Introduction to Hyperparameter Optimization and AutoML

Ondřej Sotolář
xsotolar@fi.muni.cz

Faculty of Informatics, Masaryk University

October 17, 2022

Presentation Content

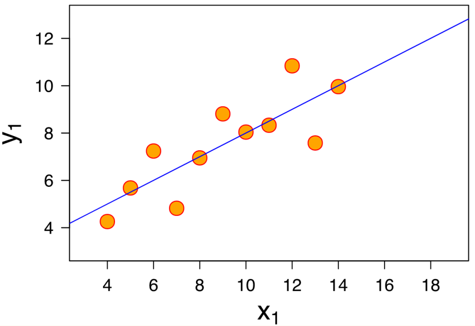
1. Introduction
2. Hyperparameter Optimization (HPO)
3. AutoML and Implementations
4. HPO and AutoML for NLP

Introduction

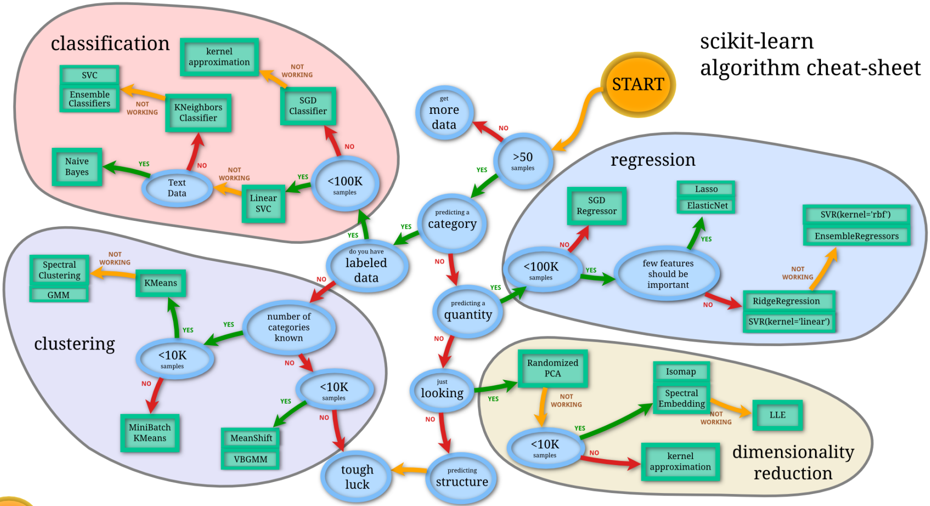
Machine learning is the science of getting computers to act without being explicitly programmed. (A. Ng)

Machine learning is concerned with predictive analytics, in contrast with many other fields that utilize statistical models [1]. Models (e.g. regression, classification, etc.) are the basic ML tools.

Exercise: What is the model here?

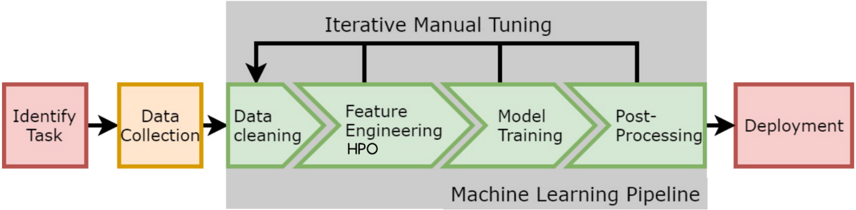


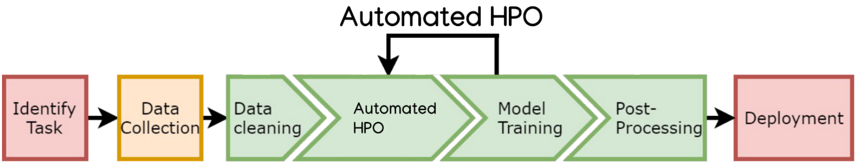
Motivation

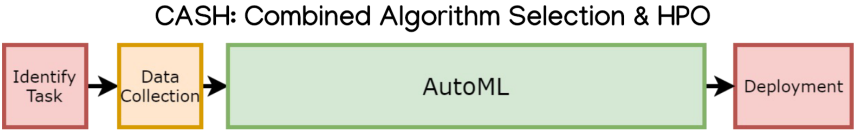


Back

AutoML: HPO, CASH







Model Parameters vs. Hyperparameters I

It is critical to understand the difference between model parameters and hyperparameters.

Model parameters are optimized during training, typically via loss minimization. They are the output of the training.

Examples:

- coefficients θ of a linear model $f(x) = \theta^T x$
- the splits and terminal node of a decision tree-based model

Model Parameters vs. Hyperparameters II

In contrast, hyperparameters (HPs) are not decided during training. They must be specified before the training, they are an input of the training.

Hyperparameters can in principle influence any structural property of a model or computational part of the training process.

Examples:

- Tree: the maximum depth of a tree
- k Nearest Neighbours: Number of neighbours k and distance measure
- Neural networks: number and type of layers, activation functions, regularization, and many more.

Types of Hyperparameters

- Real-valued parameters, e.g.:
  - Minimal error improvement in a tree to accept a split
- Integer parameters, e.g.:
  - Neighbourhood size k for k-NN
- Categorical parameters, e.g.:
  - Split criterion for classification trees
- Hierarchical dependencies
  - Cast as nested problem or Multi-criteria optimization

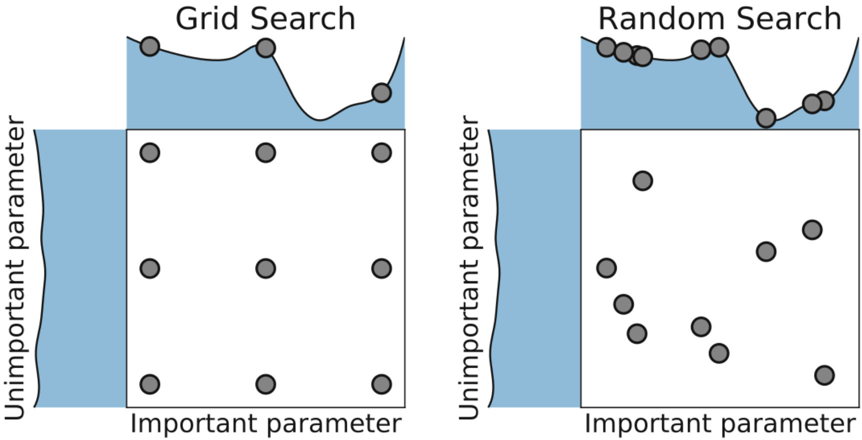
Definition

We summarize all hyperparameters we want to tune over in a vector

$$\boldsymbol{\lambda} \in \Lambda,$$

where Λ is the space of all possible hyperparameter value combinations. It has as many dimensions as there are hyperparameters.

Manual Search Strategies I: Grid serach & Random search

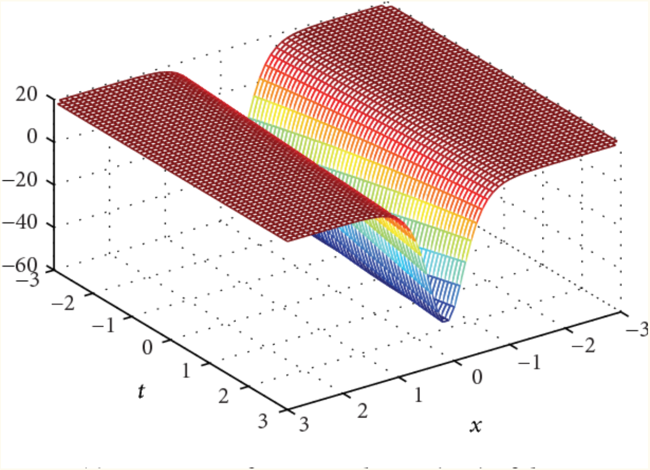


[1]

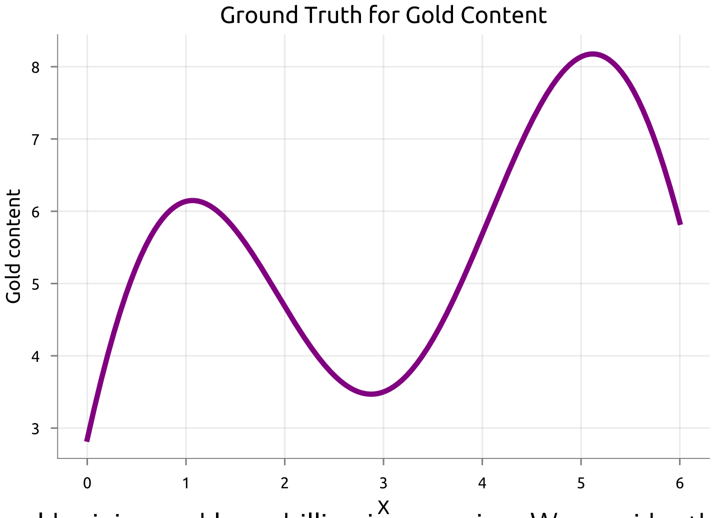
[1] Figure from [2]

Manual Search Strategies II

Exercise: Given this 2D loss surface in, would you prefer Grid search or random search?



Bayesian Optimization: Intuition & True function



In our gold mining problem, drilling is expensive. We consider the real distribution as the **True Function** (or Ground Truth).

Bayesian Optimization: Active Learning, Surrogate model, and Acquisition Functions

We need a **surrogate model** to approximate the True Function.[2]

Active learning minimizes sampling costs while maximizing performance via uncertainty reduction. This method proposes sampling at the point where the surrogate model uncertainty is the highest. We use **variance** as the measure of uncertainty.

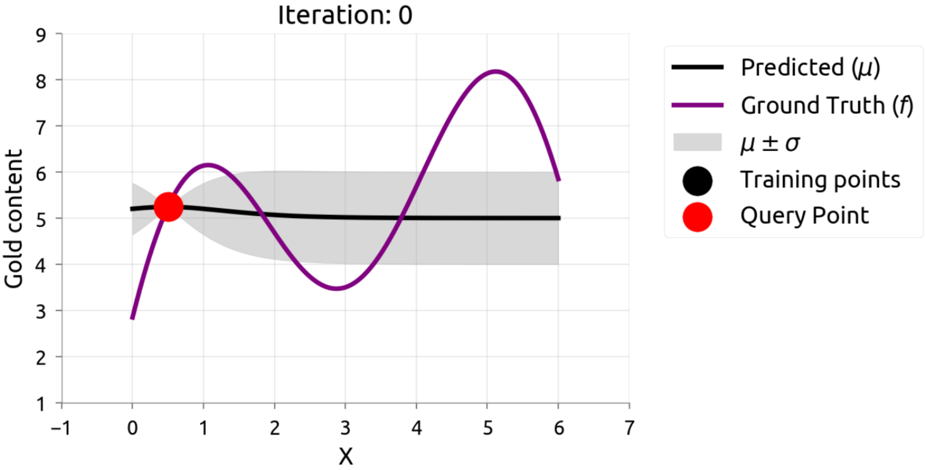
Furthermore, we need an **acquisition function** for selecting the location in the configuration space where the next sample will be taken.

Examples of Acquisition functions:

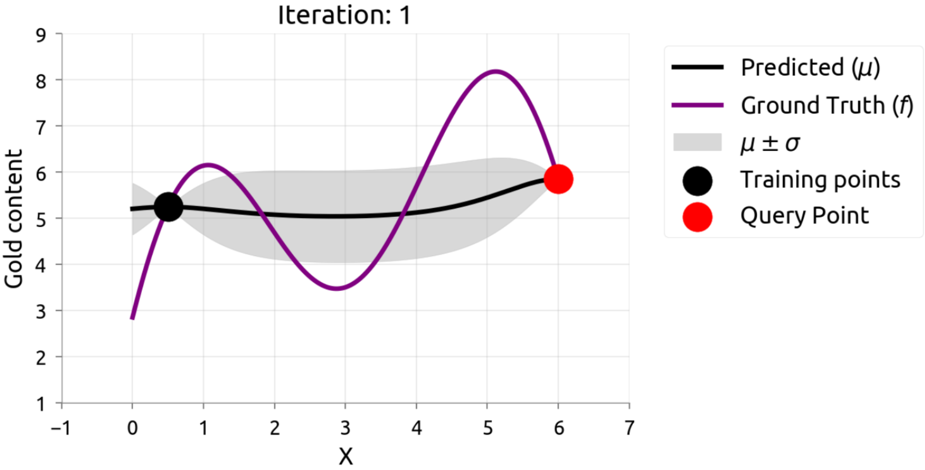
Probability of Improvement (PI), Expected Improvement (EI), Upper/Lower Confidence Bound, Thompson Sampling, etc.

[2] Figure in the next slide from [3]

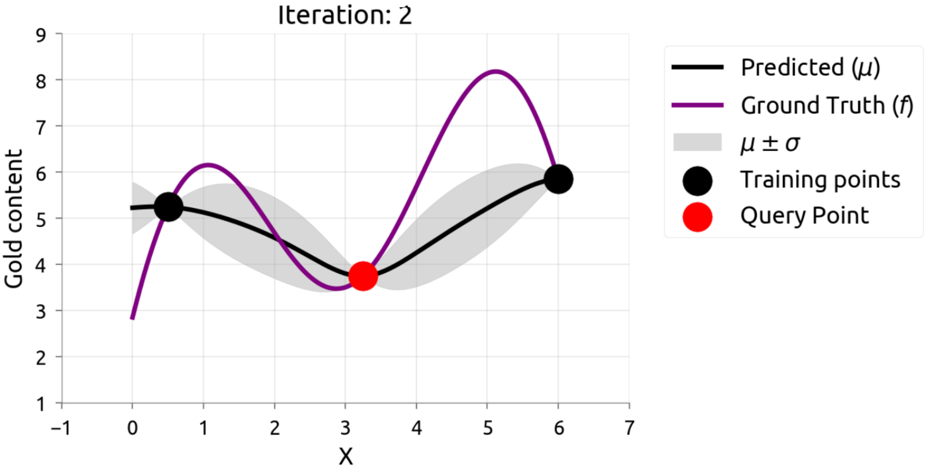
Bayesian Optimization with a Gaussian Process



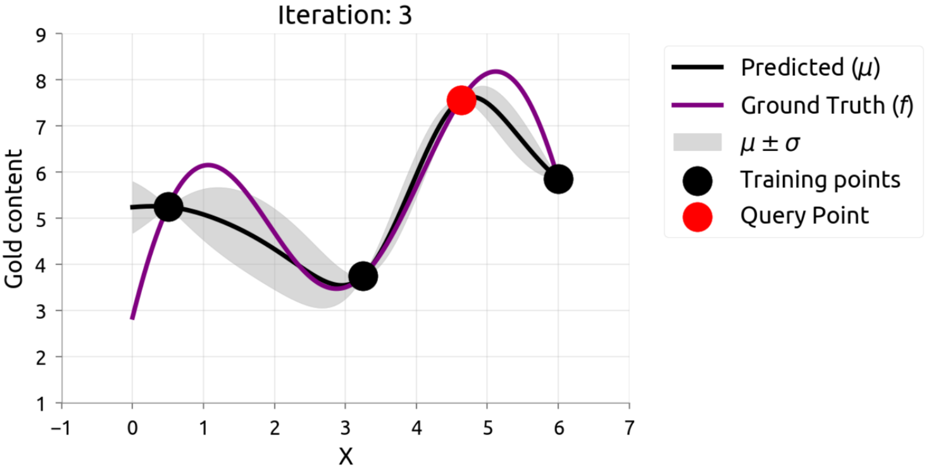
Bayesian Optimization with a Gaussian Process



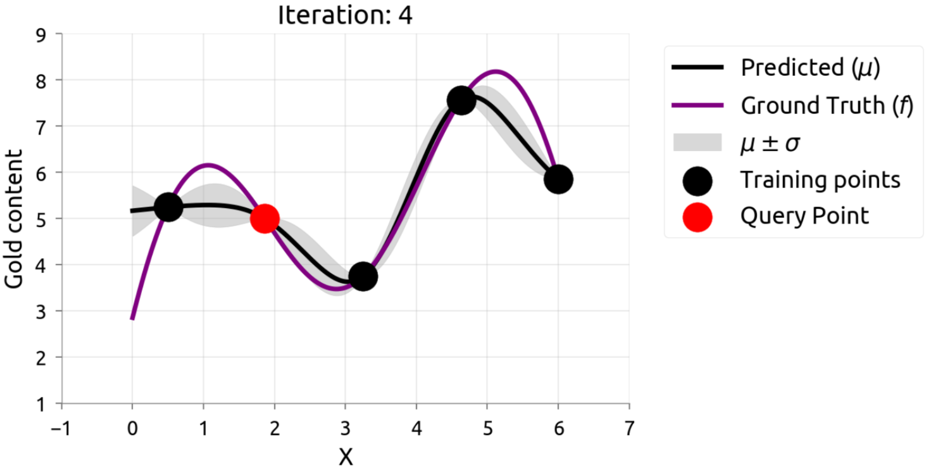
Bayesian Optimization with a Gaussian Process



Bayesian Optimization with a Gaussian Process



Bayesian Optimization with a Gaussian Process



A (very minimal) Formal Definition of a Gaussian Process

A function $f(x)$ is generated by a Gaussian process G if for any finite set of inputs $\{x_1, ..., x_n\}$, the associated vector of function values has a Gaussian distribution:

$$\boldsymbol{f} = (f(x_1), ..., f(x_n)) \sim \mathcal{N}(m, K),$$

with:

$$\boldsymbol{m} = m(x_i)_i, \boldsymbol{K} = k(x_i, x_j)_{i,j},$$

where $m(x)$ is called mean function and $k(.,.)$ is called covariance function.

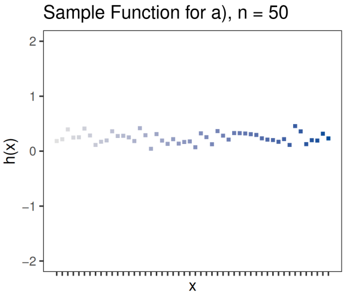
Finally, we denote a GP by:

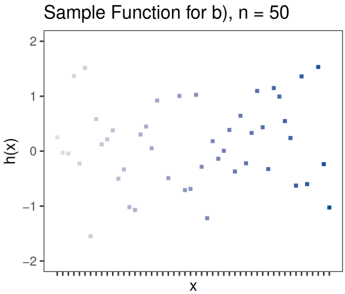
$$f(x) \sim \mathcal{G}(m(x), k(x, x'))$$

Covariance Function: Intuition

The covariance controls the shape of drawn functions. Consider two extreme cases where function values are:[3]

- strongly correlated: $\boldsymbol{K} = \begin{pmatrix} 1 & 0.99 & \dots & 0.99 \\ 0.99 & 1 & \dots & 0.99 \\ 0.99 & 0.99 & \ddots & 0.99 \\ 0.99 & \dots & 0.99 & 1 \end{pmatrix}$
- uncorrelated: $\boldsymbol{K} = \boldsymbol{I}$





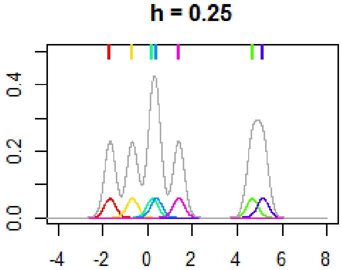
[3] Figure from [4].

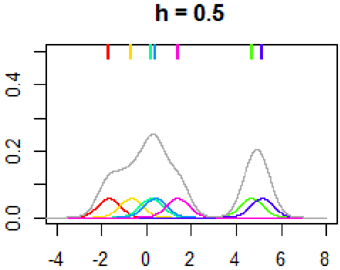
Tree-Parzen Estimators I

Parzen density estimation is defined as:

$$p(x) = \frac{1}{nh} \sum_{i=1}^{n} \boldsymbol{K}\left(\frac{x_i - x}{h}\right),$$

where n is number of elements in the vector, x is a vector, $p(x)$ is a probability density of x, h is the scale of x_i, and $\boldsymbol{K}$ is the kernel (product for TPE). The main idea to approximate f by a **mixture** of kernels.[4]





[4] Figure from [5]

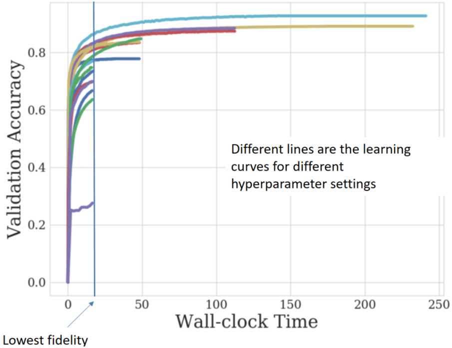
Tree-Parzen Estimators II

- TPE uses two estimators, one for **good** $l(\lambda)$ and one for **bad** $g(\lambda)$ HP configurations
- It iterates, and in each iteration, the estimates are improved by moving samples between $\boldsymbol{l}$ and $\boldsymbol{g}$
- $\boldsymbol{l}$ and $\boldsymbol{g}$ have thresholds (15% best/worst HP configurations)
- $EI(l(\lambda)/g(\lambda))$ is used to sample the next configuration.

Speedup Techniques for HPO

Multi-fidelity optimization: Successive Halving (SH)

1. Sample N configurations uniformly at random evaluate them on the cheapest fidelity
2. Keep the best half (or third), move them to the next fidelity
3. GOTO 1. (until original fidelity)[5]



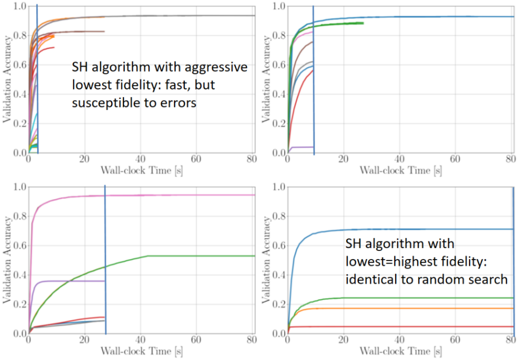
[5] Figure from [2].

Speedup Techniques for HPO

Extension to SH: Hyperband

What is the problem with SH?

Some HP configuration might start slow yet reach the highest performance on higher fidelities. The solution is to run multiple copies of SH in parallel, starting at different cheapest fidelities. (Fig from [2])



Speedup Techniques for HPO
BOHB

- BOHB combines the advantages of Bayesian Optimization and Hyperband
- BOHB replaces the random selection of configurations at the beginning of each HB with a model
- The Model is variant of the Tree Parzen Estimator, with a product kernel

HPO Frameworks

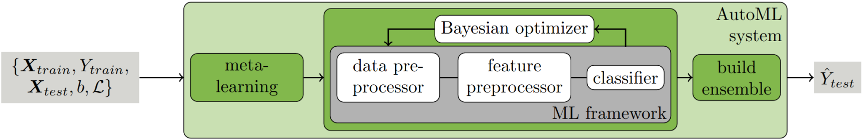
Optuna, SMAC, Hyperopt, FastText optimize, end others.

```
import optuna

# Define a simple 2-dimensional objective function
# whose minimum value is -1 when (x, y) = (0, -1).
def objective(trial):
    x = trial.suggest_float("x", -100, 100)
    y = trial.suggest_categorical("y", [-1, 0, 1])
    return x**2 + y

if __name__ == "__main__":
    # Let us minimize the objective function above.
    study = optuna.create_study()
    study.optimize(objective, n_trials=10)
    print(study.best_params)
```

AutoML Frameworks



Auto-Sklearn process. Figure from [6].

Libraries and Frameworks for AutoML

- Auto-Sklearn, AutoWeka, Auto-Keras
- {Google, Azure, Amazon} AutoML

Auto-Sklearn

Classification Case

name	#λ	cat (cond)	cont (cond)
AdaBoost (AB)	4	1 (-)	3 (-)
Bernoulli naïve Bayes	2	1 (-)	1 (-)
decision tree (DT)	4	1 (-)	3 (-)
extreml. rand. trees	5	2 (-)	3 (-)
Gaussian naïve Bayes	-	-	-
gradient boosting (GB)	6	-	6 (-)
kNN	3	2 (-)	1 (-)
LDA	4	1 (-)	3 (1)
linear SVM	4	2 (-)	2 (-)
kernel SVM	7	2 (-)	5 (2)
multinomial naïve Bayes	2	1 (-)	1 (-)
passive aggressive	3	1 (-)	2 (-)
QDA	2	-	2 (-)
random forest (RF)	5	2 (-)	3 (-)
Linear Class. (SGD)	10	4 (-)	6 (3)

(a) classification algorithms

name	#λ	cat (cond)	cont (cond)
extreml. rand. trees prepr.	5	2 (-)	3 (-)
fast ICA	4	3 (-)	1 (1)
feature agglomeration	4	3 ()	1 (-)
kernel PCA	5	1 (-)	4 (3)
rand. kitchen sinks	2	-	2 (-)
linear SVM prepr.	3	1 (-)	2 (-)
no preprocessing	-	-	-
nystroem sampler	5	1 (-)	4 (3)
PCA	2	1 (-)	1 (-)
polynomial	3	2 (-)	1 (-)
random trees embed.	4	-	4 (-)
select percentile	2	1 (-)	1 (-)
select rates	3	2 (-)	1 (-)
one-hot encoding	2	1 (-)	1 (1)
imputation	1	1 (-)	-
balancing	1	1 (-)	-
rescaling	1	1 (-)	-

(b) preprocessing methods

Figure from [6].

Auto-Sklearn: Classification Example

```
# example of auto-sklearn usage for a classification problem
data = load_breast_cancer()
X, y = data[:, :-1], data[:, -1]
X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=0.33, random_state=1)

# define search
model = AutoSklearnClassifier(
    time_left_for_this_task=5*60,
    per_run_time_limit=30, n_jobs=8)

# perform the search and evaluate
model.fit(X_train, y_train)
print(accuracy_score(y_test, model.predict(X_test)))
```

HPO and AutoML for NLP

- FastText HPO (2017) [7]

HPO and AutoML for NLP

- FastText HPO (2017) [7]
- Zhu, Wang [8]: Multi-Stage HPO with SVM and Boosted Regression Trees
  - Published in 2015, before Hyperband (2017)
  - First uses SH, then the standard BO. After running all fidelities, it outputs best config. from all HPs explored at all fidelities.

Algorithm	Hyper-parameters
SVM	Bias, cost parameter, and regularization parameter
Boosted regression trees	Feature sampling rate, data sampling rate, learning rate, # trees, # leaves, and minimum # instance per leaf

Hyper-parameters	Values
n_{min}	$\{1,2,3\}$
n_{max}	$\{n_{min},\cdots,3\}$
Weighting scheme	{tf, tf-idf, binary}
Remove stop words?	True, false
Regularization	l_1, l_2
Regularization strength	$\{10^{-5}, 10^{5}\}$
Convergence tolerance	$\{10^{-5}, 10^{-3}\}$

- TextNAS (2020) [9]

Closing Remarks
Interesting Resources

- TWIML Podcast: AutoML Zero with Quoc Le [10]
- Distill articles on HPO and GP [3, 11]
- Hutter's Book [2]
- Documetation of Optuna, SMAC, Auto-Sklearn [12, 13, 14]

Bibliography I

[1] Galit Shmueli. "To explain or to predict?" In: *Statistical science* 25.3 (2010), pp. 289–310.

[2] Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren. *Automated machine learning: methods, systems, challenges*. Springer Nature, 2019.

[3] Apoorv Agnihotri and Nipun Batra. "Exploring Bayesian Optimization". In: *Distill* (2020). https://distill.pub/2020/bayesian-optimization. doi: 10.23915/distill.00026.

[4] Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren. *Automated ML*. 2022. url: https://learn.ki-campus.org/courses/automl-luh2021.

Bibliography II

[5] *Parzen Estimators*. 2022. url: https://stats.stackexchange.com/questions/244012/can-you-explain-parzen-window-kernel-density-estimation-in-laymans-terms.

[6] Matthias Feurer et al. "Efficient and robust automated machine learning". In: *Advances in neural information processing systems* 28 (2015).

[7] Armand Joulin et al. "FastText.zip: Compressing text classification models". In: *arXiv preprint arXiv:1612.03651* (2016).

Bibliography III

[8] Lidan Wang et al. "Efficient hyper-parameter optimization for NLP applications". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. 2015, pp. 2112–2117.

[9] Yujing Wang et al. "Textnas: A neural architecture search space tailored for text representation". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 05. 2020, pp. 9242–9249.

[10] *TWIML AI Podcast: AutoML Zero*. 2022. url: https://www.youtube.com/watch?v=XesVQIjQb18&ab_channel=TheTWIMLAIPodcastwithSamCharrington.

Bibliography IV

[11] Jochen Görtler, Rebecca Kehlbeck, and Oliver Deussen. "A Visual Exploration of Gaussian Processes". In: *Distill* (2019). https://distill.pub/2019/visual-exploration-gaussian-processes. doi: 10.23915/distill.00017.

[12] *Optuna*. 2022. url: https://optuna.org/.

[13] *SMAC*. 2022. url: https://github.com/automl/SMAC3.

[14] *AutoSklearn*. 2022. url: https://automl.github.io/auto-sklearn/master/.

Thank You for Your Attention!

MUNI
FACULTY
OF INFORMATICS