

Elements of Machine Learning & Data Science

Winter semester 2025/26

Lecture 23 – Performance Optimization I

27.01.2026

Prof. Bastian Leibe

slides by Prof. Holger Hoos and Prof. Wil van der Aalst

Empirical Analysis and Performance Evaluation Topics

- 15. Data Quality and Preprocessing
- 16. Responsible Data Science
- 17. Evaluation
- 18. Performance Optimization**

Key Questions

- How good **could** an ML model be?
 - Are we *using the best possible ML method / model?*
 - Have we *configured and trained it in the best possible way?*
 - Can we *further improve* performance?

Empirical Analysis and Performance Evaluation Topics

- 15. Data Quality and Preprocessing
- 16. Responsible Data Science
- 17. Evaluation
- 18. Performance Optimization**

Key Questions

- How good could an **ML model** be?
- How can we *ensure* we are using a good *ML method / model*?
- How can we *configure and train* it for optimized performance?
- How can we *further improve* performance?

Performance Optimization Topics

1. Hyperparameter Optimization
2. Model Selection

What Is Hyperparameter Optimization?

Background:

- Learning algorithms come with a set of **hyperparameters (HPs)** which govern their behavior.
 - E.g., Neural Networks: learning rate, regularization factor, ...
 - E.g., Random Forests: number of trees, tree depth, ...
- HP settings have a high impact on the performance of the learning algorithm.

Goal of HPO:

- Find a hyperparameter configuration which yields optimal performance of a learning algorithm, i.e., which minimizes generalization error (= expected loss)
- Cross-validation: we estimate the expected loss via a mean over a validation set.
- Different types of HPs: numerical (real- or integer-valued), categorical...

Grid Search

- Brute-force approach (full factorial design)
- Each HP configuration λ is mapped to the performance (loss) of the learning algorithm
 - HP response function (response surface) Ψ that we wish to minimize
- ⇒ Strategy: sample different trial points, evaluate them and select the one that minimizes $\Psi(\lambda)$
- This is often already implemented in ML packages
 - E.g., default option for SVM optimization

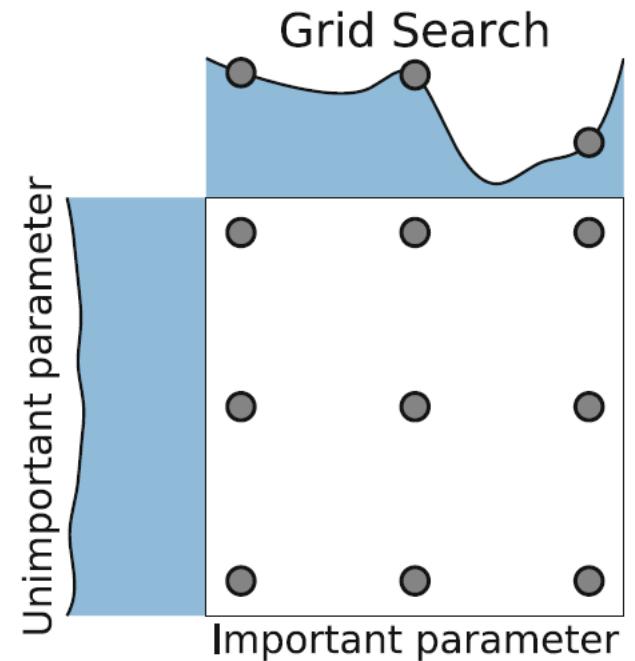


Figure taken from [Hutter et al. 2019 (eds),
Automated Machine Learning]

Hyperparameter Optimization

- **What are fundamental problems when using a grid (and manual) search for HPO?**
- *Let's again collect your ideas here...*
- *What are difficulties in the practical implementation?*
- *What are difficulties when we want to compare to or replicate results by different authors?*

Grid Search Challenges

Challenges:

- We select a set or range of values for each HP, and we consider all possible combinations between them as trial points (Cartesian product over sets)
⇒ Curse of dimensionality
- Manual search:
 - Reliance on expert knowledge, bias,
⇒ Difficult to reproduce results
- Increase of grid search granularity
⇒ Increase in complexity

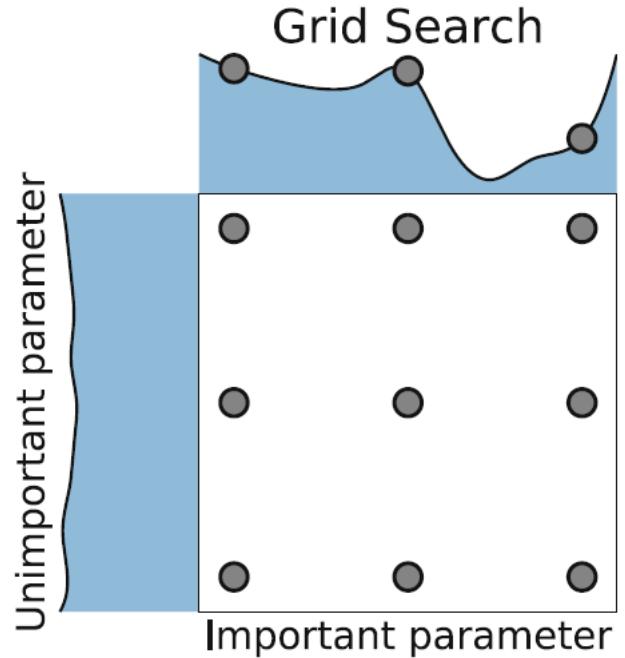
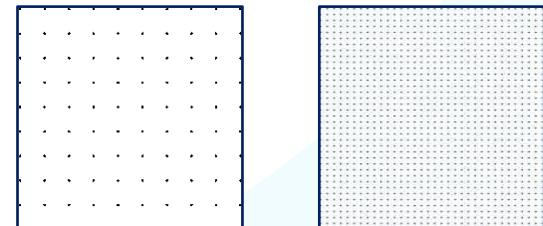
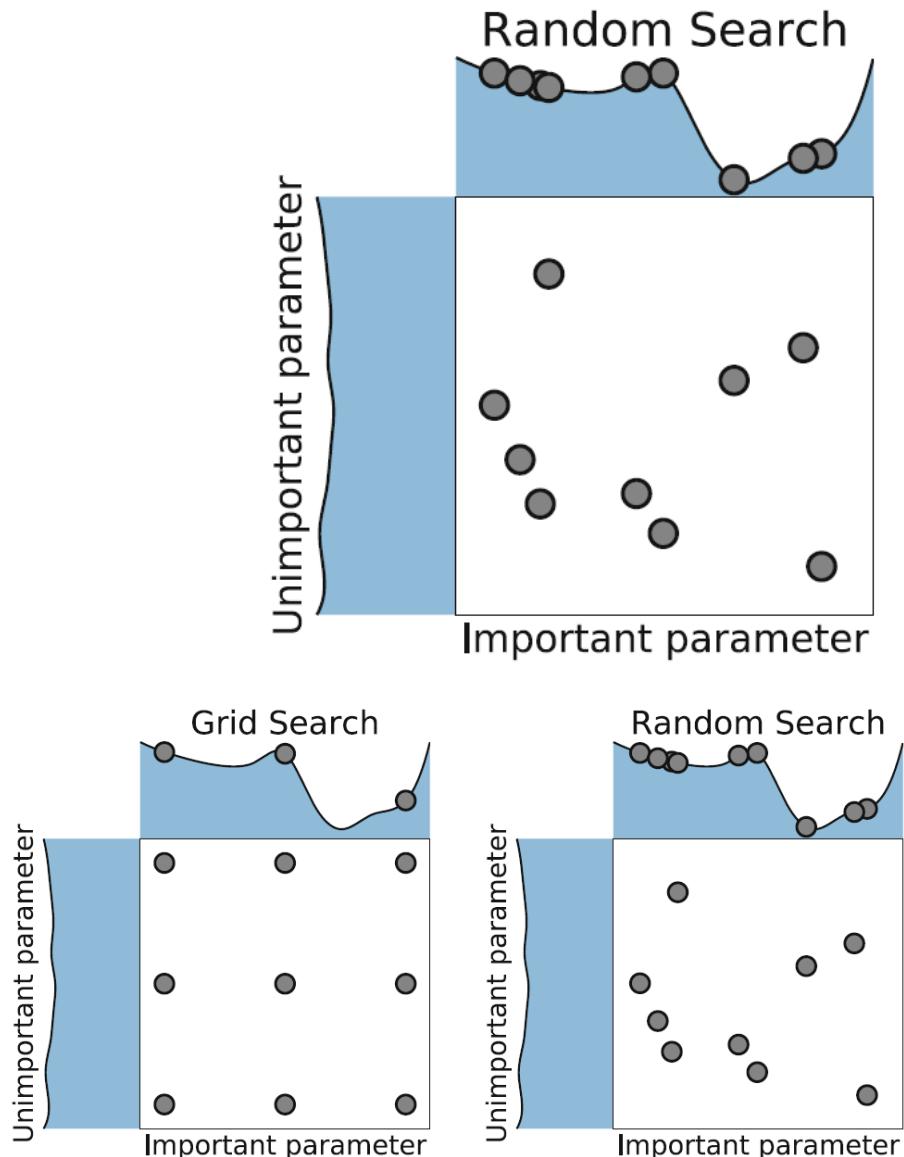


Figure taken from [Hutter et al. 2019 (eds),
Automated Machine Learning]



Alternative: Random Search

- We draw independent trial points uniformly at random from given sets or ranges of possible HP values (that would be spanned by a grid)
- Advantages
 - Easy to implement/parallelize/add and remove trials
- Disadvantages / Challenges
 - **Low effective dimensionality**: not all dimensions are equally important (i.e., sensitive to change)
⇒ RS efficient in high dimensions

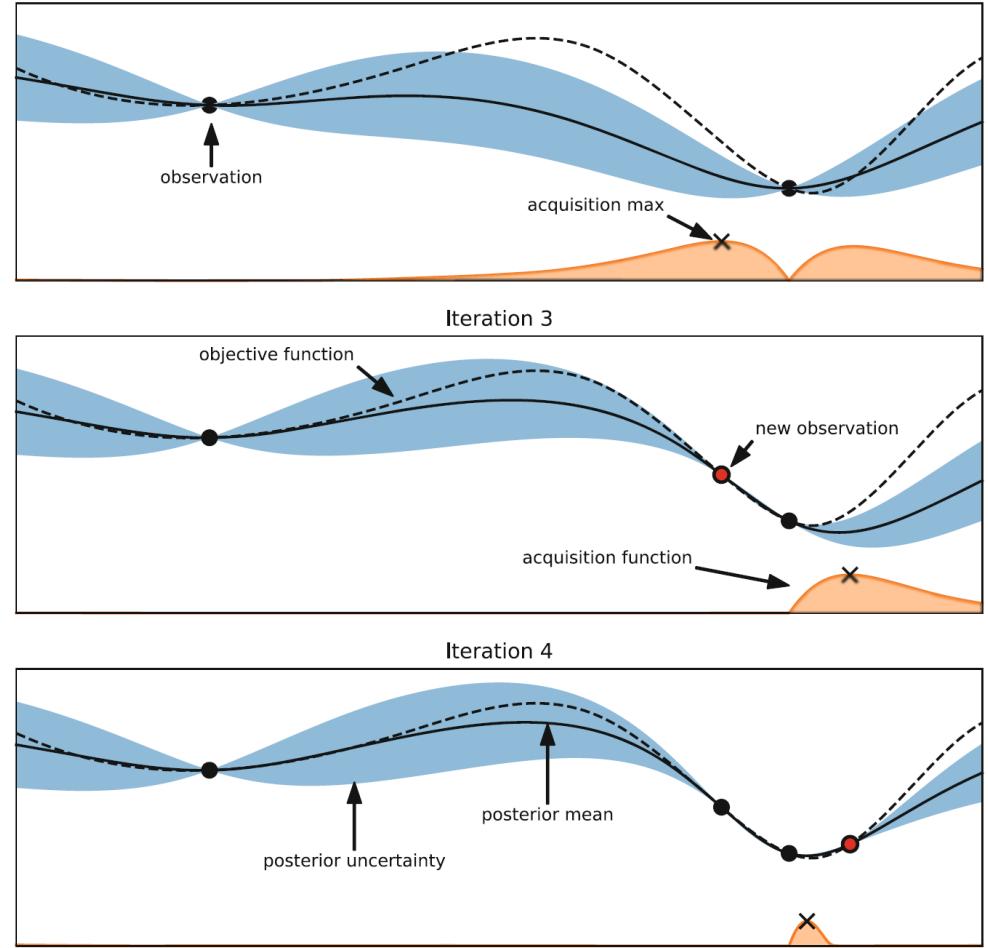


Towards Adaptive HPO Methods

- Drawback of [Random Search](#):
 - No informed decisions (all samples drawn ‘blindly’)
 - Although in expectation RS asymptotically converges to the optimum
- [Sequential framework](#):
 - Initialize the search, then iteratively make informed decisions about the next trial point to evaluate based on previously gathered knowledge.
- A non-exhaustive classification of adaptive HPO methods:
 - Black-box optimization:
 - Model-free: evolutionary algorithms, genetic algorithms, evolution strategies, ...
 - Model-based: [Bayesian optimization](#), ...
 - Multi-fidelity optimization:
 - Bandit approaches: [successive halving](#), Hyperband, ...

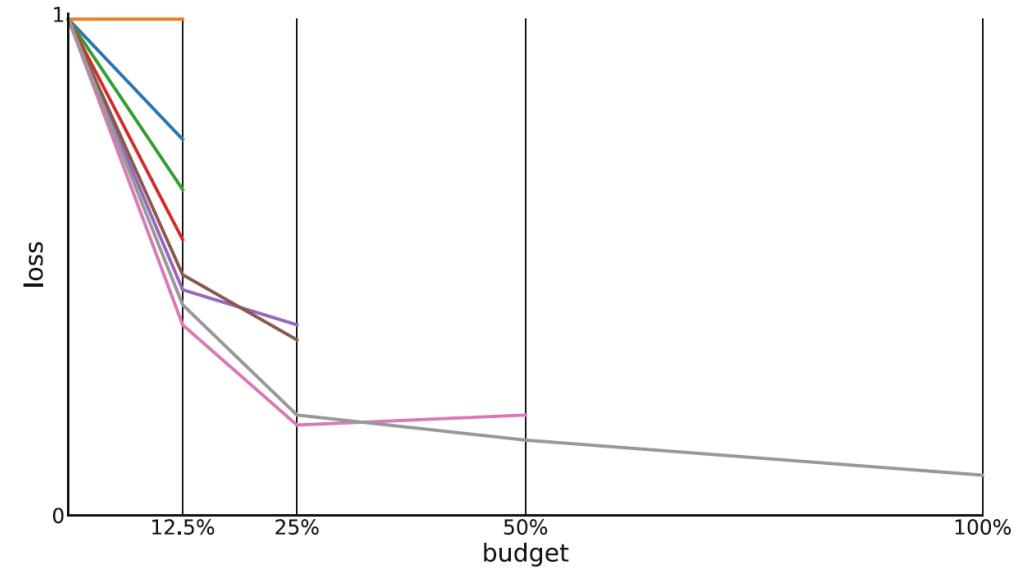
Bayesian Optimization

- Iterative optimization mechanism with 2 key ingredients:
 - A **probabilistic surrogate model** (fitted to previously observed trials)
 - An **acquisition function** (that decides which point to evaluate next)
- Best-suited for continuous domains
 - Not reliable for dimensions over 20
 - Tolerates stochastic noise
- **Surrogate models:** Quantify the uncertainty
 - Gaussian process, Random Forest, ...
- **Acquisition functions:** Determine what areas are worth exploring / exploiting
 - Probability of Improvement (PI)
 - Expected Improvement (EI)
 - Upper/Lower Confidence Bound (UCB/LBC), ...



Multi-Fidelity Approaches

- Techniques that combine a large number of cheap low-fidelity evals (on subsets of data / on small scale) and a small number of expensive high-fidelity evals (on full data)
- Allow for speed-ups that outweigh the approximation error
- Successive halving:
 - Run all trials with a budget B and rank them
 - Discard the worst half of trials
 - Double the budget and run the remaining half of trials
 - Repeat until 1 single configuration is left



Multi-Fidelity Approaches

- Problem:
 - Trade-off between the number of configurations and the number of ‘cuts’ we need.
- Hyperband:
 - Hedging technique that optimizes the search space when selecting from randomly sampled configurations.
 - We partition a given budget into random combinations of (# configurations, assigned budget) and we call successive halving on each of the combinations.
- Question: why use random sampling when we can do better?
- BOHB:
 - Method that combines the speed of Hyperband (due to successive halving) with the guidance and convergence guarantees of Bayesian optimization (thus replacing random sampling)

Key Concepts Covered Today

- HPO problem definition
- Grid search and Random Search
- Bayesian Optimization
- Multi-fidelity Approaches

Hoeffding Races: Accelerating Model Selection Search for Classification and Function Approximation

Oded Maron
Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139

Andrew W. Moore
Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

Selecting a good model of a set of input points by cross validation is a computationally intensive process, especially if the number of possible models or the number of training points is high. Techniques such as gradient descent are helpful in searching through the space of models, but problems such as local minima, and more importantly, lack of a distance metric between various models reduce the applicability of these search methods. Hoeffding Races is a technique for finding a good model for the data by quickly discarding bad models, and concentrating the computational effort at differentiating between the better ones. This paper focuses on the special case of leave-one-out cross validation applied to memory-based learning algorithms, but we also argue that it is applicable to any class of model selection problems.



Advances in Neural Information Processing Systems 6 (NIPS 1993): 59-66, 1993.
(The paper is available online at <https://proceedings.neurips.cc/>)