

Clarke's 2nd and 3rd Laws

- “Any sufficiently advanced technology is indistinguishable from magic”
- “The only way of discovering the limits of the possible is to venture a little way past them into the impossible.”

Artificial general intelligence

From Wikipedia, the free encyclopedia

Artificial general intelligence (AGI) is the intelligence of a machine that has the capacity to understand or learn any intellectual task that a human being can. It is a primary goal of some artificial intelligence research and a common topic in science fiction and future studies. Some researchers refer to Artificial general intelligence as "**strong AI**",^[1] "**full AI**"^[2], "**true AI**" or as the ability of a machine to perform "general intelligent action";^[3] others reserve "strong AI" for machines capable of experiencing consciousness.

- Even though it still in its infancy, AutoML is well past the “magic” stage as defined by Clarke.
- We argue that AutoML is the *means by which we will achieve approximate AGI*.

AutoML Patent I: Novel Non-Parametric Statistical Behavioral Identification Ecosystem for Electricity Fraud Detection

Abstract

Embodiments of the disclosure are directed towards electricity fraud detection systems that involve a behavioral detection ecosystem to improve the detection rate of electricity fraud while reducing the rate of false-positives.

More specifically, machine learning algorithms are eschewed in favor of two separate models that are applied sequentially.

The first model is directed to improving the detection rate of electricity fraud through the use of detectors to identify customers engaging in suspicious behavior based on the demand profiles of those customers.

The second model is directed to reducing the rate of false-positives by identifying potential legitimate explanations for any suspicious behavior.

Subtracting away the suspicious behavior with legitimate explanations leaves only the identified, unexplained suspicious behavior that is highly likely to be associated with fraudulent activity.

Electricity Fraud Detection

- <https://patentimages.storage.googleapis.com/fe/e3/80/256475eb967399/US20180299495A1.pdf>
- <https://patents.google.com/patent/US20180299495A1/en?inventor=Hussain+Abbas>

Pipe Leak Prediction

- <https://patentimages.storage.googleapis.com/6a/51/18/73b41405b381bd/US20180300639A1.pdf>
- <https://patents.google.com/patent/US20180300639A1/en?inventor=Hussain+Abbas>

MBI Bayesian Causal Inference Workshop at The Ohio State University

- <https://mbi.osu.edu/events/bayesian-causal-inference-workshop>



MATHEMATICAL BIOSCIENCES INSTITUTE



THE OHIO STATE UNIVERSITY
COLLEGE OF ARTS AND SCIENCES

Bayesian Causal Inference Workshop

AutoML Patent II: Novel Autonomous Artificially Intelligent System to Predict Pipe Leaks

Abstract

Embodiments of the disclosure are directed towards pipe leak prediction systems configured to predict whether a pipe (e.g., a utility pipe carrying some substance such as water) is likely to leak.

The pipe leak prediction system may include one or more predictive models based on one or more machine learning techniques, and a predictive model can be trained using data for the characteristics of various pipes in order to determine the patterns associated with pipes without leaks and the patterns associated with pipes with leaks.

A predictive model can be validated, used to construct a confusion matrix, and used to generate insights and inferences associated with the determinant variables used to make the predictions. The predictive model can be applied to data for various pipes in order to predict which of those pipes will leak.

Any pipes that are identified as likely to leak can be assigned for further investigation for potential repair or preventative maintenance.

What is AutoML?

AutoML is the process of automating the end-to-end Machine Learning pipeline.

A Machine Learning pipeline consists of:

- Business problem definition
- Data gathering
- Data cleansing
- Data pre-processing
- Feature extraction
- Feature engineering
- Feature selection
- **Algorithm selection**
- **Hyperparameter optimization**

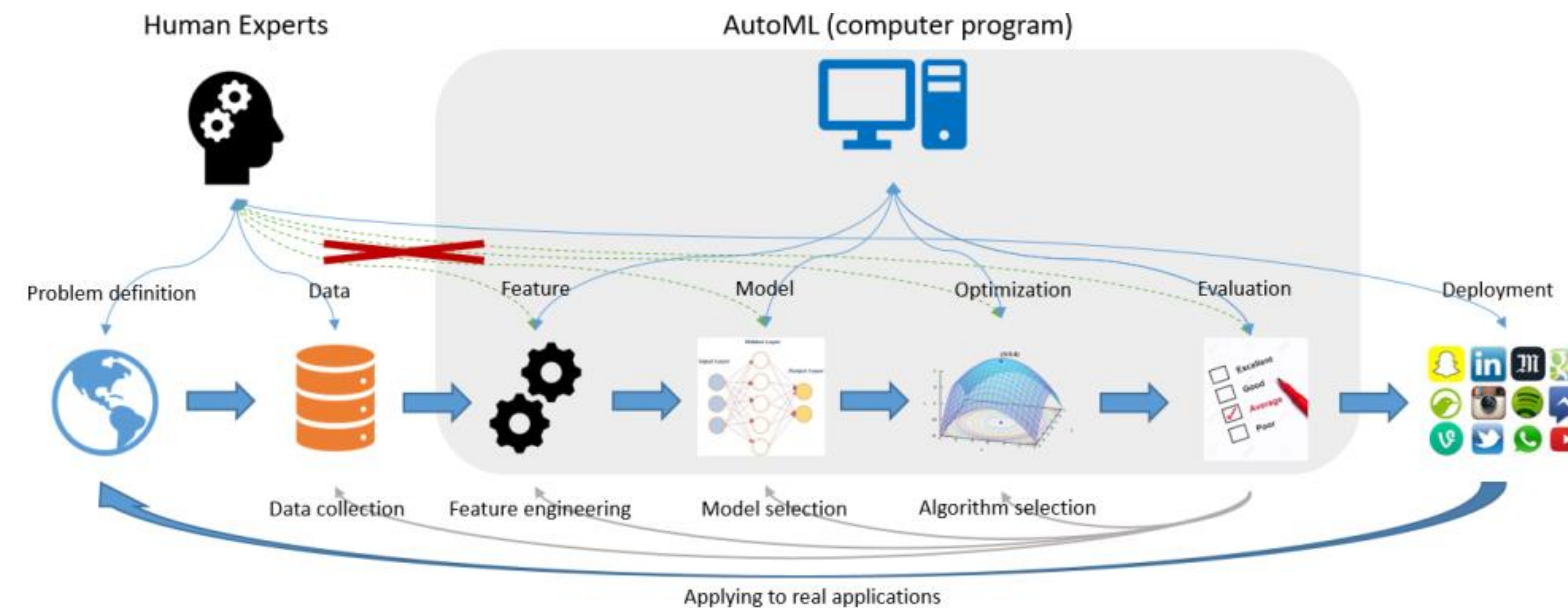


Fig. 1. To use machine learning techniques and obtain good performance, humans usually need to be involved in data collection, feature engineering, model and algorithm selection. This picture shows a typical pipeline of machine learning application, and how AutoML can get involved in the pipeline and minimize participation of humans.

A Taxonomy of Current Approaches to AutoML

Current approaches to configuration selection can be described as a structure consisting of a “malleable” outer and an inner loop:

- The outer loop selects the candidate model and the inner loop searches through the set of hyperparameter choices for the model.
- As this process can be quite computationally burdensome, a method that can reduce this burden is preferred.
- Bayesian Optimization is such a method.

Open-Source AutoML Libraries

Current AutoML frameworks seek to automate a subset of the above tasks.

Some of the most popular frameworks are:

- Auto-WEKA (Java/GUI)
- Auto-SKlearn (Python)
- SmartML (R)
- TPOT (Python)
- Auto-Keras (Python) Neural Networks only

What they use under the hood:

- Bayesian Optimization (BO): Auto-WEKA, Auto-SKlearn
- Neural Architecture Search: Auto-Keras
- Genetic Algorithms: TPOT

How Bayesian Methods Play A Critical Role in AutoML

- Bayesian Optimization (BO) is a derivative-free sequential sample-based optimization (SBO) procedure that can best be described as a fire and forget method analogous to how a homing missile zeroes in on its target.
 - BO solves black-box optimization problems by forming a probabilistic model of the loss function.
 - Example black box problem: Find the configuration (algo + param) which minimizes cross-validation loss.
 - BO is derivative free since it does not require the use of derivatives.
 - This differs from gradient descent, which utilizes gradient information to find a minimum of a loss function.
- Advantages of BO:
 - BO excels in finding optimal solutions in highly conditional spaces (such as algorithm/parameter space) which are computational burdensome to evaluate, such as the CASH problem which Auto-WEKA seeks to solve.
 - BO is particularly useful in situations in which evaluation takes a long time (minutes or hours to evaluate).
- Area of Active Research:
 - BOHB (2018) finds a solution 20x faster than traditional BO (2016) and 55x faster than Random Search (2011).

How Auto-WEKA uses BO under the Hood

- Auto WEKA uses BO under the hood to solve the CASH (Combined Algorithm Selection and Hyperparameter optimization) problem, since BO is capable of iteratively solving it.
- **To kick off the algorithm**, an initial **configuration** (An Algorithm and its corresponding Hyperparameter set) is selected using Meta-Learning:
 - Meta Learning maps configuration performance to historical meta-data (data about data) from past data.
 - Logic: Applying a configuration to datasets with similar meta-data ought to result in similar performance.
- The Algorithm **stops** when it has “found” the **configuration** which minimizes cross-validation loss.

How AutoML is poised to upend the Data Science Industry

- Current State: A typical Data scientist earns a salary of \$120,000 yet spends 80% of their time gathering and cleansing data and only 20% of their time conducting actual science.
- Future State: AutoML obviates the issue and changes the game by enabling data scientists to go deeper and broader than hitherto otherwise without increasing labor costs.
- More importantly, AutoML will enable SMB's to participate in the benefits of AI
- Rather than spending time writing code to create models, the data scientist now simply leapfrogs off what the AutoML has built!
- With this newfound time, the data scientist is now free to:
 - Focus on interpreting and communicating the AutoML output to stakeholders, thereby enabling better decision making.
 - Iterate faster through the pipeline of planned projects and deliver results.

$$A_{\lambda^*}^* \in \operatorname{argmin}_{A^{(j)} \in \mathcal{A}, \lambda \in \Lambda^{(j)}} \frac{1}{k} \sum_{i=1}^k \mathcal{L}(A_{\lambda}^{(j)}, \mathcal{D}_{\text{train}}^{(i)}, \mathcal{D}_{\text{test}}^{(i)}) \rightarrow$$

High level Overview of BO

- Assume that the loss function follows some prior distribution.
- Your data are your “function evals” or model runs with a specific configuration.
- Using the posterior distribution, use an “acquisition function” to figure out which configuration to run next.

Examples of BO Optimizers

- SMAC
- TPE
- Gaussian Processes
- BOHB

How to kickoff Bayesian Optimization

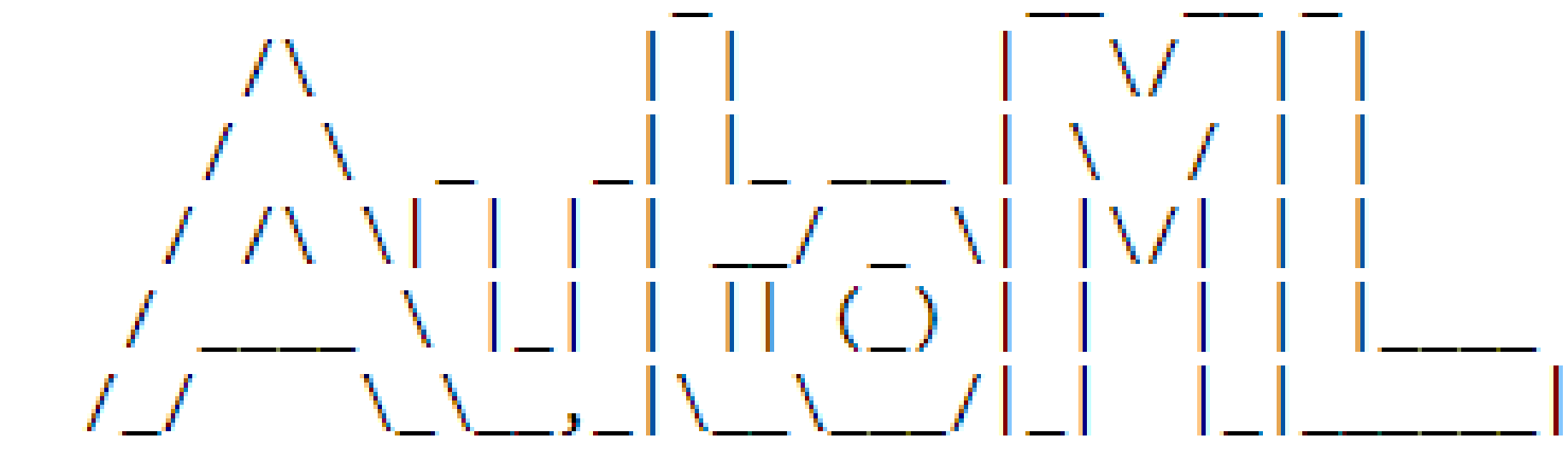
- Random configurations – test everything approach.
- Meta-learning – takes the place of a domain expert in defining the best starting point.
Uses past experience to guide where to start.
- Literally teleporting closer to the needle by “standing on the shoulders of giants”.

- Choose an algorithm based on dataset characteristics, e.g. for the Iris dataset this could be an SVM
- Manual tuning -> fiddling with hyperparameters.
- Better: Use automated methods like PSO, GA or SMBO
- Best: AutoWeka

Metafeatures



- # training examples: 150
- # classes: 3
- # features: 4
- # numerical features: 4
- # categorical features: 0
- missing values? No



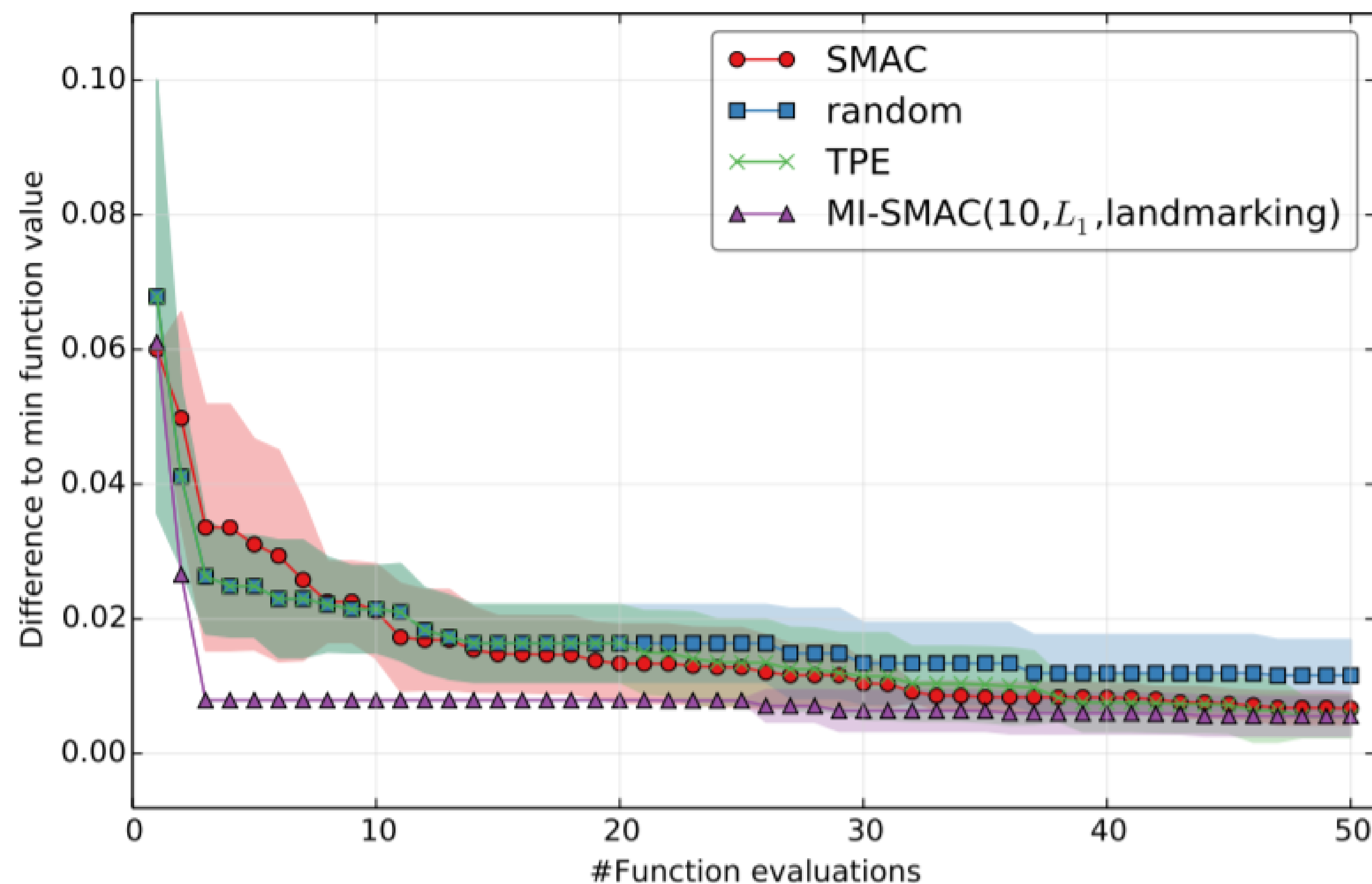
AutoML: taking the human expert out of the loop

Distance Metric on Metafeatures
 “Manhattan Distance”

Commonly used in literature, the L_1 norm:

<https://ml.informatik.uni-freiburg.de/papers/14-METASEL-MI-SMBO-slides.pdf>

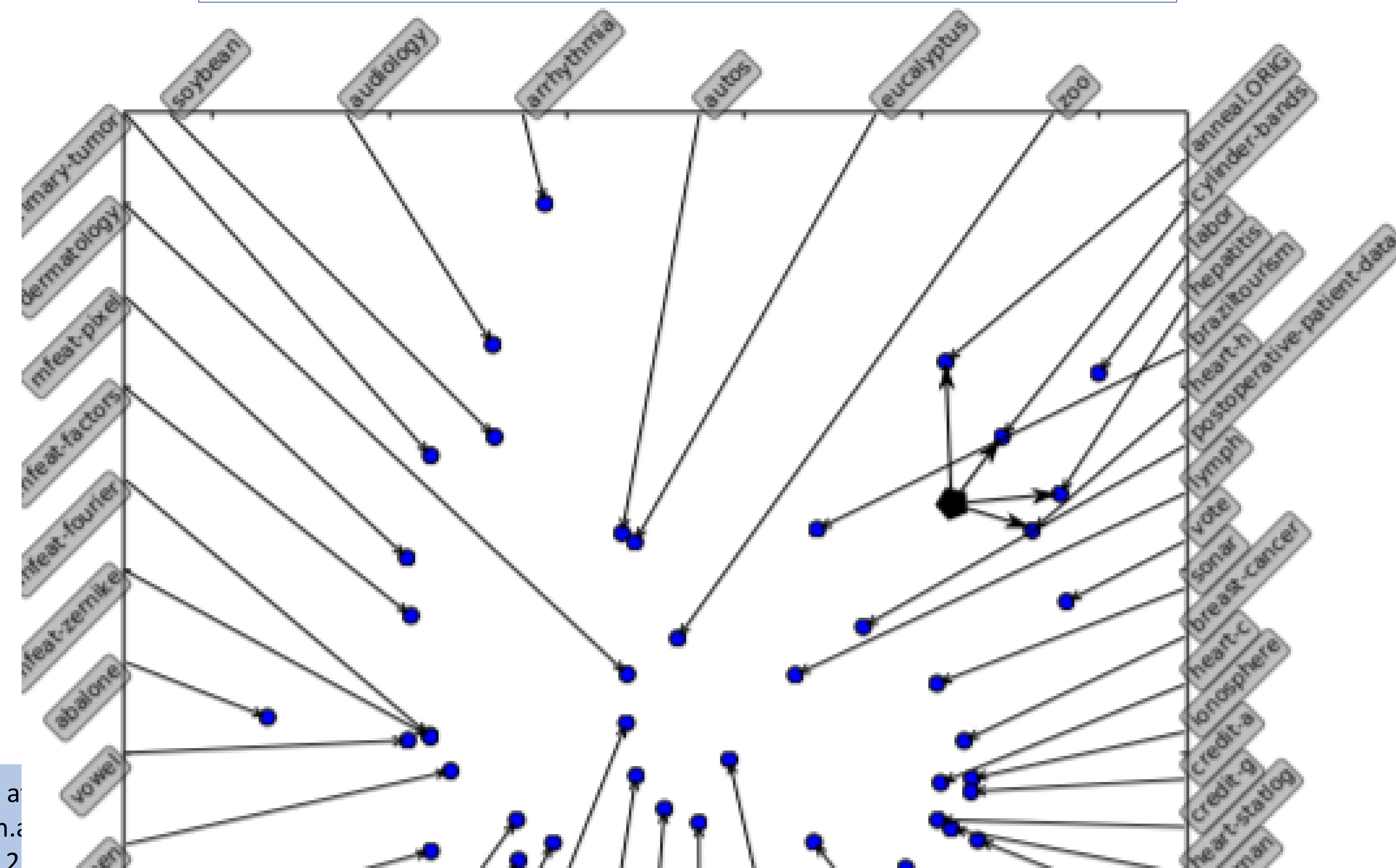
AutoSklearn: Results (1)



How to choose the best starting configurations.

For a new dataset D_{new} :

- Sort known datasets $D_{1:N}$ by distance to D_{new} .
- For each of these datasets, extract the best known hyperparameter configuration $\lambda_{D_i}^*$.
- Initialize SMBO with the first k hyperparameter configurations from the sorted list.



	SmartML	Auto-Weka	AutoSklearn	TPOT
Language	R	Java	Python	Python
API	Yes	No	No	Yes
Optimization Procedure	Bayesian Optimization (SMAC)	Bayesian Optimization (SMAC and TPE)	Bayesian Optimization (SMAC)	Genetic Programming, and Pareto Optimization
Number of Algorithms	15 classifiers on top of R	27 classifiers on top of WEKA	15 classifiers on top of scikit learn	15 classifiers on top of scikit learn
Support Ensembling	Yes	Yes	Yes	No
Use Meta-Learning	Yes (incrementally updated KB)	No	Yes (Static)	No
Feature preprocessing	Yes	Yes	Yes	No
Model Interpretability	Yes	No	No	No

Table 1: Comparison between State-of-the-art Automated Machine Learning Frameworks

HPO faces several challenges which make it a hard problem in practice:

- Function evaluations can be extremely expensive for large models (e.g., in deep learning), complex machine learning pipelines, or large datasets.
- The configuration space is often complex (comprising a mix of continuous, categorical and conditional hyperparameters) and high-dimensional. Furthermore, it is not always clear which of an algorithm's hyperparameters need to be optimized, and in which ranges.
- We usually don't have access to a gradient of the loss function with respect to the hyperparameters. Furthermore, other properties of the target function often used in classical optimization do not typically apply, such as convexity and smoothness.
- One cannot directly optimize for generalization performance as training datasets are of limited size.

- [Sources:](#)
- https://www.automl.org/wp-content/uploads/2019/05/AutoML_Book.pdf
- https://openproceedings.org/2019/conf/edbt/EDBT19_paper_235.pdf

Classification Algorithm	Categorical parameters	Numerical parameters	Package
SVM	1	4	e1071
NaiveBayes	0	2	klaR
KNN	0	1	FNN
Bagging	0	5	ipred
part	1	2	RWeka
J48	1	2	RWeka
RandomForest	0	3	randomForest
c50	3	2	C50
rpart	0	4	rpart
LDA	1	1	MASS
PLSDA	1	1	caret
LMT	0	1	RWeka
RDA	0	2	klaR
NeuralNet	0	1	nnet
DeepBoost	1	4	deepboost

Table 3: Integrated Classifier Algorithms

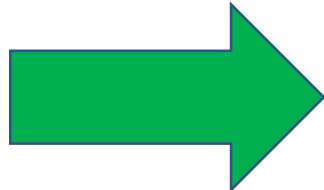
Auto-WEKA¹ addresses this problem by treating all of WEKA as a single, highly parametric machine learning framework, and using Bayesian optimization to find a strong instantiation for a given dataset. Specifically, it considers the combined space of WEKA's learning algorithms $\mathcal{A} = \{A^{(1)}, \dots, A^{(k)}\}$ and their associated hyperparameter spaces $\Lambda^{(1)}, \dots, \Lambda^{(k)}$ and aims to identify the combination of algorithm $A^{(j)} \in \mathcal{A}$ and hyperparameters $\lambda \in \Lambda^{(j)}$ that minimizes cross-validation loss,

$$A_{\lambda^*}^* \in \operatorname{argmin}_{A^{(j)} \in \mathcal{A}, \lambda \in \Lambda^{(j)}} \frac{1}{k} \sum_{i=1}^k \mathcal{L} \left(A_{\lambda}^{(j)}, \mathcal{D}_{\text{train}}^{(i)}, \mathcal{D}_{\text{test}}^{(i)} \right),$$

where $\mathcal{L}(A_{\lambda}, \mathcal{D}_{\text{train}}^{(i)}, \mathcal{D}_{\text{test}}^{(i)})$ denotes the loss achieved by algorithm A with hyperparameters λ when trained on $\mathcal{D}_{\text{train}}^{(i)}$ and evaluated on $\mathcal{D}_{\text{test}}^{(i)}$. We call this the *combined algorithm selection and hyperparameter optimization (CASH)* problem. CASH can be seen as a blackbox function optimization problem: determining $\operatorname{argmin}_{\theta \in \Theta} f(\theta)$, where each configuration $\theta \in \Theta$ comprises the choice of algorithm $A^{(j)} \in \mathcal{A}$ and its hyperparameter settings $\lambda \in \Lambda^{(j)}$. In this formulation, the hyperparameters of algorithm $A^{(j)}$ are *conditional* on $A^{(j)}$ being selected. For a given θ representing algorithm $A^{(j)} \in \mathcal{A}$ and hyperparameter settings $\lambda \in \Lambda^{(j)}$, $f(\theta)$ is then defined as the cross-validation loss $\frac{1}{k} \sum_{i=1}^k \mathcal{L}(A_{\lambda}^{(j)}, \mathcal{D}_{\text{train}}^{(i)}, \mathcal{D}_{\text{test}}^{(i)})$.²

Source: <https://www.cs.ubc.ca/labs/beta/Projects/autoweka/papers/16-599.pdf>

Why AutoWEKA uses SMAC instead of GP or TPE

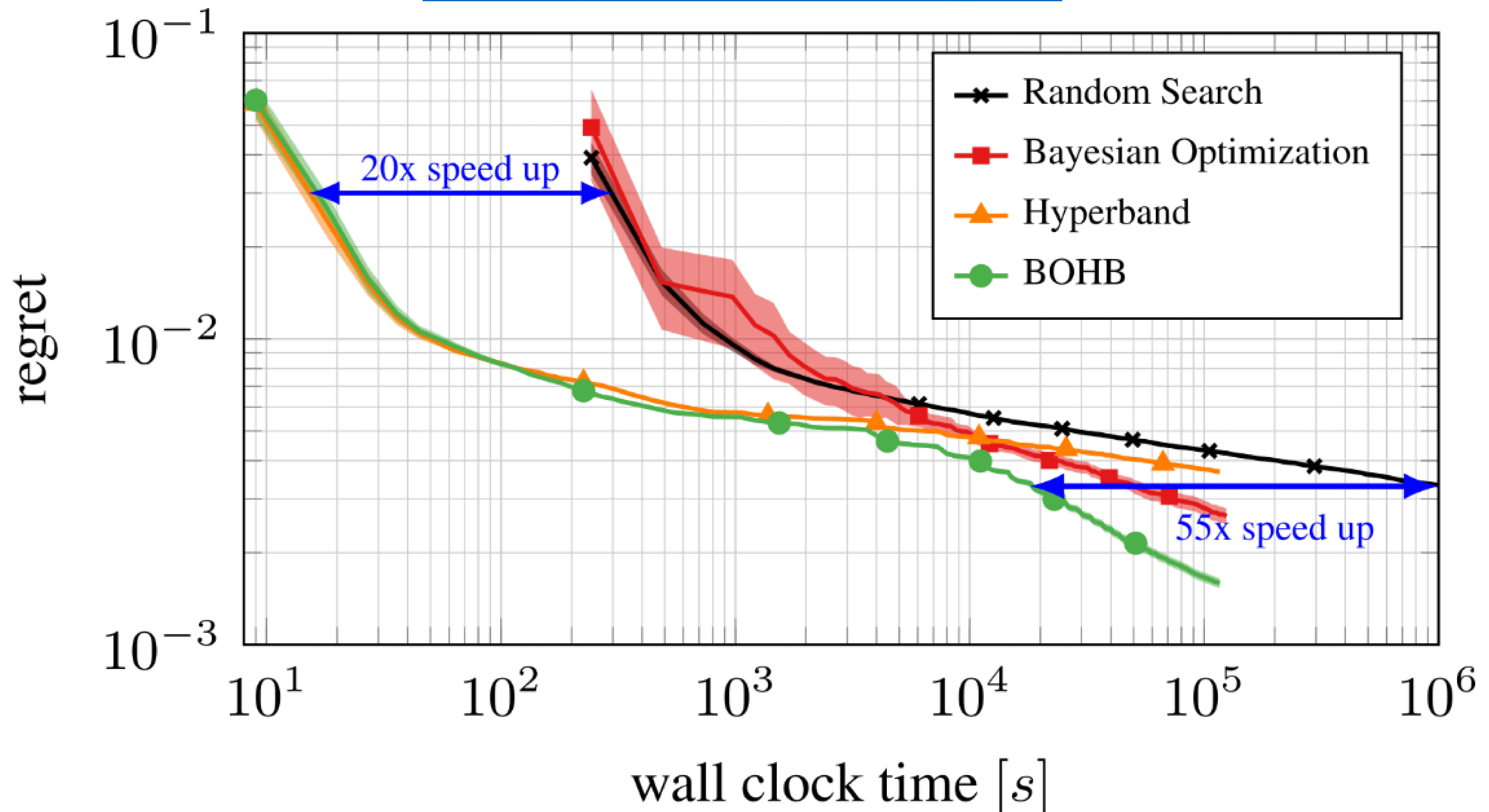


Bayesian optimization (see, e.g., Brochu et al., 2010), also known as sequential model-based optimization, is an iterative method for solving such blackbox optimization problems. In its n -th iteration, it fits a probabilistic model based on the first $n - 1$ function evaluations $\langle \theta_i, f(\theta_i) \rangle_{i=1}^{n-1}$, uses this model to select the next θ_n to evaluate (trading off exploration of new parts of the space *vs* exploitation of regions known to be good) and evaluates $f(\theta_n)$. While Bayesian optimization based on Gaussian process models is known to perform well for low-dimensional problems with numerical hyperparameters (see, e.g., Snoek et al., 2012), tree-based models have been shown to be more effective for high-dimensional, structured, and partly discrete problems (Eggenberger et al., 2013), such as the highly conditional space of WEKA's learning algorithms and their corresponding hyperparameters we face here.³ Thornton et al. (2013) showed that tree-based Bayesian optimization methods yielded the best performance in Auto-WEKA, with the random-forest-based SMAC (Hutter et al., 2011) performing better than the tree-structured Parzen estimator, TPE (Bergstra et al., 2011). Auto-WEKA uses SMAC to determine the classifier with the best performance on the given data.

Source: <https://www.cs.ubc.ca/labs/beta/Projects/autoweka/papers/16-599.pdf>

search, but that is very inefficient. This post presents BOHB, a new and versatile tool for hyperparameter optimization which comes with substantial speedups through the combination of Hyperband with Bayesian optimization. The following figure shows an exemplary result: BOHB starts to make progress much faster than vanilla Bayesian optimization and random search and finds far better solutions than Hyperband and random search.

Source: https://www.automl.org/blog_bohb/



Example results of applying BOHB (freely available under <https://github.com/automl/HpBandSter>) to optimize six hyperparameters of a neural network. The curves show the immediate regret of the best configuration found by the methods as a function of time. BOHB combines the best of BO and Hyperband: it yields good performance (20x) faster than BO in the beginning, and converges much faster than Hyperband in the end.

HPOLIB

Our **publicly available** software wraps various benchmarks and the Bayesian optimizers SMAC, Spearmint and TPE. This is just a first step towards a canonical benchmark library. You can help us:

- **Contribute your algorithm**
to add another benchmark for future research
- **Include your optimizer**
to find out its strengths and weaknesses

Source:

<https://www.automl.org/automl/hpolib/>

SMAC (Sequential Model-based algorithm configuration) is based on random forests and can handle continuous, discrete and conditional hyperparameters.

[Hutter, Hoos, and Leyton-Brown, 2011]

Spearmint uses Gaussian Process (GP) models and performs slice sampling over the GP's hyperparameters. Can handle continuous and discrete hyperparameters.

[Snoek, Larochelle, and Adams, 2012]

TPE (Tree Parzen Estimator) is based on Gaussian Mixture Models. Supports conditional, continuous and discrete parameters and also priors over them.

[Bergstra, Bardenet, Bengio, and Kégl, 2011]

Algorithm for BO via GP

Algorithm 1 Basic pseudo-code for Bayesian optimization

Place a Gaussian process prior on f

Observe f at n_0 points according to an initial space-filling experimental design. Set $n = n_0$.

while $n \leq N$ **do**

 Update the posterior probability distribution on f using all available data

 Let x_n be a maximizer of the acquisition function over x , where the acquisition function is computed using the current posterior distribution.

 Observe $y_n = f(x_n)$.

 Increment n

end while

Return a solution: either the point evaluated with the largest $f(x)$, or the point with the largest posterior mean.

The acquisition function is like your “compass”. It tells you which configuration you should check next.

Expected Improvement is the usual Acquisition function that is used in practice, but there are an infinite number of acquisition functions.

Source: <https://arxiv.org/pdf/1807.02811.pdf>