What's inside the black-box? A genetic programming method for interpreting complex machine learning models

Morgan Jones
Department of Computer Science
Aberystwyth University
mwj7@aber.ac.uk

ABSTRACT

In this work, we present a new approach to interpretable machine learning; using multi-objective genetic programming to construct simple representations of complex black-box Machine learning models. Our results are far simpler with comparable reconstructive performance than that of existing state-of-the-art methods. Demonstrated on a range of datasets our resulting interpretable model (a single decision tree) approximates the knowledge of 200 layer neural networks and ensembles of 500 trees.

CCS CONCEPTS

• **Computing Methodologies → Genetic Programming**;

KEYWORDS

Explainable Artificial Intelligence, Interpretable Machine Learning, Evolutionary Multi-objective Optimisation

ACM Reference Format:
Morgan Jones. 2019. What's inside the black-box? A genetic programming method for interpreting complex machine learning models. In *Proceedings of the Genetic and Evolutionary Computation Conference 2019 (GECCO '19).* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3321707.3321726

1 INTRODUCTION

Understanding the decisions behind Machine Learning (ML) techniques is important for their adoption into new areas where an AI's decisions have greater consequences. This need for interpretability has created the push for explainable AI (XAI).

Traditional tree-based methods from XAI suffer from limitations, due to the greedy nature of tree construction. Using Genetic Programming (GP) [5] we can evolve trees instead of constructing them in a greedy manner.

There are multiple approaches to XAI [4], one approach is the interpretable machine learning (IML) method of model extraction [1]. In this work we propose a new multi-objective GP-based method for model extraction that is applicable to any black-box (arbitrarily complex) classifier.

2 THE NEW METHOD

We use NSGA-II [3], paired with strongly-typed GP (STGP) [7], to evolve decision tree-like structures, which simultaneously balance the complexity and the accuracy of the trees. Our goal is to outperform greedy methods, while still being computationally feasible. Another benefit of this approach is that a Pareto front of non-dominated trees is produced instead of just a single tree. This is useful for XAI as it allows a user to select a tree for visualisation at a desired complexity to accuracy trade-off point along the front.

2.1 Overall Algorithm

The overall training algorithm is shown in Fig. 1. The black-box classifier is trained once only on the original data (x and y values), then the evolutionary process is performed based on the resulting predictions ($\hat{y}$) from this black-box model. The evolutionary algorithm never sees the original labels (y), as this is instead attempting to recreate the predicted labels ($\hat{y}$). At the end of the evolutionary run, the result is a set of Pareto optimal models/trees which approximate the complex black-box model. Only the model with the highest reconstructive ability is used here. The overall evolutionary process is similar to NSGA-II. When selecting individuals, the non-dominated sorting in NSGA-II algorithm is used to rank the individuals.

2.1.1 *Objective Functions.*

- **Reconstruction ability** (*maximisation*): The average weighted f1 metric as result of an internal (training set only) 3-fold (k=3) cross-validation on the tree
- **Complexity** (*minimisation*): Number of splitting points (nodes) in the tree

2.1.2 Representation. The evolved trees are decision tree-like, meaning the input is at the root, the output at a leaf, and the internal nodes are splitting points. Traditionally with GP (parse-trees), the inputs are at the leaves (terminals), the output is at the root, and the internal nodes are functional components (functions). To get around this, data is passed in at the leaves but only functions are returned until the root node (only one branch will end up returning values for a given input), and then the functions are executed once we are at the root. For visualisation and use-cases, the two can be treated uniformly.

Some simple patterns have needlessly complex representations in decision trees with axis-parallel splits. To solve this, the trees can construct features (as mathematical expressions) implicitly and check if those constructed features are greater than or equal to zero. For example:

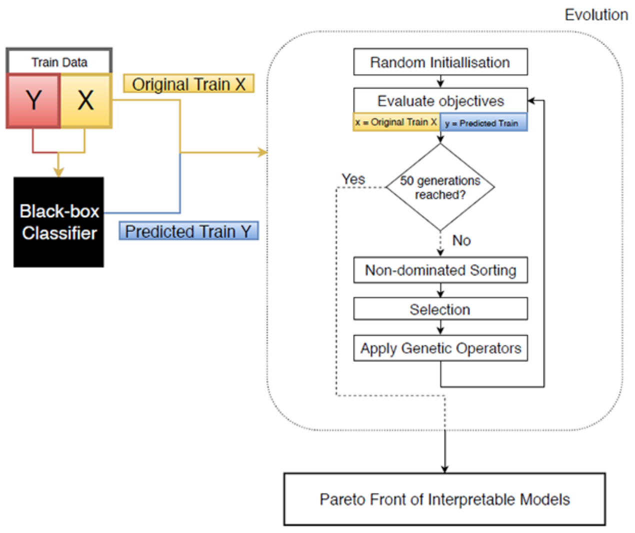

Figure 1: Evolutionary Training Process.

Consider the condition if f1 >= f2 then class 1 else class 0. Without constructed features the resulting tree would be complex. With constructed features we can simply use one split as f1-f2>=0.

Using constructed features is beneficial to XAI because simpler rules are learnt.

3 EXPERIMENTS

3.1 Experiment Details

3.1.1 Datasets. A broad range of 30 datasets from the OpenML repository [10] were used for comparison. These were the 20 most run binary datasets, and 10 most run multiclass datasets. The datasets were restricted to less than 15000 instances, less than 5 classes, and no missing values. These datasets are from a variety of domains, and have a varying number of features (both categorical and numeric), classes, and instances.

3.1.2 Comparison Methods. Current state-of-the-art approaches to model extraction were trialled:

- Bayesian rule lists [11] (as pysbrl on pip)
- A (h2o) decision tree [**?**]
- A simplified (scikit-learn) [8] decision tree [6]
- Logistic regression with L1-regularization (from scikit-learn)

Pre-Processing: One-hot-encoding was required for the scikit-learn methods to support categorical features. Multi-interval discretization is applied for the Bayesian rule lists to support continuous features.

3.1.3 GP Parameter Settings. The evolutionary search was run for 50 generations, with 100 trees in each generation. Trees were limited to a maximum height of 17. A crossover rate of 0.8 was used, and a mutation rate of 0.2. Top performing individuals are never lost, as the $\mu + \lambda$ algorithm [2] is used, with both values set to the population size.

3.1.4 Evaluation Measures.

Classification Performance. For measuring performance, we use a weighted f1-measure scaled to the range 0...100.

Complexity. We define complexity as the number of splitting points in the tree. If constructed features are used they count as multiple splits (f1+f2<=0 would count as 2). For Bayesian rule lists the complexity is measured as the number of rules + the number conjunctions in these rules. For logistic regression complexity is measured as the number of non-zero coefficients.

3.1.5 Black-box Methods. We chose three high performing black-box models, all implemented in h2o [**?**]: Random Forests, Gradient Boosting (both with 500 trees) and a deep neural network (200 hidden layers with 200 neurons each).

3.1.6 Significance Tests. To compute whether the difference in reconstruction ability across datasets for each method was statistically significant, we used Friedman tests paired with Nemenyi post-hoc analysis. To show the average reconstruction ability for each method we present the average accuracy of a 10-fold cross-validation, where each method gets the same train-test sets. The averages are also across the 3 black-box methods therefore for each method, 30 runs are executed for each of the 20 datasets.

3.2 Results

We compare methods based on the number of times each method was dominated. [1] Where domination is defined as another method achieves a simpler representation with the same (or improved) recreation ability. GP (the proposed method) was not dominated on any of the datasets (Fig. 2). The resulting p-values from the significance tests are visualised in Fig. 3. [2]

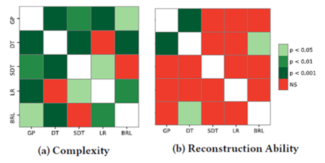

Figure 2: Statistical significance testing.

REFERENCES

[1] Osbert Bastani, Carolyn Kim, and Hamsa Bastani. 2017. Interpretability via model extraction. *arXiv preprint* 3, 0977 (2017). arXiv:cs/170609773
[2] Hans-Georg Beyer and Hans-Paul Schwefel. 2002. Evolution strategiesâĂŞA comprehensive introduction. *Natural computing* 1, 1 (2002), 3–52.
[3] Kalyanmoy Deb, Truyen Tran, and Aditya Ghose. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation* 6, 2 (2002), 182–197.
[4] Filip Karlo Dosilovic, Mario Brcic, and Nikica Hlupic. 2018. Explainable artificial intelligence: A survey. *Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO).* 41 (2018), 0210–0215.

[1] For our method we select a single solution from our resultant frontier with the highest reconstruction ability to represent our methods performance.
[2] One caveat is that analysing the dominated counts alone is not a comprehensive indicator of performance, since both the simplest possible model (majority class) and the black-box model itself would never be dominated.

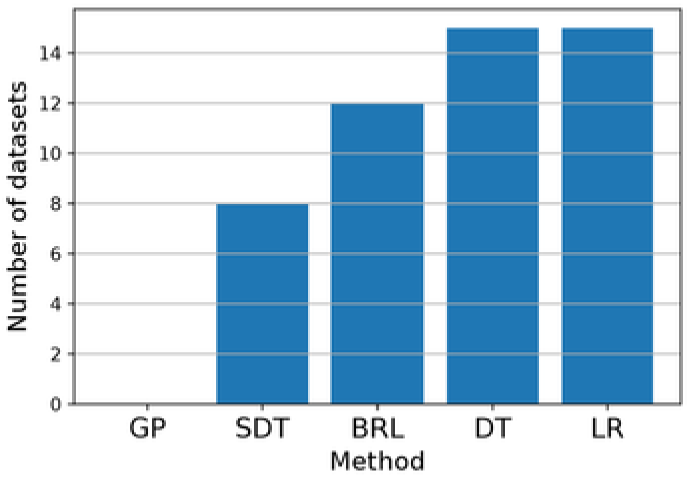

Figure 3: Method domination count.

[5] John R Koza. 1994. Genetic programming as a means for programming computers by natural selection. *Statistics and computing* 4, 2 (1994), 87–112.

[6] Tamas Madl. 2018. Sklearn interpretable tree. (2018). https://github.com/tmadl/sklearn-interpretable-tree

[7] David J Montana. 1995. Strongly typed genetic programming. *Evolutionary Computation* 3, 2 (1995), 199–230.

[8] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[] The H2O.ai team. 2015. h2o: Python Interface for H2O. (2015). http://www.h2o.ai

[10] Joaquin Vanschoren, JanN. van Rijn, Bernd Bischl, and Luis Torgo. 2013. OpenML: Networked Science in Machine Learning. *SIGKDD Explorations* 15, 2 (2013), 49–60.

[11] Hongyu Yang, Cynthia Rudin, and Margo Seltzer. 2016. Scalable Bayesian rule lists. *arXiv preprint* 0, 0861 (2016). arXiv:cs/160208610