Recommending Machine Learning Pipelines Based on Cumulative Metadata

Maxim Aliev, Sergey Muravyov
NRU ITMO
St.Petersburg, Russian Federation
imaxaliev@gmail.com, mursmail@gmail.com

Abstract—**The problem of automated machine learning pipeline design for a given supervised learning task is usually solved by various optimization methods. However, this entails high time complexity. There is a solution called meta-learning, which consists in training a certain model with metadata of the results of solving similar problems [1]. Nevertheless, this approach also has a limitation: the need for a large amount of knowledge to achieve high efficiency of the model. Based on the literature analyzed by the authors, this problem still remains relevant. In particular, auto-sklearn, one of the most popular solutions, uses a set of metadata that is predetermined and does not change based on new run results [2]. The ontological data model proposed by the authors, together with the mechanism of automated knowledge enrichment, are designed to reduce the impact of the above restriction. Currently, the pipeline recommendation process includes two scenarios: the scenario of having a hash representation of the original data set in storage; the reverse scenario, in which the pipeline is recommended based on Bayesian optimization over the global space of machine learning algorithms and their associate hyper-parameters. As part of the experiment, the pipeline inference time was measured for both scenarios. The results confirmed the superiority of the metadata-driven recommendation and the increase in this advantage as the dimension of the input data increased.**

I INTRODUCTION

There are many various approaches to solving the problem of automated machine learning (ML) pipeline design, including, but not limited to: evolutionary optimization, reinforcement learning, Bayesian optimization, hierarchical planning and others [3]. For analysis, we divide them into two categories: those based solely on optimization and those that reuse existing knowledge (in most cases in the form of metadata for warm start optimization). For approaches from the first category, the search is carried out in the global space based on the improvement in the pipeline quality metric, meanwhile for approaches from the second category, metadata allows to indicate the optimization starting point and thereby narrowing the search space or even composing it from scratch. The advantage of approaches from the second category is the low time complexity of the pipeline search process, which in most cases is achieved through the meta-learning mechanism, i.e. training some model on the results of solving similar problems. Combined with some optimization strategy it can be a powerful tool, because it can reduce the search space dozens or even hundreds of times. The existing limitations for successful adaptation of meta-learning are: the level of generalization of the model and the amount of the training instances. In this paper, an attempt is made to eliminate the latter limitation by designing a knowledge base and a mechanism for its automated population.

II. COMBINED ALGORITHM SELECTION AND HYPER-PARAMETER OPTIMIZATION

First of all, we define a method for algorithm selection and hyper-parameter tuning based on optimization for the scenario of no result in the knowledge base. To do this, we analyze the following candidates: grid and random search; Bayesian, evolutionary, early stopping-based and gradient-based optimization; and reinforcement learning.

1) Grid search. Represents a brute-force search in the given parameter space. For effective use, manual setting of bounds and discretization of parameters may be required. The drawback is that it suffers from the curse of dimensionality.

2) Random search. Randomly selects a combination of parameters from a given search space. By default, it is applicable for both discrete and continuous (including mixed) parameter spaces. It can be more efficient than the previous method when only a small number of parameters affect the quality of the ML algorithm.

3) Bayesian optimization. It is a method for the global optimization of an unknown function with noise. The algorithm of work is to build a probabilistic model and iteratively update it in search of the optimum.

4) Evolutionary optimization. It is yet another method for the global optimization of a black-box function with noise. In the first stage the population of individuals (candidate pipelines) is selected randomly. Next, all candidates are evaluated using the fitness function. Then, the top best candidates are chosen for reproduction, which takes place by means of cross-over and mutation operators. Finally, the least-fit candidates from the previous population are substituted by new ones and the process is repeated (from the second stage) until termination.

5) Early stopping-based optimization. Specifically designed for large search spaces. One of the implementations is the successive halving algorithm, which in general is a random search with the capability of pruning low performing models.

6) Gradient-based optimization. The idea behind the basic version is to compute the gradient for specific ML algorithms with respect to their hyper-parameters by means of gradient descent, which is an iterative optimization strategy for searching the local optimum of a differentiable function.

7) Reinforcement learning. The key idea is to build so-called surrogate model to predict validation loss for a candidate solution. The next value is selected based on heuristic function of the expected value and model uncertainty.

For now, as a pipeline design method for the scenario of no result in the knowledge base, our choice fell on Bayesian optimization, due to the fact that, compared to others, it allows you to achieve the highest quality result for the lowest amount of calculations [4]. But in the future, we will consider combining different optimization strategies.

III. Ontology-based Data Model

An ontology was chosen as a knowledge base for storing metadata for solving the problem of ML pipeline automated design at the current stage [5]. In the ontology, all data are presented as triplets: subject→predicate→object. This choice is justified by the flexibility of knowledge representation of the subject area and its logical inference feature, which will potentially be useful for the pipeline recommendation. The hierarchy of key classes of designed ontology and the structure of relations between them are shown in Fig. 1.

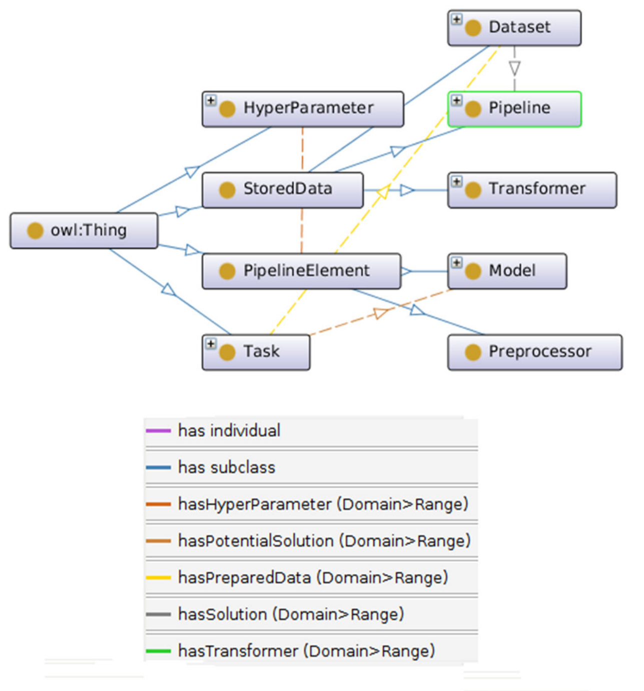
Dataset
HyperParameter
Pipeline
StoredData
Transformer
owl:Thing
PipelineElement
Model
Task
Preprocessor
has individual
has subclass
hasHyperParameter (Domain>Range)
hasPotentialSolution (Domain>Range)
hasPreparedData (Domain>Range)
hasSolution (Domain>Range)
hasTransformer (Domain>Range)

Fig. 1. Ontology class hierarchy

To begin with, we have an entity of the *Task* class. It is related to the entity set of the *Model* class through the *hasPotentialSolution* relationship. Also, each of the models can be associated with the entity set of the *HyperParameter* through the *hasHyperParameter* relationship. The considered relations are intended to provide a model search space for the optimization method based on a task type. Each potential solution is retrieved as code through the *hasPythonImplementation* property.

In addition, another sequence of relationships arises from the entity of the *Task* class. Specifically, the task is linked to the *Dataset* class through the *hasStoredData* relationship. The idea is to store the results of solving a particular task, such as a regression. Each data set is associated with one *Pipeline* class entity, resulting from the optimization method. The pipeline has a *hasEvaluatedAccuracy* property containing the accuracy estimate on the data. The pipeline instance is also associated with one or more instances of the *Transformer* class. Each transformer is either a learning model or a preprocessing algorithm and, as in the case of the search space, is extracted as a piece of software. Fig. 2 illustrates a fragment of such a relationship.

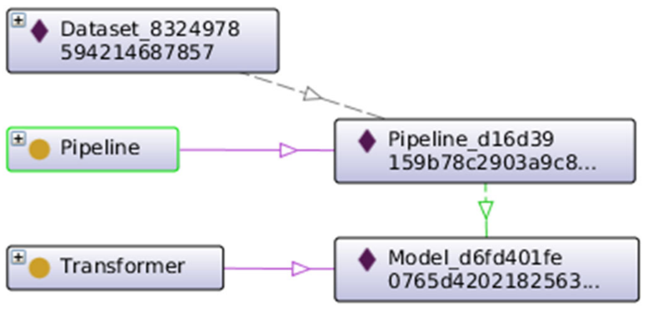
Dataset_8324978 594214687857
Pipeline
Pipeline_d16d39 159b78c2903a9c8...
Transformer
Model_d6fd401fe 0765d4202182563...

Fig. 2. Ontology entity relationship

The developed ontology contains the implementation of algorithms from the scikit-learn Python package [6]. In addition, we are currently implementing the cumulative data submodel, which will take into account data set meta-features [7].

IV. Method of Automated Population of the Knowledge Base

So, for the successful application of the developed ontology to the problem of automated ML pipeline design, a mechanism is needed that would improve the efficiency of meta-learning by accumulating knowledge [8]. The algorithm of the proposed method is shown in Fig. 3.

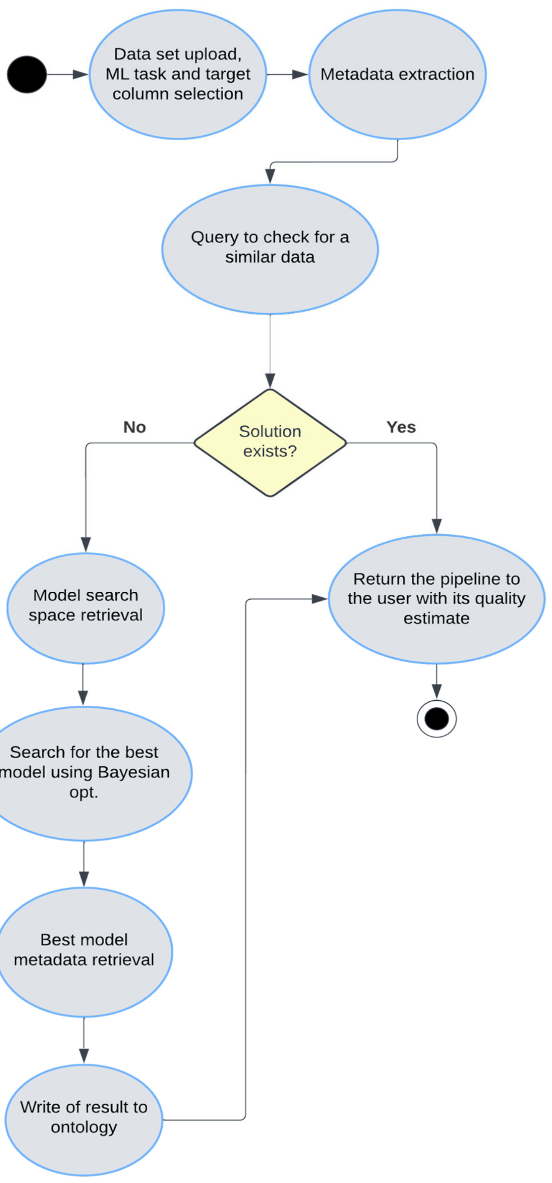
Data set upload, ML task and target column selection
Metadata extraction
Query to check for a similar data
Solution exists?
No
Yes
Model search space retrieval
Search for the best model using Bayesian opt.
Best model metadata retrieval
Write of result to ontology
Return the pipeline to the user with its quality estimate

Fig. 3 Current application workflow

V. Experimental Estimation of Pipeline Inference Time

Next, we will experimentally evaluate whether the proposed solution can improve the optimization based approach. To do this, we will run the developed program twice to cover scenarios characterized by the presence and absence of a result for the input data in the knowledge base. As test data, we will generate a data set through the *sklearn.datasets.make_regression* utility function.

Let us describe in detail the course of one of the experiments. At the first start, the pipeline for input was absent in the ontology. The search space extracted from the storage covers the common algorithms for solving the regression problem and their associated hyper-parameter ranges. Based on the results of Bayesian optimization, the following pipeline was obtained: *make_pipeline(StandardScaler(with_mean=False), LinearRegression())*, whose quality based on accuracy metric was 1.0. Subsequently, it was recorded to the knowledge base and, during second program run, was inferred from there immediately. Experimental results for datasets with different numbers of samples are presented in Table I.

TABLE I. Comparing the execution time of different scenarios

Number of samples	Execution time, s	
	Bayesian optimization	Metadata-based inference
1000	5	1
10000	88	
100000	16888 (**4.7h.**)	

As a result, metadata-based inference is always performed in constant time, and for optimization, there is a dependence of the pipeline search time on the size of the data set, which leads to a long waiting time on a high-dimensional input data.

VI. Conclusion

In the course of the work, the following significant results were achieved: a method was chosen for combined algorithm selection and hyper-parameter tuning; a data model was developed to improve the quality of the ML pipeline recommendation based on meta-learning; the method of automated population of the knowledge base was implemented; an experimental comparison was made between different ML pipeline inference scenarios, which confirmed the need of metadata warm-starting procedure for any optimization strategy to work effectively in terms of computational time.

In the future, it is planned to develop a meta-learning mechanism to recommend a pipeline for arbitrary data, as well as to gather enough metadata, so it will be possible to compete with the state-of-the-art solutions, for instance auto-sklearn 2.0, in a concrete task [9].

References

[1] J. Vanschoren, “Meta-learning”, Automated machine learning: methods, systems, challenges, 2019, pp. 35-61.

[2] M. Feurer et al., “Auto-sklearn: Efficient and Robust Automated Machine Learning”, Automated machine learning: methods, systems, challenges, 2019, pp. 113–134.
[3] Xin H. et al., “AutoML: A survey of the state-of-the-art”, Knowledge-Based Systems, vol. 212, 2021.
[4] R. Turner et al., “Bayesian optimization is superior to random search for machine learning hyperparameter tuning: Analysis of the black-box optimization challenge 2020”, PMLR, vol. 133, 2021.
[5] X. Wang et al., “Ontology based context modeling and reasoning using OWL”, in Proc. IEEE Annual Conference on Pervasive Computing and Communications Workshops, 2004, pp. 18-22.
[6] F. Pedregosa et al., "Scikit-learn: Machine learning in Python.", The Journal of Machine Learning Research, vol. 12, 2011, pp. 2825-2830.
[7] A. Rivolli et al., "Meta-features for meta-learning.", Knowledge-Based Systems, vol. 240, 2022.
[8] Petasis et al., “Ontology population and enrichment: State of the art”, Knowledge-Driven Multimedia Information Extraction and Ontology Evolution: Bridging the Semantic Gap, 2011, pp. 134-166.
[9] M. Feurer et al., "Auto-sklearn 2.0: Hands-free automl via meta-learning.", The Journal of Machine Learning Research, vol. 23, 2022, pp. 11936-11996.