Machine Learning VII

Model Selection and Parameter Optimization

Model Selection

Motivation:

- Modeling a learning problem, we have many parameters to set:
  - Feature extraction algorithm (see Block)
  - Feature selection and reduction (see Block 7)
  - Choice of the **learning algorithm**
  - **Hyper-Parameters** of the learning algorithm
  - …
- **How to find the best model ?**

Model Selection

Hochschule Offenburg
offenburg.university

How to find the best model ?

- We already have quality measures to compare models, like
  - Accuracy
  - F-Measure
  - ROC
  - …

- **But there are two problems:**
  - Do not over fit on the test data (→ can't test too often to stay unbiased)
  - Computational complexity (→ can't try every possible combination)

Cross-Validation

Hochschule Offenburg
offenburg.university

Tuning a model without test data

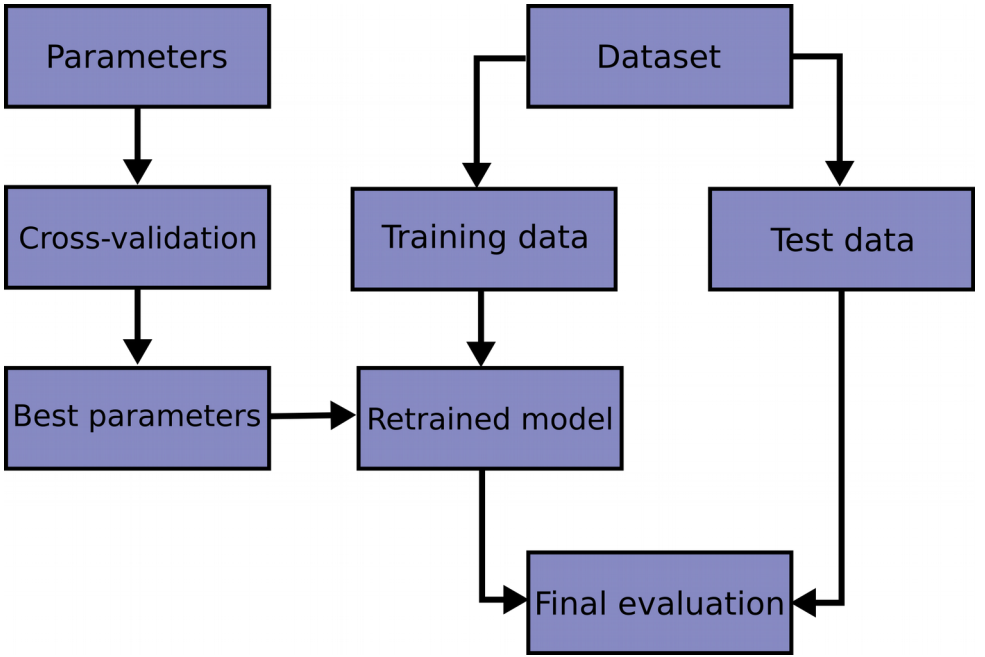
Simple approach:
n-fold cross validation

- Split train data into n parts
- Train on n-1 parts – test on the left out part
- Repeat n-times, leaving out a different part each time
- Average test results

Cross-Validation

Hochschule Offenburg
offenburg.university

Workflow



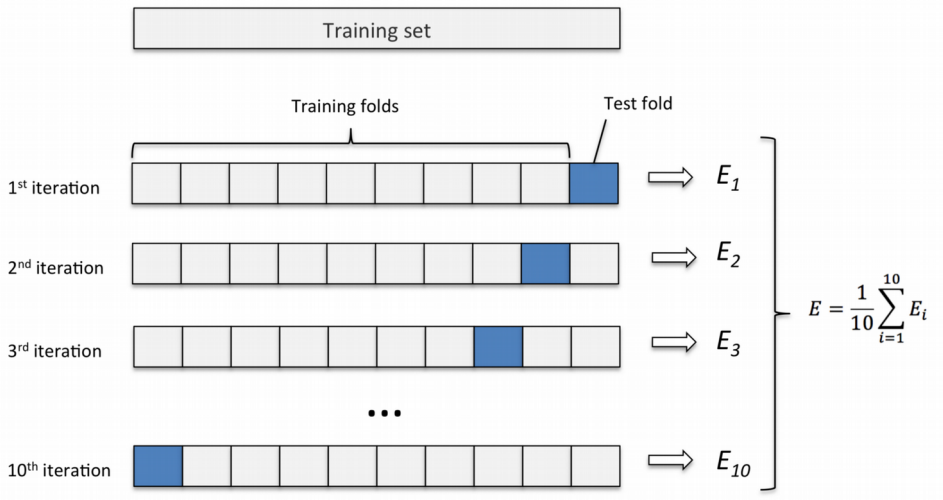
https://scikit-learn.org/stable/modules/cross_validation.html

Cross-Validation

Hochschule Offenburg
offenburg.university

In Scikit-Learn

```
>>> from sklearn import metrics
>>> scores = cross_val_score(
...     clf, X, y, cv=5, scoring='f1_macro')
>>> scores
array([0.96..., 1.  ..., 0.96..., 0.96..., 1.        ])
```



Hyper-Parameter Optimization

Hochschule Offenburg
offenburg.university

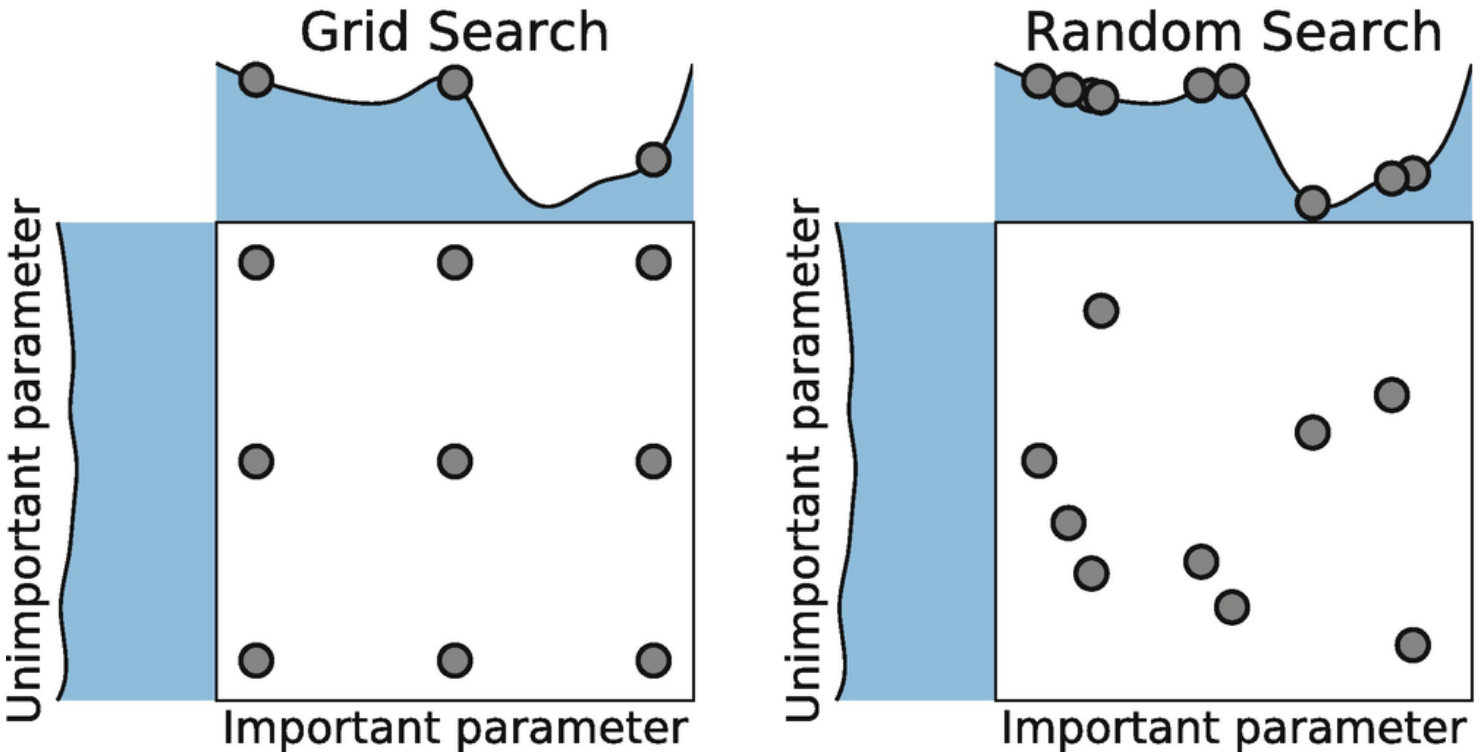
How to find the best model ?

- Grid-search over the parameter space
  - Very expensive
  - How to space the grid ?
- **Random Search**
  - Cheaper than grid-search
  - Quite effective
- Bayesian optimization
  - “optimal” next parameter set for testing

Hyper-Parameter Optimization

Hochschule Offenburg
offenburg.university

Grid Search vs Random Search



Hyper-Parameter Optimization

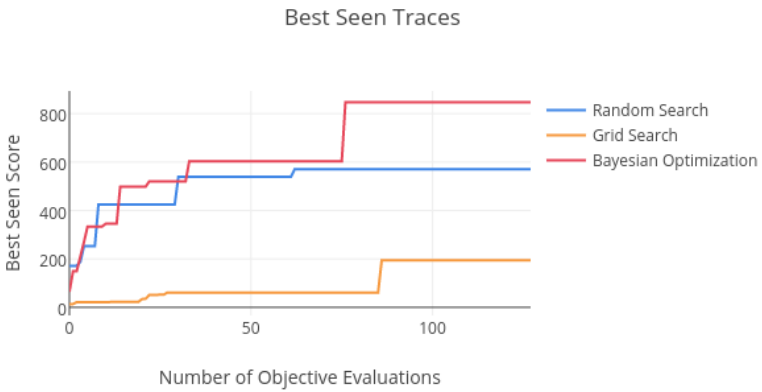
Hochschule Offenburg
offenburg.university

Random Search combined with Cross-Validation

```
>>> from sklearn.datasets import load_iris
>>> from sklearn.linear_model import LogisticRegression
>>> from sklearn.model_selection import RandomizedSearchCV
>>> from scipy.stats import uniform
>>> iris = load_iris()
>>> logistic = LogisticRegression(solver='saga', tol=1e-2, max_iter=200,
...                               random_state=0)
>>> distributions = dict(C=uniform(loc=0, scale=4),
...                      penalty=['l2', 'l1'])
>>> clf = RandomizedSearchCV(logistic, distributions, random_state=0)
>>> search = clf.fit(iris.data, iris.target)
>>> search.best_params_
{'C': 2..., 'penalty': 'l1'}
```

Bayesian Optimization

Method	Effectiveness	Parallelization
grid search	- -	trivial
random search	+	trivial
informed search (e.g. Bayesian Optimization)	+ +	non-trivial



https://sigopt.com/blog/using-bayesian-optimization-for-reinforcement-learning/

Basics of Bayesian Optimization (BO)

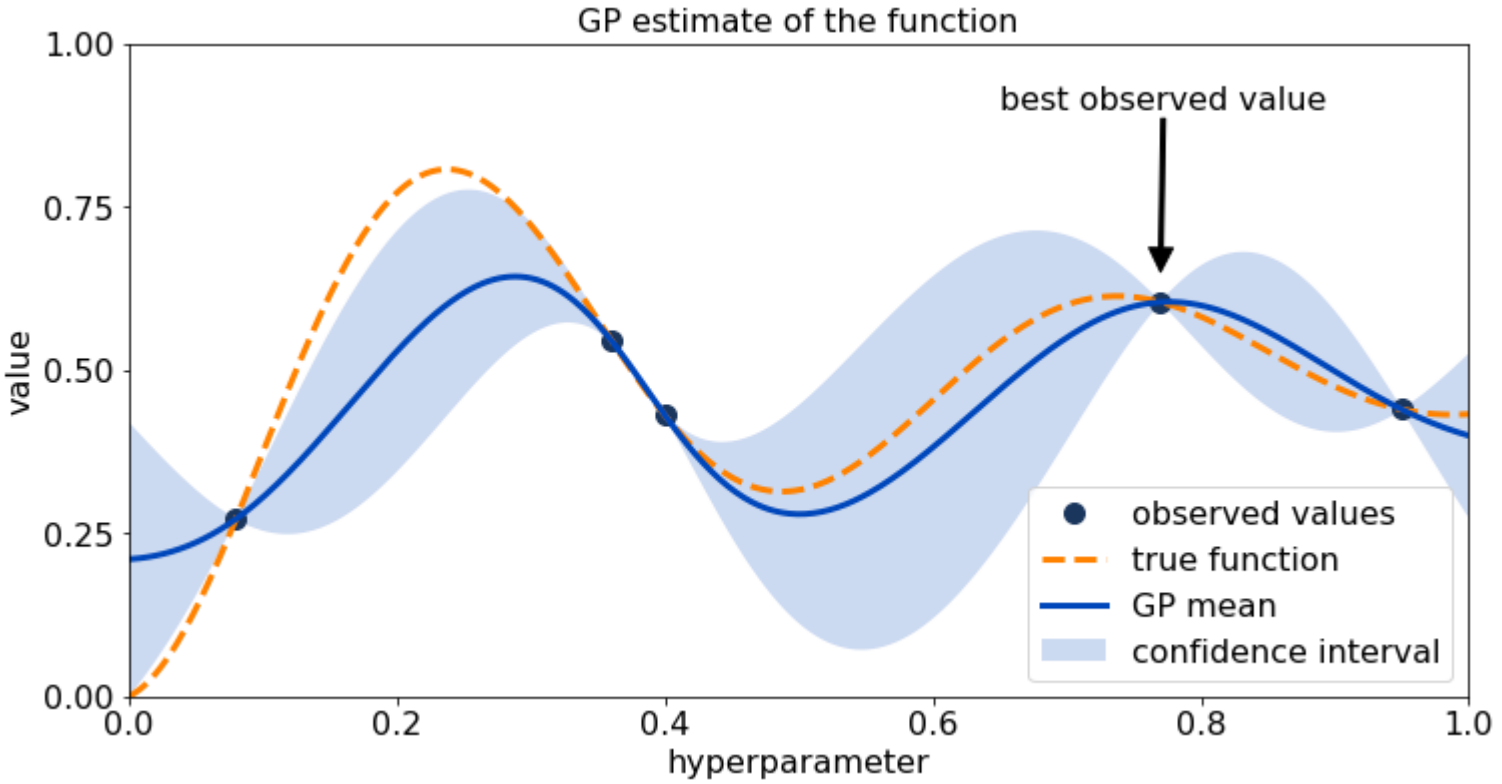
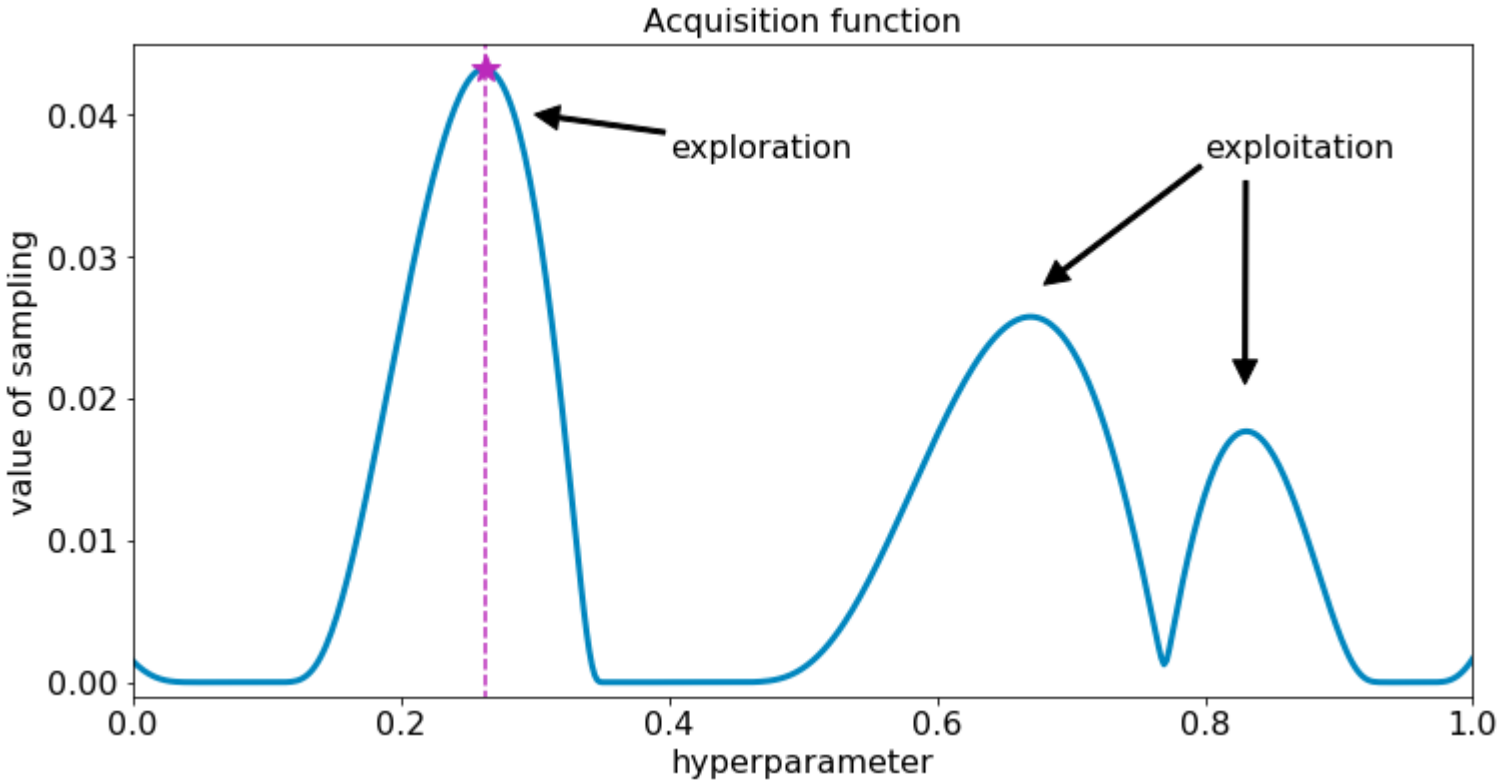
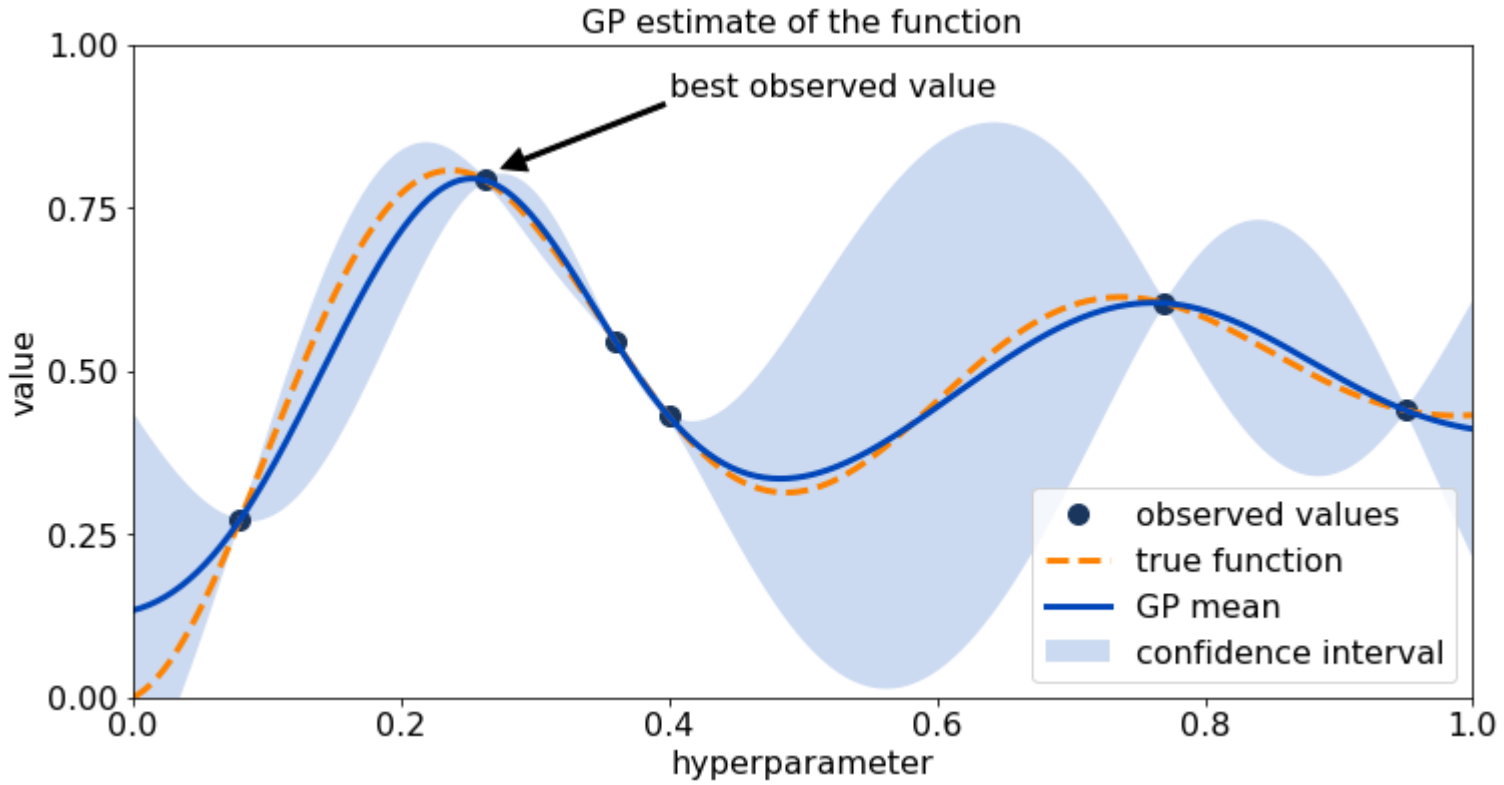


Figure from: https://mlconf.com/blog/lets-talk-bayesian-optimization/

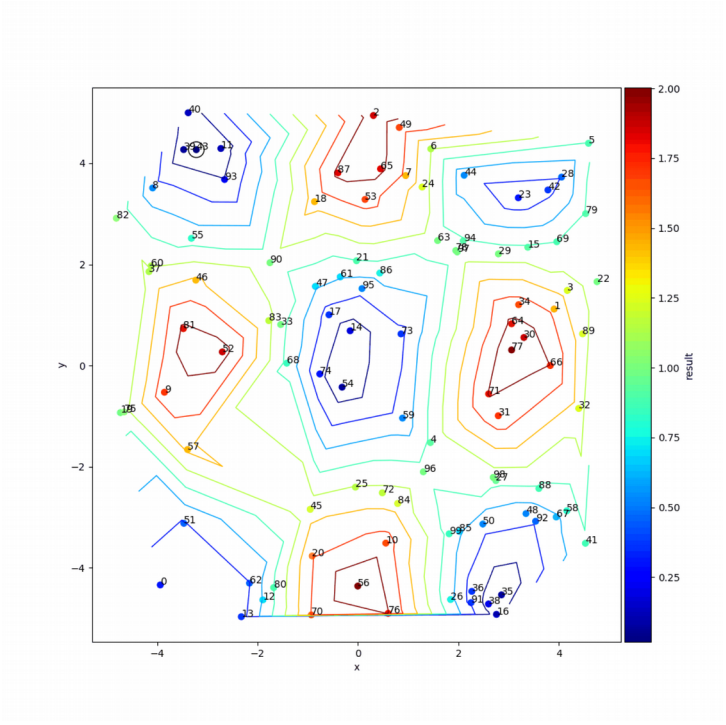
Basics of Bayesian Optimization (BO)



Basics of Bayesian Optimization (BO)



Bayesian Optimization



PHS: A Toolbox for Parallel Hyperparameter Search

1st Peter Michael Habelitz
Competence Center for High Performance Computing
Fraunhofer ITWM
Kaiserslautern, Germany
peter.michael.habelitz@itwm.fraunhofer.de

2nd Janis Keuper
Institute for Machine Learning and Analytics (IMLA)
Offenburg University, Germany
keuper@imla.ai

***Abstract*—We introduce an open source python framework named *PHS - Parallel Hyperparameter Search* to enable hyperparameter optimization on numerous compute instances of any arbitrary python function. This is achieved with minimal modifications inside the target function. Possible applications appear in expensive to evaluate numerical computations which strongly depend on hyperparameters such as machine learning. Bayesian optimization is chosen as a sample efficient method to propose the next query set of parameters.**

Source code: https://github.com/cc-hpc-itwm/PHS

***Index Terms*—Hyperparameter Optimization; Bayesian Optimization; Machine Learning**

I. INTRODUCTION

In recent years, the requirements in the field of machine learning have changed drastically: There is an enlarging amount of data and the computing power is constantly increasing. Research is also dealing with more complex and more difficult questions, such that the associated algorithms are becoming more complicated. This growing complexity also affects the number of hyperparameters to be configured. These range from

II. CURRENT STATE OF THE ART

The variety of different approaches to hyperparameter searches is very broad: From random search [1] to random forests [2], to particle swarm optimization [3], up to Bayesian optimization [4], [5]. Especially the latter is a useful environment for the optimization of expensive functions. It naturally handles the balancing between exploration and exploitation by means of its acquisition function.

III. METHODS AND KEY RESULTS

Many algorithms have some *free* parameters which are set at the beginning of an evaluation. *free* means the impact on the result is in theory and practice not fully understood and can therefor be set in a reasonable range. In machine learning and deep learning these values are called *hyperparameters* whose relevance are explained in the following.

- Research in that field has proven to be a highly empirically driven task. This means that progress is often rooted in experiments which showed improvements in some specific aspect. Thereafter researchers hop onto and try to find theoretical explanations. One nice example is (batch normalization). This is not a bad situation in general but it emphasizes the lack of a broad and profound mathematical

https://github.com/cc-hpc-itwm/PHS

https://arxiv.org/abs/2002.11429

Discussion

Hochschule Offenburg
offenburg.university

Lab exercises coming up ...