Models set pre-processing for genetic programming based evolvement of models of models

To cite this article: Mariia Semenkina et al 2020 IOP Conf. Ser.: Mater. Sci. Eng. 734 012108

View the article online for updates and enhancements.
Models set pre-processing for genetic programming based evolvement of models of models

Mariia Semenkina\textsuperscript{1,2,3}, Bogdan Burlacu\textsuperscript{1,2}, Michael Affenzeller\textsuperscript{1,2} and Erik Pitzer\textsuperscript{1}

\textsuperscript{1}Heuristic and Evolutionary Algorithms Laboratory (HEAL), University of Applied Sciences Upper Austria, Softwarepark 11, 4232 Hagenberg, Austria
\textsuperscript{2}Johannes Kepler University, Altenberger Straße 69, 4040 Linz, Austria
\textsuperscript{3}E-mail: Mariia.Semenkina@fh-hagenberg.at

Abstract. We describe in this paper an extension of standard Genetic Programming where the terminal set of the algorithm is expanded with a set of basic models generated offline using a deterministic approach. The new algorithm called EMM-GP (Genetic Programming based Evolvement of Models of Models) uses a specialized mutation operator to sample this set during the recombination phase in order to create models that reuse valuable building blocks. In this work, a preprocessing of model set by means of clustering is considered.

1. Introduction
Symbolic regression (SR) is one of the tasks that can be solved with Genetic Programming (GP) [2]. Genetic Programming can generate solutions of an arbitrary form using components from functional and terminal sets. However, a practical limitation of GP is that it typically performs a long search for relatively simple bundles of variables.

The idea to use some “good parts of a tree” as parts of new generated trees during the GP run for GP boosting was first time considered in [2]. One of the most common variants is to use so-called Automatically Defined Functions (ADF) [3]. Most research in this direction focuses on using some kind of coevolution of GP trees and ADFs or independently train ADFs and then use them inside of GP trees [4]. In both variants ADFs inherits the same disadvantages and problems of GP such as bloat, redundancy, lack of phenotypic robustness (small changes in tree leading to big changes in model behaviour) and introduce additional complexity (in particular, when performing evaluation of trees with ADFs). Fast Function Extraction (FFX) algorithms [5, 6] do not suffer from these disadvantages, because they are deterministic and provide simpler models with more generalization ability. In this paper we introduce an approach to use deterministic algorithm [6] to generate symbolic regression formulas and to use them as building blocks for GP.

In [1] we consider two possible model set variants and show that the choice of model set significantly affects to algorithm efficiency. If the problem contains more than two variables the size of model set variants will increase dramatically. So, in this paper we consider pre-processing step for simplification of model set and it’s size limitation.

The paper is organized as follows: in Section 2, we explain how a map of models can be created and used during the GP run; Section 3 describes step of model set preprocessing and its challenges, then EMM-GP with preprocessing is compared with the previous version without preprocessing; lastly, our findings and conclusions are discussed in Section 4.
2. Genetic Programming based Evolvement of Models of Models (EMM-GP)
Genetic Programming based Evolvement of Models of Models (EMM-GP) is the approach that is discussed in [1]. The main scheme of the algorithm is presented below and the description of main steps.

| EMM-GP Algorithm scheme |
|-------------------------|
| 1. Create models using deterministic approach (offline) |
| 2. Pre-processing of model set (offline) |
| 3. Create a map of models based on distances among models (offline) |
| 4. Genetic programming algorithm run: |
| 4.1. Initialization (using additional terminal nodes from the model set) |
| 4.2. Fitness function calculation |
| 4.3. Selection |
| 4.4. Crossover |
| 4.5. Mutation (with special operations for nodes from models set) |
| 4.6. Survivor selection |

Initially (step 1), a set of symbolic regression models is created using a deterministic approach such as [6], to be subsequently used as leafs in GP trees. Afterward (step 2), the model set is simplified by deleting duplicate models according to a given distance measure. Certain difficulties arise on the step of mutation because the most obvious way of models replacement (replace on a random one from the set) can produce very big changes in phenotype that contradict the idea of mutation. We use a distance (metric as in [8]) based approach to cluster [7] the models created in the first step, and use them to create a map of buildings blocks (step 3). In this case, on the stage of mutation operator (step 4.5) works every model can be randomly replaced only on a model from the same.

All sub-models in this paper were created with the help of the algorithm based on Exhaustive Search from [6]. One of the challenges arising from this deterministic model generation scheme is how to select an appropriate complexity level for the generated models. In [1] we consider two possible variants: a set of models created using all available elements of a functional set (+, -, *, /, sin, cos, exp, log) as components (1200 possible sub-models for 2 variables case) and a set of models that were created using only basic elements of a functional set as components (+, -, *, /) and contains only 120 models for 2 variables case. If the problem contains more than two features the size of model set will increase dramatically. As shown in [1], the choice of model set significantly affects to algorithm efficiency. For these reasons the step of model set preprocessing was added to the algorithm.

Table 1. Examples of models from models set.

| Example 1 | Example 2 | Example 3 | Example 4 | Example 5 | Example 6 | Example 7 | Example 8 | Example 9 |
|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| (0 + 1/(0 + ((SIN((0 + ('X' * 1))) * SIN((0 + ('Y' * 1)))) * 1)) * 1)) | (0 + 1/(0 + ((LOG((0 + ('X' * 1))) * SIN((0 + ('Y' * 1)))) * 1)) * 1)) | (0 + 1/(0 + ((LOG((0 + ('X' * 1))) * EXP(('Y' * 1)))) * 1)) * 1)) | (0 + (1/(0 + (LOG((0 + ('Y' * 1))) * 1)) * EXP((('X' * 'X') * 1))) * 1)) | (0 + (1/(0 + (LOG((0 + ('Y' * 1))) * 1)) * EXP((('X' * 'X') * 1))) * 1)) | (0 + (1/(0 + (LOG((0 + ('Y' * 1))) * 1)) * EXP((('X' * 'X') * 1))) * 1)) | (0 + (1/(0 + ('X' * 1)) * 'X') * 1)) | (0 + ('X' * 1)) + ('Y' * 1)) | (0 + (1/(0 + ('X' * 1)) * 'X')) * 1)) + ('X' * 1)) |

3. Pre-processing of model set
The main idea of this step is to reduce a large, possibly redundant set of models (more than ten thousand models in the case of three input features) to a smaller, sufficiently representative set of models.
Figure 1. Clusterization result example for MSE distance (common picture and its zoom).

We employ clustering to reduce the size of the initial set of models, such that only one representative from each cluster is included in the reduced set of models. We initially tested the k-Means algorithm on a distance matrix based on the MSE distance between average model behaviour (averaged over ten randomly selected parameter set values). The resulting clusters presented in figure 1 (for visualization was used FastMap algorithm from [8]) for a small data set with two features indicate that this approach does not fulfil the main goal of representability.
In addition, the results reveal that numerical parameter values significantly influence model behaviour, to the point where two different parameter sets for the same model can produce completely different behaviours. An example of this phenomenon is presented in table 2 and correspondent graphs are presented on figure 2.

| Number | Model variant |
|--------|---------------|
| 0      | \((0 + (1/((0 + ('X' * 1)) * 'X') * 1)) + ('X' * 1))\) |
| A      | \(((0.657) + ((11.124 \times 1/((-12.187) + ((-0.05*X) * 4.816)) \times (-0.071*X)) \times (-17.079))) + ((-0.061*X) \times (-1.478)))\) |
| B      | \(((0.093 + (17.018 \times 1/((-12.301) + ((0.026*X) * 15.351)) \times (-0.038*X))) \times (-10.096)) + ((0.0264*X) \times 18.855))\) |
| C      | \(((13.468 + (((-0.513) \times 1/(5.541 + ((-0.109*X) \times (-18.018))) \times (-0.071*X)) \times 13.129)) + ((-0.023*X) \times 1.378))\) |

**Figure 2.** Examples of one model behavior for different parameter values.

In previous experiments we used ten different random sets of constants for each function and averaged the results. Simple averaging can lead to errors, but we cannot solve this problem with help of abs or quadratic functions (they a part of model sets to, and in case of their using functions x and |x| cannot be distinguished). Therefore, we use another variant:

1. Generate 10 randomly created sets of constants for each model in models set.
2. Create a new model set with 10 versions for each model (constants inside);
3. For this 10 times larger model’s set we make a calculation of distances to each other model according to metrics;
4. For i-th model in the initial set of models distance to j-th model in the initial subset is calculating like average distance from 10 variants of i-th to 10 variants of j-th (100 numbers that will be averaged then).

This variant describes the character ofcommons the behavior of two models on average more correctly.

Table 3. Average distances between clusters and in cluster.

| Average distances between clusters       | Average distance in cluster |
|-----------------------------------------|-----------------------------|
| 0                                       | 1302 1E+06 28383 282,47 242,7 111,05 160,6 207,7 167,9 740,6 95860 781 4079 655 1869 193 289 1200 90 |
| 1E+06 28383 0 229,82 1163 230,7 9E+06 298,7 142,6 3633,8 115 202 267 181 227 306 390 101 145 |
| 41404 282,4 229,8 0 3504 107602 3847 6E+07 5E+10 48,177 14672 18669 7502 95860 981 |
| 23257 242,6 1163 3503,6 0 36412 21738 9E+07 1E+11 97,069 1302 0 28383 0 229,82 1163 230,7 9E+06 298,7 142,6 3633,8 |
| 8249 111,0 230,7 107602 36412 0 22964 2E+08 65838 60,21 19 14672 160,6 9E+06 3847,3 21738 22964 0 58674 71070 42582 289 |
| 18669 207,7 298,7 6E+07 9E+07 2E+08 58674 0 6E+07 77,775 7502 167,8 142,6 5E+10 1E+11 65838 71070 6E+07 0 143,1 1575 90 |
| 7502 167,8 142,6 5E+10 1E+11 65838 71070 6E+07 0 143,1 1575 95860 740,6 3634 48,177 97,07 60,21 42582 77,77 143,1 0 90 |

Second problem that we can see on the Figure 1 is "outliers", that are far enough from all other models. This follow us to situations when clusterization is not correct like in table 3. We want to make an automatic selection of this "outliers" during clusterization run, so we tried to use different variants of clusterization algorithms:
- Usual K-means for real-valued vectors (based on different metrics). This did not bring any essential difference with our discrete based on distance matrices k-means algorithm that was considered previously.
- Different clusterization techniques (SOM (Figure 3), DBSCAN, X-Means) for real vectors, all of them cannot do anything with "outliers" in automatic mode.

Then, we made some modifications of discrete k-means:
- Modification, that tries to delete clusters with one only member and put this member to the closest one from big clusters; it decreases number of clusters but do not solve the problems with "outliers".
- Second modification, that for creation new cluster choose a point from cluster with biggest average distance in cluster. This version work more adequate then initial version and previous one. But good results were produced not on every run.
- Third modification, that in the selected cluster choose not random point, but a point that is far enough from others in this cluster (the distance from this point to cluster center should be bigger than median distance to cluster center). In addition, this modification do not put small cluster in big cluster if the distance to closest other cluster center is bigger than average distance in this cluster, and have dynamically value of K. This version produces the most adequate results that are shown on the figure 4.
Figure 3. Clusterization result example SOM.

Figure 4. Clusterization result example for MSE distance with new clusterization algorithm.
Therefore, the composition of the new variant of distance calculation and modified k-means gave us explainable results.

Investigation of EMM-GP effectiveness was performed on several test problems [9] and one real-world problem (Prognosis of turbine vibration characteristics). In this paper, we consider the influence of pre-processing step on the algorithm efficiency. All other parameters of EMM-GP were fixed. We use Standard GP and Standard GP with coefficient optimization as a baseline for comparison. During all tests, each algorithm made equal number of fitness function calculations. All results below are presented for the test part of a data set in terms of Pearson R² correlation coefficient. All results were received for 100 algorithm runs.

Table 4. Efficiency Comparison for different algorithm variants for two variables problem

| Algorithm                  | Average | SD  |
|----------------------------|---------|-----|
| Without pre-processing     | 0.8721  | 0.1643 |
| With pre-processing        | 0.9356  | 0.1213 |
| Standard GP                | 0.8997  | 0.2602 |

Table 5. Efficiency Comparison for different algorithm variants for five variables problem

| Algorithm                  | Average | SD  |
|----------------------------|---------|-----|
| Without pre-processing     | 0.7641  | 0.1747 |
| With pre-processing        | 0.9103  | 0.1312 |
| Standard GP                | 0.7431  | 0.2213 |

A summary of results for all tested problems is presented on Fig. 5.

4. Conclusion

Genetic Programming based evolvement models of models work quicker than conventional GP. In majority cases, EMM-GP with pre-processing step outperforms EMM-GP without pre-processing. Advantage of EMM-GP with pre-processing step is more noticeable for tasks with bigger amount of variables. Probably, other variants of using distance information in a map can be useful too.
This decision will not affect the running time of the algorithm because all preparation steps (sub-models extraction, distances calculation, simplification, clustering and map creation) can be done offline (before the algorithm is started) and only one time for each problem (algorithm for simple models creation [6] is deterministic).

References

[1] Semenkina M, Burlacu B and Affenzeller M 2019 Genetic Programming Based Evolvement of Models of Models Computer Aided Systems Theory, EUROCAST’2019 (Las Palmas de Gran Canaria: Universidad de Las Palmas de Gran Canaria) pp 104-105

[2] Koza J 1993 Simultaneous discovery of reusable detectors and subroutines using genetic programming Proceedings of the 5th International Conference on Genetic Algorithms, ICGA-93, (Morgan Kaufmann, University of Illinois at Urbana-Champaign) pp 295–302

[3] Koza J 1994 Genetic Programming II: Automatic Discovery of Reusable Programs (Cambridge: MIT Press)

[4] Niimi A and Tazaki E 2000 Genetic Programming Combined with Association Rule Algorithm for Decision Tree Construction Proceedings of Fourth International Conference on Knowledge-based Intelligent Engineering Systems and Allied Technologies (Brighton, UK) pp 746 – 749

[5] McConaghy T 2011 FFX: Fast, Scalable, Deterministic Symbolic Regression Technology Genetic Programming Theory and Practice IX (Springer Science) pp 235-260

[6] Kammerer L, Kronberger G, Burlacu B, Winkler S M, Kommenda M and Affenzeller M 2019 Symbolic Regression by Exhaustive Search – Reducing the Search Space Using Syntactical Constraints and Efficient Semantic Structure Deduplication Genetic Programming Theory and Practice XVII (Springer Science)

[7] Zojaji Z and Ebadzadeh M 2018 Semantic schema modeling for genetic programming using clustering of building blocks Appl Intell 48 (Springer Science) pp 1442-1460

[8] Faloutsos C 1995 FastMap: a fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets Proceedings of the 1995 ACM SIGMOD international conference on Management of data Vol 24 2 (USA, California, San Jose: ACM SIGMOD Record) pp 164-174

[9] Liang J, Qu B, Suganthan P, Chen Q 2015 Problem Definitions and Evaluation Criteria for the CEC 2015 Competition on Learning-based Real-Parameter Single Objective Optimization CEC 2015 Competition on Learning-based Real-Parameter Single Objective Optimization