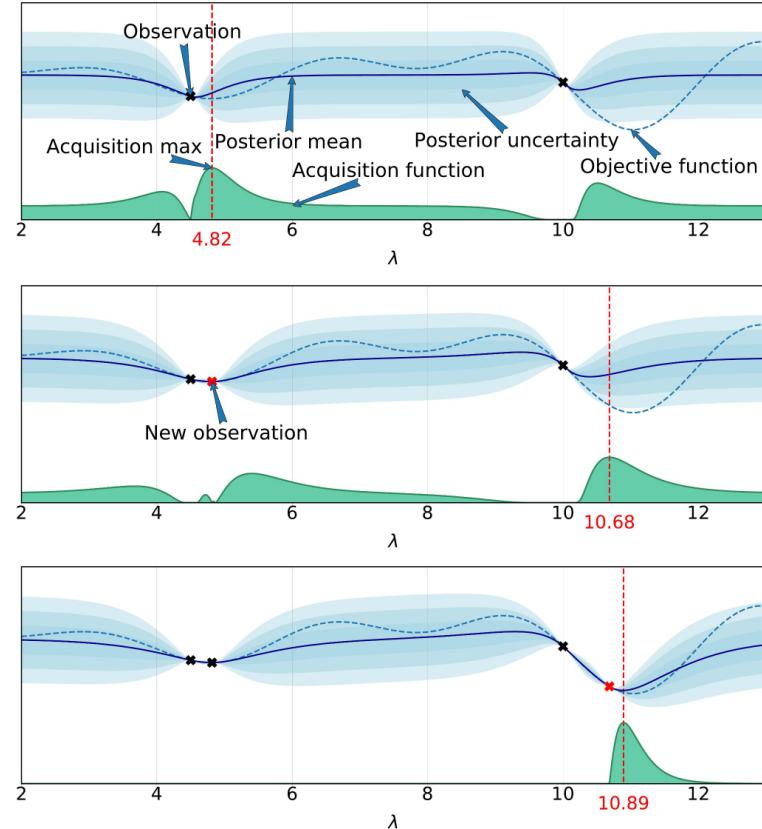


General approach

- Fit a **probabilistic model** to the collected function samples $\langle \lambda, c(\lambda) \rangle$
- 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]





BO loop

Require: Search space Λ , cost function c , acquisition function u , predictive model \hat{c} , maximal number of function evaluations T
Result : Best configuration $\hat{\lambda}$ (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: $\lambda^{(t)} \in \arg \max_{\lambda \in \Lambda} u(\lambda; \mathcal{D}^{(t-1)}, \hat{c}^{(t)})$
- 5 Query $c(\lambda^{(t)})$
- 6 Update data: $\mathcal{D}^{(t)} \leftarrow \mathcal{D}^{(t-1)} \cup \{\langle \lambda^{(t)}, c(\lambda^{(t)}) \rangle\}$



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:

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

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



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

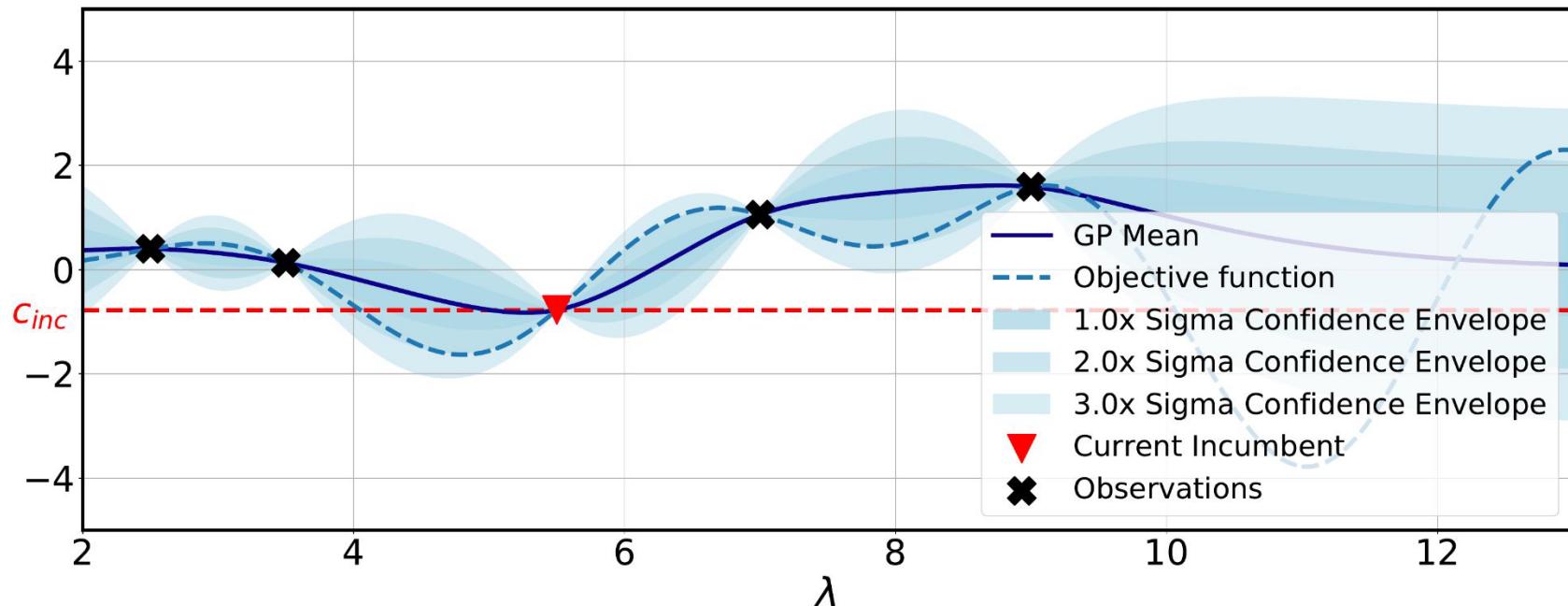


The acquisition function:

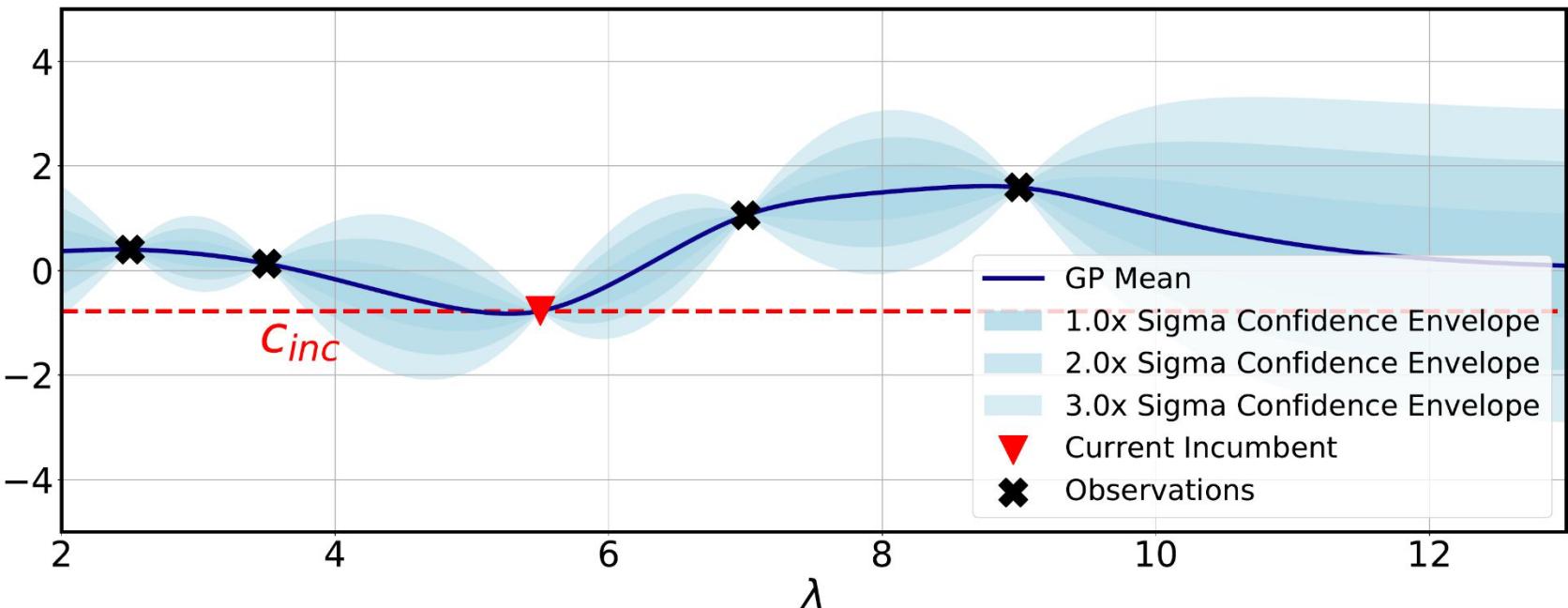
- decides which configuration to evaluate next
- judges the **utility** (or **usefulness**) of evaluating a configuration (based on the surrogate model)

→ 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



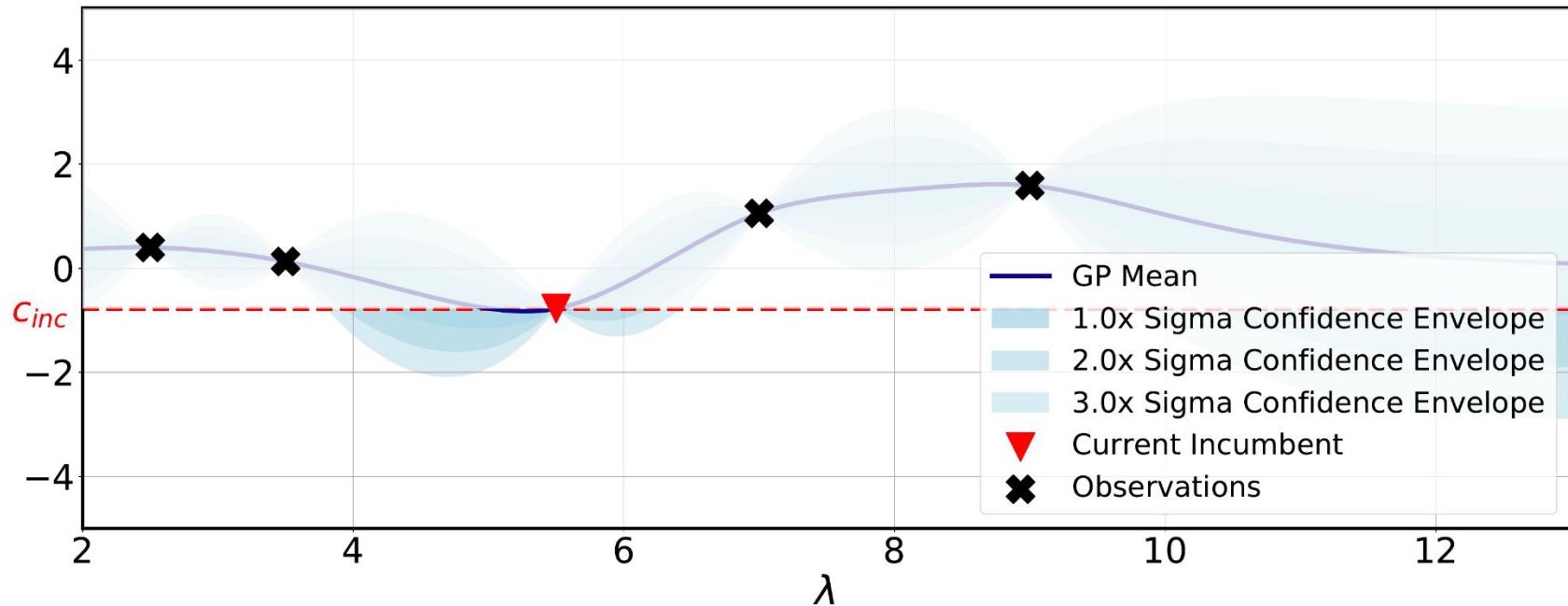
Given some observations and a fitted surrogate,



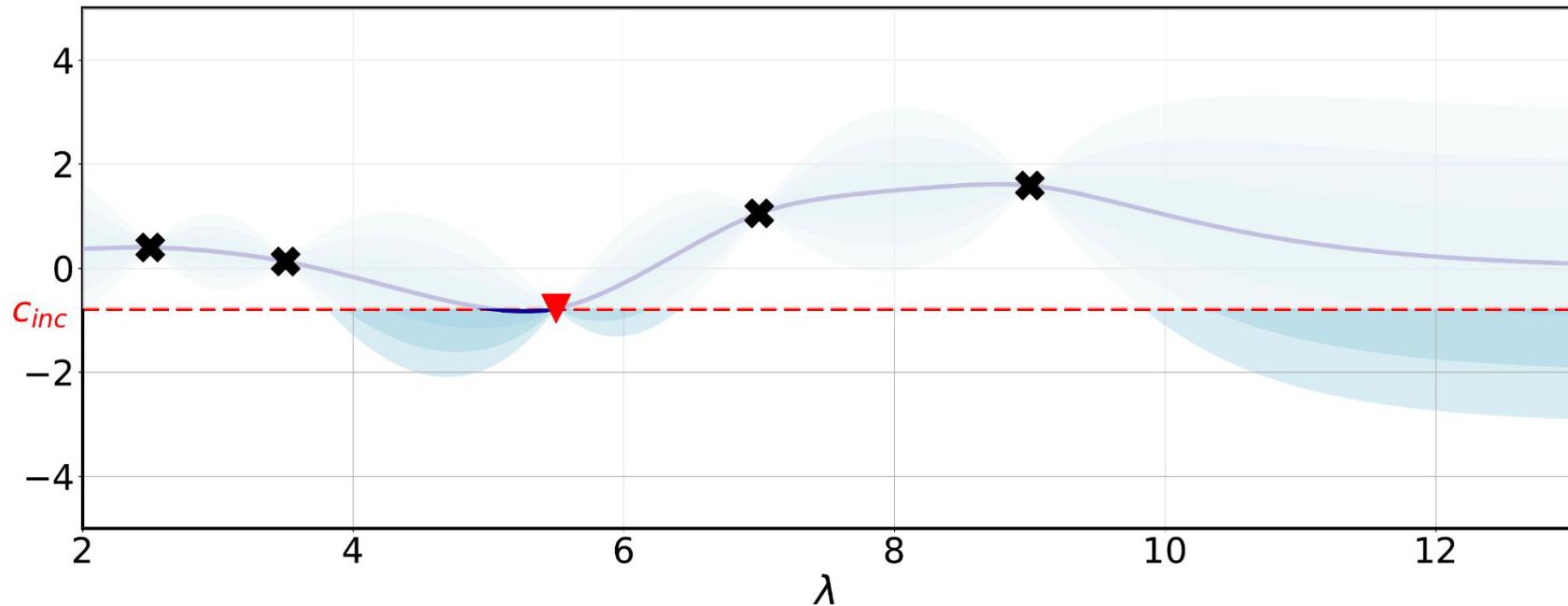
Given some observations and a fitted surrogate,



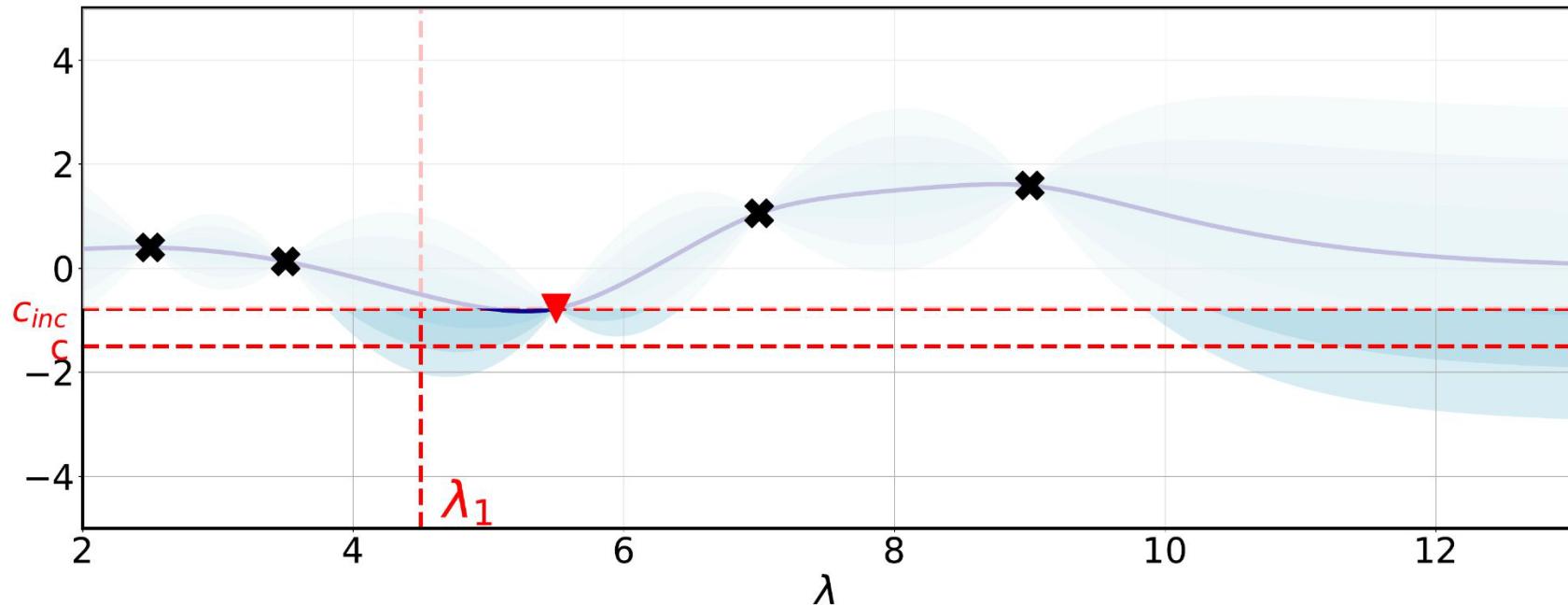
Expected Improvement (EI)



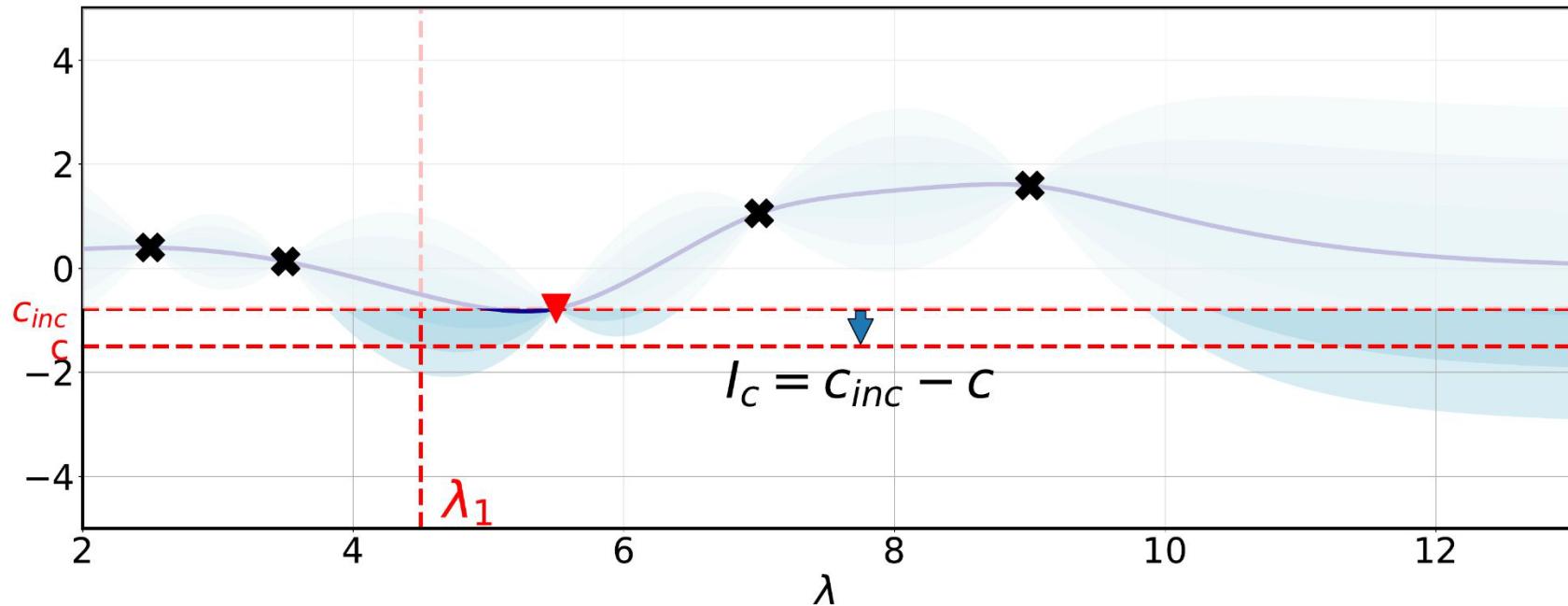
We care about *improving* over the c_{inc} .



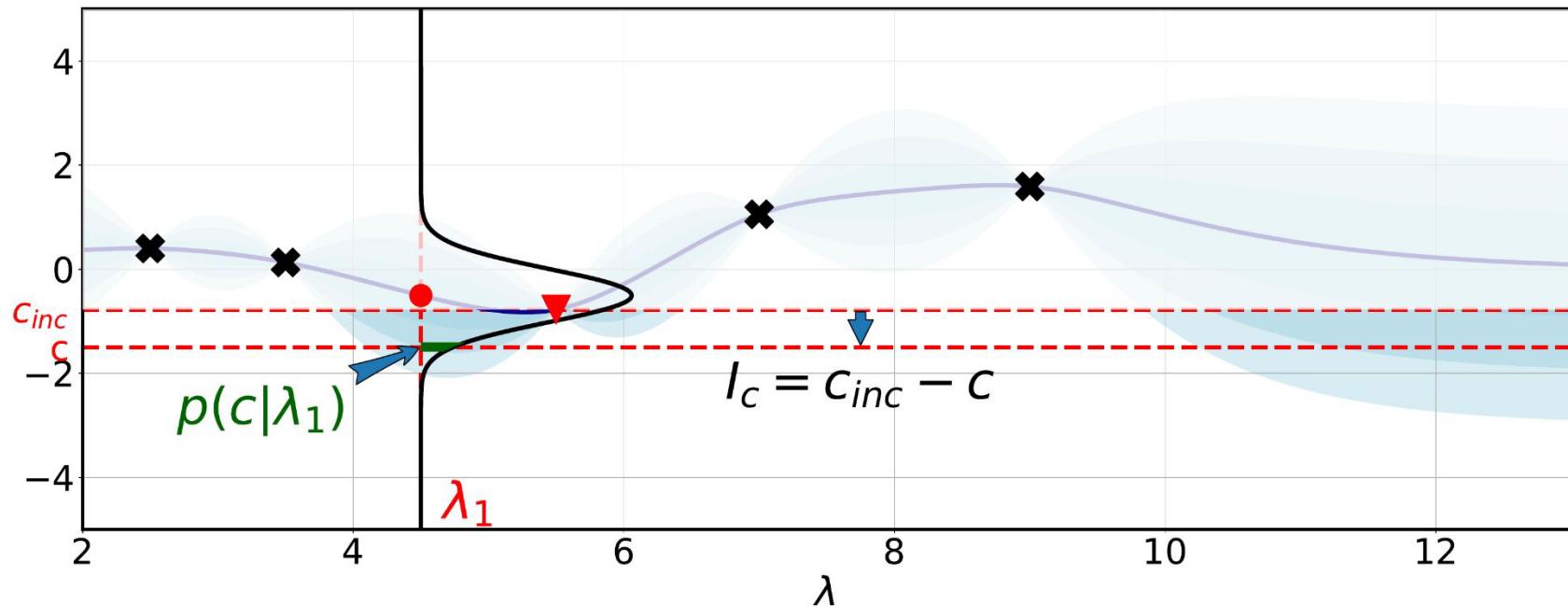
We care about *improving* over the c_{inc} .



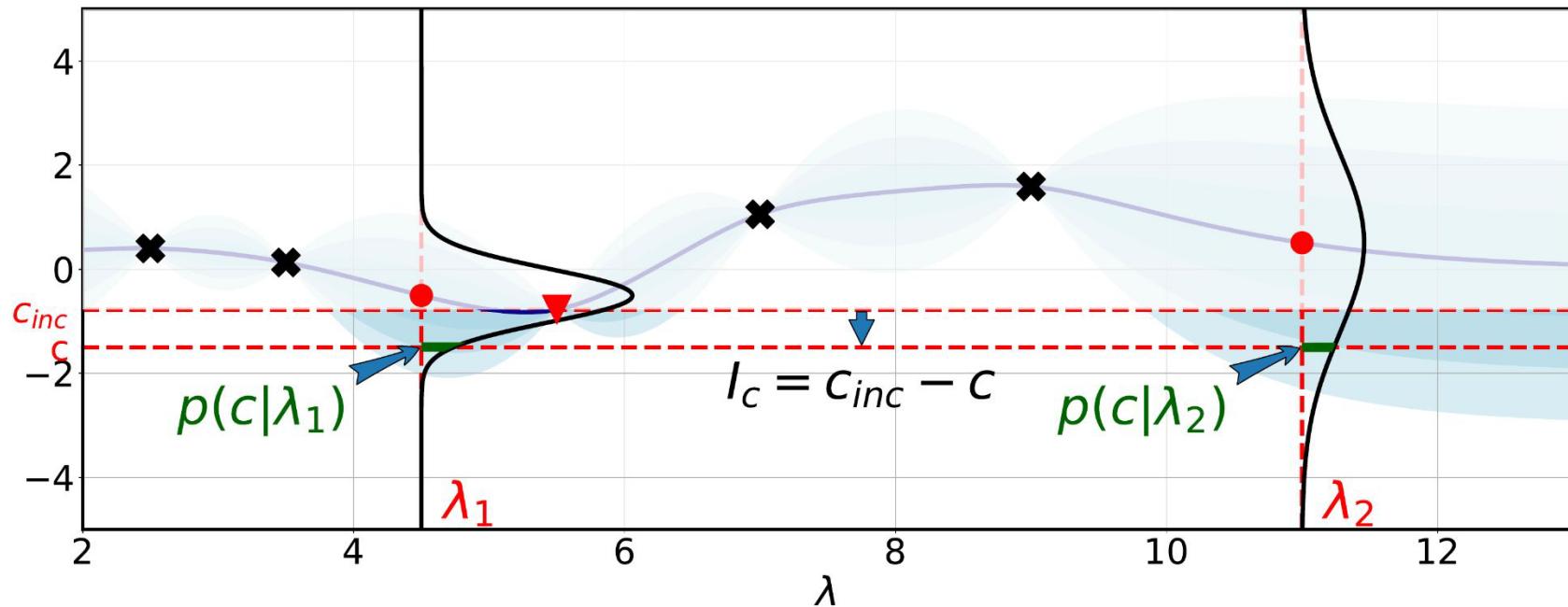
Let's look at a candidate configuration λ_1 and its hypothetical cost c .



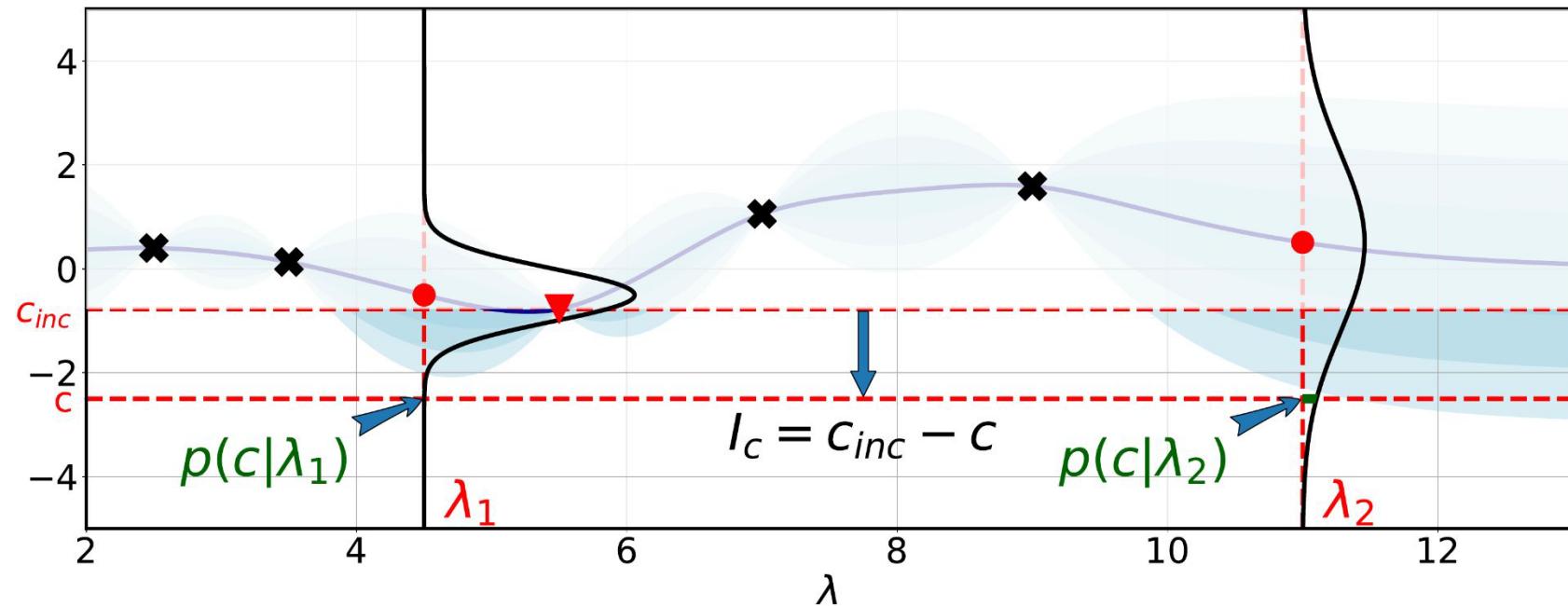
We can compute the improvement $I_c(\lambda_1)$. But how likely is it?



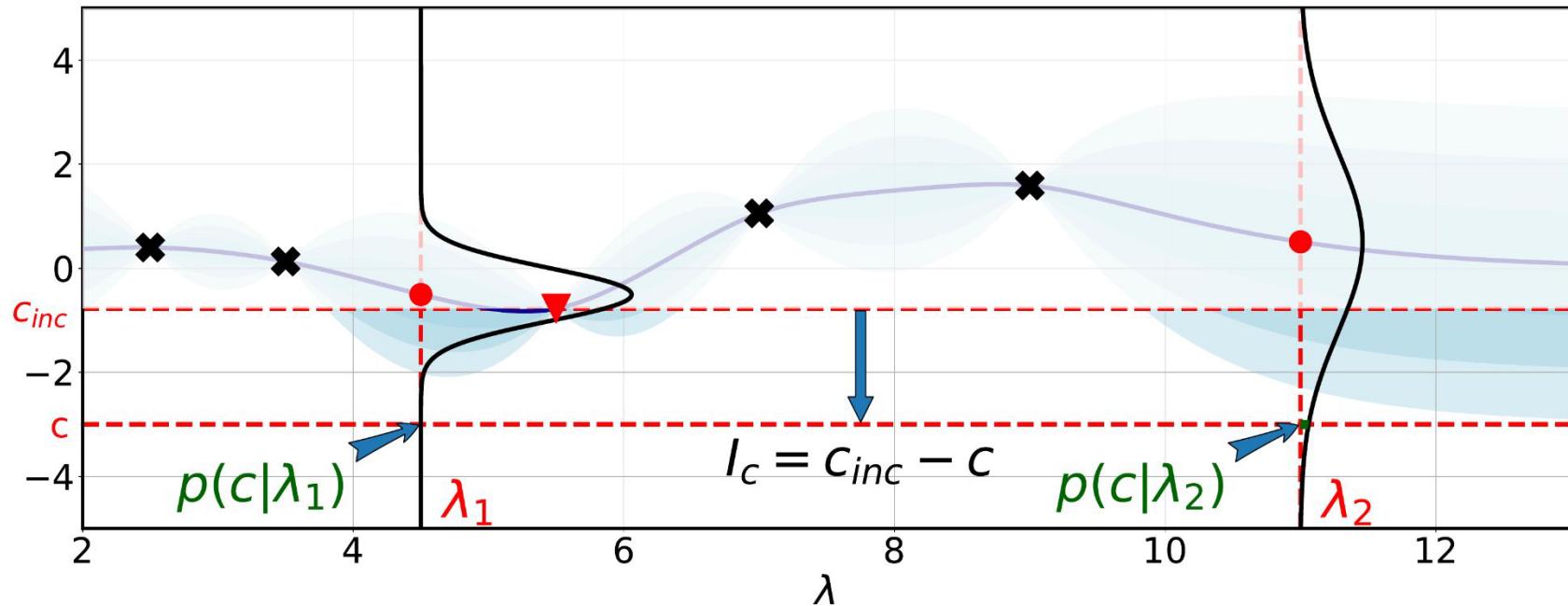
Knowing that $\hat{c}(\lambda) = \mathcal{N}(\mu(\lambda), \sigma^2(\lambda))$, we can compute $p(c|\lambda)$



Comparing this for different configurations



and costs.



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



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

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

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



- **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



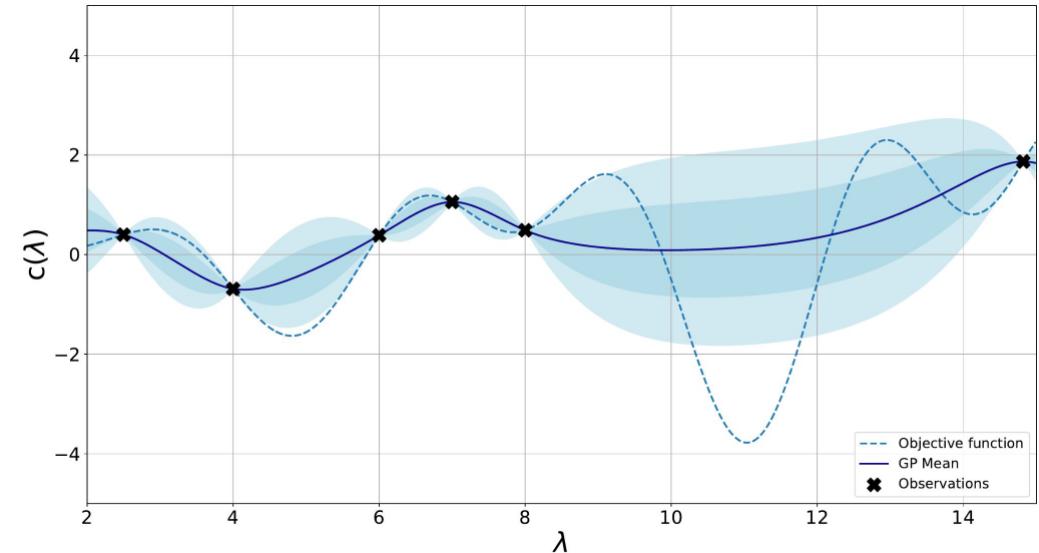
Questions?

**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





- Gaussian Processes
- Random Forests
- Bayesian Neural Networks

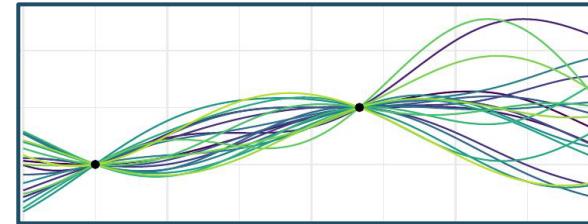


Photo by [Filip Zrnzević](#) on [Unsplash](#)

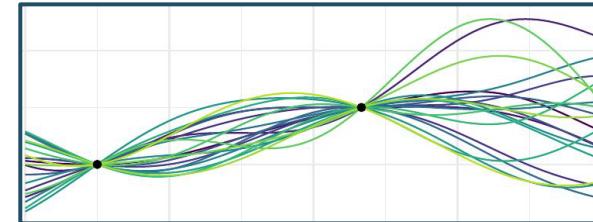
Photo by [Alina Grubnyak](#) on [Unsplash](#)



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

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

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



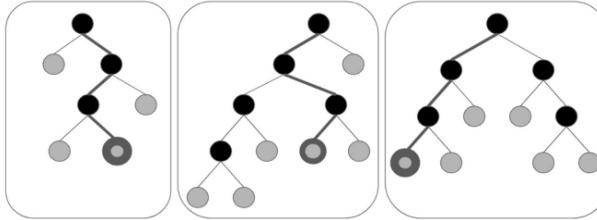
Advantages

- 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



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 Zrnjević](#) on [Unsplash](#)

Photo by [Alina Grubnyak](#) on [Unsplash](#)

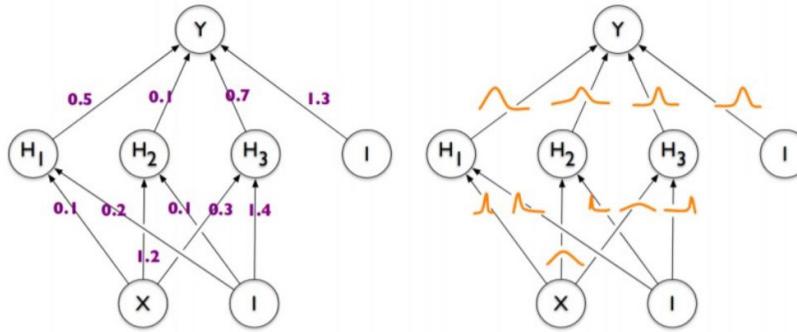


Image source: [Blundell et al. 2015]

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 Zrnjević](#) on [Unsplash](#)

Photo by [Alina Grubnyak](#) on [Unsplash](#)



Questions?

How to evaluate ML models when using HPO?

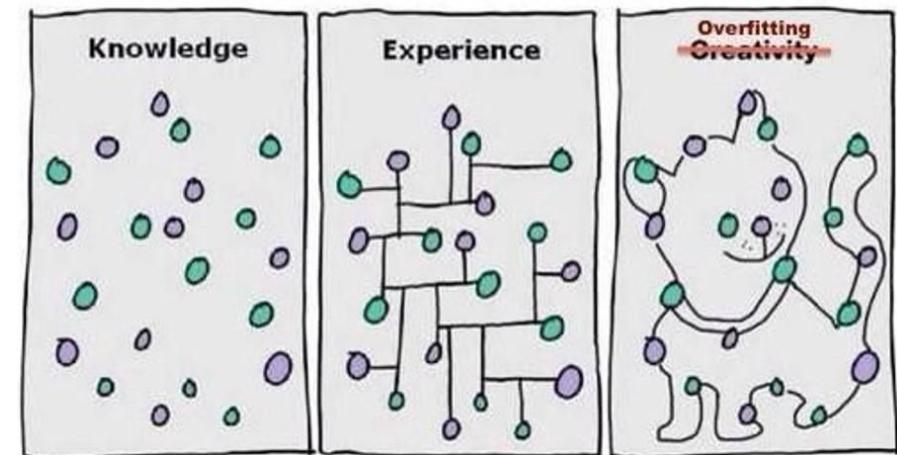


We want solutions that **generalize to new data!**

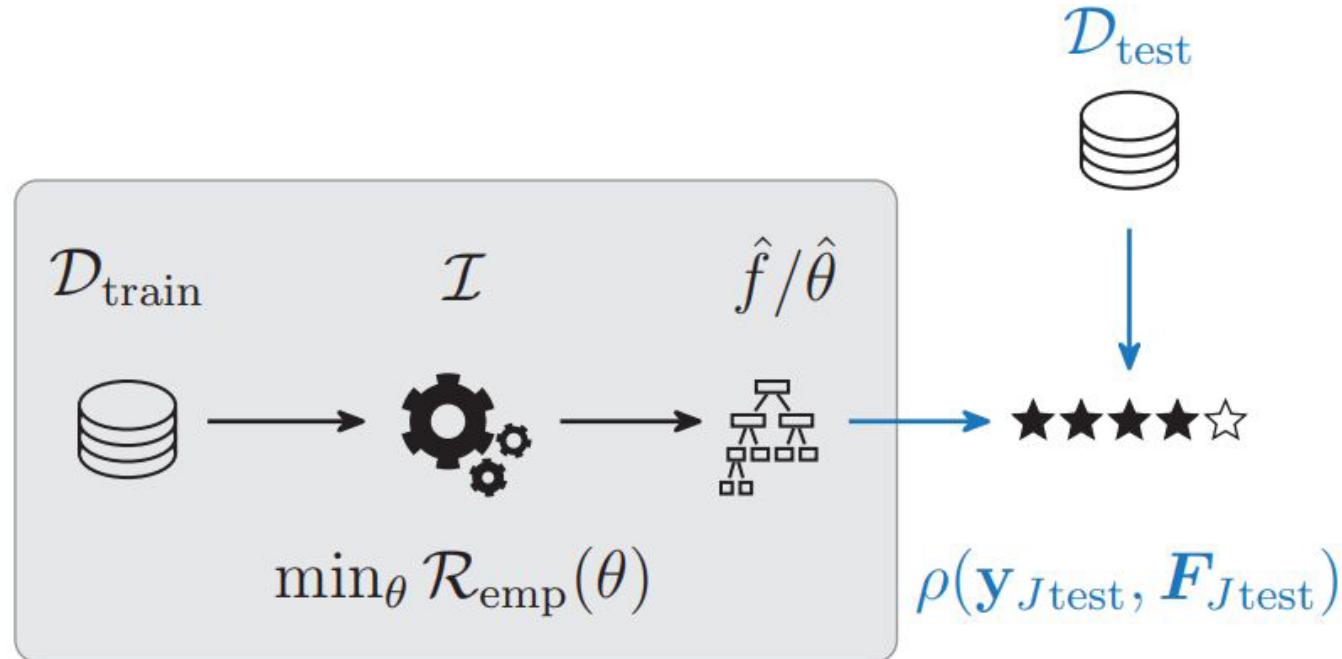
→ “reasonable” predictions
on **new data**

This might include:

- ignoring outliers
- smooth
- capturing general trend

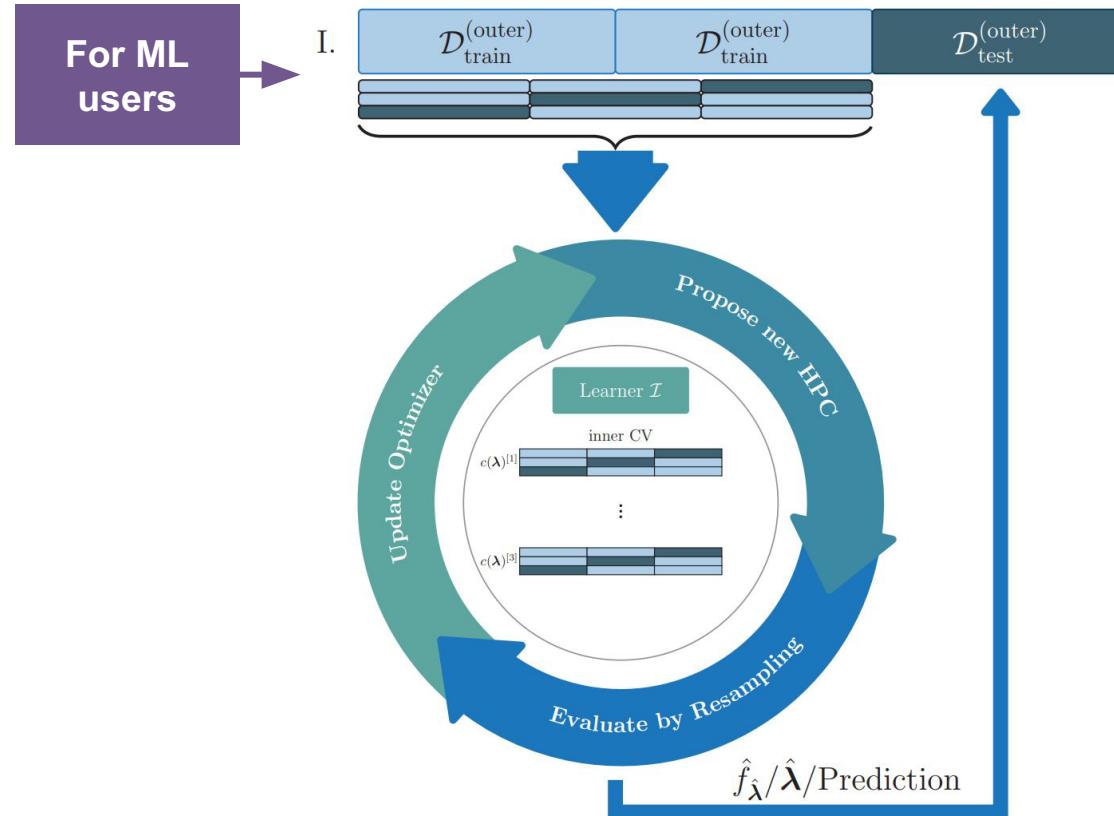


Source: Kaushik, 2016

Image: [Bischl et al. 2023](#)



Nested crossvalidation for AutoML

Image: [Bischl et al. 2023](#)



Nested CV for AutoML

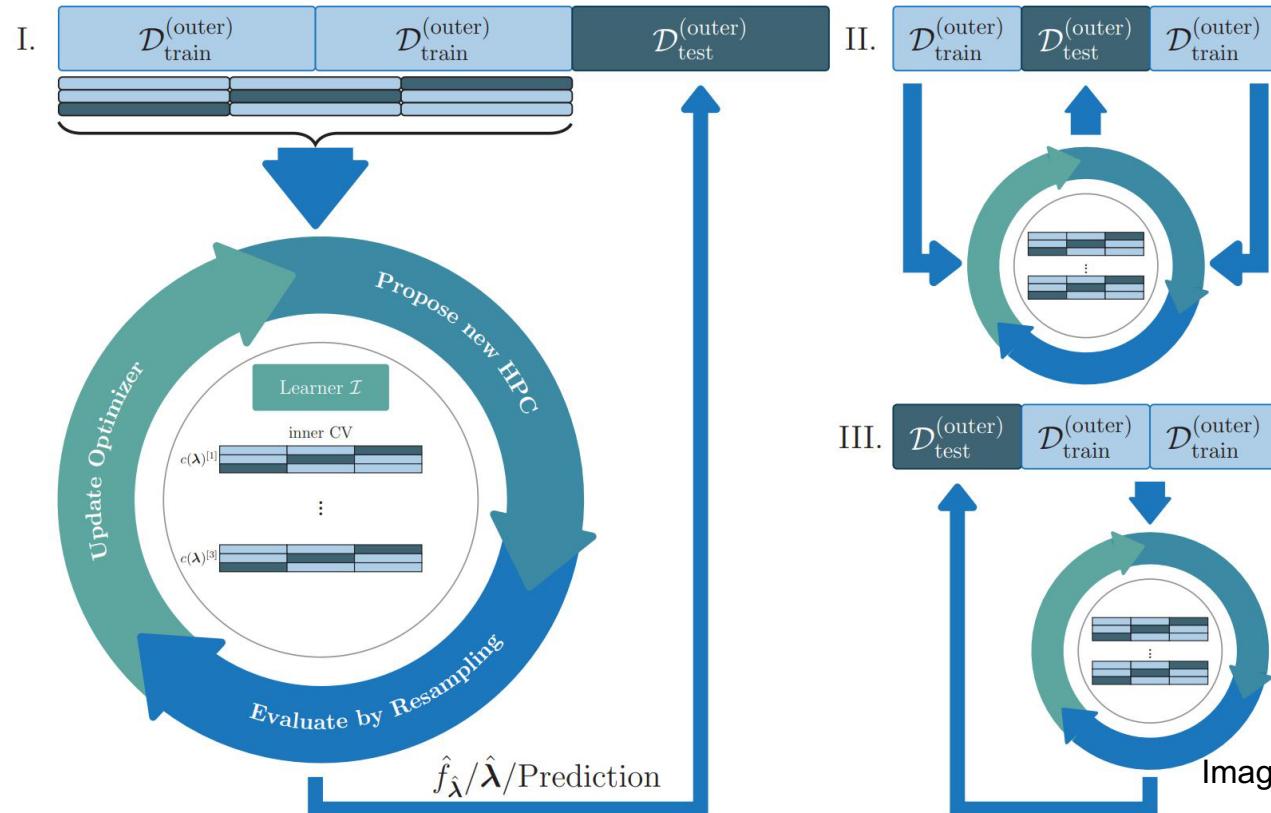
Image: [Bischl et al. 2023](#)



Illustration: Resampling vs. Nested Resampling

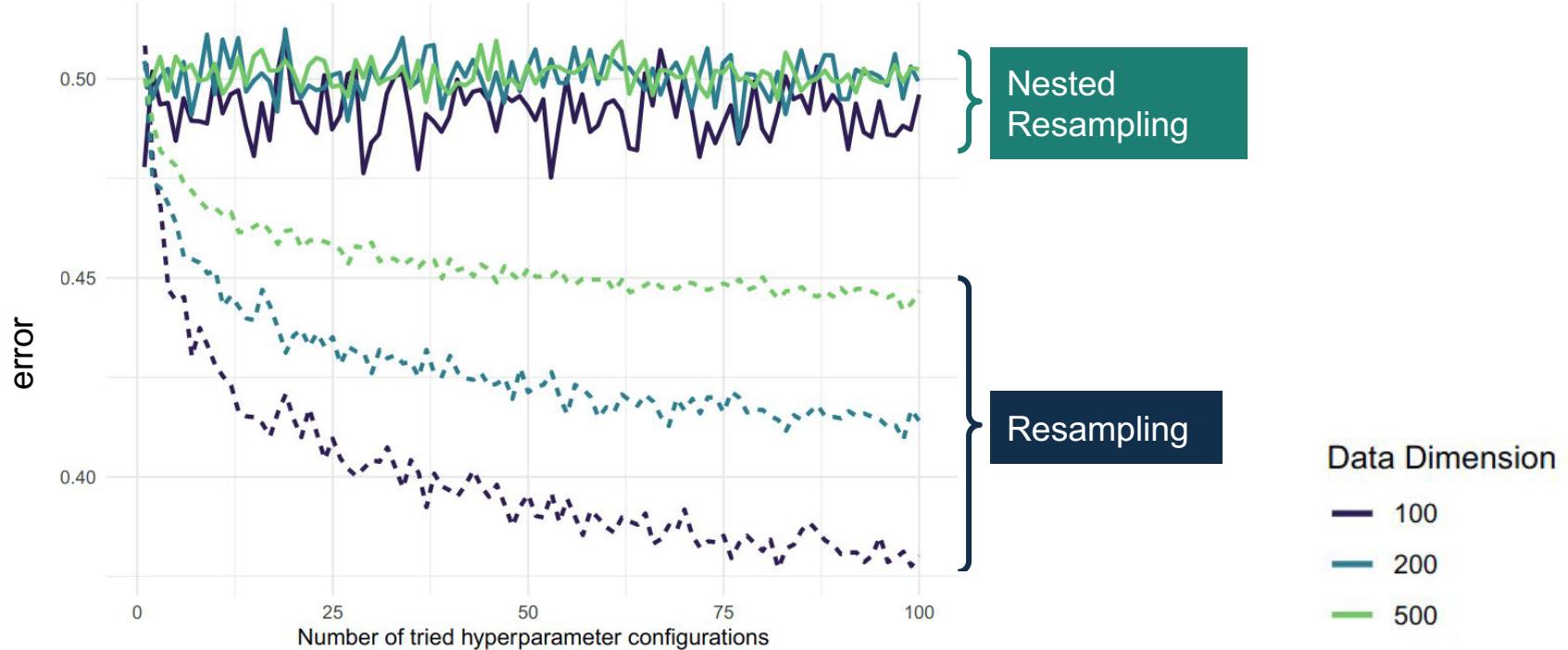


Image: [Bischl et al. 2023](#)



Questions?



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 Practices and Open Challenges” - <https://arxiv.org/pdf/2107.05847.pdf>