Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 178 (2020) 414–423

Procedia Computer Science

www.elsevier.com/locate/procedia

9th International Young Scientist Conference on Computational Science (YSC 2020)

Structural Evolutionary Learning for Composite Classification Models

Nikolay O. Nikitin[a,*], Iana S. Polonskaia[a,*], Pavel Vychuzhanin[a], Irina V. Barabanova[a], Anna V. Kalyuzhnaya[a]

[a]*ITMO University, 49 Kronverksky Pr. St. Petersburg, 197101, Russian Federation*

Abstract

In this paper, we propose an evolutionary learning approach for flexible identification of custom composite models for classification problems. To solve this problem in an efficient way, the problem-specific evolutionary operators are proposed and the effectiveness of different modifications of the common genetic programming algorithm is investigated. Also, several implementations of caching for the fitted models were compared from the performance point of view. To verify the proposed algorithm, both synthetic and real-world classification cases are examined. The implemented solution can identify the structure of the composite models from scratch, as well as be used as a part of automated machine learning solutions.

Keywords: evolutionary learning; data-driven models; AutoML; genetic programming; machine learning; classification

1. Introduction

Nowadays, the identification of the data-driven models with complex heterogeneous structure is still an unsolved problem. It is closely related to an automatic machine learning task (also named AutoML), but at the same time primarily focused on the identification of the optimal models instead of data prepossessing selection or hyperparameter optimization. However, the common goal of the vast majority of automatic learning techniques is to improve the quality of analysis of various natural, technical, and social processes by observation-based identification of the optimal composite models [6]. AutoML solutions provide the opportunity to save a large amount of time, which specialists spend by perform many routine operations to select appropriate model. There are a lot of various solutions for the automatic learning problem that were proposed in recent years.

The existing AutoML solutions mostly deal with a fixed-shape structure (or an execution pipeline) of the composite model [29]. The optimization of the variable-shape structure at the appropriate time is still an open problem. The

* Corresponding author.
E-mail address: nnikitin@itmo.ru, ispolonskaia@itmo.ru

10.1016/j.procs.2020.11.043

structure of the model can be represented as a directed acyclic graph (DAG) [15] and the best suitable variant of the structure can be found using optimization approaches. The elements of this graph are machine learning models (intended for classification or supplementary models for side problems - i.e. clustering), that raises the computational cost problem. The other issue is the difficulty of the optimization metric choice since it should take into account both predictions quality and structural complexity of the composite models.

For these reasons, we decided to develop the evolutionary-based method for the identification of the composite models with the custom structure. Also, the various ways to improve the effectiveness of structural evolutionary learning are analyzed in the paper: caching, regularisation, etc. The experimental studies were restricted with classification models set only, however, the proposed concepts can be generalized to other machine learning tasks too. The main research question of this paper can be formulated as: in which ways the results of structural evolutionary learning for the classification models can be improved to be comparable with state-of-the-art solutions. The developed framework effectiveness was investigated on credit scoring problem. The obtained result was compared with the results of other algorithms and models which were found with most popular AutoML frameworks.

The paper is structured as follows. Sec. 2 provides the mathematical formulation of structural learning for the composite models and related optimization tasks. Sec. 3 contains the analysis of existing techniques that can be applied for structural learning, including the appropriate AutoML solutions. Sec. 4 describes the proposed approach based on the genetic programming algorithm. Sec. 5 highlights the problem of the quality assessment for structural learning and provides several ways to solve it. Sec. 6 contains the results of experimental studies based on both synthetic and real-world datasets. Finally, Sec. 7 provides an analysis of the obtained results.

2. Problem statement

The mathematical formalization of the structural identification problem for the composite model can be represented as follows. The structure of composite model can be represented as a chain C, that consists of a set of various machine learning models $M = \{M_1, ..., M_n\}$ and a directed link between them $L = \{L_1, ..., L_k\}$.

Each link L represents the data flow from the output of model M_i to one of the M_j inputs. The model M_j can receive several data flows, so its output can be represented as $Y_j{=}M_j(M_{i1}, ...M_{ik})$. The initial (primary) models in the chain are initialized by input data X directly, and secondary models (located on second and third layers of graph in Fig. 1) receive previous models predictions as input features. Also, each model M_i is initialised by a set of hyperparameters P_i. The structure of the chain $C(\boldsymbol{M}, \boldsymbol{L}, \boldsymbol{P})$ can be represented as a directed acyclic graph (DAG), where the vertices correspond to models M and the edges correspond to links L. The search of the suitable chain for the observations Y_{obs}, can be represented as an optimization problem:

$$\begin{aligned} \boldsymbol{M_{opt}}, \boldsymbol{L_{opt}}, \boldsymbol{P_{opt}} &= \underset{\boldsymbol{M},\boldsymbol{L},\boldsymbol{P}}{\operatorname{argmin}} F(C(\boldsymbol{M}, \boldsymbol{L}, \boldsymbol{P}), \boldsymbol{X}, \boldsymbol{Y_{obs}}), \\ F(\boldsymbol{M}, \boldsymbol{L}, \boldsymbol{P}, \boldsymbol{X}) &= \mathcal{G}(Q(C(\boldsymbol{M}, \boldsymbol{L}, \boldsymbol{P}, \boldsymbol{X})), S(\boldsymbol{M}, \boldsymbol{L}, \boldsymbol{P})), \end{aligned} \tag{1}$$

where F is the objective function, Q is a quality metric for the composite model, S is a structural complexity metric, $\boldsymbol{Y_{obs}}$ - simulation result for the whole chain, $\boldsymbol{Y}$ - observation for the simulated process, $\mathcal{G}(\bullet)$ is an operator for multiobjective transformation. Also, the implicit time constraint T should be considered. The multi-objective transformation represents fitness function in a Pareto form or implement the complexity-based penalty term in a frame of a single-objective optimization task. The structure of the chain is represented with the notation proposed in Fig. 1 together with the main aspects of structural identification tasks - target criteria and optimization tool (composer).

The models M can be represented as both target-specific models (for example, the classifier model for the classification problem) and supplementary models (regression and clustering). The additional restriction is imposed to M_{final}, since it should be target-specific to produce the correct prediction.

It should be noted that the problem of structural identification of the composite model is close to the machine learning pipeline generation problem [3]. However, the machine learning pipeline is typically based on a fixed high-level structure, while the links in the graph-based structure of the composite model can be designed arbitrarily. Also, the

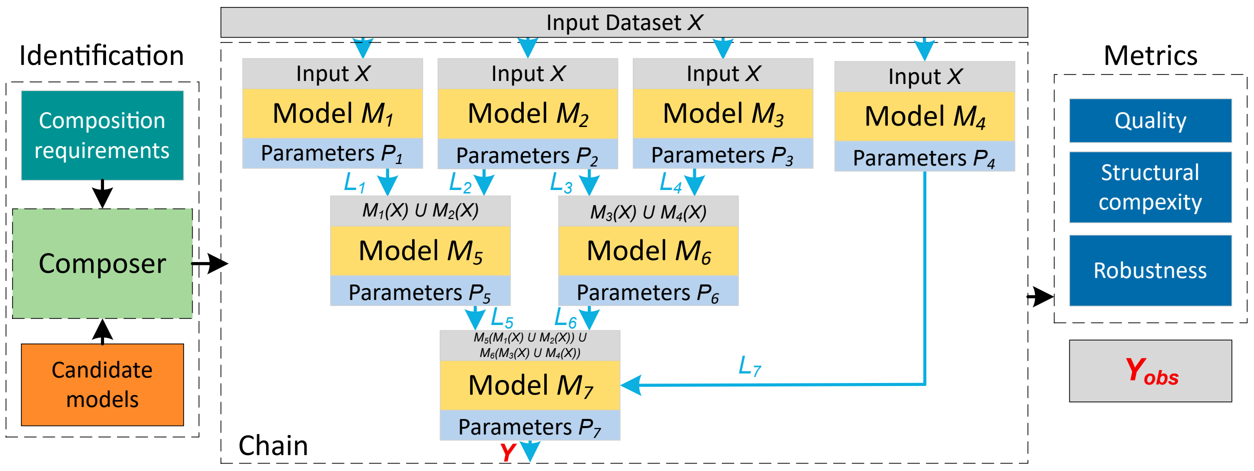


Fig. 1: The main aspects of the identification task, quality assessment approaches, and the internal structure of the chain.

pipelines include not just models, but also pre-processing (as feature selection, etc) and hyperparameters optimization steps.

3. Related work

There are several classes of approaches that can be used to solve the structural learning problem for the composite model. The first is an automatic machine learning solution, that solves the problems of the machine learning pipeline identification, which is often referred to as CASH: combined model selection and hyperparameters optimization [20]. However, it's frequently restricted by a limited set of models with fixed connections [21], since the pipeline generation problem is quite complex to be resolved directly [3].

TPOT is one of the most popular open-source AutoML tools. TPOT stands for the Tree-based Pipeline Optimization Tool. It is built on top of scikit-learn so the interface might look familiar. One of the main traits of the tool is using a genetic search algorithm to create pipelines. It creates a pipeline, estimates its performance, and randomly changes parts of it in search of better-performing algorithms [12]. At the end of the estimation, the tool generates a python script with the best found pipeline so it can be run at any convenient time. Considering that running such an algorithm can take hours or days to finish, the execution can be interrupted and the middle-step performance metric obtained.

Another AutoML framework H2O provides data preprocessing capabilities, a random grid of algorithms, and two stacked ensembles of models [21]. In the context of using H2O stacking or super-learning means to include second-level 'meta-learner' to find the optimal combinations of base learners. H2O automates the steps like specifying the base models and specifying the 'meta-learning' algorithm. The framework trains each of the base models then performs k-fold cross-validation and saves cross-validated predicted values to a matrix. The execution of the grid search stopped as soon as faces with a defined maximum number of models or defined time limit.

The other AutoML solution that can be used to identify the optimal pipeline is MLBox. It uses bayesian Tree Parzen Estimator to optimize a pipeline [1]. The quite promising approach for the evolutionary generation of the composite model represented as a directed acyclic graph (DAG) is used in the DarwinML framework [15]. The claimed error metrics for this solution outperform the state-of-the-art AutoML solutions in a set of benchmarks.

Among all considered auto-ml frameworks, the developed approach has most similarity with TPOT, since in the proposed approach an evolutionary algorithm is also used to identify the structure, and models set for chains composing contains scikit-learn models. However, all considered frameworks are of interest for comparison.

The second research area, that is similar to the pipeline identification problem, is the search of the optimal architecture neural network. It is applicable for the various real-world problems (i.e. pain intensity classification problem [8]). The state-of-art solution for the automated generation of the neural network-based models is AutoKeras. The framework uses Neural Architecture Search (NAS) algorithm to tune the hyperparameters of deep neural networks

(NN) and identify it's structure [13]. In the context of the composite classification model identification, NAS aims to obtain the best architecture of the neural network. The Search module generates architecture using the Bayesian optimization algorithm. There are two levels developed to accommodate different user's needs. The first one is the task-level, where a user specifies the task, e.g. image classification. The second search-level is for advanced users who can choose different types of neural networks (structural patterns). The opportunity for neural network architecture search in perspective can be added to developed framework as an separate module.

Moreover, there are some other approaches to improve and the quality of structural learning. Thus, in [2] the approach to reduce human participation in the previously-noted TPOT optimization process is proposed. The idea is to resize population size during evolution by the Fibonacci sequence. The maximum time for optimization is used as an algorithm stopping criterion. The ML-Plan took [9] tries to reduce the search space using hierarchical patterns of the pipeline's segments. In [18] methods based on k-means for simplification and size limitation of a model set for genetic programming as preprocessing steps were investigated. The main idea is to construct clusters of models, and take one appropriate model from each cluster to construct result models set. Reinforcement learning also can be used to solve the pipeline search problem [19]. The involvement of pre-defined grammar can be useful in some cases [11] too. The effectiveness of the surrogate modeling [28] and the co-evolutionary approach [7] is investigated in some studies. Some existing solutions are focused on the evolutionary optimization of classifier ensembles [25]. The RECIPE framework, which a based at grammar-based genetic programming [17], allow incorporating the existing knowledge about successful structural patterns of ML models into the optimization process. So, the evolutionary-based search is a quite promising approach in the identification of the structure of the composite models.

4. Structural learning by the genetic programming approach

Since the identification of the model structure can be considered as a graph optimization problem, it requires specific techniques that are able to operate with solutions in the form of directed acyclic graphs or tree-based structures. We decided to develop the specialized optimizer for the chain composition problems (hereinafter the 'composer') based on the modified genetic programming (GP) algorithm [24]. For each specified problem, an algorithm finds a chain consists of several ML models contained in pre-defined models set. The algorithm generates the initial population of chains using the method "Grow" [14]: each chain is recursively generated from the root node. After that, the population is evolved using crossover and mutation operators during the few generations until the convergence-based termination criteria are not satisfied.

Unlike the majority of GP tasks, the elements of the chain are models, that can be fitted partially independently. So, it is possible to save fitted nodes of the chain into the cache and use it to evaluate the fitness for the modified version of this chain. Also, it allows obtaining the fitness function value for some sub-graphs inexpensively, if it was previously trained as a part of another chain. Since the similar combinations of the models occur in the set of chains repeatably, the population of the genetic programming algorithm can be evaluated faster if the multi-chain caching will be applied to the fitted models.

Besides this, there is a problem of chains' structural over-complication. The smaller chains can provide solutions with fitness values equivalent to larger chains. Instead of the direct involvement of the penalty term for complexity (as it implemented in [15]), it is possible to implement a regularization operator, which reduces the chain's graph to the simpler subgraph, and either replace the original or add one or more copies into the current population.

To avoid the incorrectly structured chain in the results of GP composition, the validation functions were implemented. Validation strategies include well-known checks for graph structures like cycle detection, isolated nodes detection, and self-cycled nodes detection. These checks were implemented with the usage of the NetworkX python library for the graph analysis [5]. To use its functions directly, the composed chain transforms into the NetworkX-supported representation of graphs. Since the composed chain has to have two types of models some specific validation functions were developed. Any chain is checked for the presence of a primary model (that receives data as an input), so the chain cannot consist of only secondary models. The composed chain is also checked for the presence of only one root (final) node.

The pseudocode of the composer algorithms is presented in Alg. 1. The generational and steady-state evolutionary scheme was implemented in a frame of the algorithm. When using the first scheme, the generated offspring completely replaces the parent population. In the second scheme, the new population is formed by selection from the union of offspring and the previous population, which allows saving the ancestors and their perspective genotype. According to results of comparison on evolutionary schemes, steady-state scheme performed better. Therefore, we used it in our further experiments.

```
Data: populationSize, nodesSet, crossoverRate, mutationRate, trainData, testData, maxDepth, maxArity,
minArity, offspingSize, eliteCount
Result: best chain
pop ← InitPopulation(populationSize, nodesSet, maxDepth, maxArity)
while not ConvergenceCriterion() do
    for individ in pop do
        fitnessValue ← CalculateMetric(individ, trainData, testData)
    end
    bestIndividuals ← BestInds(pop, eleteCount)
    regularizedPopulation ← Regularization(pop, populationSize)
    matingPool ← TournamentSelection(pop ∪ regularizedPopulation, offspringSize)
    SelectedCrossover ← RandomChoice(SubtreeCrossover, OnePointCrossover)
    offspring ← SelectedCrossover(matingPool, crossoverRate, maxDepth)
    SelectedMutation ← RandomChoice(SimpleMutation, SubtreeMutation, ReduceMutation)
    offspring ← SelectedMutation(offspring, mutationRate, maxDepth, minArity, maxArity)
    pop ← TournamentSelection(pop ∪ offspring, populationSize, eliteCount) ∪ bestIndividuals
end
```

Algorithm 1: The pseudocode of the implemented genetic programming-based composer.

4.1. Crossover and mutation

In the proposed algorithm, we used three variants of mutation operators and two variants of crossover. If more than one variant is selected, the randomly chosen option is applied in each generation. Subtree crossover is performed by the replacement of random subtree in first selected parent to random subtree from the second parent. One-point crossover finds common structural parts between two trees, and after that randomly chooses the location of nodes, subtrees of which will be swapped. This approach allows saving established important structures of the graph instead of random changes (Fig. 2 a)). The other variant that was examined during the experiments is the genetic algorithm without crossover at all (in the modern NAS solutions it is practiced to used mutation-only algorithms [16]).

The first variant of mutation (simple) is passed over all nodes of the tree started from the root node and changes nodes' models with a specified probability. Subtree mutation selects a random node in a tree, generates new subtree, and replaces the selected node's subtree. This mutation is aimed to improve the exploration of search space by the generation of new subtrees. This type of mutation can lead to increase depth or width of the tree. Therefore, we added the reducing mutation which selects a random node in a tree, then removes its subtree. If the current arity of the node's parent is more than the specified minimal arity, then the selected node is also removed. Otherwise, it is replaced by a random primary node. The goal of this mutation is to perform exploitation and decrease tree depth. The mutation options demonstrated at Fig. 2 b).

Comparison of evolutionary operators on credit scoring problem (see the description in Sec. 6.2) is presented in Fig. 3. In the first three experiments (in Fig. 3 a)) specific options of mutation operators were used, while the crossover operator was chosen randomly from all possible options. However, the mutation also was chosen randomly in the last experiment. For experiments with crossover (presented in Fig. 3 b)), in the first two experiments specific options of crossover operator were used, while the mutation operator was chosen randomly. In the third experiment, the crossover was chosen randomly, and, in the fourth experiment, the crossover was excluded.

According to the results of experiments, it can be concluded that reduce mutation demonstrates the most effective and stable results. It can be related to the fact that composing of large trees is redundant for this problem. Regarding the crossover comparison, the worst result was demonstrated in the experiment with excluded crossover, other crossover options demonstrate similar results (but sub-tree crossover is the most effective variant in the early steps of optimization).

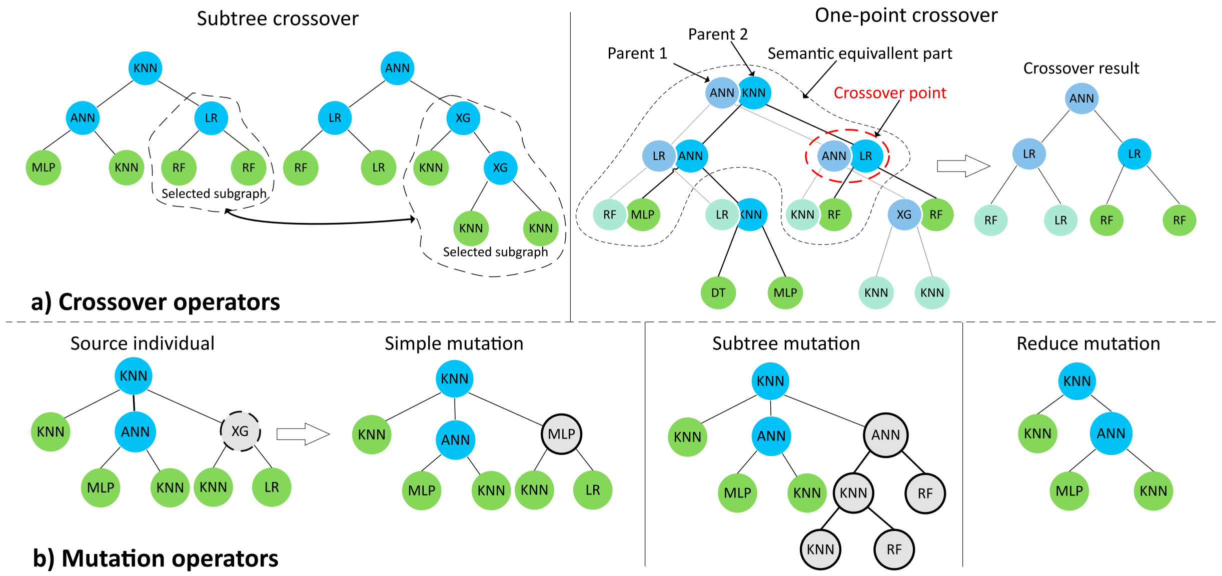


Fig. 2: Different variants of evolutionary operators for the graph representation of models structure: (a) crossover (b) mutation. Primary and secondary nodes are noted by green and blue colors respectively.

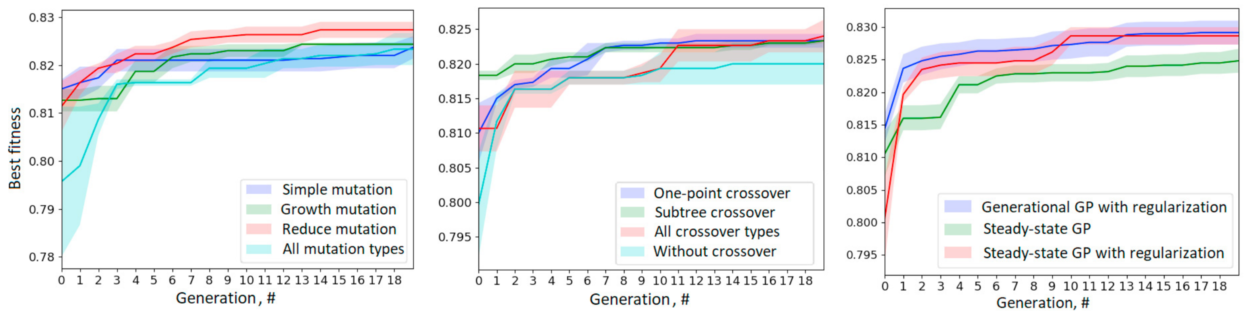


Fig. 3: Comparison of model structure optimisation convergence with: (a) simple, growth and reduce mutation (b) one-point and subtree crossover and without it at all (c) different types of heredity with regularization and without it.

4.2. Regularization of the chains

As a part of the composing algorithm, we developed a regularization operator (see Fig. 4) which is applied to the population before the parents selection procedure to provide more sustainable behavior of the evolutionary algorithm. Its mathematical formulation can be represented as Eq. 2.

The regularization suppresses unnecessarily growing of trees by the finding competitive subgraphs (rooted tree) among all possible unique subtrees of current population individuals and adding them to parents set (in a case of large trees, during regularization it is possible to subtract the only fixed number of random subtrees for a tree). The number of additional subtrees was set equals to population size. Experiments show that the regularization provides a positive influence for composing algorithm and improves its convergence (Fig. 3 c)).

$$\begin{gathered} R(\boldsymbol{G}) = (f_{best} \circ f_{sort} \circ f_{sub})(\boldsymbol{G}), \\ F_{reg}(\boldsymbol{M}, \boldsymbol{L}, \boldsymbol{P}, \boldsymbol{X}) = Q(C(\boldsymbol{M}, \boldsymbol{L}, \boldsymbol{P}, \boldsymbol{X})) + \lambda S(\boldsymbol{M}, \boldsymbol{L}, \boldsymbol{P}), \end{gathered} \tag{2}$$

where f_{sub} - the function, set decomposes the graph into the set of the primary sub-trees, f_{sort} - the function, that sorts the sub-trees according the F_{reg}, f_{best} selects the N best sub-trees, F_{reg} - regularized fitness function, λ - complexity penalty term, the other designations are derived from Eq. 1.

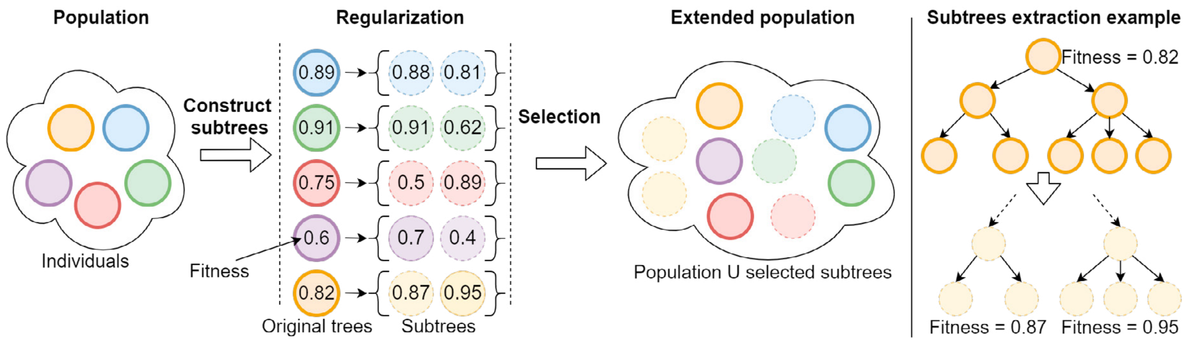


Fig. 4: The concept of regularization operator and example of its application to the models graph.

5. Metrics and benchmarks

The task of structural learning for the composite model can be considered from two points of view. On the one hand, it is an AutoML problem, where the goal is to identify a suitable set of model configurations and data preprocessing strategies that allow one to achieve a reliable value of the performance metric. On the other hand, it can be attributed to the symbolic regression task, which in turn is closely related to genetic programming. Therefore, the benchmarks can be split into two logical groups: data-oriented and structural benchmarks.

5.1. *Data-oriented benchmarks*

The common approach to measuring the effectiveness of AutoML systems is to obtain performance metrics on various ML tasks and datasets. For instance, the set of classification problems can be solved, during which the system automatically tries to find the best model or the ensemble of models, and also performs hyperparameter optimization. Then, the performance metric, such as the area under the ROC curve on the train and validation dataset is obtained. Finally, the results are compared with some reference solution. The benchmarking is usually done using both synthetic and real-world datasets, for instance, from OpenML [22], which is popular in the data science community. However, the results obtained with such type of benchmarking may be controversial and not useful for ML pipelines because it is mostly focused on a comparison with a single model result [4]. But it is of interest to measure the effectiveness of AutoML pipeline systems on the tasks where the dataset can be predicted mostly with a set or hierarchy of models.

5.2. *Structural benchmarks*

The second group refers to the field of genetic programming field, where researchers also note the problem of a correct benchmarking of algorithms [23]. Most often, GP-algorithms are aimed at finding a tree-based structure (for example, a symbolic expression that can be a solution of a task). In this case, one of the benchmarking methods is the task of finding a previously known tree or the symbolic expression. Thus, the convergence of the algorithm can be estimated measuring the difference between the ground-truth and the current solution. Also, as the quantitative metrics, some properties of the obtained trees from graph theory can be used. For instance, the complexity of a solution can be estimated based on a tree depth, number or the arity of nodes.

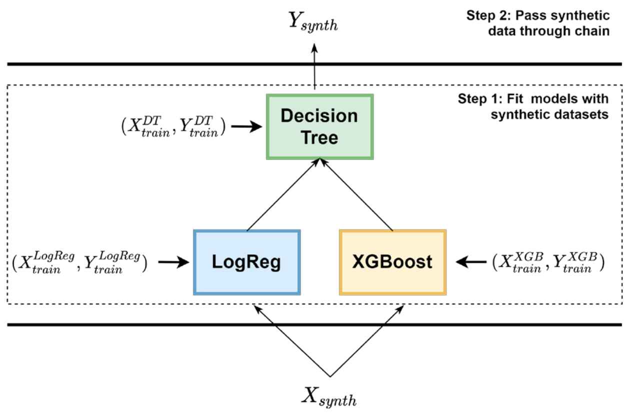
In Sec. 6 experimental results on both types of the proposed benchmarks are shown. The synthetic examples are structural benchmarks where the algorithm should find a ground-truth chain structure. While the real-world case is the simple data-oriented benchmark where the implemented composition algorithm is compared with several AutoML frameworks on the binary classification task.

6. Experimental studies

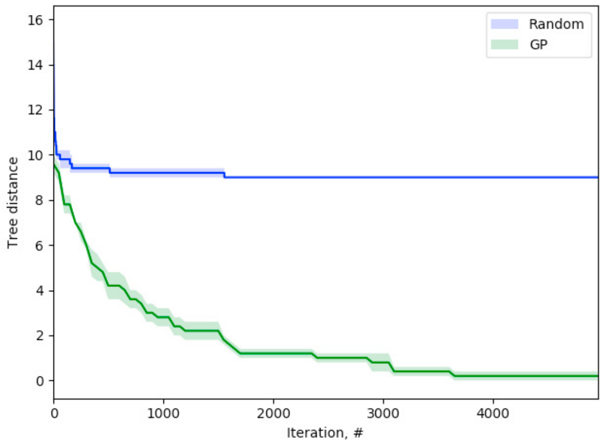
6.1. Synthetic cases

In order to test the framework, we implemented a synthetic case described below. Firstly, we generated a chain with a balanced binary tree structure with a fixed value of a tree depth. Each model in the nodes was selected randomly from a predefined set of ML models. Then, each model was fitted independently with synthetically generated datasets (we used scikit-learn functions that allow one to generate binary classification datasets with given parameters). Next, using the same approach we generated a dataset with X_{synth} features and passed it through the chain obtaining Y_{synth} predictions. Fig. 5a shows the proposed scheme of a synthetic dataset generation.

Further, we used (X_{synth}, Y_{synth}) in the ground-truth chain structure identification with our GP composition algorithm (also called as GP-composer). To simply put, we checked the ability to find the source ML pipeline with GP-composer based on the generated data. The experiments were conducted on chains with depth equals to four and limited set of ML models that included Logistic Regression, XGBoost, KNN, and a Decision Tree. The results is shown in Fig. 5b, where we compared GP-composer with the Random Search algorithm. In this case, the fitness function corresponded to Zhang-Shasha tree edit distance [27] between the source chain and the current solution. The experimental results indicated that GP-composer outperformed Random Search and allow one to find the ground-truth chain structure.


Y_{synth}
Step 2: Pass synthetic data through chain
Step 1: Fit models with synthetic datasets
$(X^{DT}_{train}, Y^{DT}_{train})$
Decision Tree
$(X^{LogReg}_{train}, Y^{LogReg}_{train})$
LogReg
XGBoost
$(X^{XGB}_{train}, Y^{XGB}_{train})$
X_{synth}

(a)



(b)

Fig. 5: The benchmarking on the synthetic cases: (a) synthetic dataset generation scheme (b) optimisation results for the synthetic benchmark (GP-based and Random Search-based composers) obtained after 30 runs

6.2. Real-world cases

The credit scoring problem was selected as real-world benchmark for the implemented composition algorithm. It is a well-known problem that can be reduced to binary classification. The effective scoring models can be created using ensembling [26]. The optimal structure of the scoring ensemble can be selected using evolutionary algorithms [10]. We choose the area under ROC curve (ROC AUC) as a quality metric for the scoring case. The scoring dataset (application data with default flag as target variable) was obtained from Kaggle competition named ”GiveMeSomeCredit”. It consists of 11 features (numerical and categorical) and the binary target variable.

We compared the implemented method of pipelines' composition with the random search and single-model baseline (xgboost). Also, the additional comparison was performed against the models obtained with the several AutoML frameworks: H2O, TPOT, MLBox. The optimization time limit was restricted by thirty minutes of real-time. The structure of the best model generated by the composer is presented in Fig. 6. The quality metrics for the obtained models are represented in Table 1.

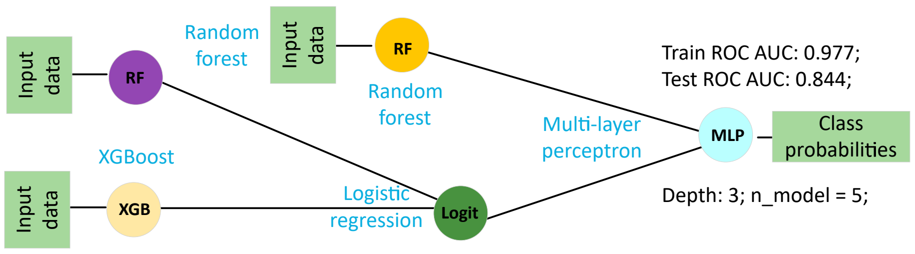


Fig. 6: The structure and metrics of the best model found by composer for the credit scoring problem.

Table 1: The ROC AUC metrics of the scoring models, generated by proposed algorithm and various AutoML approaches (the test sample is used for assessment).

Metric	GPComp	Random Search	XGBoost	H2O	TPOT	MLBox
ROC AUC, test sample	0.859	0.822	0.812	0.856	0.846	0.83
F1, test sample	0.43	0.26	0.34	0.4	0.32	0.31

It can be seen, the implemented GP-based composer outperforms the baseline and some of the state-of-art solutions. The obtained performance is competitive with the TPOT and H2O frameworks. Also, a similar quality (0.844 ROC AUC) can be achieved by GPComp even without the application of the highly-effective hyper-parameter optimization scheme (the default parameters for all sub-models in a composite model can be used). Besides, according to the empirical results using GPComp algorithm allow one to achieve approximately 0.16% and 0.3% metrics score increasing per minute of optimization relative to the baseline.

7. Conclusion

In this paper, we propose the flexible evolutionary approach to the structural learning of the composite model. This solution can be applied as a part of the AutoML framework, as well as a meta-optimization approach for the specific problem. The experimental studies with test problems (both synthetic and real-world) confirm that the proposed solution is comparable with the state-of-the-art methods even without hyperparameter optimization implemented. The experimental confirmation of the implemented specialized evolutionary operators is also provided. The source code of the algorithm is available in the open repository [1] as a part of the FEDOT framework, that implements the concept of the knowledge-enriched AutoML. The future development of this research will extend the proposed solution to the regression problem and the broader set of models (including domain-specific models). Also, in future works we are planing more extensive investigations of approaches to improve the efficiency of evolutionary search. For example, better results can be achieved by self-configuration the hyperparameters of the evolutionary algorithm and models during the evolution, using the additional evolutionary operators which take into account problems related to automatical models chains composition and etc.

8. Acknowledgements

This research is financially supported by The Russian Scientific Foundation, Agreement №19-11-00326.

References

[1] Elshawi, R., Maher, M., Sakr, S., 2019. Automated machine learning: State-of-the-art and open challenges URL: `http://arxiv.org/abs/1906.02287`, `arXiv:1906.02287`.

[1] https://github.com/nccr-itmo/FEDOT

[2] Evans, B.P., Xue, B., Zhang, M., 2020. An adaptive and near parameter-free evolutionary computation approach towards true automation in automl. arXiv preprint arXiv:2001.10178 .

[3] Garciarena, U., Santana, R., Mendiburu, A., 2018. Analysis of the complexity of the automatic pipeline generation problem, in: 2018 IEEE Congress on Evolutionary Computation (CEC), IEEE. pp. 1–8.

[4] Gijsbers, P., LeDell, E., Thomas, J., Poirier, S., Bischl, B., Vanschoren, J., 2019. An open source automl benchmark. arXiv preprint arXiv:1907.00909 .

[5] Hagberg, A., Swart, P., S Chult, D., 2008. Exploring network structure, dynamics, and function using NetworkX. Technical Report. Los Alamos National Lab.(LANL), Los Alamos, NM (United States).

[6] Kovalchuk, S.V., Metsker, O.G., Funkner, A.A., Kisliakovskii, I.O., Nikitin, N.O., Kalyuzhnaya, A.V., Vaganov, D.A., Bochenina, K.O., 2018. A conceptual approach to complex model management with generalized modelling patterns and evolutionary identification. Complexity 2018.

[7] Larcher Jr, C.H., Barbosa, H.J., 2019. Auto-cve: a coevolutionary approach to evolve ensembles in automated machine learning, in: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 392–400.

[8] Mamontov, D., Polonskaia, I., Skorokhod, A., Semenkin, E., Kessler, V., Schwenker, F., 2018. Evolutionary algorithms for the design of neural network classifiers for the classification of pain intensity, in: IAPR Workshop on Multimodal Pattern Recognition of Social Signals in Human-Computer Interaction, Springer. pp. 84–100.

[9] Mohr, F., Wever, M., Hüllermeier, E., 2018. Ml-plan: Automated machine learning via hierarchical planning. Machine Learning 107, 1495–1515.

[10] Nikitin, N.O., Kalyuzhnaya, A.V., Bochenina, K., Kudryashov, A.A., Uteuov, A., Derevitskii, I., Boukhanovsky, A.V., 2018. Evolutionary ensemble approach for behavioral credit scoring, in: International Conference on Computational Science, Springer. pp. 825–831.

[11] Nyathi, T., Pillay, N., 2018. Comparison of a genetic algorithm to grammatical evolution for automated design of genetic programming classification algorithms. Expert Systems with Applications 104, 213–234.

[12] Olson, R.S., Bartley, N., Urbanowicz, R.J., Moore, J.H., 2016. Evaluation of a tree-based pipeline optimization tool for automating data science, in: Proceedings of the Genetic and Evolutionary Computation Conference 2016, ACM, New York, NY, USA. pp. 485–492. URL: http://doi.acm.org/10.1145/2908812.2908918, doi:10.1145/2908812.2908918.

[13] Pham, H., Guan, M.Y., Zoph, B., Le, Q.V., Dean, J., 2018. Efficient Neural Architecture Search via parameter Sharing. 35th Int. Conf. Mach. Learn. ICML 2018 9, 6522–6531.

[14] Poli, R., Langdon, W.B., McPhee, N.F., Koza, J.R., 2008. A field guide to genetic programming. Lulu. com.

[15] Qi, F., Xia, Z., Tang, G., Yang, H., Song, Y., Qian, G., An, X., Lin, C., Shi, G., 2018. Darwinml: A graph-based evolutionary algorithm for automated machine learning. arXiv preprint arXiv:1901.08013 .

[16] Real, E., Aggarwal, A., Huang, Y., Le, Q.V., 2019. Regularized evolution for image classifier architecture search, in: Proceedings of the aaai conference on artificial intelligence, pp. 4780–4789.

[17] de Sá, A.G., Pinto, W.J.G., Oliveira, L.O.V., Pappa, G.L., 2017. Recipe: a grammar-based framework for automatically evolving classification pipelines, in: European Conference on Genetic Programming, Springer. pp. 246–261.

[18] Semenkina, M., Burlacu, B., Affenzeller, M., Pitzer, E., 2020. Models set pre-processing for genetic programming based evolvement of models of models, in: IOP Conference Series: Materials Science and Engineering, IOP Publishing. p. 012108.

[19] Sun, X., Lin, J., Bischl, B., 2019. Reinbo: Machine learning pipeline search and configuration with bayesian optimization embedded reinforcement learning. arXiv preprint arXiv:1904.05381 .

[20] Thornton, C., Hutter, F., Hoos, H.H., Leyton-Brown, K., 2013. Auto-weka: Combined selection and hyperparameter optimization of classification algorithms, in: Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 847–855.

[21] Truong, A., Walters, A., Goodsitt, J., Hines, K., Bruss, C.B., Farivar, R., 2019. Towards automated machine learning: Evaluation and comparison of AutoML approaches and tools. Proc. - Int. Conf. Tools with Artif. Intell. ICTAI , 1471–1479.

[22] Vanschoren, J., van Rijn, J.N., Bischl, B., Torgo, L., 2013. Openml: Networked science in machine learning. SIGKDD Explorations 15, 49–60. URL: http://doi.acm.org/10.1145/2641190.2641198, doi:10.1145/2641190.2641198.

[23] White, D.R., Mcdermott, J., Castelli, M., Manzoni, L., Goldman, B.W., Kronberger, G., Jaśkowski, W., O'Reilly, U.M., Luke, S., 2013. Better gp benchmarks: community survey results and proposals. Genetic Programming and Evolvable Machines 14, 3–29.

[24] Winkler, S., Affenzeller, M., Wagner, S., 2005. New methods for the identification of nonlinear model structures based upon genetic programming techniques. Systems Science 31, 5–13.

[25] Xavier-Júnior, J.C., Freitas, A.A., Feitosa-Neto, A., Ludermir, T.B., 2018. A novel evolutionary algorithm for automated machine learning focusing on classifier ensembles, in: 2018 7th Brazilian Conference on Intelligent Systems (BRACIS), IEEE. pp. 462–467.

[26] Xia, Y., Liu, C., Da, B., Xie, F., 2018. A novel heterogeneous ensemble credit scoring model based on bstacking approach. Expert Systems with Applications 93, 182–199.

[27] Zhang, K., Shasha, D., 1989. Simple fast algorithms for the editing distance between trees and related problems. SIAM journal on computing 18, 1245–1262.

[28] Zöller, M.A., Gabrys, B., . Avatar-machine learning pipeline evaluation using surrogate model, in: Advances in Intelligent Data Analysis XVIII: 18th International Symposium on Intelligent Data Analysis, IDA 2020, Konstanz, Germany, April 27–29, 2020, Proceedings, Springer Nature. p. 352.

[29] Zöller, M.A., Huber, M.F., . Benchmark and survey of automated machine learning frameworks .