AutoML Review

Cohen Leonardo
lkoen@uth.gr

Peteinarelis Charalampos
pcharalampos@.uth.gr

Panagiotidis Ioannis
panioannis@uth.gr

Abstract

Machine learning has proved successful in a variety of scientific fields and industries. Despite that, the challenges faced when selecting an ML algorithm remain unsolved. The choice of method and its corresponding hyperparameters still remains an unsolved problem today. Because such a task is the objective of only experts of the field but widespread adoption requires the use of ML by non-experts, it is imperative to create algorithms that automate such process. The automation of the hyperparameter tuning procedure is widely used and this led to the idea of the method choice automation. This is the so called Auto ML, which is an automated method that builds optimal ML pipelines with given data chooses the optimal algorithm and also performs hyperparameter tuning. The latter belongs among others to the CASH (Combined Algorithm Selection and Hyperparameter Optimization Algorithms).

1. Introduction

In recent years, machine learning and neural network methods have reached remarkable effectiveness on industries such as finance as well as energy and have seen a fast increase in their popularity. This success was only possible due to decades of machine learning (ML) research into many aspects of the field, ranging from learning strategies to new architectures. Such architectures are used to create pipelines of data processing that perform necessary preprocessing and then train a machine learning model with such data and predefined hyperparameters. Due to the overwhelming variety of methods and schemes it became essential to seek algorithms that further automate the pipeline design process.[3]

Such algorithms used to focus on hyperparameter tuning but are now turning to the optimization of the machine learning algorithm selection as well as the selection of data preprocessing steps.[4] The length and difficulty of ML research prompted a new field, named AutoML, that aims to automate such progress by spending machine compute time instead of human research time. This endeavor has been fruitful but, so far, modern studies have employed constrained search spaces heavily reliant on human design.

2. Definition of the Pipeline Optimization Problem

An ML pipeline is a sequential combination of various algorithms that transforms a feature vector $x \in X^d$ into a target value $y \in Y$, i.e a class label for a classification problem, or a Numerical Value for a regression problem. These algorithms can be grouped in three sets: data preparation, feature engineering, and modeling (model generation).

For each step a finite set of Algorithms is available $\{A_{prep}, A_{feature}, A_{model}\}$ as well as their corresponding hyperparameters λ.

Therefore the Machine Learning Pipeline is defined as a directed acyclic graph (DAG) , where each node represents an algorithm as defined above along with its corresponding hyperparameters. The Pipeline is denoted as

$$Pipeline = P_{g,A,\lambda} \tag{1}$$

Where g is the Pipeline Structure A is the vector of Algorithms and λ the Vector of Hyperparameters corresponding to each algorithm.

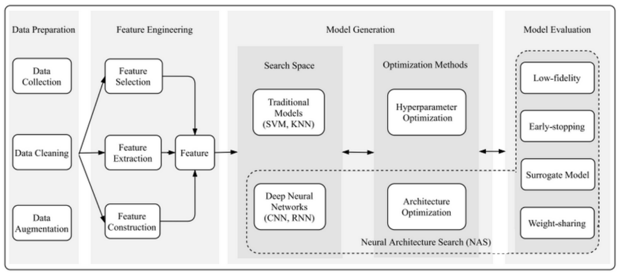

Figure 1. Ml Pipeline

Since the ML Pipeline is defined we should now define a cost or objective function as a performance indicator. This is no different than standard performance metrics but is tailored to the pipeline optimization problem. Let a pipeline $P_{g,A,\lambda}$ be given, a feature vector $x \in X^d$ as well as $y \in Y$ the desired outputs. We denote performance of the pipeline P as the Expected Error of the Loss function $L(\cdot,\cdot)$:

$$\pi = \frac{1}{n}\sum_{i=1}^{n} L(\hat{y}_i, y_i) \tag{2}$$

Where $\hat{y}_i$ is the predicted output of the Pipeline with input x_i

The loss function depends on whether we are dealing with a Regression or Classification ML problem.

We have therefore formulated the problem that AutoMl tries to solve; We search for the optimal parameters i.e. Algorithms , their hyperparameters and their dependency structure to minimize our cost.

3. Redundant approaches

It is usual practice to attempt to optimize such pipelines manually and iteratively. This has been the work of Ml engineers and researchers alike. There have also been attempts to find systematic solutions to such equations though a closed form solution is not possible nor are gradient -based or standard optimization techniques as the input data affect the solution in an unpredictable manner.

4. New directions and Approaches

The first task for building an ML pipeline is creating the pipeline structure. Then follows the selection of algorithm for every step of the pipeline and then the tuning of its hyperparameters. The latter two are reffered to as the CASH Problem.

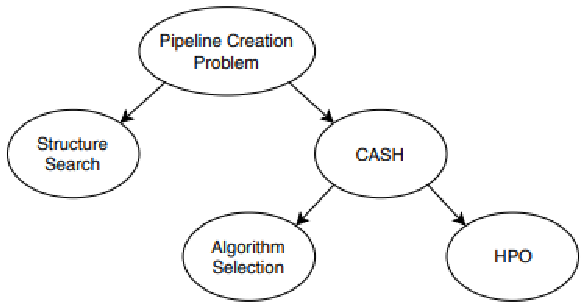

Figure 2. Pipeline Creation Problem

5. Pipeline Structure Creation

This topic has received a lot of attention in the context of designing neural networks referred to as architecture search. Regarding classical machine learning, this topic was tackled for various specific problems like natural language processing. Surprisingly, basically no publications exist treating general pipeline construction. Yet, common best practices suggest a basic ML pipeline layout.1 At first, the input data is cleaned in multiple distinct steps, like imputation of missing data and one-hot encoding of categorical input. It also creates new Next, relevant features are selected and new features created in a feature engineering stage. This stage highly depends on the underlying domain. Finally, a single model is trained on the previously selected features.

6. Definition of the CASH Problem

We first review the formalization of AutoML as a Combined Algorithm Selection and Hyperparameter optimization (CASH) problem . Two important problems in AutoML are that (1) no single machine learning method performs best on all datasets and (2) some machine learning methods (e.g., non-linear SVMs) crucially rely on hyperparameter optimization. Instead of selecting an algorithm first and optimizing its hyperparameters later, both steps are executed simultaneously. The two problems can efficiently be tackled as a single, structured, joint optimization problem:[3]

Definition 2 (CASH) Let A be a set of algorithms, and let the hyperparameters of each algorithm be λ. We want to find the optimal Algorithms given our architecture as well as their corresponding optimal parameters. This defines a joint search space of both hyperparameters and algorithms called the Configuration space.

Further, let $D_{train} = \{(x_1, y_1), ..., (x_n, y_n)\}$ be a training set which is split into K cross-validation folds. Finally, let $L(A_\lambda^{(j)}, D_{train}^{(i)}, D_{valid}^{(i)})$ be the loss that algorithm $A^{(j)}$ achieves on $D_{valid}^{(i)}$ valid when trained on $D_{train}^{(i)}$ with hyperparameters λ. Then, the Combined Algorithm Selection and Hyperparameter optimization (CASH) problem is to find the joint algorithm and hyperparameter setting that minimizes this loss:

$$\mathrm{A}^*, \lambda_* \,\epsilon\, argmin\, \frac{1}{K}\sum_{i=1}^{K} L(A_l^{(j)}, D_{train}^{(i)}, D_{valid}^{(i)}))$$

An exhaustive search of this space is not feasible, but now we have defined the space that we want to optimize 3

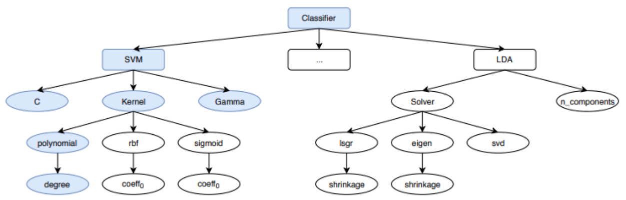

Figure 3. Configuration Space

6.1. Methods for Structure Search optimization

- Genetic Programming
- Hierarchical Planning
- Self Play

6.2. Methods for CASH optimization

- Grid search
- Random Search
- Sequential Model-Based Optimization
- Gaussian Process
- Random Forests
- Tree Structured Parzen Estimator
- Evolutionary Algorithms
- Genetic Programming
- Particle Swarm Optimization
- Multi-Armed Bandit Learning
- Gradient Descent

7. Data Preprocessing

Automated Data cleaning

Data cleaning is the initial part of the pipeline and an important aspect of building an ML pipeline. The purpose of data cleaning is improving the quality of a data set by removing data errors. Common error classes are missing values in the input data, invalid values or broken links between entries of multiple data sets. Most existing AutoML frameworks recognize the importance of data cleaning and include various data cleaning stages in the fitted ML pipeline.[8]

However, these data cleaning steps are usually hard-coded and not generated based on some metric during an optimization period. These fixed data cleaning steps usually contain imputation of missing values, removing of samples with incorrect values, like infinity, or outliers and scaling features to a normalized range. Sometimes, high requirements for specific data qualities are introduced by later stages in an ML pipeline. For example, consider categorical features in a classification task. If a SVM is used for classification, all categorical features have to be numerically encoded. In contrast, a random forest is able to handle categorical features. These additional requirements can be detected by analyzing a candidate pipeline and matching the prerequisites of every stage with meta-features of each feature in the data set.

All currently used data cleaning steps are domain-agnostic. Consequently, these general purpose steps are not able to catch domain specific errors. However, by incorporating domain knowledge, the data quality can be heavily increased. Using different representations of expert knowledge, like integrity constraints or first order logic, low quality data can be automatically detected and corrected . However, these potentials are not used by current AutoML frameworks as they aim to be completely data-agnostic to be applicable to a wide range of data sets. Consequently, advanced and domain specific data cleaning is conferred to the user.[7]

8. Feature Engineering

Feature engineering is the process of generating and selecting features from a given data set for the subsequent modeling step. This step is crucial for the complete ML pipeline, as the overall model performance highly depends on the available features. By building good features, the performance of an ML pipeline can be increased many times over. Yet, having too many unrelated feature increases the training time (curse of dimensionality) and could also lead to overfitting. The task of feature engineering is highly domain specific and very difficult to generalize. Even for a data scientist assessing the impact of a feature is difficult, as domain knowledge is necessary.[3]

Consequently, feature engineering is a mainly manual and time-consuming task driven by trial and error. Feature selection is the easier part of feature engineering as the search space is very limited: each feature can either be included or excluded. By removing redundant or misleading features, an ML pipeline can be trained faster with a lower generalization error. Furthermore, the interpretability of the trained model is increased. By examining the features from an information theoretic or statistics viewpoint no domain knowledge is necessary for feature selection.

Therefore, many different algorithms for feature selection exist—like univariate selection, variance threshold, feature importance or correlation matrices and are already integrated in modern AutoML frameworks. More interesting is the task of feature generation as the number of new features that can be derived from a given data set is unbounded. As being domain-agnostic is a core goal of AutoML, domain knowledge cannot be used for feature generation. Approaches to enhance automatic feature generation with domain knowledge,[6] are therefore not considered. Still, some features—like dates or addresses—can be easily parsed without domain knowledge to create more meaningful features. However, for arbitrary numeric or categorical features such rule-based feature generation is not applicable requiring different measures

9. Ensemble Learning

A well-known concept in ML is ensemble learning. Ensemble methods combine multiple ML models to create predictions. Depending on the diversity of the combined models, the overall accuracy of the predictions can be significantly increased. The cost of evaluating multiple ML models is often neglectable considering the performance improvements. During the search for a well performing ML pipeline, AutoML frameworks create a large number of different pipelines. Instead of only yielding the best performing configuration, the set of best performing configurations can be used to create an ensemble. [1]

As a consequence, AutoML procedures are capable of yielding a better overall performance. An interesting approach for ensemble learning is stacking . A stacked ML pipeline is generated in multiple layers, each layer being a normal ML pipeline.[10] The predicted output of each previous layer is appended as a new feature to the training data of subsequent layers. This way, later layers have the chance to correct wrong predictions of earlier layers. In the end, all predictions are combined via bagging [3].

10. Meta-Learning

Given a new unknown ML task, AutoML methods usually start from scratch to build an ML pipeline. However, a human data scientist does not always start all over again but learns from previous tasks. Meta-learning is the science of learning how ML algorithms learn. Based on the observation of various configurations on previous ML tasks, meta-learning builds a model to construct promising configurations for a new unknown ML task leading to faster convergence with less trial and error . [9]

Meta-learning can be used in multiple stages of automatically building an ML pipeline to increase the efficiency: Search Space Refinements All presented CASH methods require an underlying search space definition. Often these search spaces are chosen arbitrary without any validation leading to either bloated spaces or spaces missing well-performing regions.

In both cases the AutoML procedure is unable to find optimal results. Meta-learning can be used to asses the importance of single hyperparameters allowing to remove unimportant hyperparameters from the configuration space . Filtering of Candidate Configurations Many AutoML procedures generate multiple candidate configurations usually selecting the configuration with the highest expected improvement. Meta-learning can be used to filter empirically bad performing candidate configurations . Consequently, the risk of superfluous configuration evaluations is minimized. Warm-Starting Basically all presented methods have an initialization phase where random configurations are selected. However, the same methods as for filtering candidate configurations can be applied to initialization.

Warm-starting can be used for many aspects of AutoML, yet most research focuses on model selection and tuning. Pipeline Structure Meta-learning is also applicable for pipeline structure search. Using information which preprocessor and model combination perform well together, potentially better performing pipelines can obtain a higher ranking.

11. Existing Frameworks

This section provides an table to the most popular open-source CASH frameworks. At first, implementations of CASH algorithms are presented and analyzed.

Algorithm	Solver	Λ	Parallel	Timeout
Grid Search	Grid Search	no	Local	no
Random Search	Random Search	no	Local	no
SPEARMINT	SMBO with Gaussian process	no	Cluster	no
RoBO	SMBO with various models	no	no	no
BTB	Bandit learning and Gaussian process	yes	no	no
HYPEROPT	SMBO with TPE	yes	Cluster	no
SMAC	SMBO with random forest	yes	Local	yes
BOHB	Bandit learning and TPE	yes	Cluster	yes
OPTUNITY	Particle Swarm Optimization	yes	Local	no

Figure 4. Existing CASH Frameworks

12. AutoML Frameworks

These are frameworks for creating complete ML pipelines, where the CASH optimizer used is mentioned as well.

Framework	CASH Solver	Structure	Ensem.	Meta	Parallel	Timeout
ML-PLAN	Grid Search	Variable	no	no	Local	no
TPOT	Genetic Prog.	Variable	no	no	Local	yes
HYPEROPT-SKLEARN	HYPEROPT	Fixed	no	no	no	yes
AUTO-SKLEARN	SMAC	Fixed	yes	yes	Cluster	yes
ATM	BTB	Fixed	no	no	Cluster	no
AUTO_ML	Grid Search	Fixed	yes	no	Local	no

Figure 5. Existing Integrated AutoMl Frameworks

13. Auto-Sklearn

Auto-sklearn's machine learning pipeline comprises of 15 classification algorithms, 14 preprocessing methods, and 4 data preprocessing methods. We parameterized each of them, which resulted in a space of 110 hyperparameters. Most of these are conditional hyperparameters that are only active if their respective component is selected. We note that SMAC can handle this conditionality natively.

All 15 classification algorithms in Auto-sklearn are as follows: They fall into different categories, such as general linear models (2 algorithms), support vector machines (2), discriminant analysis (2), nearest neighbors (1), naïve Bayes (3), decision trees (1) and ensembles (4). In contrast to Auto-WEKA , we focused our configuration space on base classifiers and excluded meta-models and ensembles that are themselves parameterized by one or more base classifiers. While such ensembles increased Auto-WEKA's number of hyperparameters by almost a factor of five (to 786), Auto-sklearn "only" features 110 hyperparameters. [5]

14. Discussion and Future Research Directions

Currently, AutoML is completely focused on supervised learning. Furthermore, all existing high-level AutoML frameworks only support classification and regression tasks. The majority of all publications currently treats the CASH problem either by introducing new solvers or adding performance improvements to existing approaches. A possible explanation could be that CASH is completely domain-agnostic and therefore comparatively easier to automate. However, CASH is only a small piece of the puzzle to build an ML pipeline automatically.

So far, researchers have focused on a single point of the pipeline creation process. Combining dynamically shaped pipelines with automatic feature engineering and sophisticated CASH methods has the potential to beat the currently available frameworks. However, the complexity of the search space is raised to a whole new level probably requiring new methods for efficient search. Nevertheless, the long term goal should be automatically building complete pipelines with every single component optimized. AutoML aims to completely automate the creation of an ML pipeline to enable domain expert to use ML. Except very few publications, current AutoML algorithms are designed as a black-box. [2]

15. Conclusion

In this review, we have introduced the notation of the pipeline creation problem as a optimization problem and presented various methods for automating each step of creating an ML pipeline . Current state-of-the-art frameworks enable users building reasonable well performing ML pipelines without knowledge about ML or statistics. Seasoned data scientists can profit from the automation of tedious manual tasks, especially model selection and Hyperparameter optimization. However, automatically generated pipelines are not able to beat human experts yet. It is likely, that AutoML will continue to be a hot research topic leading to even better, holistic AutoML frameworks in the near future.

References

[1] M. M. Alexandre Lacoste, Hugo Larochelle and F. Laviolette. Sequential model-based ensemble optimization. *In arXiv preprint arXiv:1402.0796*, 2014.

[2] M. Feurer and F. Hutter. Towards further automation in automl. *International Conference on Machine Learning AutoML Workshop*, 2018.

[3] M. F. H. Marc-Andre Zoller. Gsurvey on automated machine learning. *arXiv:1904.12054v1*, 2019.

[4] N. S. Martin Wistuba and L. Schmidt-Thieme. Hyperparameter search space pruning - a new component for sequential model-based hyperparameter optimization. in pages 104–119. *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2015.

[5] K. E. J. T. S. M. B. Matthias Feurer, Aaron Klein and F. Hutter. Chapter 6 auto-sklearn: Efficient and robust automated machine learning. 2019.

[6] R. W. Micah J. Smith and K. Veeramachaneni. Featurehub: Towards collaborative data sciencepages 590–600. *In IEEE International Conference on Data Science and Advanced Analytics*, 2017.

[7] D. Pyle. Data preparation for data mining. *Morgan Kaufmann Publishers, Inc.*, 1999.

[8] E. Rahm and H. H. Do. Data cleaning: Problems and current approaches. *IEEE Data Engineering Bulletin*, 2000.

[9] J. Vanschoren. Meta-learning. in automatic machine learning: Methods, systems, challenges, pages 39–68. *Springer,*, 2018.

[10] D. H. Wolpert. Stacked generalization 241–259. *Neural Networks*, 1992.