SOLVE MACHINE LEARNING... WITH MACHINE LEARNING: GENETIC ALGORITHM-BASED PROGRAM SYNTHESIZER FOR THE CONSTRUCTION OF MACHINE LEARNING PIPELINES

”HOW WELL DOES A PROGRAM SYNTHESIZER BASED ON A GENETIC ALGORITHM PERFORM ON CREATING MACHINE LEARNING PIPELINES?”

BACKGROUND

With the growing presence of artificial intelligence, developers are looking for more efficient methods to construct machine learning algorithms.

Program synthesizers allow us to produce algorithms consisting of scalers, feature selection and classifiers. Each pipeline is a potential solution to the given machine learning task.

The goal of this synthesizer was to find the best-suited pipeline for the problem.

METHODOLOGY

Research questions:

- *"How well does a program synthesizer based on a genetic algorithm perform in creating machine learning pipelines?"*;
  - *"What observable difference is there in the quality of different kinds of machine learning pipelines produced by the same genetic algorithm-based synthesizer?"*;
  - *"How does the genetic algorithm-based synthesizer compare to man-made pipelines and synthesizers based on different machine learning algorithms?"*;

strategy:

- Create a context-free grammar, describing the set of possible machine learning pipelines.
- Develop the algorithm that would find the best pipeline in the search space.
  - This algorithm with a genetic algorithm approach for the best solution for the machine learning task.
- Evaluate the performance of the algorithm.
  - Accuracy of the produced pipeline.
  - Speed of synthesizer and pipelines.
- Compare to other synthesizers and report results.

RESULTS

Datasets:

- Development:
  - Iris (i: 150; f: 4; c: 3)*
- Evaluation:
  - Seeds (i: 210, f: 8, c: 3)
  - Wisconsin Diagnostic Dataset Breast Cancer (i: 569, f:31, c:2)
  - Human Activity Recognition (i: 10299, f: 562, c: 6)

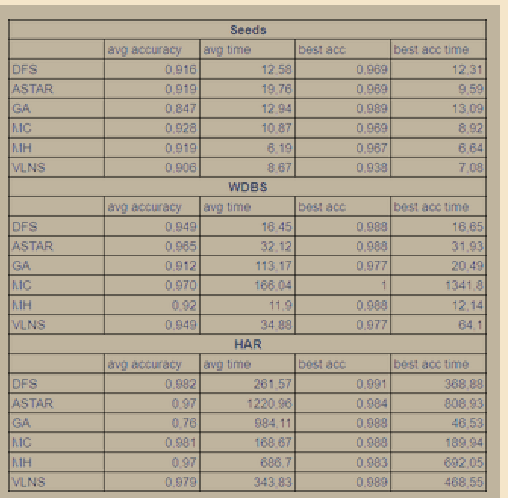
Settings

- Maximum number of fitness functions (100)
- Maximum depth pipeline (4)
- mutation probability (0.1)

* instances, features, classes

Results

- Focus on average accuracy, average time, best accuracy and time of best accuracy.
- Small variance in accuracy, great variance in time.
- note worthy observations:
  - Monte Carlo Tree search: perfect accuracy in 1341,8 seconds on WDBS
  - Genetic algorithm: two failed experiments with 0 accuracy, resulting in a smaller average.



Seeds				
	avg accuracy	avg time	best acc	best acc time
DFS	0,916	12,58	0,969	12,31
ASTAR	0,919	19,76	0,969	9,59
GA	0,847	12,94	0,989	13,09
MC	0,928	10,87	0,969	8,92
MH	0,919	6,19	0,967	6,64
VLNS	0,906	8,67	0,938	7,08

WDBS				
	avg accuracy	avg time	best acc	best acc time
DFS	0,949	16,45	0,988	16,65
ASTAR	0,965	32,12	0,988	31,93
GA	0,912	113,17	0,977	20,49
MC	0,970	166,04	1	1341,8
MH	0,92	11,9	0,988	12,14
VLNS	0,949	34,88	0,977	64,1

HAR				
	avg accuracy	avg time	best acc	best acc time
DFS	0,982	261,57	0,991	368,88
ASTAR	0,97	1220,96	0,984	808,93
GA	0,76	984,11	0,985	46,53
MC	0,981	168,67	0,988	189,94
MH	0,97	686,7	0,983	692,05
VLNS	0,979	343,83	0,989	468,55

figure 1: Experiments results

Conclusion

- High variance in time can be explained by dependency on Skicit-learn + faulty pipelines + different search algorithm configurations.
- Synthesizers do not have much better performances than a breadth-first search.
- Small pipelines perform almost as well as long pipelines on the datasets.

"Is the use of synthesizers in machine learning worth persuing?"

FUTURE WORK

The synthesizers proved not to be much more accurate than the general breadth-first algorithm. Runtimes are too irregular, most likely due to incompatibility between particular pipelines and Scikit-learn functionalities.

Future research can continue improving the algorithms with the goal of stabilizing the runtimes. Afterward, bigger classification problems for which breadth-first search algorithms are not suitable anymore can be used in further research toward the validity of machine learning synthesizers. The number of iterations, pipeline depth and number of fitness functions can be increased to increase the search space. To perform this research, resources that were not available for this research project, like more advanced computers should be used to solve the bigger classification problems.

REFERENCES

M. Charytanowicz, J. Niewczas, P. Kulczycki, P. Kowalski, and S. Lukasik. seeds. UCI Machine Learning Repository, 2012. DOI: https://doi.org/10.24432/C5H30K.
I. De Falco, âªA. Della Cioppa, and E. Tarantino. Mutation-based genetic algorithm: performance evaluation. Applied Soft Computing, 1(4):285–299, 2002.
A. de Sá, W. Pinto, L. Oliveira, and G. Pappa. Recipe: a grammar-based framework for automatically evolving classification pipelines. In Genetic Programming: 20th European Conference, EuroGP 2017, Amsterdam, The Netherlands, April 19-21, 2017, Proceedings 20, pages 246–261. Springer, 2017.
B. Filius, T. Hinnerichs, and S. Duman˘ cić. Solving ml with ml: Evaluating the performance of the monte carlo tree search algorithm in the context of program synthesis. TU Delft preprint: available from repository.tudelft.nl, 2023.
R. A. Fisher. Iris. UCI Machine Learning Repository, 1988. DOI: https://doi.org/10.24432/C56C76.
D. E. Goldberg and K. Deb. A comparative analysis of selection schemes used in genetic algorithms. volume 1 of Foundations of Genetic Algorithms, pages 69–93. Elsevier, 1991.
S. Gulwani, O. Polozov, R. Singh, et al. Program synthesis. Foundations and Trends® in Programming Languages, 4(1-2):1–119, 2017.
M. Katz, P. Ram, S. Sohrabi, and O. Udrea. Exploring context-free languages via planning: The case for automating machine learning. Proceedings of the International Conference on Automated Planning and Scheduling, 30(1):403–411, Jun. 2020.
R. Lejeune, T. Hinnerichs, and S. Duman˘ cić. Solving ml with ml: Effectiveness of a star search for synthesizing machine learning pipelines. TU Delft preprint: available from repository.tudelft.nl, 2023.
T. M. Mitchell. The discipline of machine learning, volume 9. Carnegie Mellon University, School of Computer Science, Machine Learning âŠ, 2006.
M. Negnevitsky. Artificial Intelligence: A Guide to Intelligent Systems. Addison-Wesley Longman Publishing Co., Inc., USA, 1st edition, 2001.
J. Reyes-Ortiz, D. Anguita, A. Ghio, L. Oneto, and X. Parra. Human Activity Recognition Using Smartphones. UCI Machine Learning Repository, 2012. DOI: https://doi.org/10.24432/C54S4K.
J. De Jong P. Klop S. Duman˘ cić, T. Hinnerichs. Herb.jl. https://github.com/ Herb-AI/Herb.jl, 2023.
D. Sheremet, T. Hinnerichs, and S. Duman˘ cić. Solving ml with ml: Effectiveness of the metropolis-hastings algorithm for synthesizing machine learning pipelines. TU Delft preprint: available from repository.tudelft.nl, 2023.
D. Sheremet, A. Sonneveld, R. Lejeune, B. Filius, M. Butenaerts, S. Duman˘ cić, and T. Hinnerichs. Herb.jl. https://github.com/Herb-AI/Herb.jl, 2023. 29
Michael Sipser. Introduction to the Theory of Computation. Course Technology, 2006.
A. Sonneveld, T. Hinnerichs, and S. Duman˘ cić. Solving machine learning with machine learning: Exploiting very large-scale neighbourhood search for synthesizing machine learning pipelines. TU Delft preprint: available from repository.tudelft.nl, 2023.
A. J. Umbarkar and P. D. Sheth. Crossover operators in genetic algorithms: a review. ICTACT journal on soft computing, 6(1), 2015.
W. Wolberg, O. Mangasarian, N. Street, and W. Street. Breast Cancer Wisconsin (Diagnostic). UCI Machine Learning Repository, 1995. DOI: https://doi.org/10.24432/C5DW2B.
M. Butenaerts, T. Hinnerichs, and S. Dumancić. Genetic algorithm-based program synthesizer for the construction of machine learning pipelines. TU Delft preprint: available from repository.tudelft.nl, 2023

Author
Mathieu Butenaerts
m.g.a.butenaerts@student.tudelft.nl

Professor
Sebastijan Dumančić
s.dumancic@tudelft.nl

Supervisor
Tilman Hinnerichs
T.R.Hinnerichs@tudelft.nl