Bayesian Optimization for Hyperparameter Optimization

Marius Lindauer

Leibniz Universität Hannover
Germany

/ LindauerMarius
m.lindauer@ai.uni-hannover.de

Katharina Eggensperger

Eberhard Karls Universität Tübingen
Germany

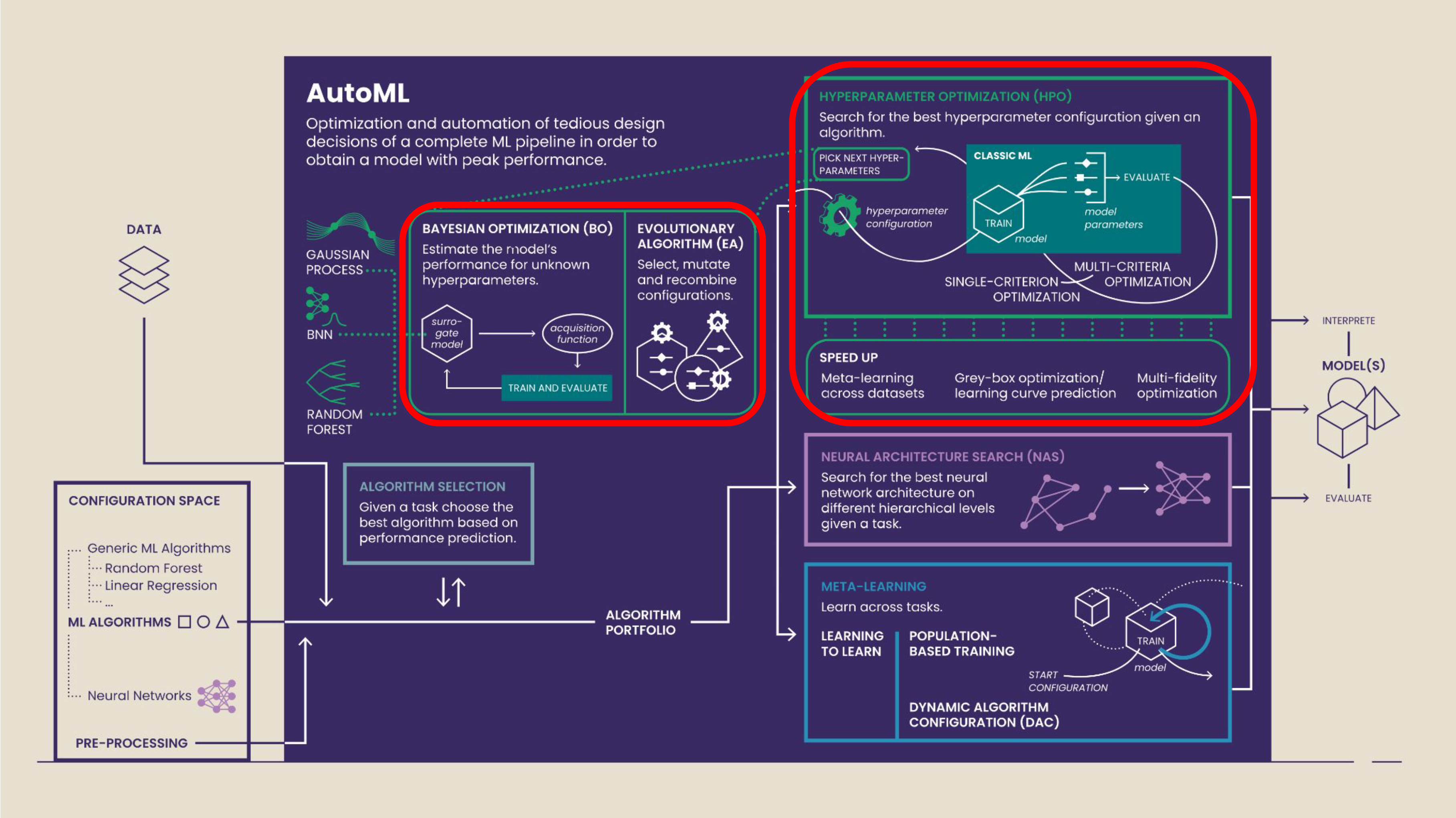
/ KEggensperger
katharina.eggensperger@uni-tuebingen.de

Questions?

Let’s use sli.do.

Please use “Day 2”!

https://app.sli.do/event/fptZhnBcuqozfF7NQ3gBm2


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
EVALUATE
hyperparameter configuration
TRAIN
model
model parameters
MULTI-CRITERIA OPTIMIZATION
SINGLE-CRITERION OPTIMIZATION
SPEED UP
Meta-learning across datasets
Grey-box optimization/ learning curve prediction
Multi-fidelity optimization
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
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)

Story Line 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
- How do we optimize it even more?
  - Multi-Fidelity Optimization using Hyperband
  - Multi-Objective Optimization using ParEGO
- Demo: SMAC

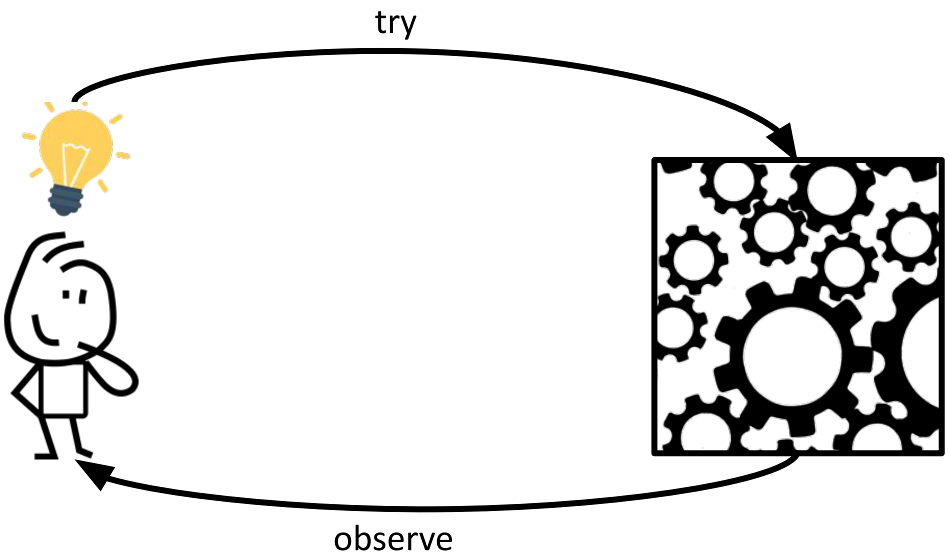
Note: This lecture is based on the free online lecture “Automated Machine Learning” at https://learn.ki-campus.org/courses/automl-luh2021

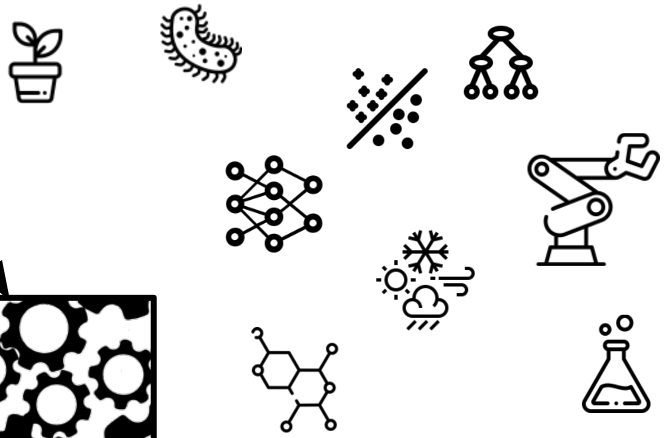
- Basics of HPO
- Bayesian Optimization for HPO
- Speedup Techniques for Hyperparameter Optimiziation
- Multi-criteria Optimization

What do we optimize?

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

Sequential Experimentation



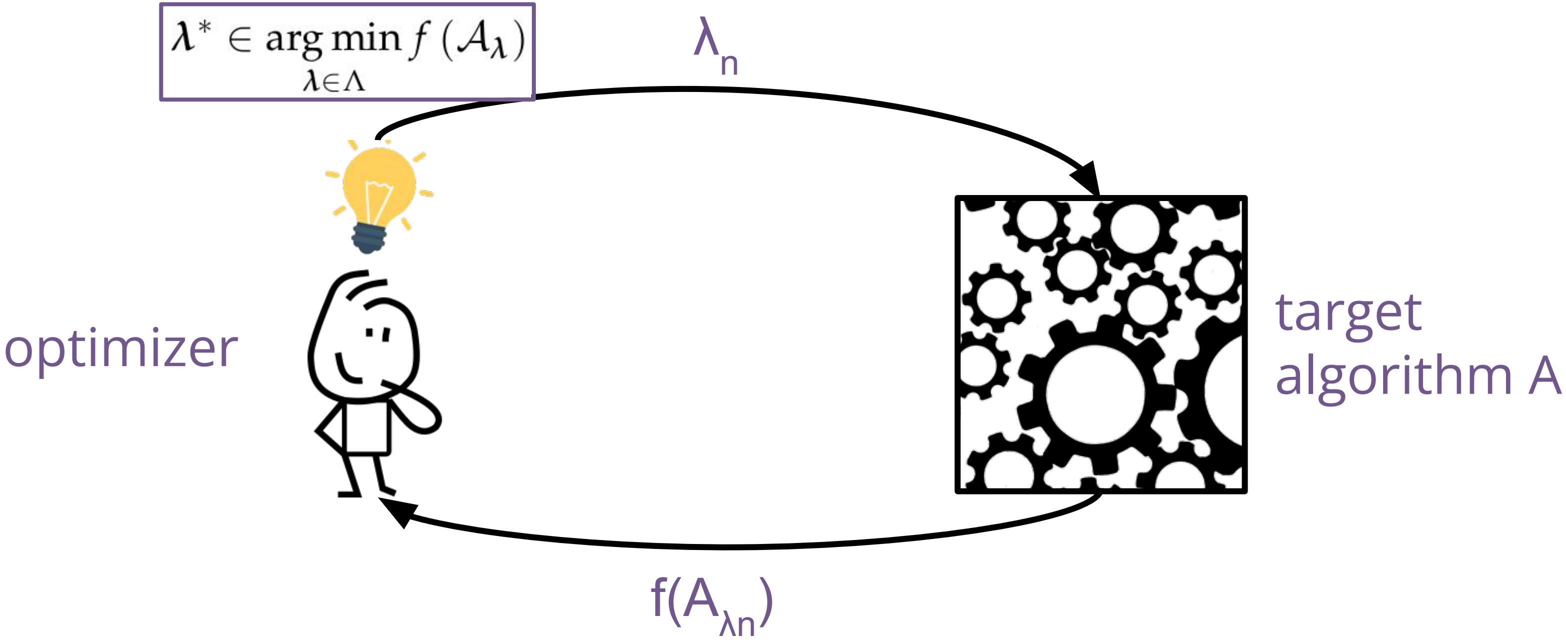


- Algorithm
- Neural Architecture
- Network Training
- Data Science Pipeline
- Simulator
- Robot

Hyperparameter Optimization

Goal: Find the best performing configuration:

$$\boxed{\lambda^* \in \underset{\lambda \in \Lambda}{\arg\min}\, f(\mathcal{A}_\lambda)}$$



Example

- **Given a dataset**, we want to train a neural network
- We need to choose a **learning rate and architecture**
- The “learner” takes the input data, and returns a fitted network

→We are interested in **generalization error**!
→ We need to look at how our trained model **performs on “unseen” data**
→ We evaluate different settings and select the one that **performs best w.r.t generalization error**.

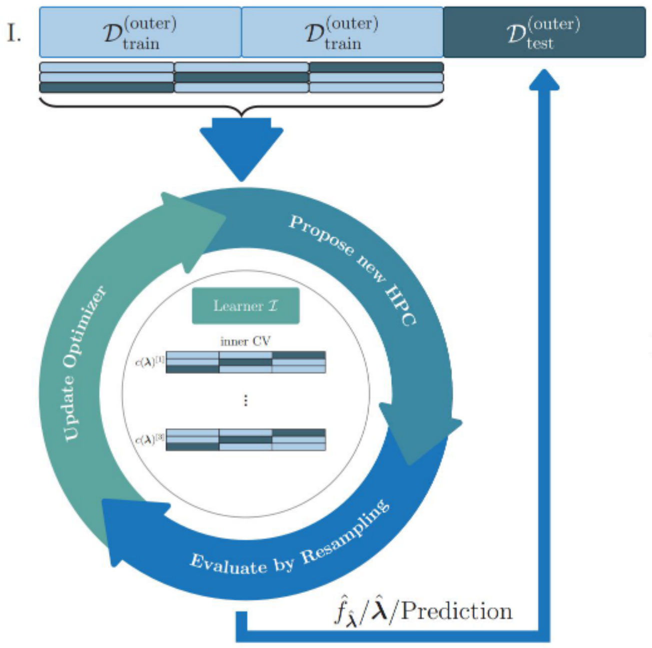


Image: Bischl et al. 2023

Hyperparameters and Parameters

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

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

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

- Max. depth of a Decision Tree
- Number of layers of a Neural Network
- K for K-Nearest Neighbours

Types of Hyperparameters

Real-valued

- Learning rate for SGD to train NNs
- Bandwidth of kernel density estimates in Naive Bayes

Integer

- #Neurons in a layer of a NN
- maximum depth of a Decision Tree

Can also be on a log-scale

Categorical

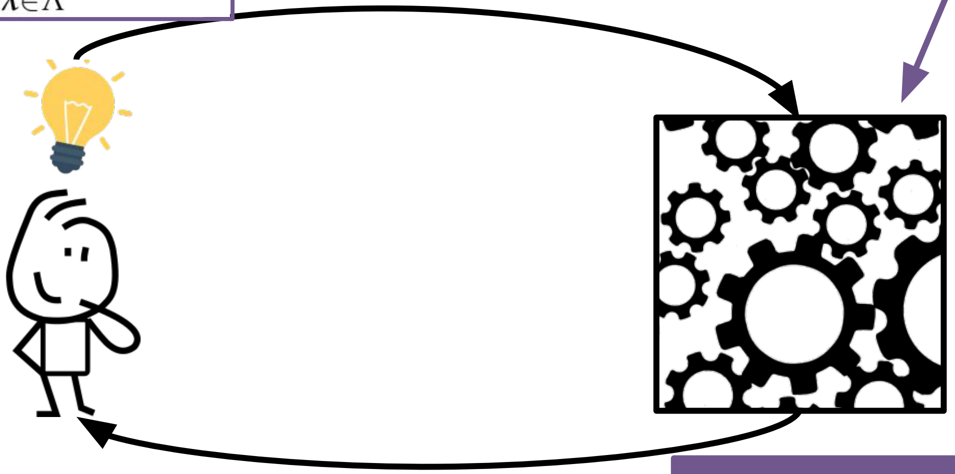
- Training Algorithm for NNs
- Split criterion for Decision Trees

+ Hyperparameters can be **hierarchically dependent** on each other

Why is Hyperparameter Optimization Challenging?

Goal: Find the best performing configuration:

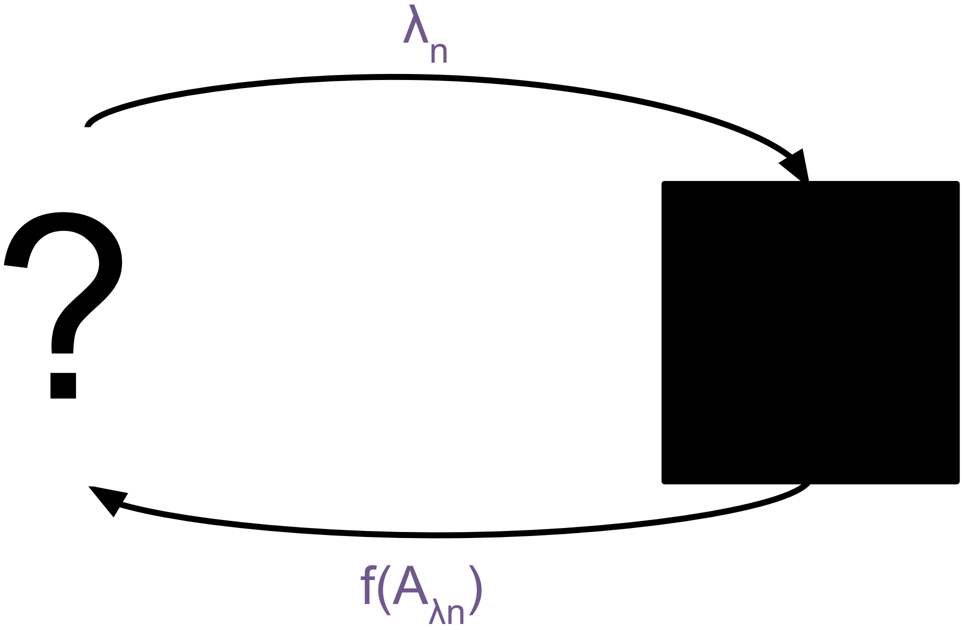
$$\lambda^* \in \arg\min_{\lambda \in \Lambda} f(\mathcal{A}_\lambda)$$



How do we optimize it?

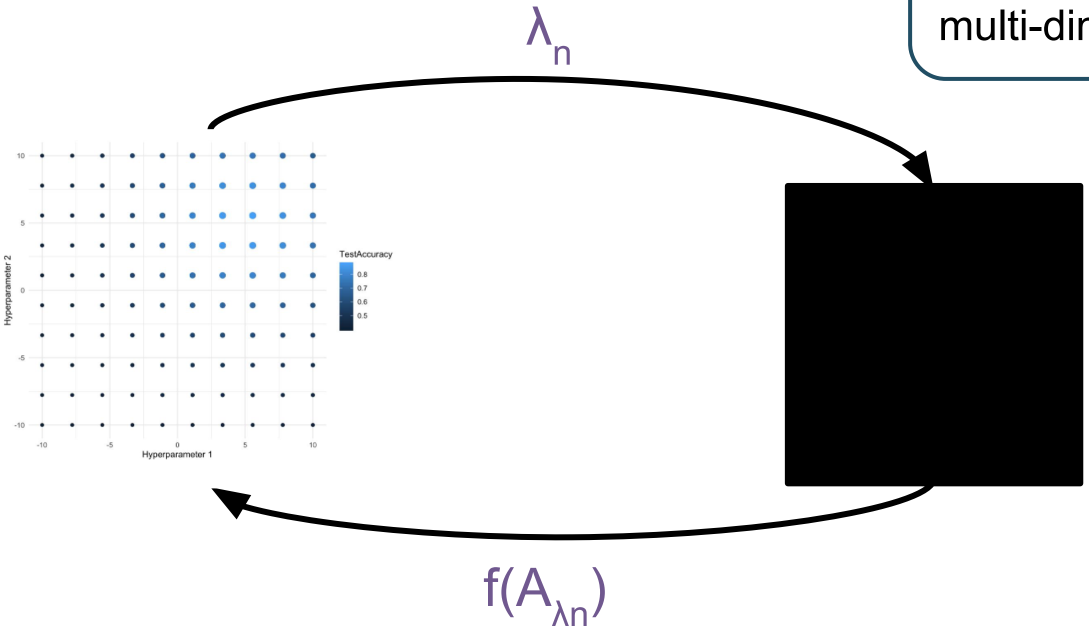
>> Here’s my algorithm, data, metric and search space, what should I do?

Black-Box Optimization Problem

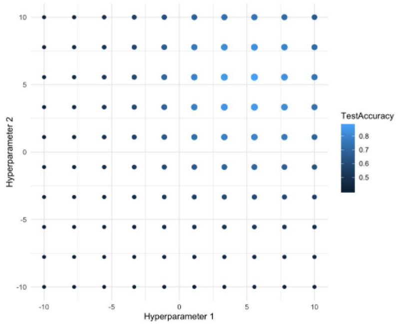


Option 1: Grid Search

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



Option 1: Grid Search II



Advantages

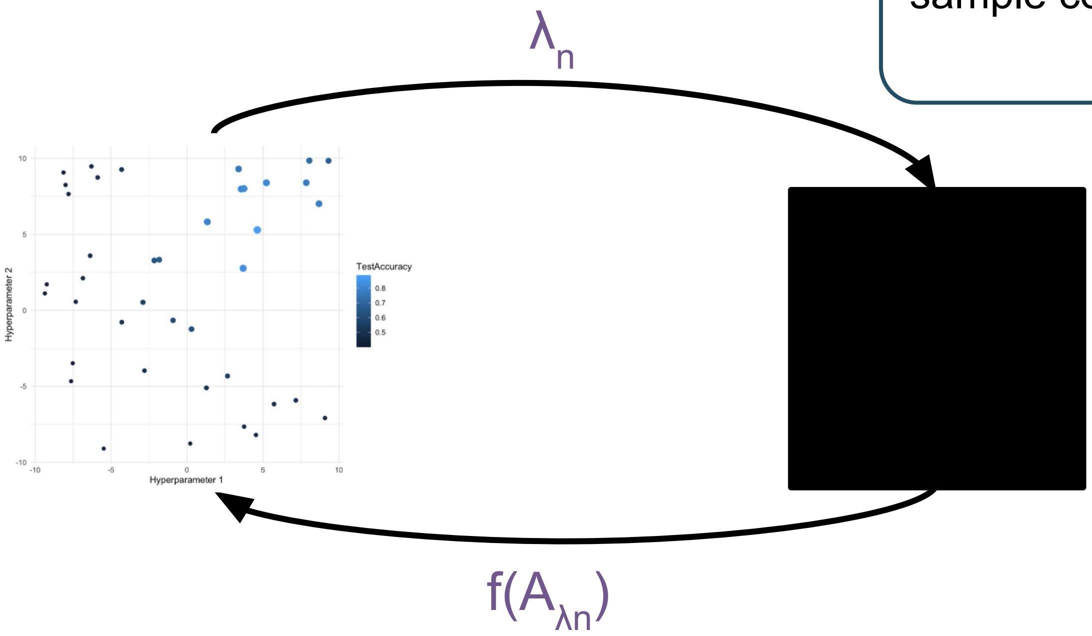
- Very easy to implement
- Very easy to parallelize
- Can handle all types of hyperparameters

Disadvantages

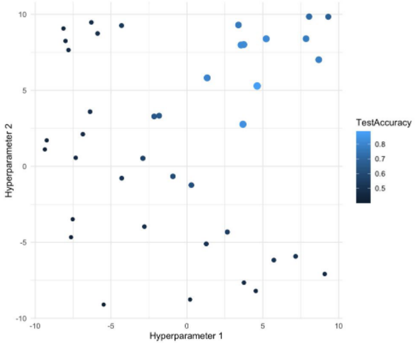
- Scales badly with #dimensions
- Inefficient: Searches irrelevant areas
- Requires to manual define discretization
- All grid points need to be evaluated

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

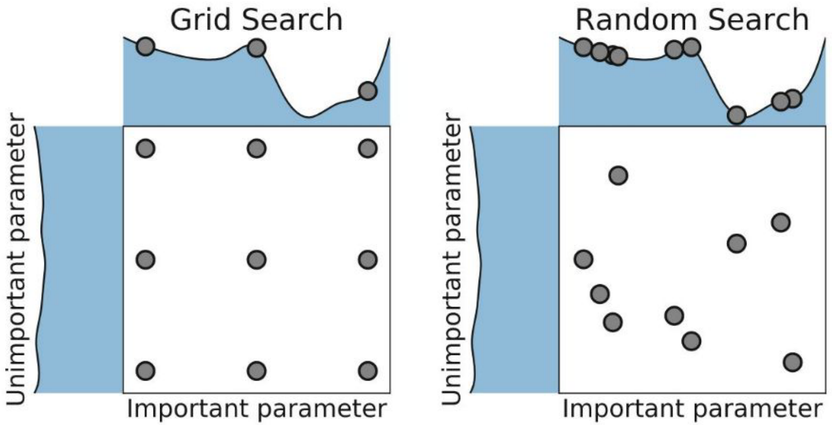


Image source: [Hutter et al. 2019]

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

Questions?

Kahoot Quiz I

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

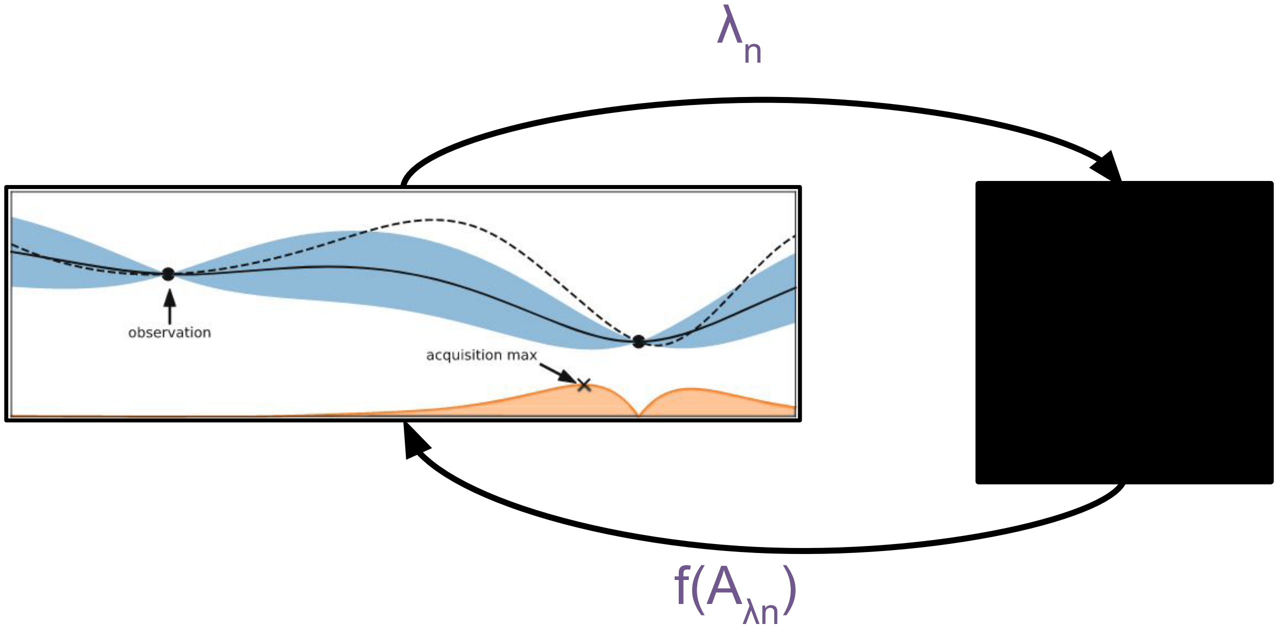
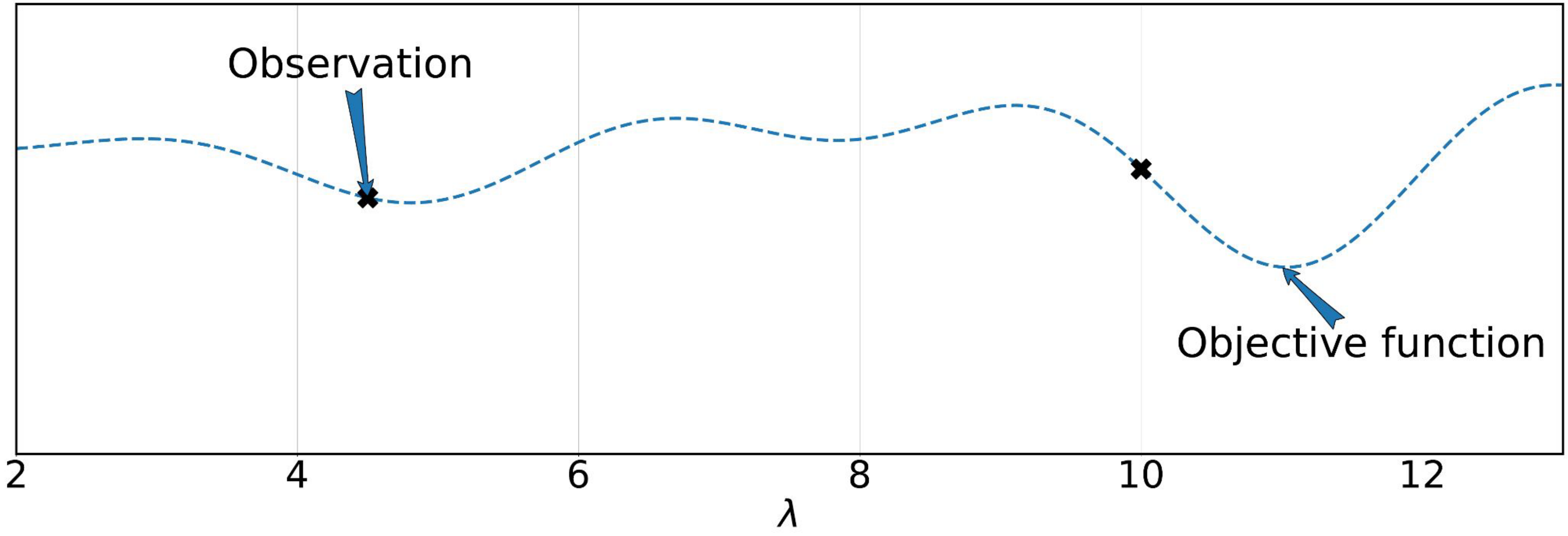
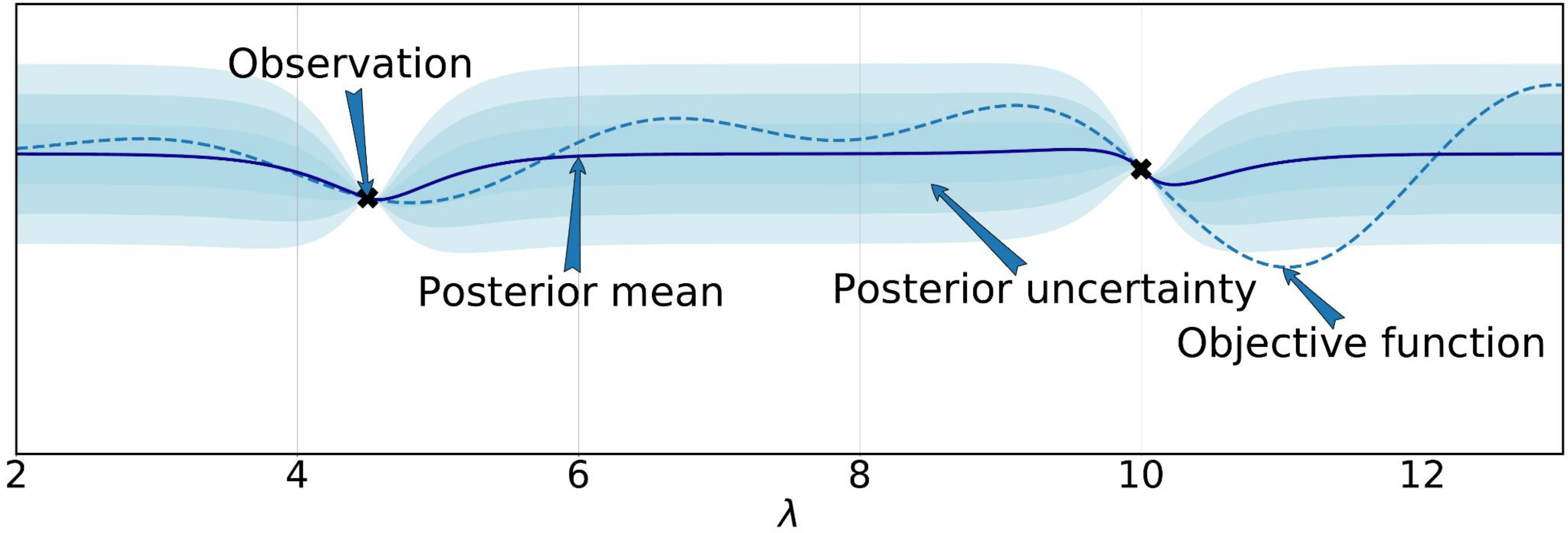


Photo by Wilhelm Gunkel on Unsplash
Image by Feurer, Hutter: Hyperparameter Optimization. In: Automated Machine Learning

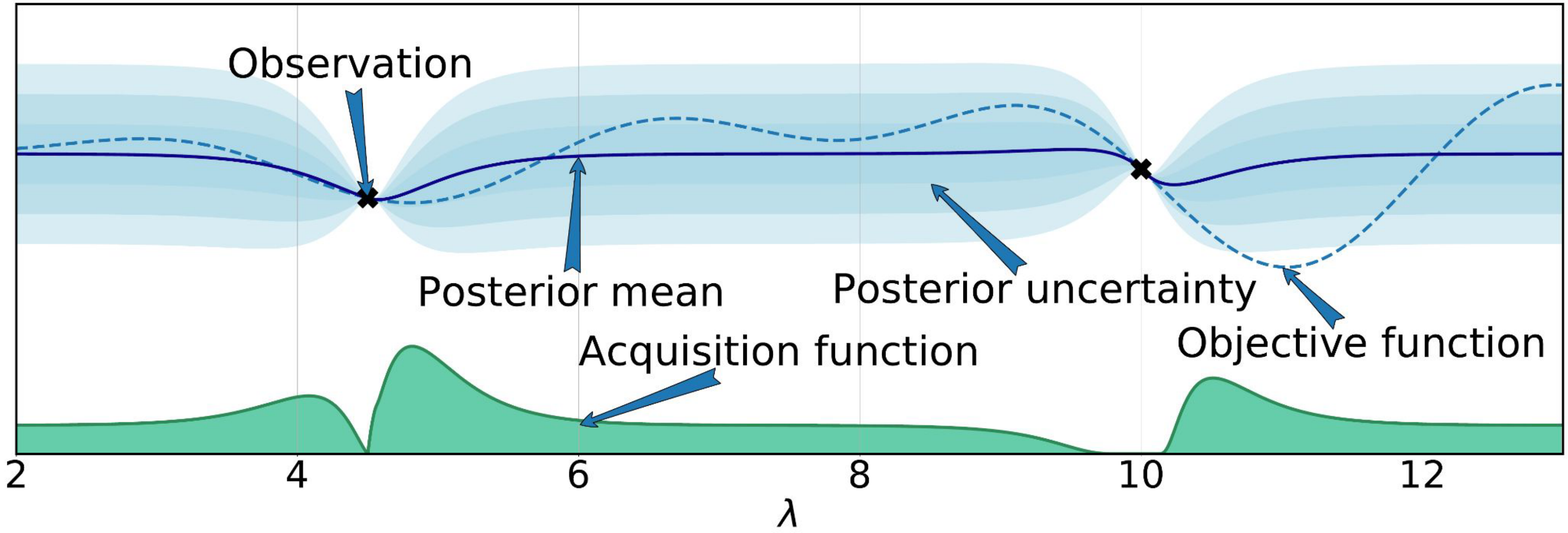
Bayesian Optimization in a Nutshell



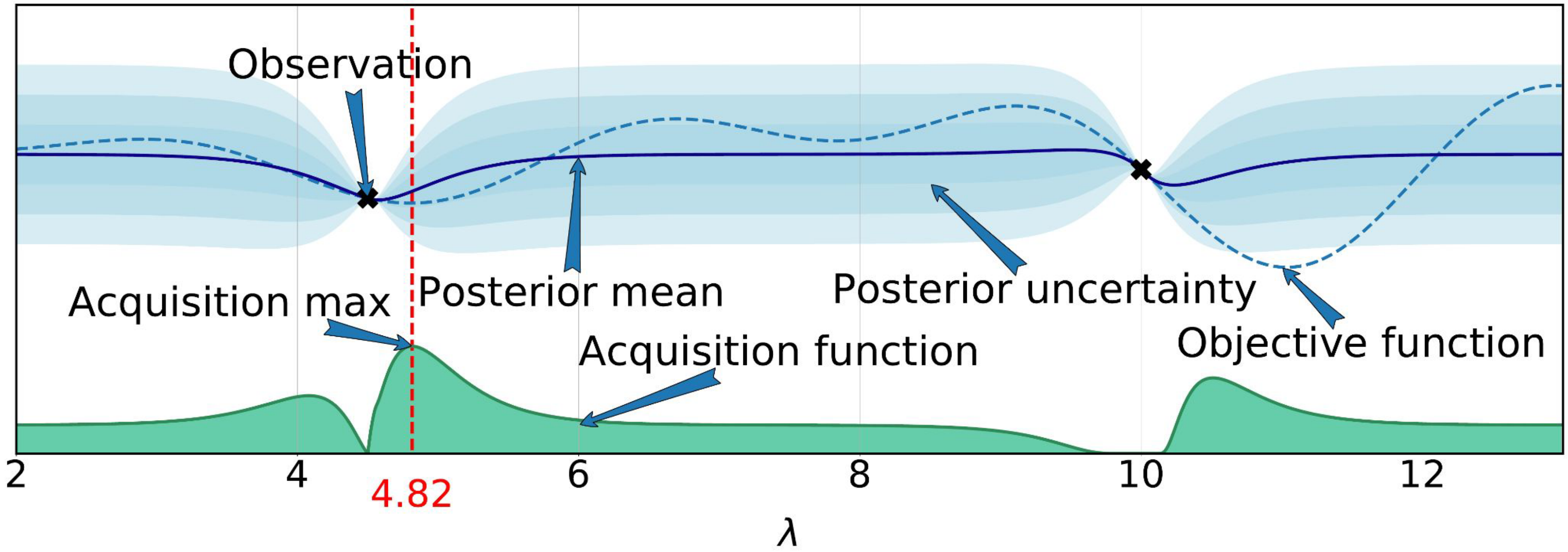
Bayesian Optimization in a Nutshell



Bayesian Optimization in a Nutshell



Bayesian Optimization in a Nutshell



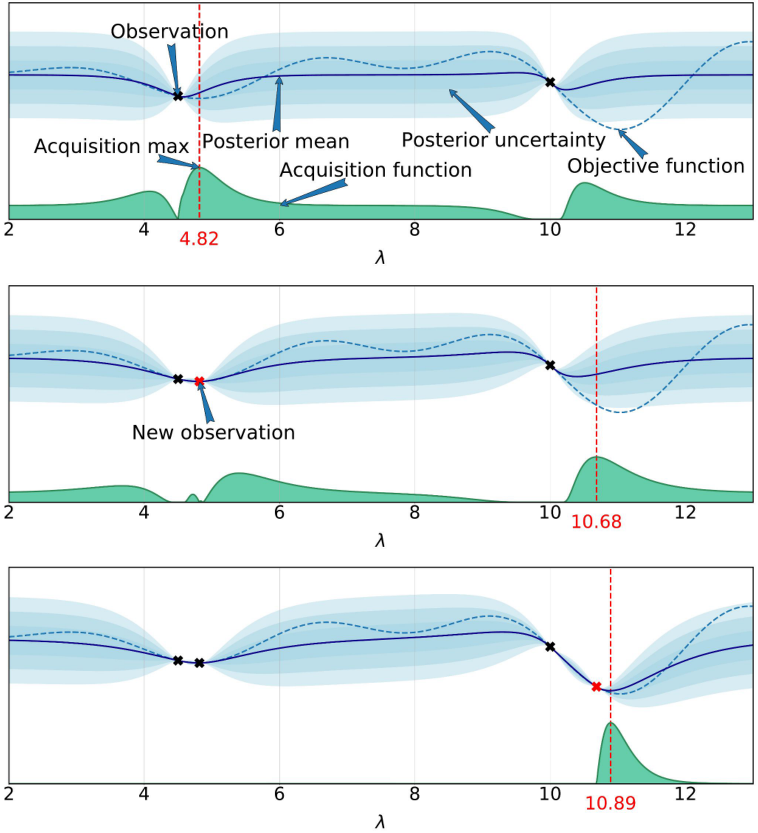
Bayesian Optimization in a Nutshell

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

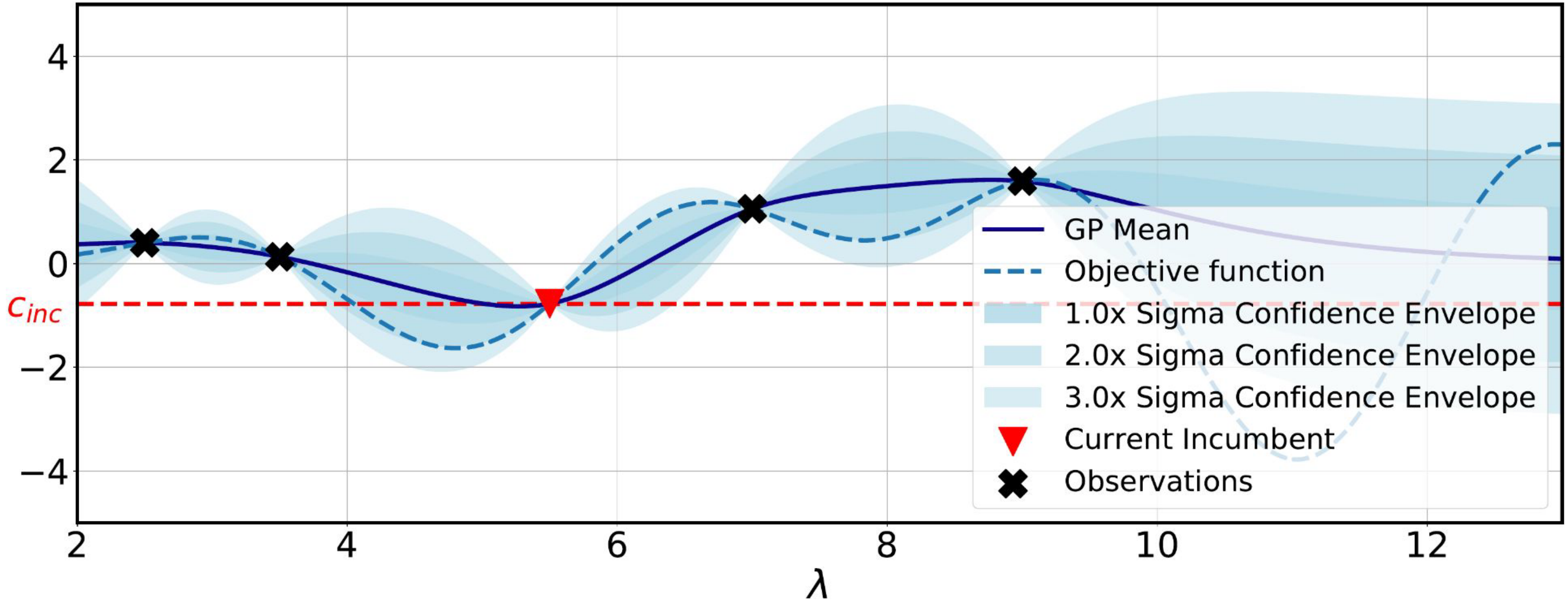
The acquisition function

- 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**

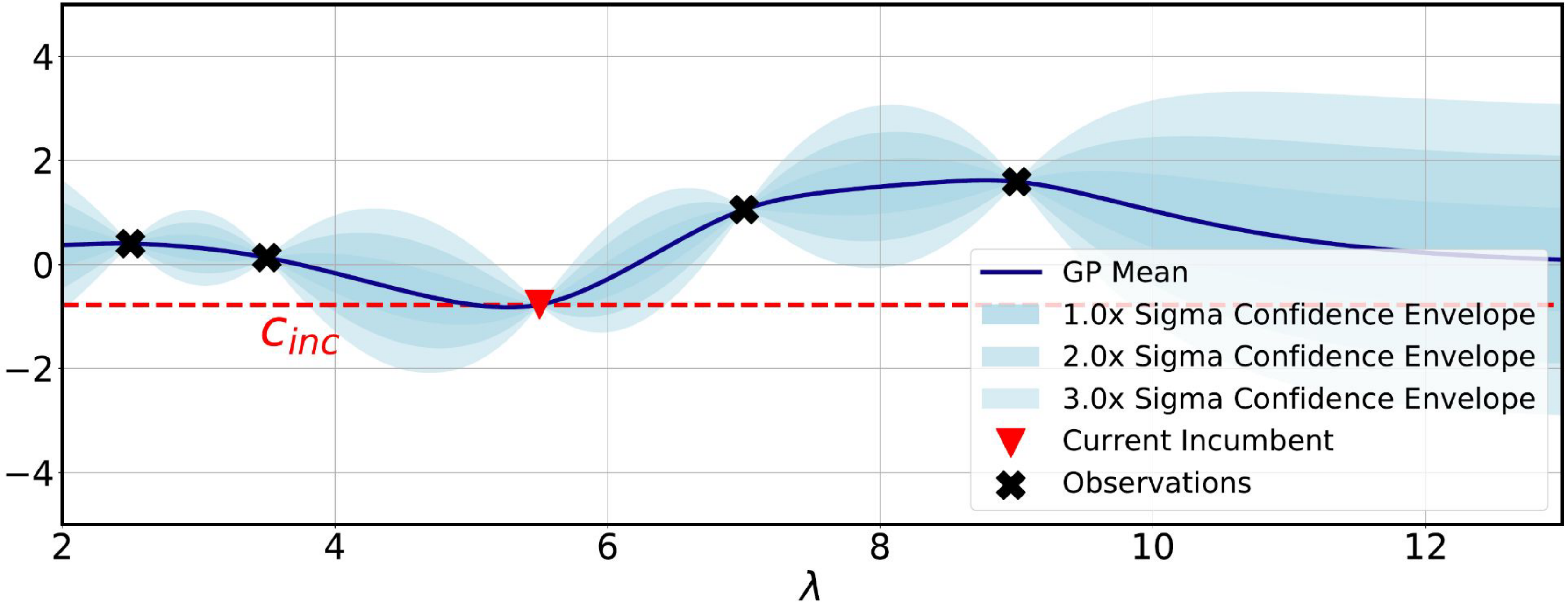
- Just picking the configuration with the lowest prediction would be too greedy
- It needs to consider the uncertainty of the surrogate model

Expected Improvement (EI)



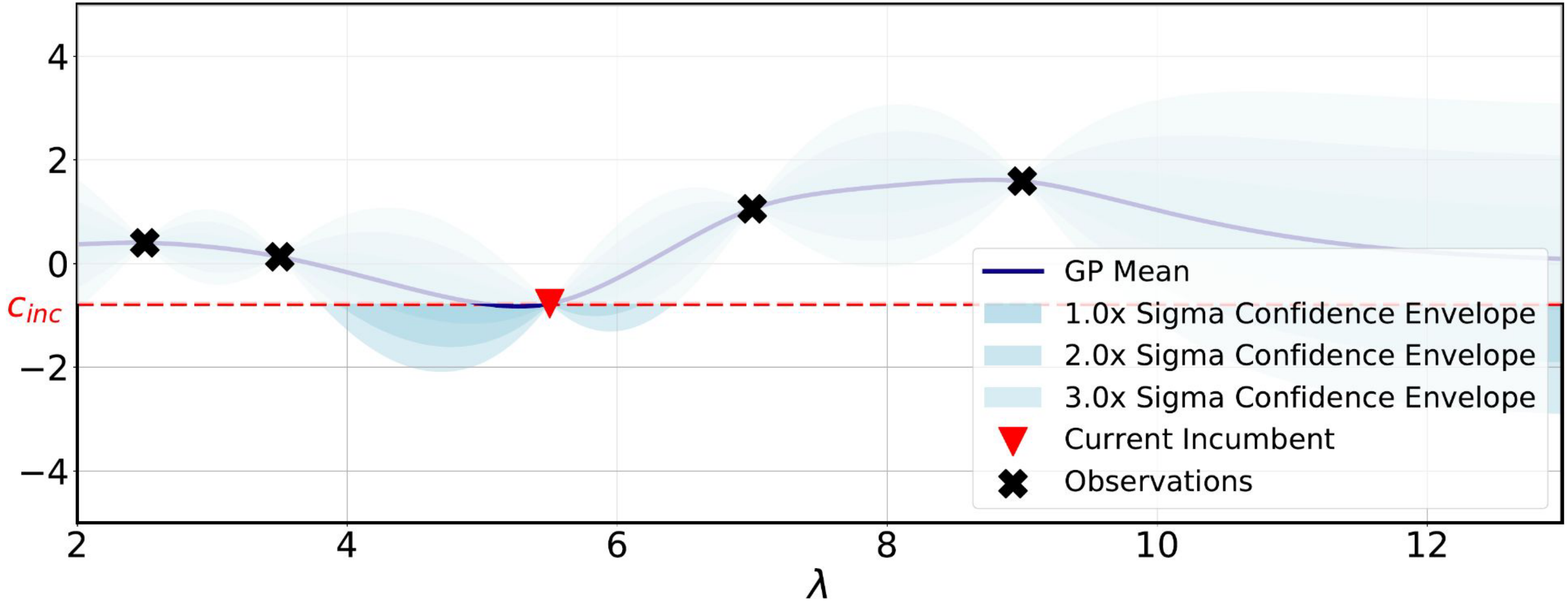
Given some observations and a fitted surrogate,

Expected Improvement (EI)



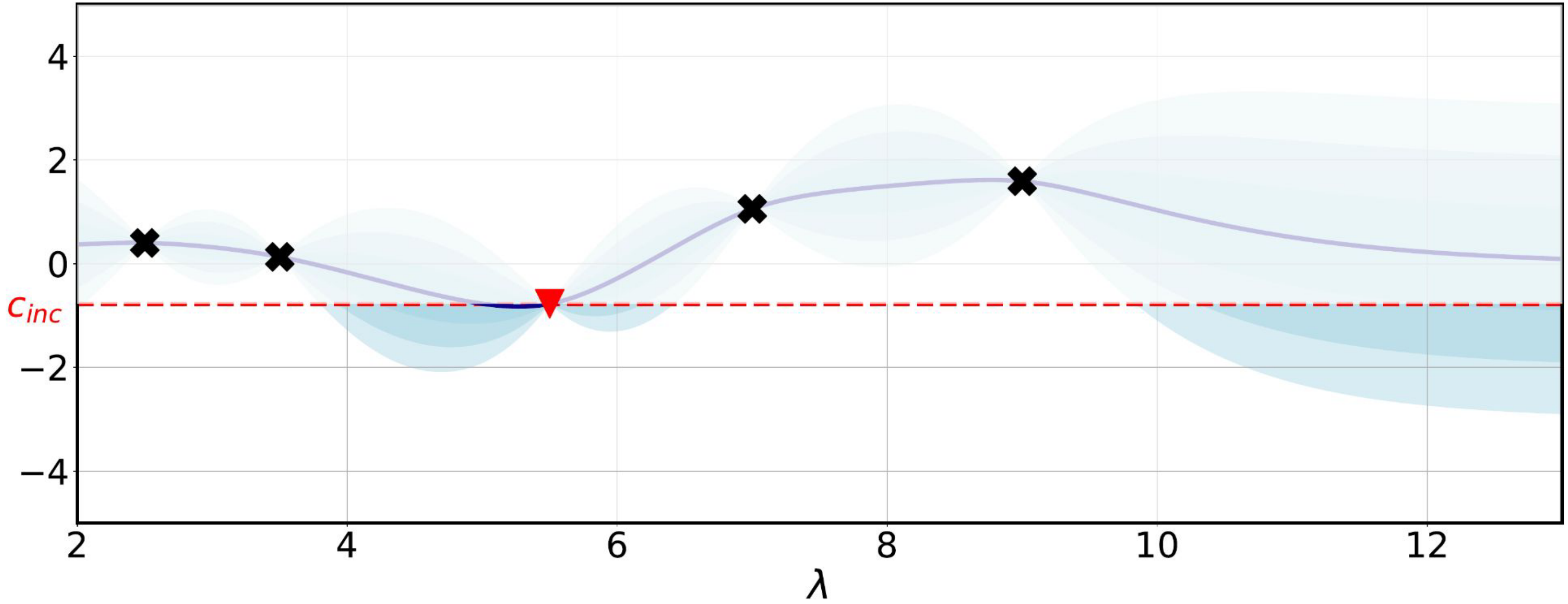
Given some observations and a fitted surrogate,

Expected Improvement (EI)



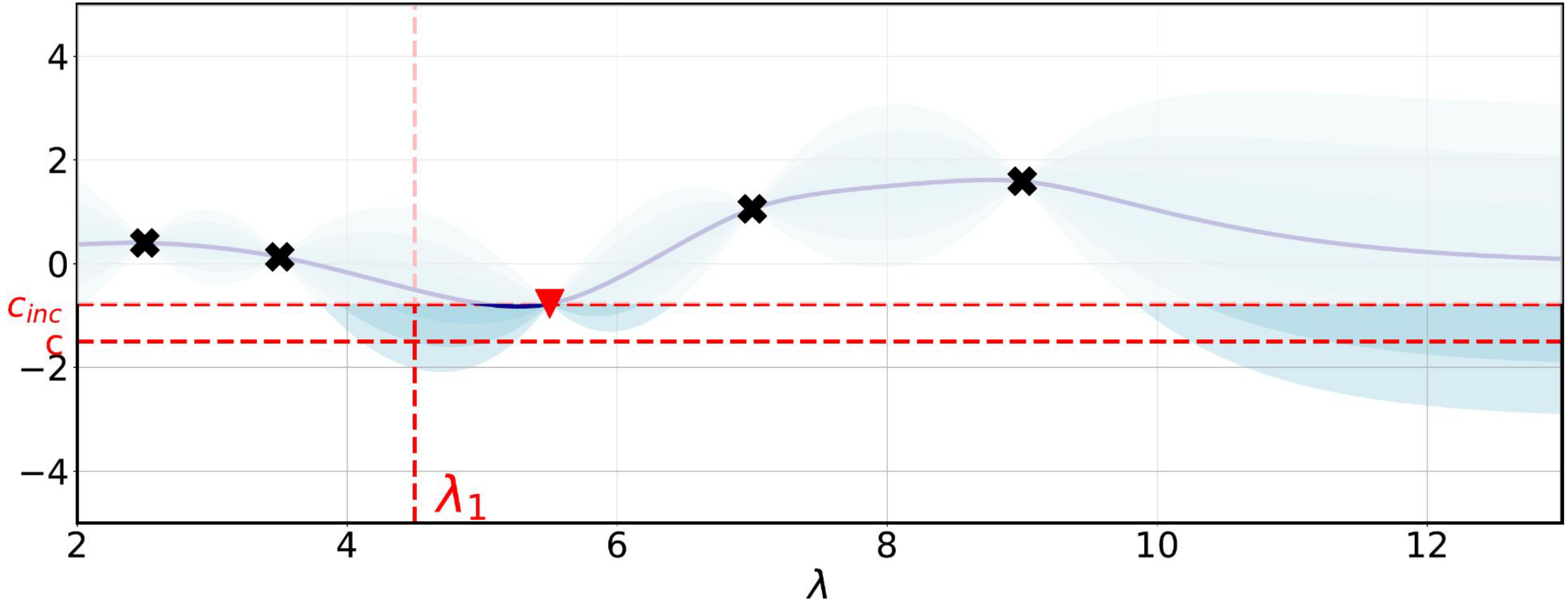
We care about *improving* over the c_{inc}.

Expected Improvement (EI)



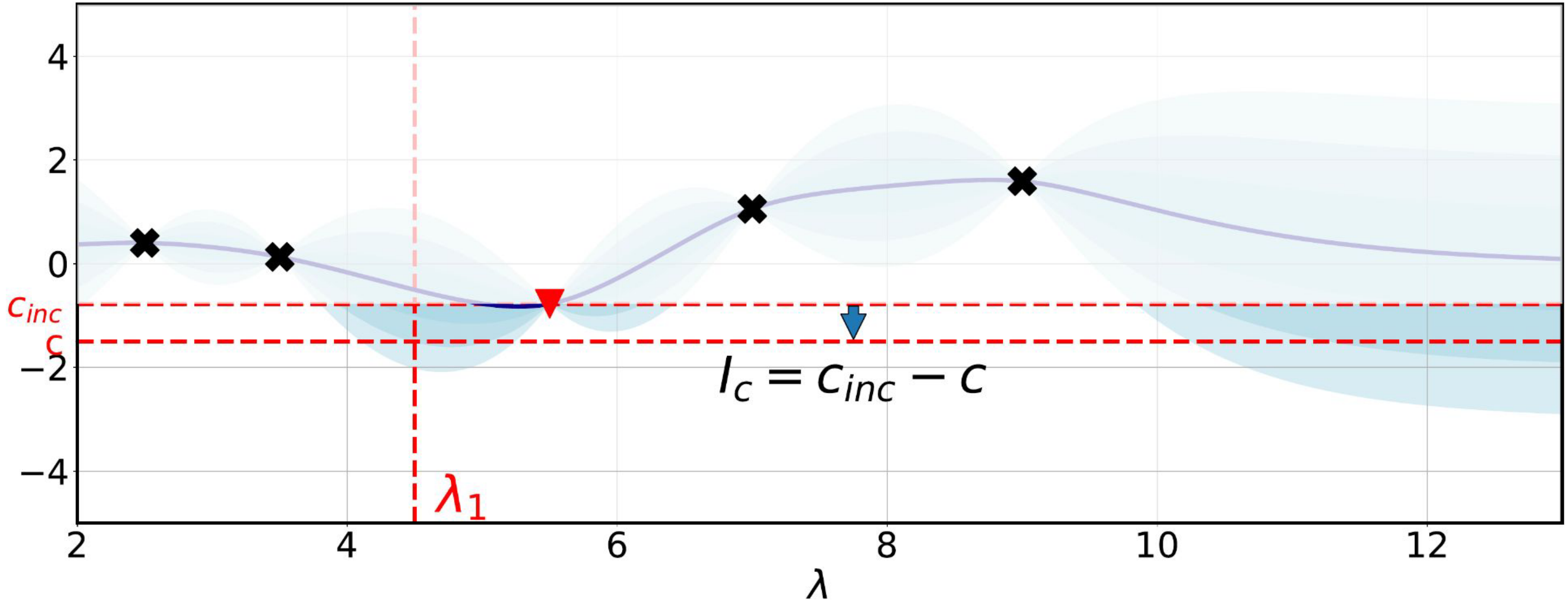
We care about *improving* over the c_{inc}.

Expected Improvement (EI)



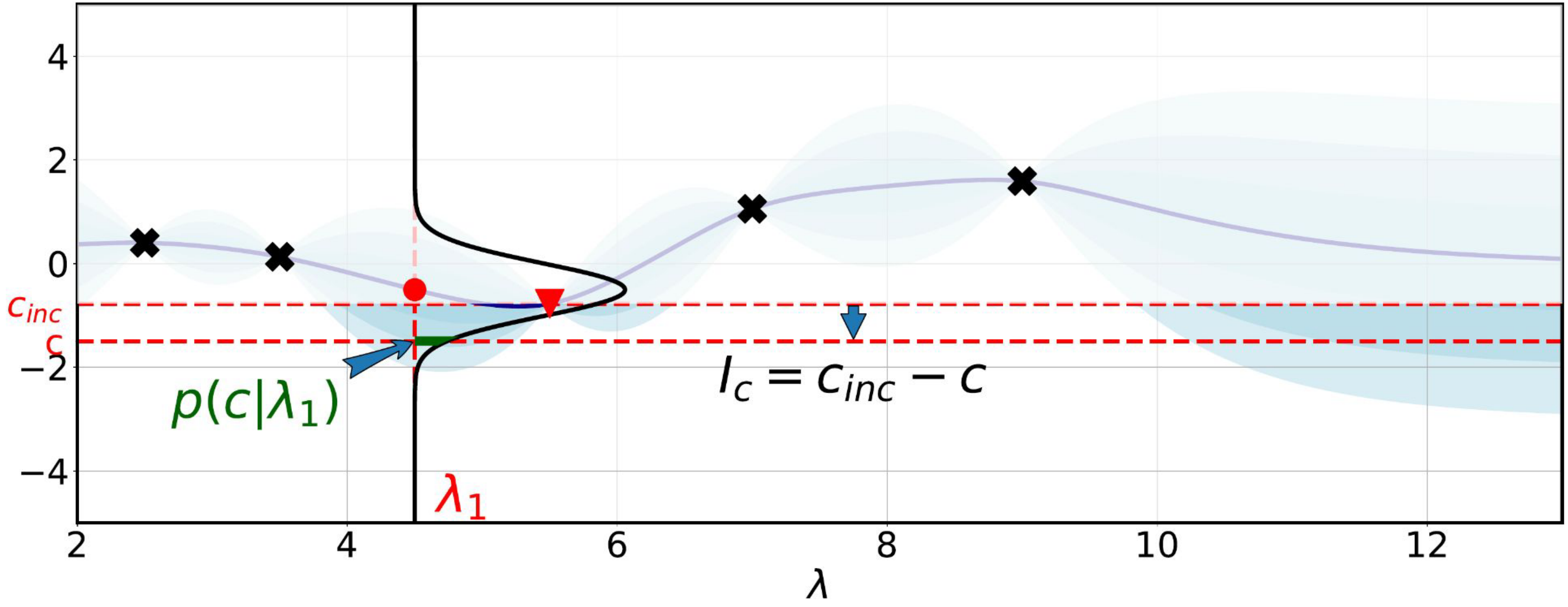
Let's look at a candidate configuration λ_1 and its hypothetical cost c.

Expected Improvement (EI)



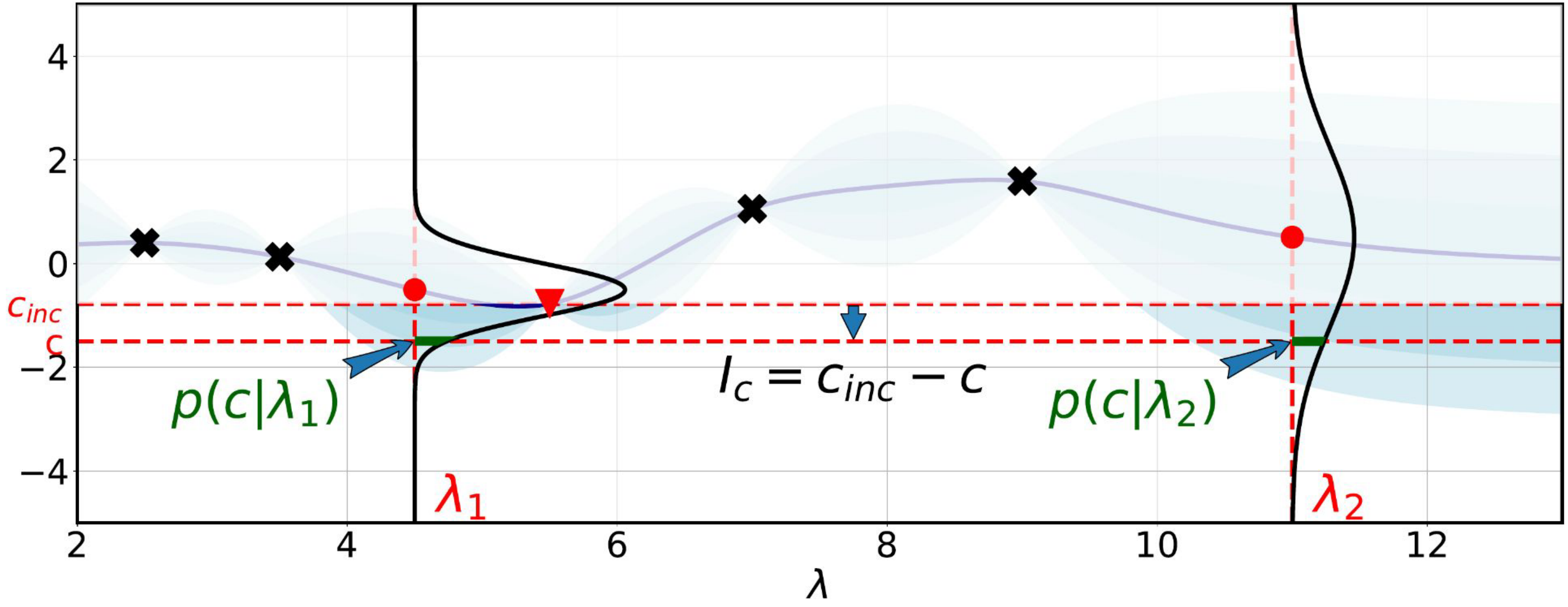
We can compute the improvement $I_c(\lambda_1)$. But how likely is it?

Expected Improvement (EI)



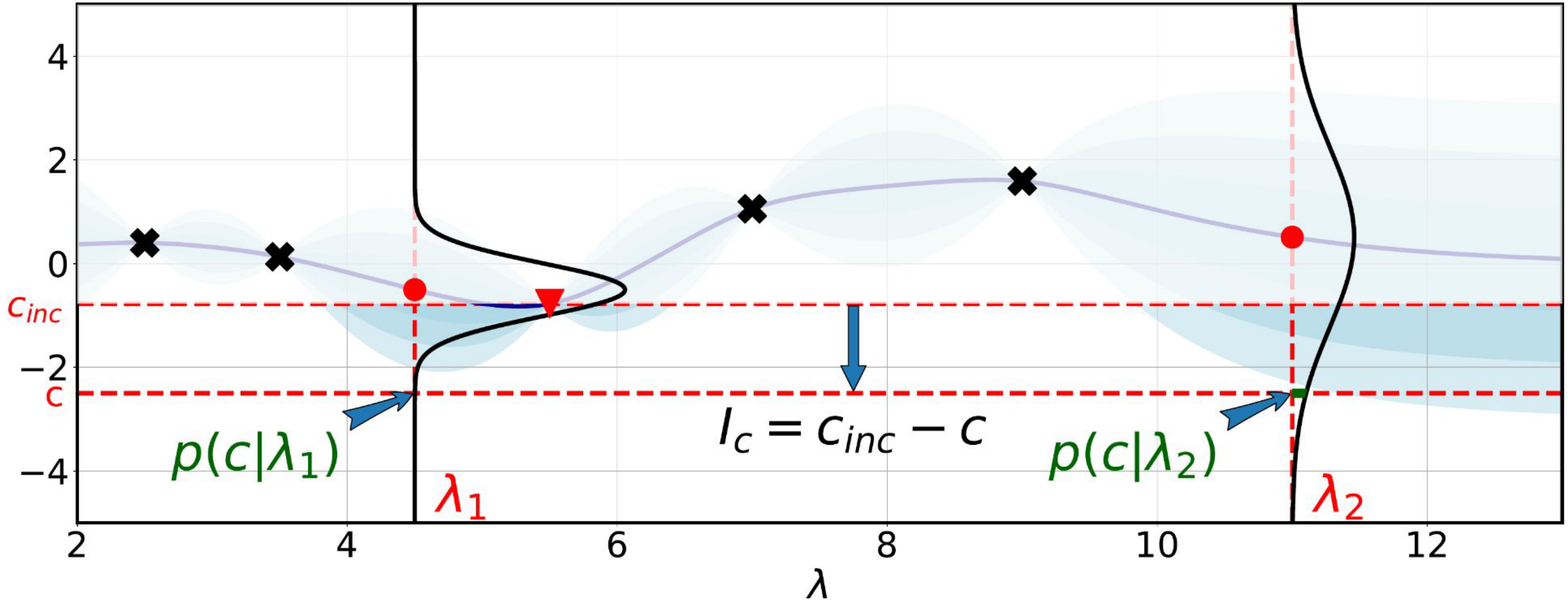
Knowing that $\hat{c}(\boldsymbol{\lambda}) = \mathcal{N}(\mu(\boldsymbol{\lambda}), \sigma^2(\boldsymbol{\lambda}))$, we can compute $p(c|\boldsymbol{\lambda})$

Expected Improvement (EI)



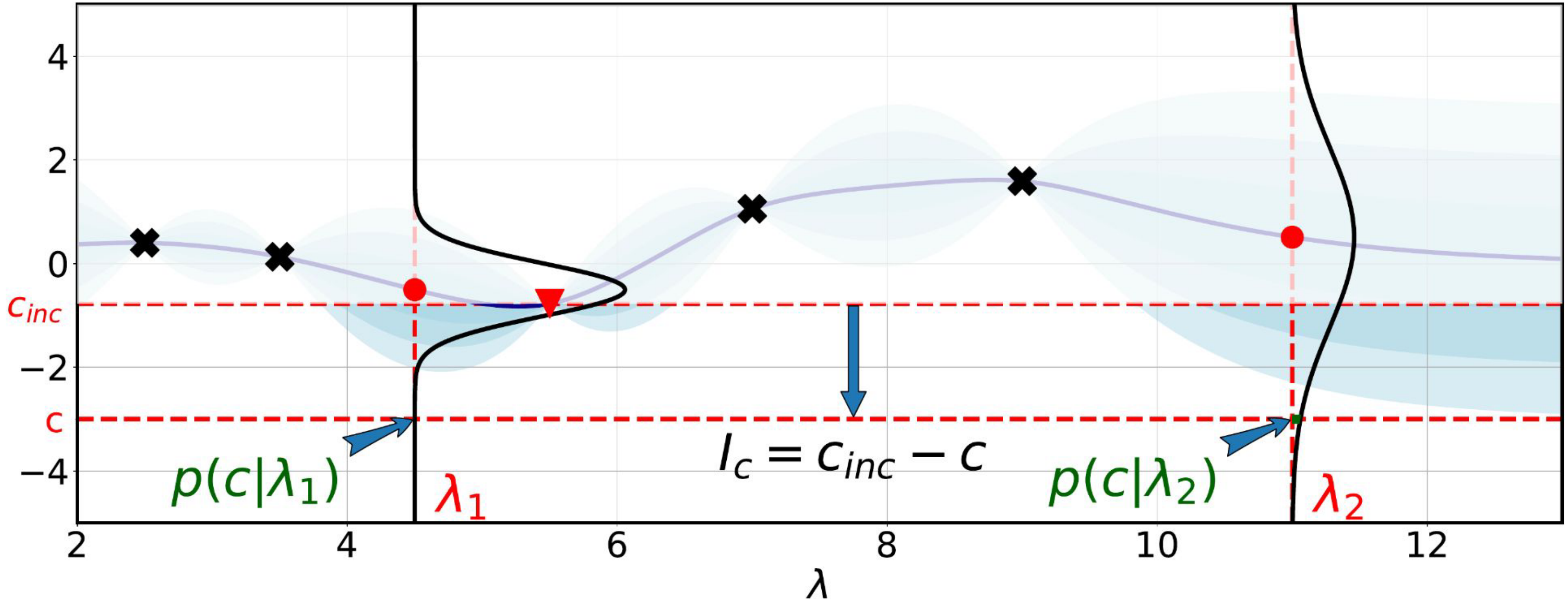
Comparing this for different configurations

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

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

Since posterior is Gaussian, EI can be computed in closed form.

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

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 [Hoffman et al. 2011; Cowen-Rivers 2022]

Questions?

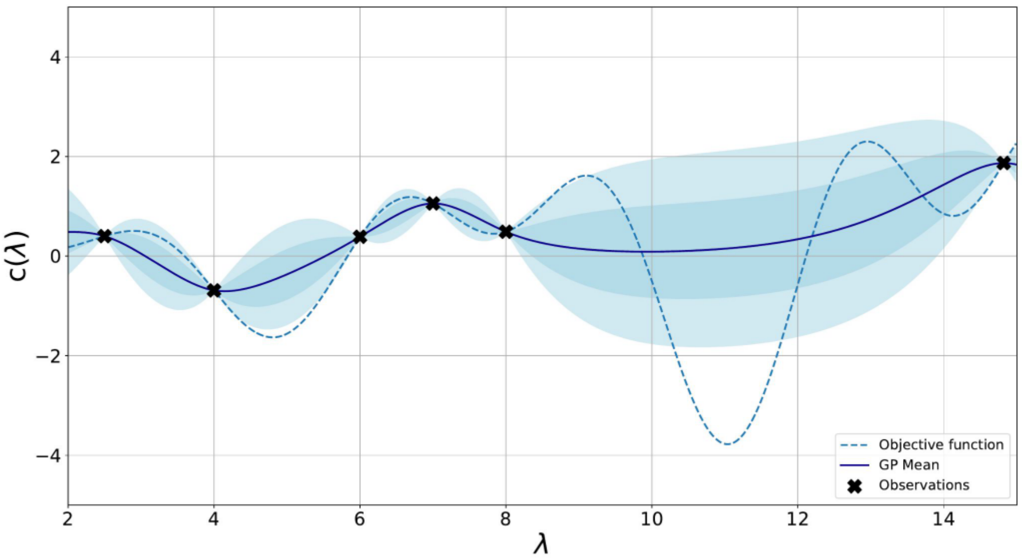
Main Ingredient II: The Surrogate Model

Required in all cases

- Regression model with uncertainty estimates
- Accurate predictions

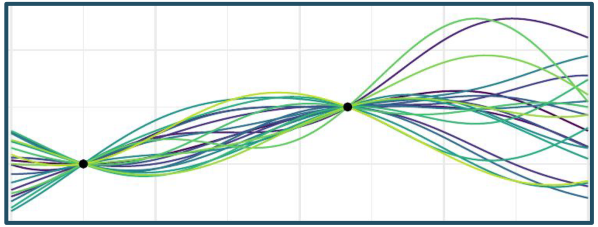
Depending on the application

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



Types of Surrogates Models

- Gaussian Processes



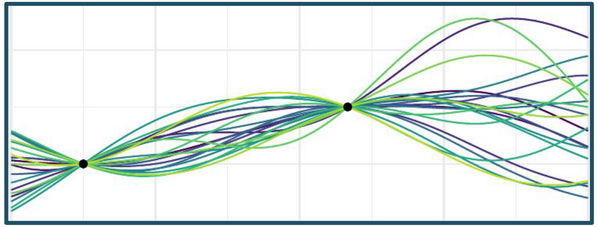
- Random Forests

- 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

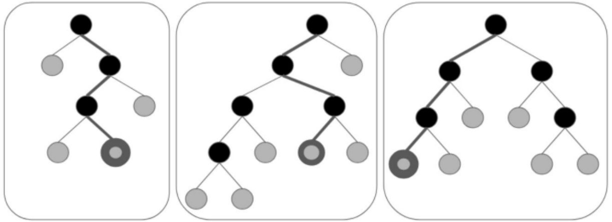
- Smooth uncertainty estimates
- Strong sample efficiency
- Expert knowledge can be encoded in the kernel
- Accurate predictions

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

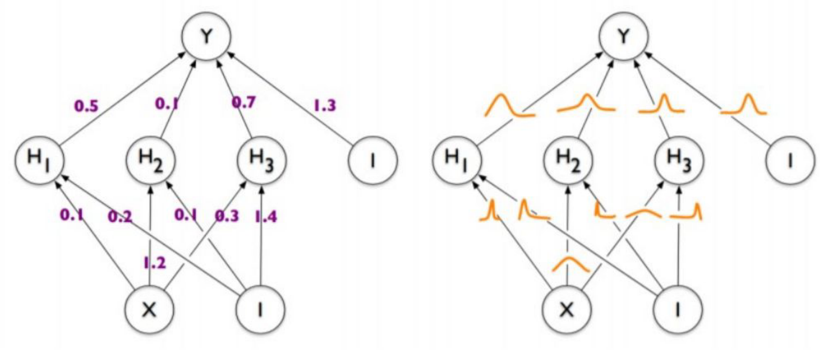


Image source: [Blundell et al. 2015]

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?

Kahoot Quiz II

How do we optimize it even more?

`>> Here’s my algorithm, data and design space, I have only limited time and I want more, what should I do?`

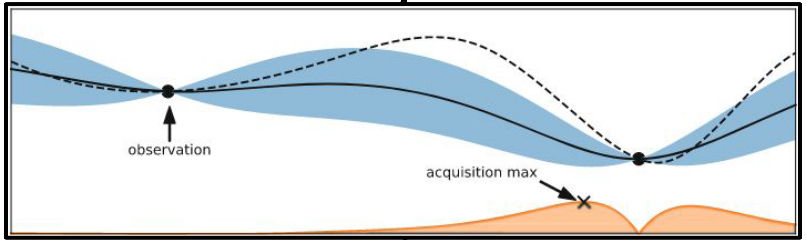
Bayesian Optimization: Extensions

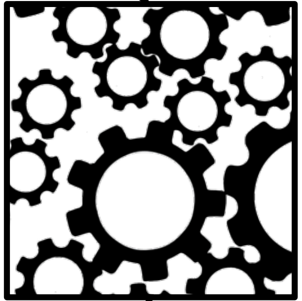
Different models to handle new design spaces

λ_n

Increase efficiency by using lower fidelities

Multi-objective





Consider constraints

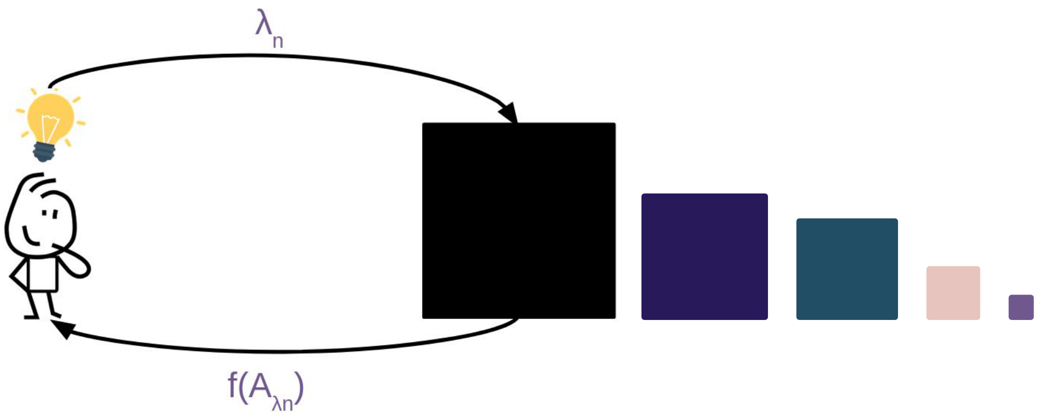
Leverage observations across experiments

$f(A_{\lambda n})$

Take user priors into account

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

Multi-Fidelity Bayesian Optimization



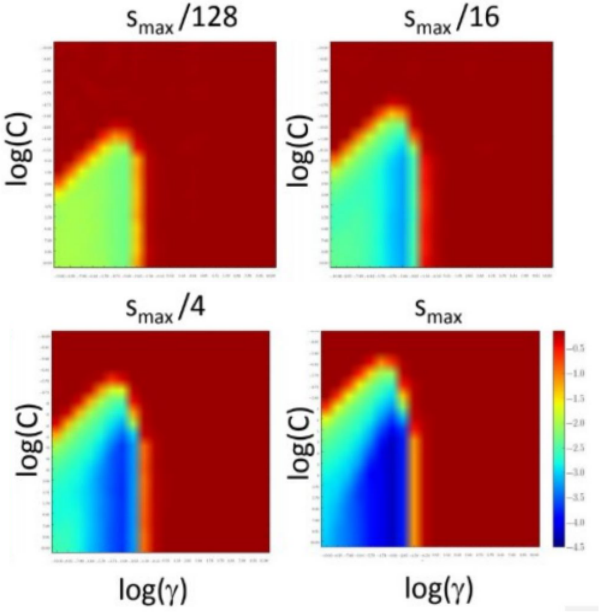
Often, the black-box

- is an **iterative** process,
- has **cheaper approximations** available,
- or can be evaluated **partially**

→ We can collect information about the actual objective value with less costly evaluations

Two Motivating Examples

Performance of a SVM on different subsets of MNIST



Learning curves of fully connected NNs on CIFAR-10

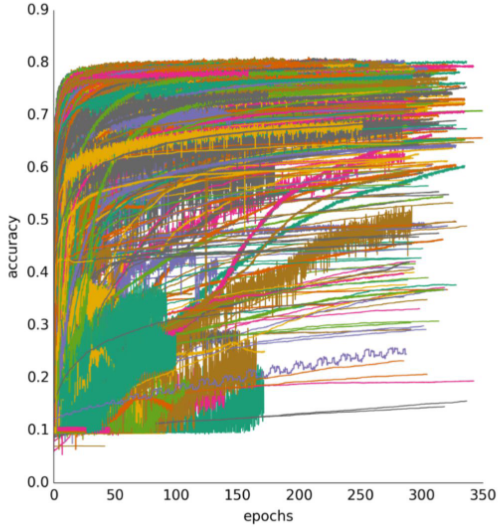
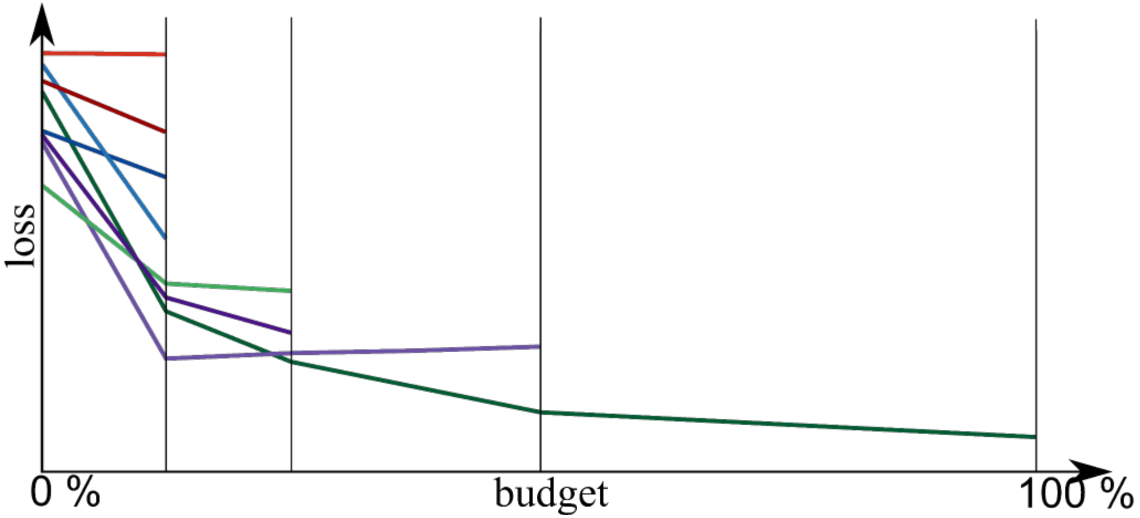


Image Source: [Domhan et al., 2015]

Successive Halving



[Jamieson and Talwalkar. 2016]

- A very simple algorithm:
  - Sample N configurations uniformly at random & evaluate them on the cheapest fidelity
  - Keep the best half (or third), move them to the next fidelity
  - Iterate until the most expensive fidelity (= original expensive black box)

Hyperband

What if the information on the lowest fidelity is not informative?

→ Run multiple iterations of SH, starting at different “lowest” fidelities.

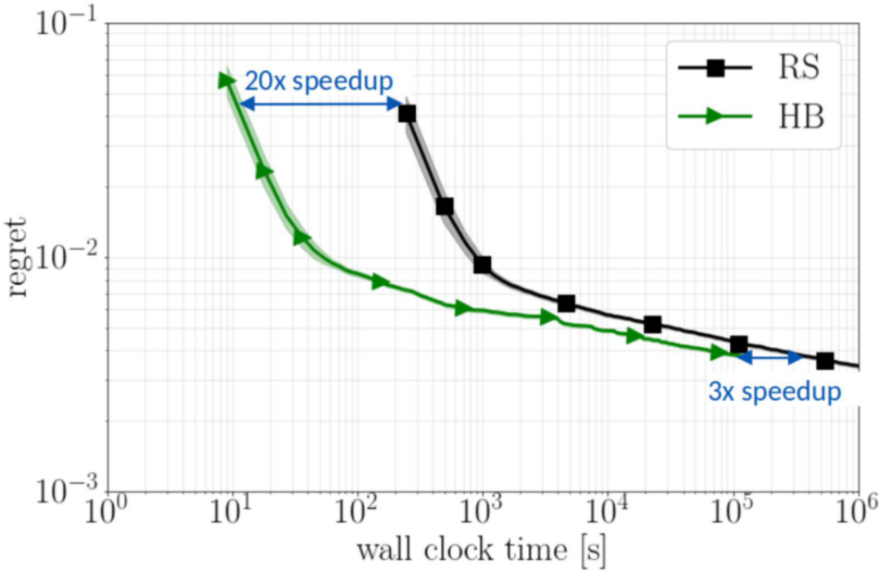


image credit: [Falkner et al. 2018]

BOHB: Hyperband X Bayesian Optimization

Idea: Use Bayesian Optimization to choose configurations [Falkner et al. 2018]

- BO to achieve strong performance
- HB to achieve good anytime performance

→ easy parallelization

→ with interleaved random sampling it keeps theoretical guarantees of HB

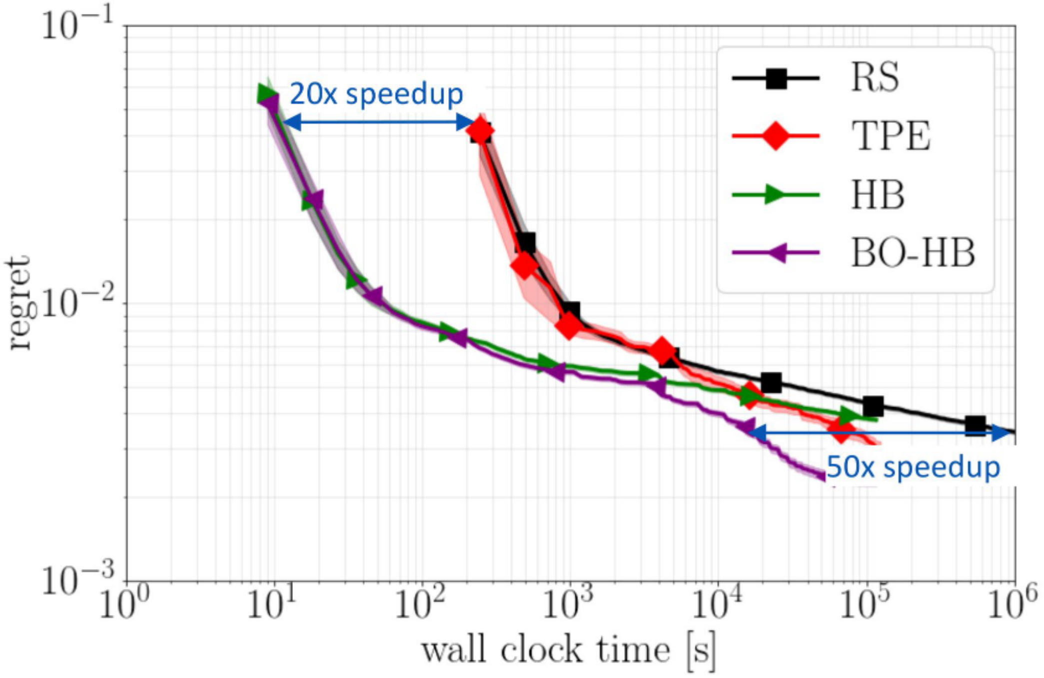


image credit: [Falkner et al. 2018]

Landscape of Multi-Fidelity HPO Methods

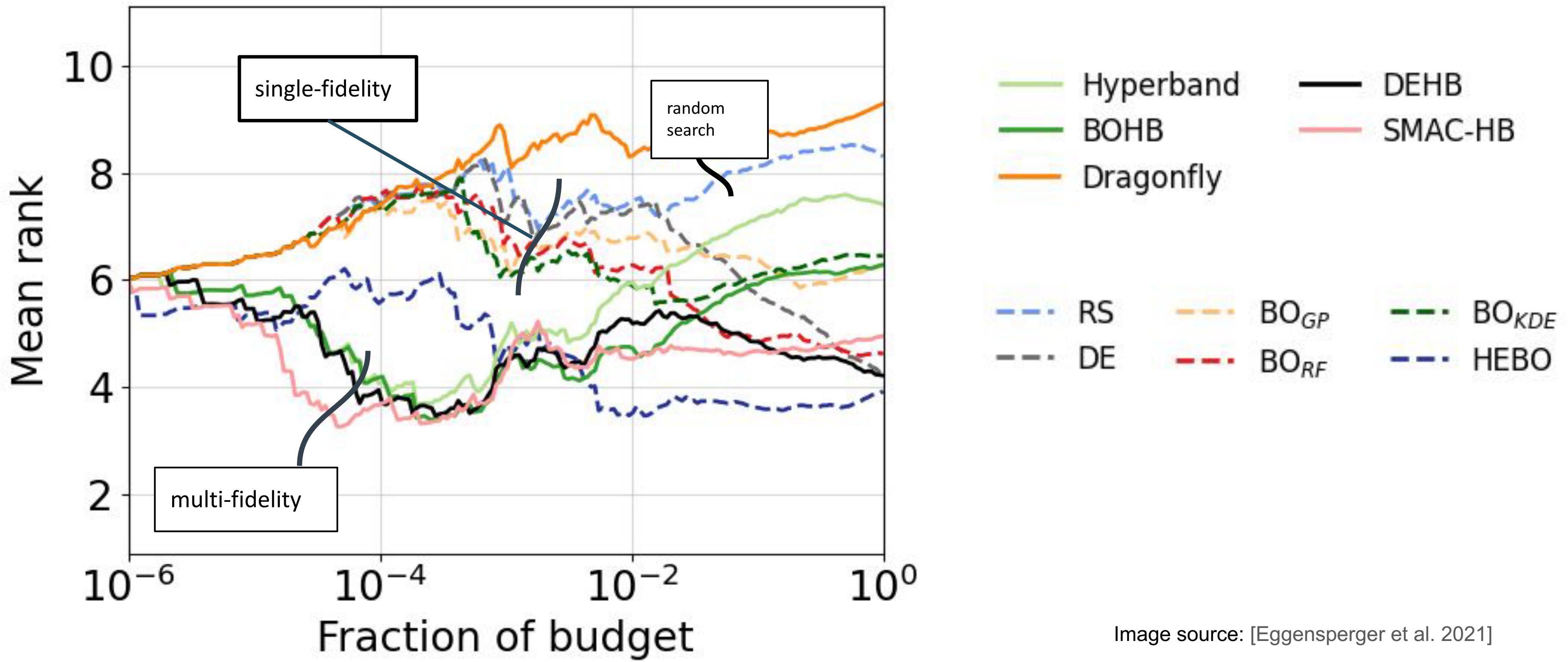
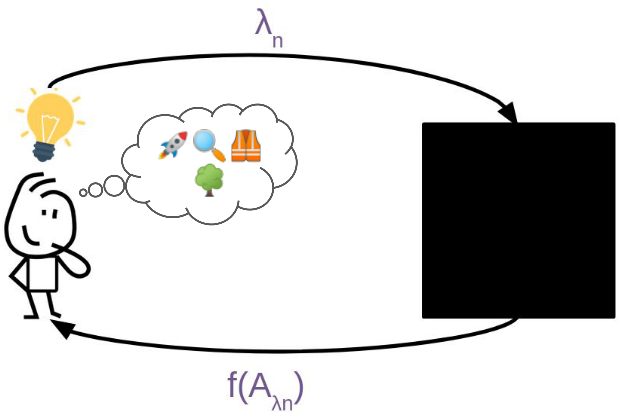


Image source: [Eggensperger et al. 2021]

Questions?

Multi-Objective Optimization



In practice, we often care about more than a single objective, e.g.

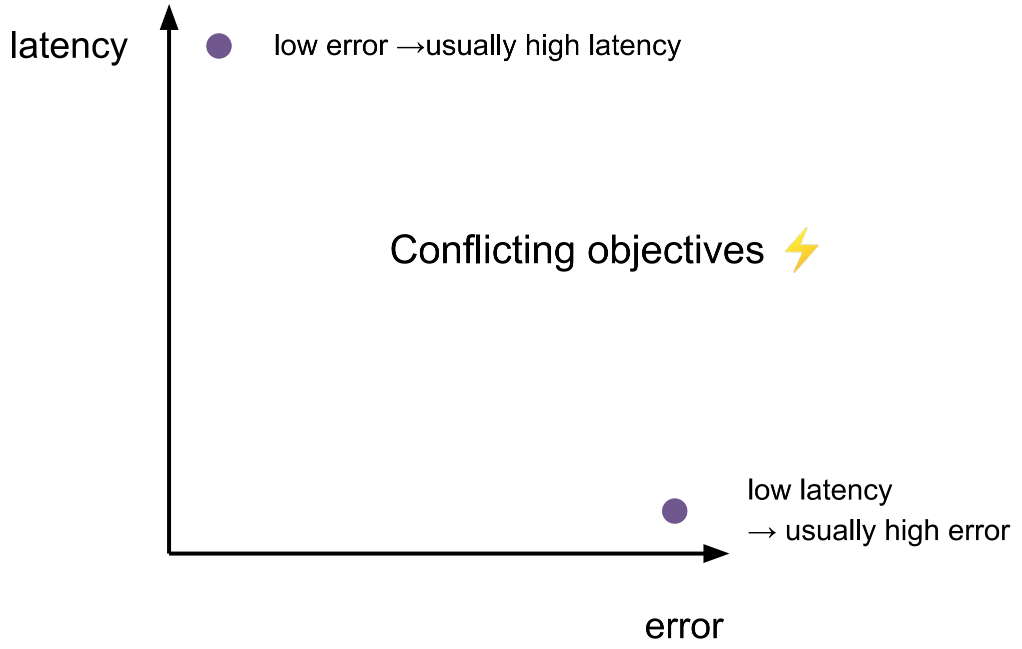
- error,
- inference time
- unfairness,
- energy consumption,
- model complexity,
- and many more

Practical Example

Goal: Find a Neural Network with **high accuracy** and **low latency**

→ Objectives are measured on different scales

→ There is no single optimal solution



Definition

A **multi-criteria optimization problem** is defined by

$$\min_{\boldsymbol{\lambda}\in\boldsymbol{\Lambda}} c(\boldsymbol{\lambda}) \Leftrightarrow \min_{\boldsymbol{\lambda}\in\boldsymbol{\Lambda}} \left(c_1(\boldsymbol{\lambda}), c_2(\boldsymbol{\lambda}), ..., c_m(\boldsymbol{\lambda})\right),$$

with $\boldsymbol{\Lambda} \subset \mathbb{R}^n$ and multi-criteria objective function $c : \boldsymbol{\Lambda} \to \mathbb{R}^m$, $m \geq 2$.

- **Goal:** minimize multiple target functions simultaneously.
- $(c_1(\boldsymbol{\lambda}), ..., c_m(\boldsymbol{\lambda}))^\top$ maps each candidate $\boldsymbol{\lambda}$ into the objecive space $\mathbb{R}^m$.
- Often no clear best solution, as objective are usually conflicting and we cannot totally order in $\mathbb{R}^m$.
- W.l.o.g. we always minimize.
- Alternative names: multi-criteria optimization, multi-objective optimization, Pareto optimization.

Pareto Sets and Pareto Optimality

Definition:
Given a multi-criteria optimization problem

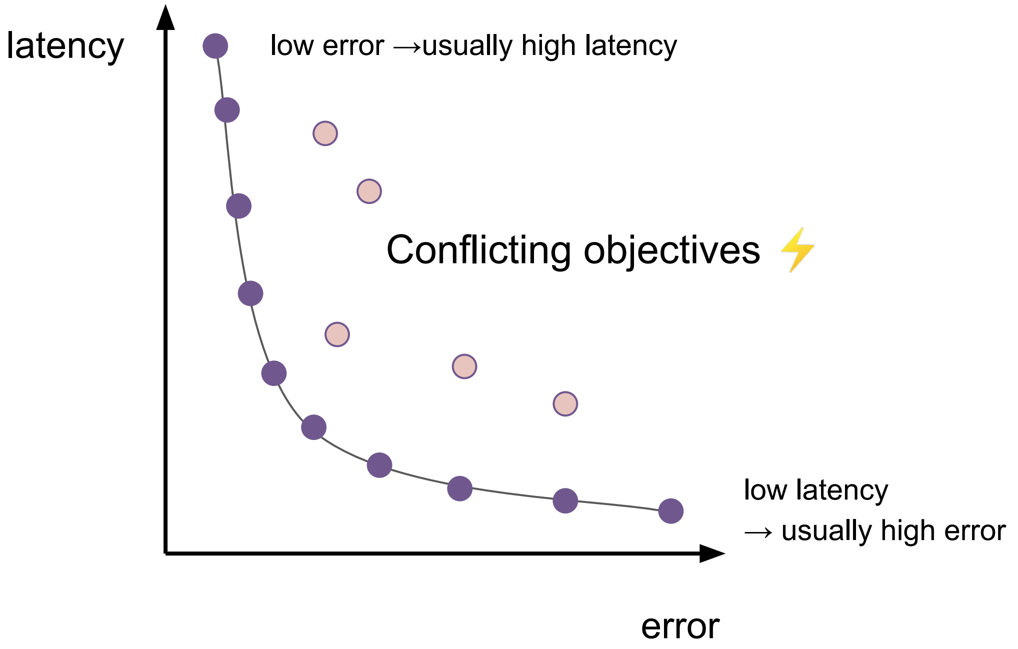
$$\min_{\boldsymbol{\lambda} \in \boldsymbol{\Lambda}} \left(c_1(\boldsymbol{\lambda}), ..., c_m(\boldsymbol{\lambda}) \right), \quad c_i : \boldsymbol{\Lambda} \to \mathbb{R}.$$

- A candidate $\boldsymbol{\lambda}^{(1)}$ **(Pareto-) dominates** $\boldsymbol{\lambda}^{(2)}$, if $c(\boldsymbol{\lambda}^{(1)}) \prec c(\boldsymbol{\lambda}^{(2)})$, i.e.
  1. $c_i(\boldsymbol{\lambda}^{(1)}) \leq c_i(\boldsymbol{\lambda}^{(2)})$ for all $i \in \{1, 2, ..., m\}$ and
  2. $c_j(\boldsymbol{\lambda}^{(1)}) < c_j(\boldsymbol{\lambda}^{(2)})$ for at least one $j \in \{1, 2, ..., m\}$
- A candidate $\boldsymbol{\lambda}^*$ that is not dominated by any other candidate is called **Pareto optimal**.
- The set of all Pareto optimal candidates is called **Pareto set**
  $\mathcal{P} := \{\boldsymbol{\lambda} \in \boldsymbol{\Lambda} | \nexists\, \tilde{\boldsymbol{\lambda}}$ with $c(\tilde{\boldsymbol{\lambda}}) \prec c(\boldsymbol{\lambda})\}$
- $\mathcal{F} = c(\mathcal{P}) = \{c(\boldsymbol{\lambda}) | \boldsymbol{\lambda} \in \mathcal{P}\}$ is called **Pareto front**.

Practical Example

~~Goal: Find a Neural Network with high accuracy and low latency~~

Goal: Find the Pareto Set of Neural Networks that balance accuracy and latency.



Multi-Objective Bayesian Optimization

BO loop

Require: Search space $\mathbf{\Lambda}$, 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: $\boldsymbol{\lambda}^{(t)} \in \arg\max_{\boldsymbol{\lambda} \in \mathbf{\Lambda}} u(\boldsymbol{\lambda}; \mathcal{D}^{(t-1)}, \hat{c}^{(t)})$

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

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

Two basic approaches:

- Simplify the problem by **scalarizing cost functions**
- Define a **new acquisition** function for multiple costs

Scalarization

Idea: Aggregate all cost functions

$$\min_{\boldsymbol{\lambda} \in \boldsymbol{\Lambda}} \sum_{i=1}^{m} w_i c_i(\boldsymbol{\lambda}) \qquad \text{with} \quad w_i \geq 0$$

- **Obvious problem:** How to choose $w_1, \ldots, w_m$?
  - Expert knowledge?
  - Systematic variation?
  - Random variation?
- If expert knowledge is not available a-priori, we need to ensure that different trade-offs between cost functions are explored.
- Simplifies multi-criteria optimization problem to single-objective
  $\longrightarrow$ Bayesian optimization can be used without adaption of the general algorithm.

Scalarization: ParEGO

Scalarize the cost functions using the augmented Tchebycheff norm / achievement function

$$c = \max_{i=1,\ldots,m} (w_i c_i(\boldsymbol{\lambda})) + \rho \sum_{i=1}^{m} w_i c_i(\boldsymbol{\lambda}),$$

- The weights $w \in W$ are drawn from

$$W = \left\{ w = (w_1, \ldots, w_m) | \sum_{i=1}^{m} w_i = 1, w_i = \frac{l}{s}, l \in 0, \ldots, s \right\},$$

  with $|W| = \binom{s+m-1}{k-1} 1.$
- New weights are drawn in every BO iteration.
- ρ is a small parameter suggested to be set to 0.05.
- s selects the number of different weights to draw from.

[Knowles et al. 2006]

ParEGO

ParEGO loop

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

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**

- Sample w from $\{w = (w_1, \ldots, w_m) | \sum_{i=1}^{m} w_i = 1, w_i = \frac{l}{s} \wedge, l \in 0, \ldots, s\}$;
- Compute scalarization $c^{(t)} = \max_{i=1,\ldots,m} (w_i c_i(\boldsymbol{\lambda})) + \rho \sum_{i=1}^{m} w_i c_i(\boldsymbol{\lambda})$;
- 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: $\mathcal{D}^{(t)} \leftarrow \mathcal{D}^{(t-1)} \cup \{\langle \boldsymbol{\lambda}^{(t)}, c(\boldsymbol{\lambda}^{(t)}) \rangle\}$

Questions?

Kahoot Quiz III

Demo: SMAC

`>> Here's my algorithm, data and design space, I have only limited time and want to use Bayesian Optimization, what should I do?`

Recommendations

- Literature
  - Bayesian Optimization
  - A Tutorial on Bayesian Optimization
- Bayesian Optimization Tools
  - SMAC3 (Colab Demo)
  - Optuna
  - Syne-Tune
  - BoTorch

Thanks.
See you tomorrow!

👋

References

- N. Awad, N. Mallik, F. Hutter (2021). DEHB: Evolutionary Hyperband for Scalable, Robust and Efficient Hyperparameter Optimization. Proceedings of International Joint Conference on Artificial Intelligence (IJCAI'21)
- A. Bull (2012). Convergence Rates of Efficient Global Optimization Algorithms. Journal of Machine Learning (JMLR'18).
- J. Bergstra, and Y. Bengio (2012). Random Search for Hyper-Parameter Optimization. Journal of Machine Learning Research
- C. Blundell, J. Cornebise, K. Kavukcuoglu, D. Wierstra (2015). Weight Uncertainty in Neural Network. Proceedings of the International Conference on Machine Learning (ICML'15)
- A. Cowen-Rivers, W. Lyu, R. Tutunov, Z. Wang, A. Grosnit, R. Griffiths, A. Maraval, H. Jianye, J. Wang, J. Peters, and H. Ammar (2022). HEBO: Pushing the limits of sample-efficient hyperparameter optimisation. Journal of Artificial Intelligence Research.
- T. Domhan, J. T. Springenberg, F. Hutter (2015). Speeding up Automatic Hyperparameter Optimization of Deep Neural Networks by Extrapolation of Learning Curves. Proceedings of International Joint Conference of Artificial Intelligence (IJCAI'15)
- K. Eggensperger, P. Müller, N. Mallik, M. Feurer, R. Sass, A. Klein, N. Awad, M. Lindauer, and F. Hutter (2021). HPOBench: A Collection of Reproducible Multi-Fidelity Benchmark Problems for HPO. Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS'2021)
- K. Jamieson, A. Talwalkar (2016). Non-stochastic Best Arm Identification and Hyperparameter Optimization. In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS'16)
- J. Mockus (1975). On Bayesian methods for Seeking the Extremum. IFIP Technical Conference on Optimization Techniques.
- N. de Freitas, A. Smola, M. Zoghi (2012). Exponential Regret Bounds for Gaussian Process Bandits with Deterministic Observations. Proceedings of the International Conference of Machine Learning (ICML'12)
- S. Falkner, A. Klein, F. Hutter (2018). BOHB: Robust and Efficient Hyperparameter Optimization at Scale. Proceedings of the International Conference on Machine Learning (ICML'18)
- R. Garnett (2022). Bayesian Optimization. Cambridge University Press.
- M. Hoffman, E. Brochu, and N. de Freitas (2011). Portfolio Allocation for Bayesian Optimization. Proceedings of the 27th conference on Uncertainty in Artificial Intelligence (UAI'11).
- K. Kawaguchi, L. Pack Kaelbling, T. Lozano-Pérez (2015). Bayesian Optimization with Exponential Convergence. Advances in Neural Information Processing Systems (NeurIPS'15)
- J. Knowles (2006). ParEGO: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. IEEE Transactions on Evolutionary Computation.
- Y. Li, T. Rudner, A. Wilson (2023). A study of Bayesian Neural Network Surrogates for Bayesian Optimization. arXiv:2305.20028v1 [cs.LG]
- L. Li, K. Jamieson, A. Rostamizadeh, E. Gonina, M. Hardt, B. Recht, A. Talwalkar (2020). A System for Massively Parallel Hyperparameter Tuning. Proceeding of Machine Learning and Systems (MLSys'20)
- N. Srinivas, A. Krause, S. Kakade, and M. Seeger (2010). Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design. Proceedings of the International Conference on Machine Learning (ICML'10).