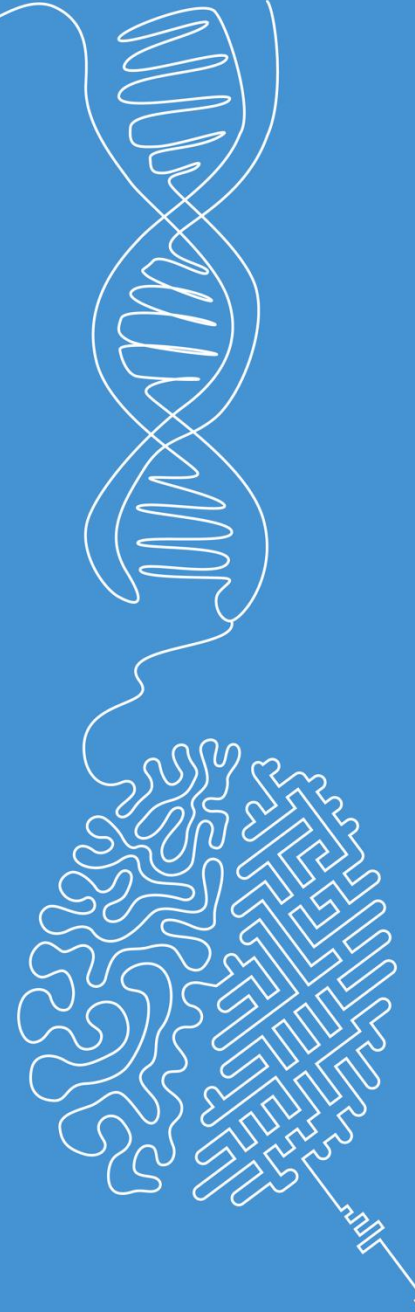


Hyperparameter Tuning

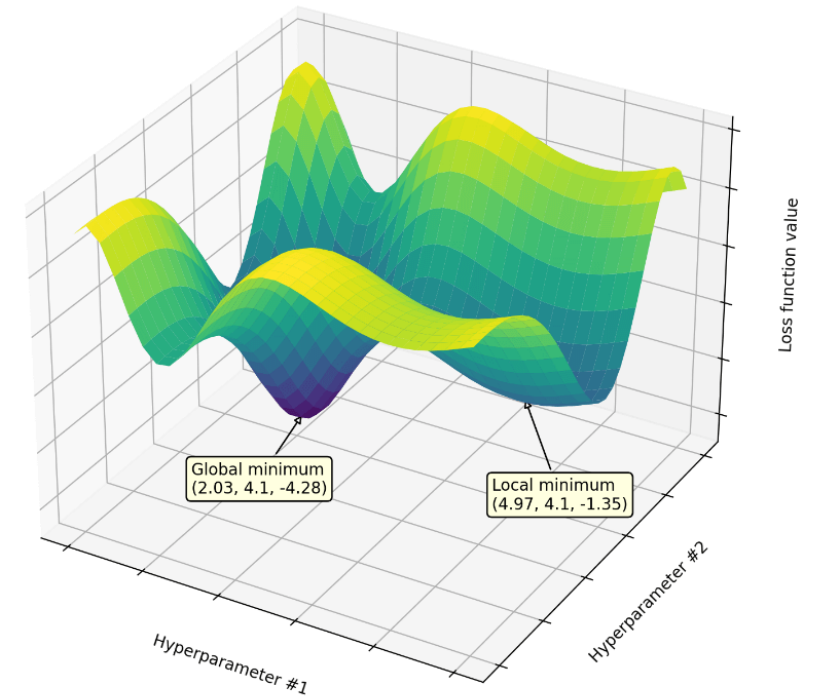
Machine Learning

Norman Juchler



Hyperparameters

- Parameters set before training a machine learning model, which govern the learning process and model architecture.
- Examples:
 - Number of clusters
 - Regularization strength
 - Learning rate
- Hyperparameters are not (directly) learned from the data but are tuned manually or through automated methods to optimize model performance.



The three levels of model parameters



(Model Design + Hyperparameters) → Model Parameters

Examples

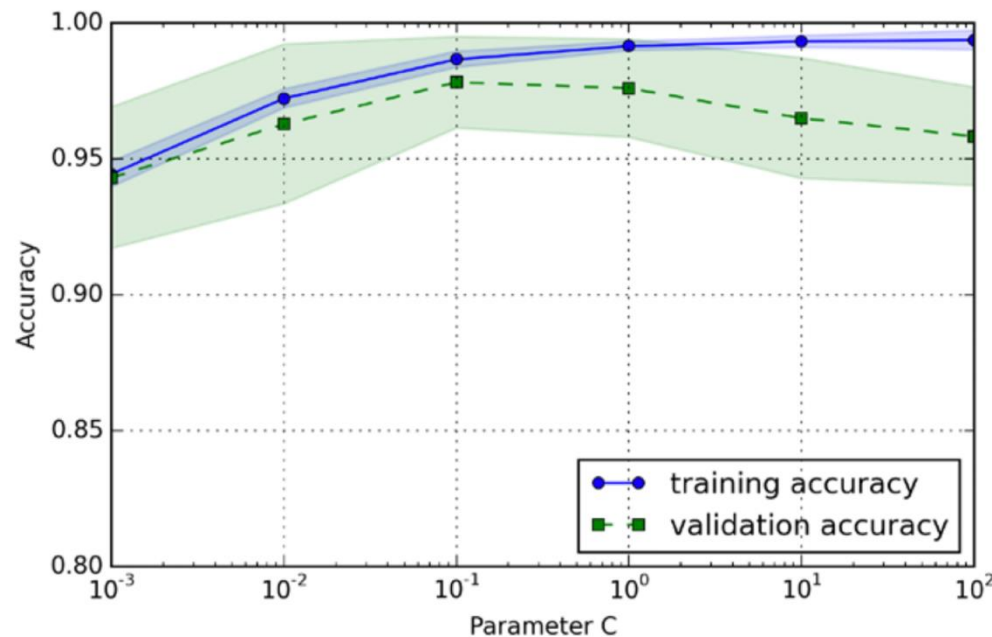
Function class
(e.g. cosign
or a polynomial)

Degree of the
polynomial

The polynomial's
parameters (a_0 , a_1 , etc.)

Validation curves

- Plot the training and validation accuracy as functions of hyper parameters.



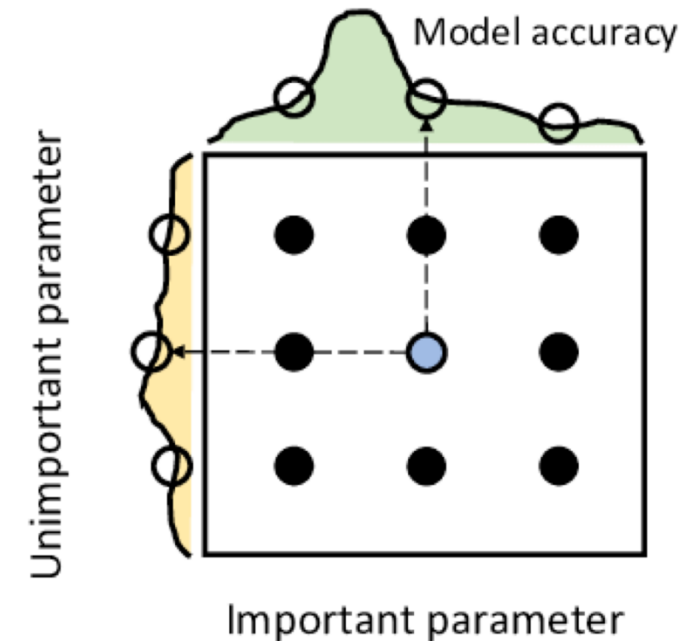
Question: How many times would we have to train the model if we have six hyperparameters and want to test three different values for each?

Answer: $3^6 = 729$

- Identify the point with the maximal validation accuracy.

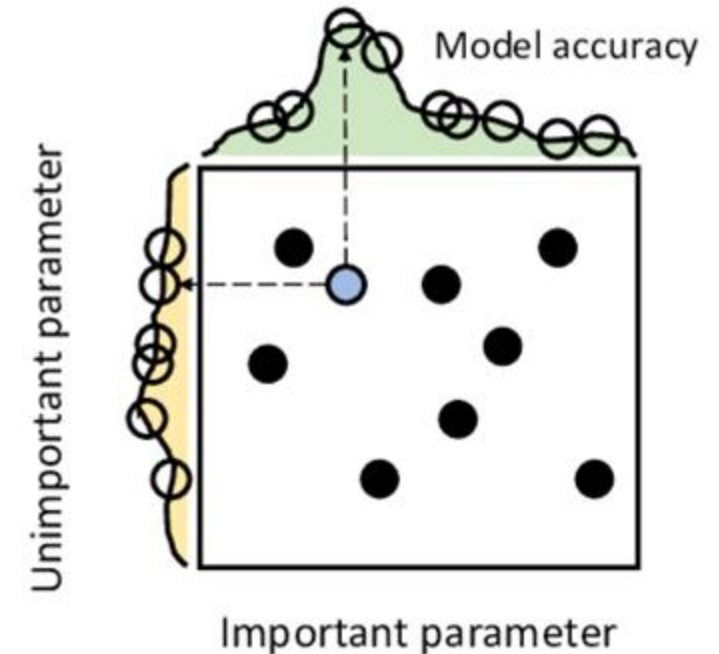
Automatization: Grid search

- Grid search simply executes all combinations of a given set of parameter values and delivers the best performing estimator.
- Advantage: No gradient needed
- Problems:
 - The number of combinations grows exponentially with the number of parameters
 - Unimportant parameters waste a lot of computation
 - Sparse sampling of the parameter space: Only few points can be tested per parameter



Random search

- Not all parameter combinations are tried out, but rather a fixed number of parameter settings is sampled from the specified distributions.
- Advantages:
 - Variations in unimportant parameters still sample new points in the other parameters
 - More parameter values are sampled than in the grid sampling (8 values vs. 3 values per parameter)



Faster search with successive halving

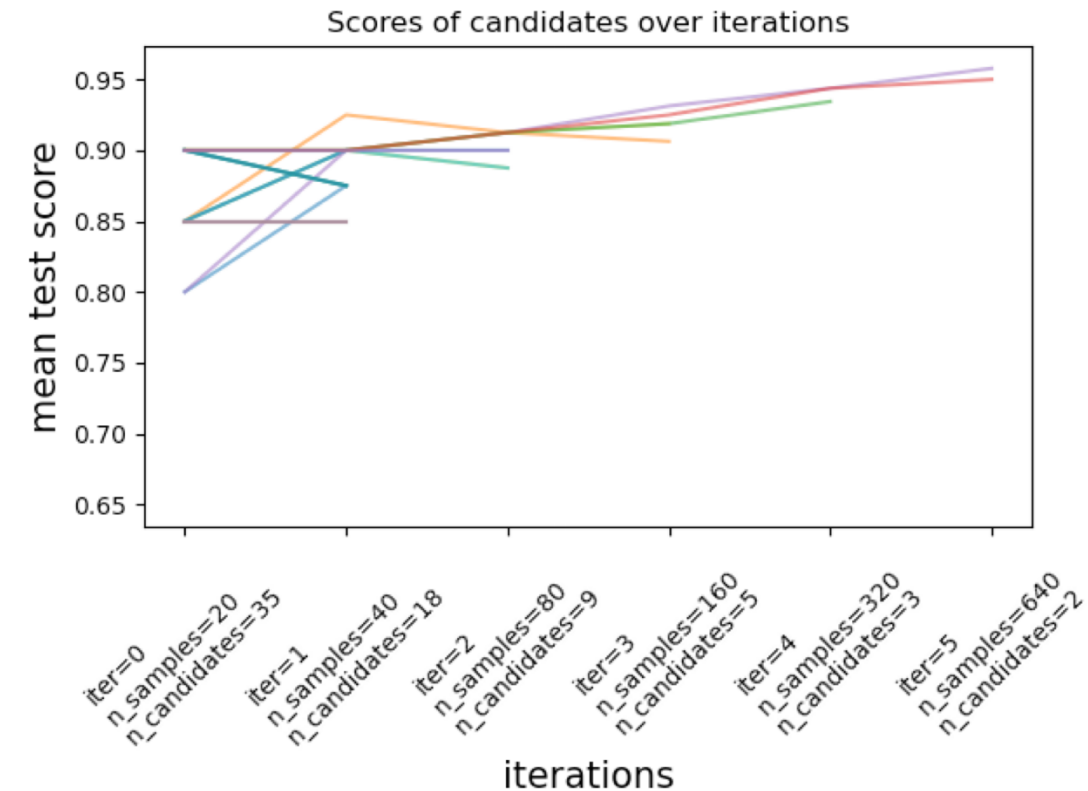
■ Outline:

- **Start broad:** Begin with a large pool of hyperparameter configurations and evaluate the different models using a small budget: small training set, few training iterations, ...
- **Select the best:** Identify the 50% of the best performing configurations
- **Increase the budget:** More data, longer training
- **Repeat:** Until the best configuration is found

■ Relevant scikit-learn classes:

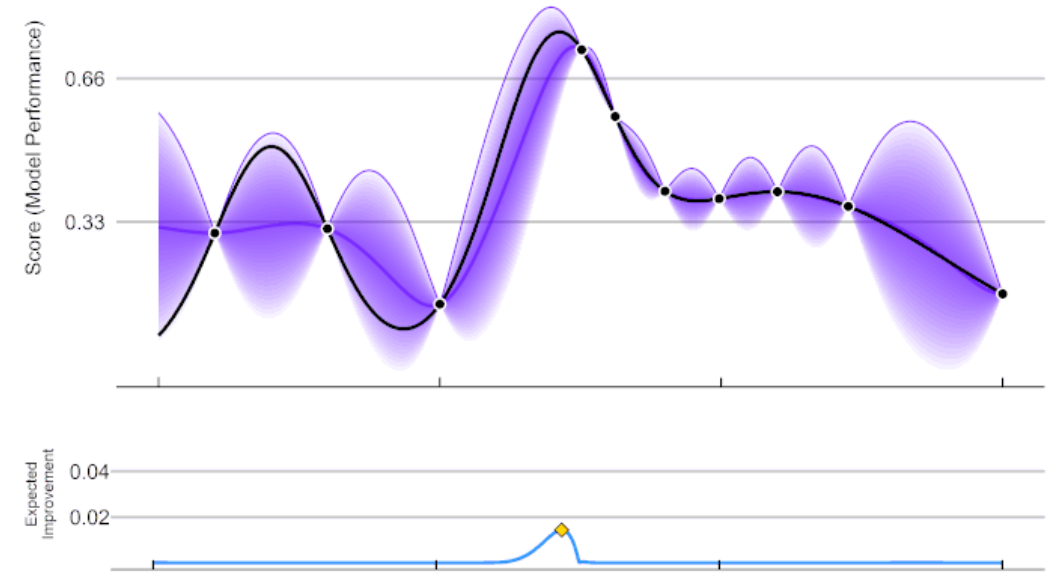
- HalvingGridSearchCV
- HalvingRandomSearchCV

■ Further reading (scikit-learn): [Link](#)



Bayesian optimization

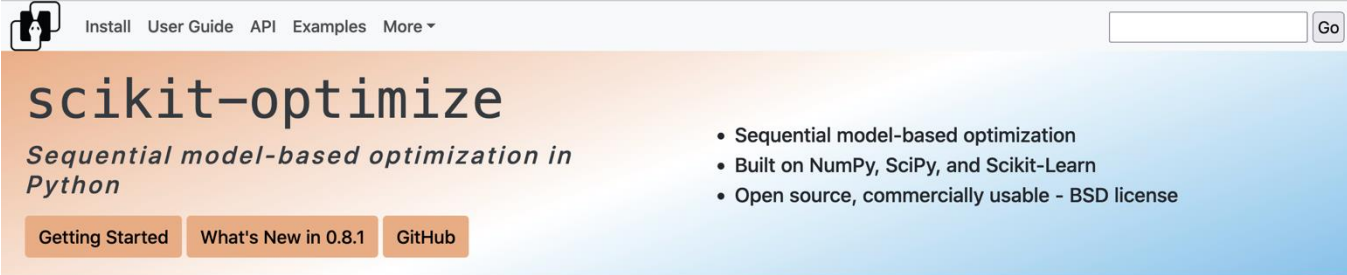
- The previous methods (like grid search) does not take advantage of the structure of a search space.
- Bayesian optimization uses a predictive “surrogate” model to model the search space and find a good parameter value combination quickly.
- The overall goal is: To evaluate the model as few times as possible, by learning from the previous evaluations.



The black line represents a model's evaluation score as a function of its hyperparameter(s). Normally, we do not know this function; we can only measure points along the function: by training a model with a specific hyperparameter and evaluating the score (black points). With probabilistic modelling (purple area) we try to learn from each evaluation and narrow down the most probable course of this unknown (black) function. Using this approach, we may find the optimal hyperparameters more efficiently, with fewer evaluations.

scikit-optimize

- Sequential model-based optimization in Python
- <https://scikit-optimize.github.io>



Install User Guide API Examples More ▾

scikit-optimize

Sequential model-based optimization in Python

- Sequential model-based optimization
- Built on NumPy, SciPy, and Scikit-Learn
- Open source, commercially usable - BSD license

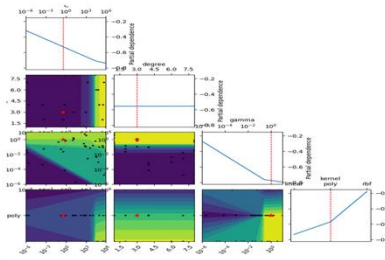
Getting Started What's New in 0.8.1 GitHub

BayesSearchCV

Scikit-learn hyperparameter search wrapper.

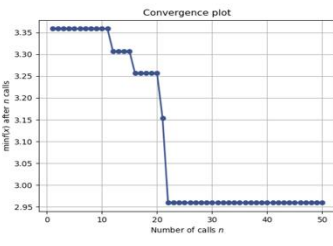
Search for parameters of machine learning models that result in best cross-validation performance

Algorithms: BayesSearchCV



Tuning

Tuning a scikit-learn estimator with skopt



Visualizing

Visualizing optimization results

