







Grammar-based Fuzzy Pattern Trees for Classification Problems

Aidan Murphy¹, Muhammad Sarmad Ali¹, Douglas Mota Dias^{1,2}, Jorge Amaral²,
Enrique Naredo¹ and Conor Ryan¹

¹University of Limerick, Limerick, Ireland

²Rio de Janeiro State University, Rio de Janeiro, Brazil

Keywords: Grammatical Evolution, Pattern Trees, Fuzzy Logic.

Abstract: This paper introduces a novel approach to induce Fuzzy Pattern Trees (FPT) using Grammatical Evolution (GE), FGE, and applies to a set of benchmark classification problems. While conventionally a set of FPTs are needed for classifiers, one for each class, FGE needs just a single tree. This is the case for both binary and multi-classification problems. Experimental results show that FGE achieves competitive and frequently better results against state of the art FPT related methods, such as FPTs evolved using Cartesian Genetic Programming (FCGP), on a set of benchmark problems. While FCGP produces smaller trees, FGE reaches a better classification performance. FGE also benefits from a reduction in the number of necessary user-selectable parameters. Furthermore, in order to tackle bloat or solutions growing too large, another version of FGE using parsimony pressure was tested. The experimental results show that FGE with this addition is able to produce smaller trees than those using FCGP, frequently without compromising the classification performance.


1 INTRODUCTION


Machine learning (ML) has great potential to solve real-world problems and to contribute to the improvement of processes, products, and research. In the last two decades, the number of applications of machine learning has been increasing due to the availability of vast collections of data and massive computer power thanks to the development of new training algorithms, the emergence of new hardware platforms based on graphics cards with GPUs and the availability of open-source libraries (Došilović et al., 2018). Such conditions provide ML systems with the ability to solve highly complex problems with performance superior to those obtained by techniques that, until then, represented state of the art. Moreover, in some specific fields of application, such as image classification, ML systems have surpassed human performance (He et al., 2015).


Although ML algorithms are successful in terms of results and predictions, they have their shortcomings. The most compelling is the absence of transparency, which identifies the so-called black-box models. In such models, it is very difficult or even impossible to understand how the ML system makes its decision or to extract the knowledge of how the decision is made. As a result, it does not allow a human being, expert, or not to check, interpret, and understand how the model reaches its conclusions.


In order to address these issues, Explainable Artificial Intelligence (XAI) (Adadi and Berrada, 2018; Arrieta et al., 2020) has appeared as a field of research focused on the interpretability of ML. The main purpose is to create a set of models and interpretable methods that are more explainable while preserving high levels of predictive performance (Carvalho et al., 2019).


Fuzzy Set theory has provided a framework to develop interpretable models (Cordón, 2011) (Herrera, 2008) because it allows the knowledge acquired from data to be expressed in a comprehensible form, close to natural language, which gives the model a higher degree of interpretability (Hüllermeier, 2005). Most developed fuzzy models are rule-based fuzzy systems (FBRS) that can represent both classification


^a  <https://orcid.org/0000-0002-6209-4642>

^b  <https://orcid.org/0000-0002-7223-5322>

^c  <https://orcid.org/0000-0002-1783-6352>

^d  <https://orcid.org/0000-0001-6580-5668>

^e  <https://orcid.org/0000-0001-9818-911X>

^f  <https://orcid.org/0000-0002-7002-5815>

and regression functions and for which there are many strategies developed for the synthesis of these models (Cordón, 2011). Obtaining fuzzy models based on easily interpretable rules may not be an easy task, because depending on the application, many rules may be necessary with many antecedents that make it difficult to understand the model.

On the other hand, a system with relatively few rules can be easily interpreted, but have its predictive accuracy compromised. In this work, a novel approach to automatically induce models applied on classification problems is introduced. It uses a method based on the theory of fuzzy sets, Fuzzy Pattern Trees (FPT), which is not based on rules, but on a hierarchical method. This work replaces the FPT learning method with Grammatical Evolution (GE).

GE is flexible enough to derive feasible models such as FPTs, and it can efficiently address different problems by changing the grammar and the evaluation function. As a result, it is possible to obtain models that can solve a classification problem and to get explainable solutions at the same time. Moreover, the combination of GE and Fuzzy Logic gives a valuable opportunity to address the new research lines in XAI. Experimental results show that GE can evolve fuzzy pattern trees to solve benchmark classification problems with competitive results against state of the art methods with better results in three of them.

The remainder of this paper is organized as follows: Section 2 reviews the main background concepts, including FPTs, Genetic Programming (GP), Cartesian GP (CGP) and GE. Section 3 explains the proposal and contributions of this work in more detail. Next, Section 4 presents the experimental setup, outlining all of the considered variants and performance measures. Section 5 presents and discusses the main experimental results of the described research. Finally, Section 6 presents the conclusions and future work derived from this research.

2 BACKGROUND

2.1 Fuzzy Pattern Trees

FPTs have independently been introduced by Huang et al. (Huang et al., 2008), and Yi et al. (Yi et al., 2009) who called this type of model Fuzzy Operator Trees. The FPT model class is related to several other model classes including fuzzy rule-based systems (FRBS), and fuzzy decision trees (FDT).

A FPT is a hierarchical, tree-like structure, whose inner nodes are marked with generalized (fuzzy) logical and arithmetic operators, and whose leaf nodes are

associated with fuzzy predicates on input variables. It propagates information from the bottom to the top: A node takes the values of its descendants as input, aggregates them using the respective operator, and submits the result to its predecessor. Thus, an FPT implements a recursive mapping producing outputs in the $[0,1]$ interval.

The following operators are used, where a and b are the inputs to the operator:

$$WTA = IF\{\}\dots ELSE() \quad (1)$$

$$MAX = \max(a,b) \quad (2)$$

$$MIN = \min(a,b) \quad (3)$$

$$WA(k) = ka + (1-k)b \quad (4)$$

$$OWA(k) = k \cdot \max(a,b) + (1-k)\min(a,b) \quad (5)$$

$$CONCENTRATE = a^2 \quad (6)$$

$$DILATE = a^{\frac{1}{2}} \quad (7)$$

$$COMPLEMENT = 1 - a \quad (8)$$

where WTA , WA & OWA denote Winner takes all, Weighted Average and Ordered Weighted Average, respectively.

Figure 1 shows an example of an FPT, which was trained from a (wine) quality dataset. It represents the fuzzy concept – a fuzzy criterion for – wine with a high quality.

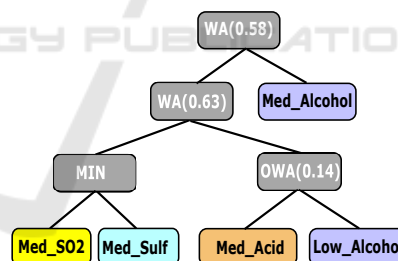


Figure 1: Tree representing the interpretable class "Good Quality Wine", showing each variable with different color.

The node labels of the tree illustrate their interpretation and not yet their implementation. In order to interpret the whole tree and grasp the fuzzy pattern it depicts, we start at the root node. It represents the final aggregation (a simple average in this case) and outputs the overall evaluation of the tree for a given instance (a wine). Then, we proceed to its children and so forth. The interpretation could be like this:

A high quality wine fulfills two criteria. We call these two criteria – the left and right subtrees of the root node – criterion I and criterion II. Criterion I is fulfilled if the alcohol concentration of the wine is high or its density is high. Criterion II is fulfilled, if the wine has a high concentration of sulfates or a

third criterion (III) is met. This is the case, if both alcohol concentration and the wines acidity is low.

The FPTs were created focusing on the representation of knowledge through a tree-shaped expression rather than representing it in the form of rules. The first FPT induction method was created by Huang, Gedeon and Nikraves (Huang et al., 2008), and refined in (Senge and Hüllermeier, 2011).

Hierarchical representation minimizes existing problems in rule-based systems, such as exponential increase in the number of rules with increasing entries and loss of interpretability when a large number of rules are required to achieve accuracy requirements. The tree is represented as a graph, favoring the human ability to recognize visual patterns, allowing the discovery of connections between the input variables and a class. These connections can be complicated to make when using models with a fixed set of rules.

To obtain a classifier one tree is created for each class, the classifier decision occurs in favor of the tree (class) that has the highest output value. Also, since each tree is considered a “logical description” of the class, it allows a more specific interpretation of the learning problem (Senge and Hüllermeier, 2011).

The FPT provides an alternative for the construction of accurate and interpretable fuzzy models.

2.2 Cartesian GP

GP concerns the automatic generation of programs inspired on the evolution theory. John Koza pioneered a tree-based GP form to represent computer programs using at that time LISP, an artificial intelligence computer language (Koza, 1992). CGP (Miller, 1999) is a flavor of GP with approximately 20 years of interesting and varied research works addressing a wide range of problem domains.

CGP uses graphs to represent solutions, and its key feature is the ability to encoding computational structures as directed graphs using redundant genes. This redundancy serves CGP to get a very adaptable representation by allowing the outputs nodes to either connect or disconnect to nodes from previous nodes in the directed graph.

The synthesis of FPTs by CGP was proposed in (dos Santos and do Amaral, 2015). The authors replaced the learning strategy proposed in (Senge and Hüllermeier, 2011) called Beam Search by CGP. The former learning strategy has a “greedy” characteristic which prevents a better exploration of the search space, increasing the possibility of the algorithm of being trapped in a sub-optimal solution. Also it suffered from the “curse of dimensionality”. If the number of input features and the width of the beam are

large, the algorithm will take a long time to evaluate all the possibilities; as a result, there will be an explosion in the number of possible combinations. The results reported in (dos Santos and do Amaral, 2015) indicated that FPTs synthesized by CGP are competitive with other classifier algorithms, and they are smaller than those obtained in (Senge and Hüllermeier, 2011).

Another example of the synthesis of FPTs by CGP can be found in (dos Santos et al., 2018). In that paper, the authors implement the improvements in CGP suggested by (Goldman and Punch, 2014) and implemented an NSGA-II strategy to deal with two conflicting objectives: the accuracy and the size of the tree.

Authors in (Wilson and Banzhaf, 2008) investigated the fundamental difference between traditional forms of Linear GP (LGP) and CGP, and their restrictions in connectivity.

The difference between graph-based LGP and CGP is the means with which they restrict the feed-forward connectivity of their directed acyclic graphs. In particular, CGP restricts connectivity based on the levels-back parameter while LGP’s connectivity is implicit and is under evolutionary control as a component of the genotype.

Experimental results show that CGP does not exhibit program bloat (Turner and Miller, 2014). However, using CGP to evolve programs in an arbitrary language can be tricky.

2.3 Grammatical Evolution

GE is a variant of GP, which differs in that the space of legal programs it can explore is described by a Backus-Naur Form (BNF) grammar (Ryan et al., 1998; O’Neill and Ryan, 2001) or Attribute Grammar (AG) (Patten and Ryan, 2015; Karim and Ryan, 2014; Karim and Ryan, 2011b; Karim and Ryan, 2011a), and it can evolve computer programs or arbitrary structures that can be defined in this way.

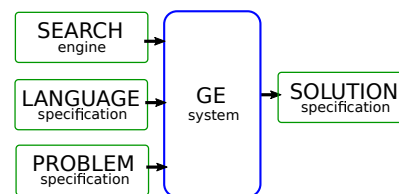


Figure 2: The GE system uses a search engine (typically a GA) to generate solutions for a given problem, by recombining the genetic material (genotype) and mapped onto programs (phenotype) according to a language specification (interpreter/compiler).

The modular design behind GE, as shown in Figure 2, means that any search engine can be used, although typically a variable-length Genetic Algorithm (GA) is employed to evolve a population of binary strings. After mapping each individual onto a program using GE, any program/algorithm can be used to evaluate those individuals.

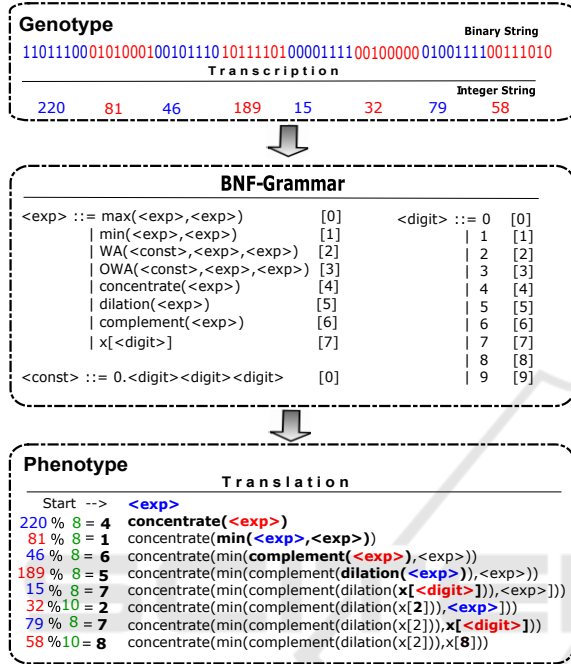


Figure 3: Example of a GE genotype-phenotype mapping process for the Iris dataset, where the binary genotype is grouped into codons (e.g. 8 bits; red & blue), transcribed into an integer string, then used to select production rules from a predefined grammar (BNF-Grammar), and finally translated into a sequence of rules to build a classifier (phenotype).

The linear representation of the genome allows the application of genetic operators such as crossover and mutation in the manner of a typical GA, unlike tree-based GP. Starting with the start symbol of the grammar, each individual’s chromosome contains in its codons (typically groups of 8 bits) the information necessary to select and apply the grammar production rules, in this way constructing the final program. The mapping process is illustrated with an example in Figure 3.

Production rules for each non-terminal are indexed starting from 0 and, when selecting a production rule (starting with the left-most non-terminal of the developing program) the next codon value in the genome is read and interpreted using the formula: $p = c \% r$, where c represents the current codon value, $\%$ represents the modulus operator, and r is the number of production rules for the left-most non-terminal.

If, while reading codons, the algorithm reaches the end of the genome, a wrapping operator is invoked and the process continues reading from the beginning of the genome. The process stops when all of the non-terminal symbols have been replaced, resulting in a valid program. If it fails to replace all of the non-terminal symbols after a maximum number of iterations, it is considered invalid and penalized with the lowest possible fitness.

3 FUZZY GE

This section introduces Fuzzy GE, an evolutionary approach to generate classifiers with linguistic labels. The aim is to create meaningful models applied to multi-classification problems.

The schematic of a fuzzy system is shown in Figure 4. The acquisition of a fuzzy rule base and the associated fuzzy set parameters from a set of input/output data is important in the development of fuzzy expert systems (Zadeh, 1965) with or without the aid of human expertise. Several automated techniques have been proposed for the solution of this knowledge acquisition problem. An advantage of using GE in the context of evolving structures for fuzzy rule base is the flexibility it gives in defining different partitioning geometries based on a chosen grammar (Wilson and Kaur, 2006).

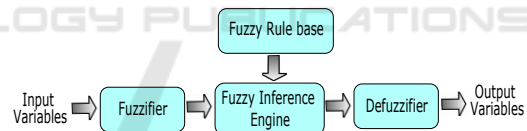


Figure 4: Fuzzy system.

3.1 Fuzzy Classification

There are many approaches to using evolve GP to classifiers (Espejo et al., 2009) and GE has been shown to be well suited for such a task (Nyathi and Pillay, 2018). One of the most popular methods for evolving a GP binary classifier is Static Range Selection, described by Zhang and Smart, who later proposed Centred Dynamic Class Boundary Determination (CDCBD) for multi-class classification (Zhang and Smart, 2004).

In binary classification, an input $\mathbf{x} \in \mathcal{R}^n$ has to be classified as belonging to one of the either two classes, ω_1 or ω_2 . In this method, the goal is to evolve a mapping $g(\mathbf{x}) : \mathcal{R}^n \rightarrow \mathcal{R}$. The classification rule \mathcal{R} states that pattern \mathbf{x} is labelled as belonging to class ω_1 if $g(\mathbf{x}) > r$, and belongs to ω_2 otherwise, where r is the decision boundary value.

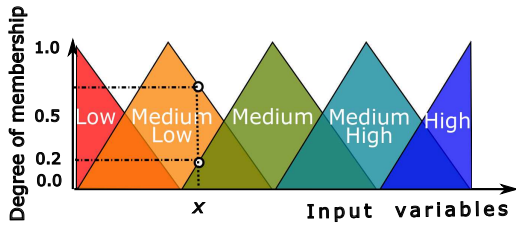


Figure 5: Fuzzy sets.

The fitness function is defined to maximize the total classification accuracy after \mathcal{R} is applied, normally setting the decision boundary to $r = 0$. A data sample is passed to the tree which yields a score. If the score is below the boundary it is labelled a particular class, and likewise it is labelled the other class if it is above the boundary. The process for CDCBD is similar, with $n-1$ boundaries existing, which can dynamically change to class each individual.

Both approaches only evolve one tree (or mapping) regardless of the number of classes and attempt to classify the individual based on its output from that tree. There are drawbacks to this approach as much effort need to be expended into designing or hand crafting class boundaries or creating systems to optimise them for each individual (Fitzgerald and Ryan, 2012), which becomes increasingly more difficult as the number of classes increases.

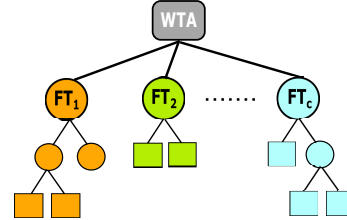
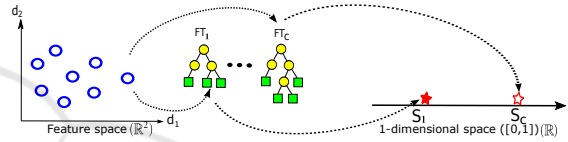
A FPT classifier requires that one FPT be evolved per class in the problem. Evolving multiple trees simultaneously adds a great deal of complexity to the problem. In general, care must be taken and special operators, particularly when using crossover, must be created (Ain et al., 2018; Lee et al., 2015). However, due to the separation between the search space and program space in GE, it is not necessary to create any special operators in FGE.

The novel method involves evolving only one large solution. This solution comprises of FPTs, with each class having its own FPT, and a decision node at its root. Each FPT can therefore be thought of as a subtree of a larger classifier tree, as seen in Figure 6, with the root node assigning the label to each individual. More formally, i mappings $f_i(\mathbf{x}) : \mathcal{R}^n \rightarrow [0,1]$, where i is the number of classes in the problem are evolved. The FPT, or subtree, $(f_1(\mathbf{x}) \dots f_i(\mathbf{x}))$ which confers the largest score to the individual is deemed the winner and the individual is labeled with the class that FPT represents.

For example, if $f_1(\mathbf{x})$ yielded the largest score the individual would be assigned to class 1. This is highlighted further in Figure 7, where the second tree yields the better score, S_c , the hollow star. The individual is therefore assigned class c . This is in contrast to the methods described above which only produce 1

score per individual and assign it a label based on the scores position relative to a boundary(s).

FGE does not require the use of any protected operators when evolving multiple trees due to the unique separation between genotype and phenotype and only needs grammar augmentation to address different problem types.

Figure 6: Pictorial representation of a multi-classifier evolved by FGE, where FT_c is the fuzzy tree for each available class, and at the root the winner take all (WTA).Figure 7: Graphical depiction of the mapping process from the feature space to a 1-dimensional space $[0,1]$ using a set of fuzzy trees FT_1 to FT_c .

3.2 Fuzzy Representation

In order to give a better interpretability to the evolved models, fuzzy logic is used to build more meaningful trees. To this end it uses the following five linguistic terms for fuzzy labels: low, medium-low, medium, medium-high, and high (see Figure 5). An uniform distribution of the input variables was considered for these fuzzy terms to have partitioning of the space.

The Fuzzy operators used are described in Section 2.1. The values of the inputs of the operated nodes are a and b . In the case of the Weighted Average (WA) and Ordered Weighted Average (OWA) operators, k will be a value created randomly within the range $[0,1]$. Only one input will be provided in the case of the concentration, dilation, and complement. Winner Takes All (WTA) will be the root node of every fuzzy tree. This function receives the score from each FPT and labels the individual corresponding to the highest scoring tree.

The genotype represents all the operators and fuzzy terms and different trees can be obtained depending on the grammar used on GE.

3.3 Lexicographic Parsimony Pressure

Accuracy, efficiency in training, and interpretability are often the dominant considerations when evolving a classifier. Maximising accuracy, or similarly minimising error, has traditionally been the main focus of research but interpretability has continued to grow in significance, with many papers and conferences now dedicated to the area (Adadi and Berrada, 2018). For a FPT to be interpretable or comprehensible, and therefore serve as a class descriptor, it is important the evolved solutions remain as small as possible and bloat is mitigated (Espejo et al., 2009). While GP’s ability to find high-dimensional, non-linear solutions is lauded, it can result in a significant loss of interpretability. Indeed, one of CGPs main advantages over standard GP and GP variants is its inherent lack of bloat (Turner and Miller, 2014).

Parsimony pressure is not GP-specific and has been used whenever arbitrarily-sized representations tended to get out of control. Such usage to date can be divided into two broad categories: parametric and Pareto parsimony pressure.

Parametric parsimony pressure uses size as a direct numerical factor in fitness, while Pareto parsimony pressure, uses size as a separate objective in a Pareto-optimization procedure.

In this work two sets of experiments were run. The first uses standard GE and the second is identical to the first, but implements a slightly modified lexicographic parsimony pressure (Luke and Panait, 2002) to bias the selection to prefer smaller solution size. The size is defined as the maximum depth of any of the n FPTs GE evolves for a particular solution.

4 EXPERIMENTAL SETUP

This section presents the experimental setup used. The approach is compared with several state of the art classification algorithms and one other FPT related method, FPT evolved using CGP (FCGP) (dos Santos et al., 2018; dos Santos, 2014). The same benchmark classification problems are used as previous for a fair comparison between the two approaches. The full experimental setup for CGP and each of the other benchmark classification techniques which FGE is compared against can be found in (dos Santos et al., 2018). The results are seen in Table 3

4.1 Datasets

The experiments are run on eight benchmark datasets, all of which can be found online in the UCI and

CMU repositories (Dua and Graff, 2017; StatLib, 2020). They include six binary classification problems and two multi-class problems. The size of the eight datasets, in addition to the number of classes and variables for each, are shown in Table 1.

Table 1: Benchmark datasets for binary and multiclass classification problems, taken from the UCI repository[†] and the CMU repository[§].

Datasets	Short	Class	Vars	Instances
Binary				
Lupus [§]	Lupus	2	3	87
Haberman [†]	Haber	2	3	306
Lawsuit [§]	Law	2	4	264
Transfusion [†]	Transf	2	4	748
Pima [†]	Pima	2	8	768
Australian [†]	Austr	2	14	690
Multiclass				
Iris [†]	Iris	3	4	150
Wine [†]	Wine	3	13	178

4.2 GE Parameters

The experiments were run for 50 generations with a population size of 500. Sensible Initialisation and effective crossover were used (Ryan and Azad, 2003). 5-fold cross-validation was used. This was repeated 5 times for a total of 25 runs.

Table 2: List of the main parameters used to run GE.

Parameter	Value
Folds	5
Runs	25 (5 per fold)
Total Generations	50
Population	500
Replacement	Tournament
Crossover	0.9 (Effective)
Mutation	0.01
Initialisation	Sensible

These values result in a higher computational cost than those of the CGP experiments (dos Santos, 2014). The full experimental setup can be seen in Table 2.

The grammar used for binary classification can be seen in Figure 8. The *WTA* node contains two $\langle exp \rangle$ non-terminals which need to be expanded. When fully expanded, these will be the FPT for each class. For binary classification two FPTs are required. For multi-class classification the grammar simply needs to be augmented by adding more $\langle exp \rangle$ symbols in the expression. Three classes re-

quire three $\langle exp \rangle$ symbols and so on. Constants were created using the standard GE approach of digit concatenation (Azad and Ryan, 2014).

```

< start > ::= WTA(< exp >, < exp >)
< exp > ::= max(< exp >, < exp >) |
           min(< exp >, < exp >) |
           WA(< const >, < exp >, < exp >) |
           OWA(< const >, < exp >, < exp >) |
           concentrate(< exp >) |
           dilation(< exp >) |
           complement(< exp >) |
           x1 | x2 | x3 | ...
< const > ::= 0. < digit > < digit > < digit >
< digit > ::= 0 | 1 | 2 | ...

```

Figure 8: Grammar used to evolve a Fuzzy Pattern Tree for a binary dataset. The *WTA* node can be augmented by adding extra $\langle exp \rangle$ to include as many subtrees as necessary, making it a multi-class grammar.

4.3 Fitness Function

The fitness function used for FGE, shown in Eq. 10, seeks to minimise the RMSE for each individual. The benchmark datasets used are reasonably balanced and therefore a non standard fitness function, such as cross entropy, was deemed not to be needed. However, future experiments run on very unbalanced data may need to modify this.

$$RMSE = \sum_{i=1}^n \frac{(\hat{y}_i - y_i)^2}{n} \quad (9)$$

$$F = 1 - RMSE \quad (10)$$

The fitness function for FGE with lexicographic parsimony pressure, seeking interpretability in solutions, is calculated by penalising the solution by its size. It is computed as follows;

$$F_L = 1 - RMSE \times 0.99 - MaxDepth \times 0.01 \quad (11)$$

Experiments using fitness function F are denoted as *FGE* in the results section, while experiments using F_L are identified by *FGE-L*

The mean depth of a solution is the average max depth of each FPT evolved. For a binary classifier C with FPT_1 and FPT_2 , the mean depth would be:

$$MeanDepth(C) = \frac{1}{2} [MaxDepth(FPT_1) + MaxDepth(FPT_2)] \quad (12)$$

5 RESULTS

The experimental results are summarized in Table 3 showing the best performance from 25 runs of FGE. Other methods performances are taken from (dos Santos et al., 2018). The best result across each problem are highlighted in bold. Datasets are sorted to first show the binary problems (1-6), followed by the multi-classification problems (7-8).

The first two columns show the results for FGE and FGE with lexicographic pressure applied, respectively. The third column shows the results for CGP, the fourth Support Vector Machine with Linear Kernel (SVM-L) and fifth Random Forest (RF). The sixth column shows Support Vector Machine with Radial Basis Function Kernel (SVM-R) and lastly column seven shows the Pattern Tree Top-Down Epsilon (PTTDE), the original technique for creating FPTs (Senge and Hüllermeier, 2011). In this paper epsilon is set to 0.25%, where epsilon controls the amount of improvement required to continue to grow the tree.

A Friedman test was carried out on the data to compare the performance of the classifiers. This test showed no evidence there was one classifier that was statistically significantly better than all others.

FGE achieves very competitive results with the previous experiments, only being glaringly outperformed in one benchmark, Wine, in which it was the worst performing classifier. FGE attained best performance of 83% on this benchmark, compared to 98% found by SVM, RF and PTTDE. It was also noticeably worse than the result achieved by CGP, which reached 90%.

FGE accomplished the best performance in 3 problems: (i) Haberman - FGE achieves 74%, outperforming all and equalling PTTDE as the best result; (ii) Australian - FGE and FGE-L score 86% for a tie in best performing classifier; and (iii) Iris - FGE and FGE-L again match the best performing classifier attaining 96%. Interestingly, FGE-L achieves best in class performance, 77%, on the Transfusion problem.

As well as these results, FGE and FGE-L attain competitive results on the rest of the classification problems with the exception of the Lupus dataset. FGE reaches similar performance as FCGP, 73% and 74% respectively, but PTTDE produced the best accuracy, 77%. FGE-L performs identically, finding 73% accuracy.

FGE-L can be seen to find substantially smaller solutions than those found by FGE as shown in 4, where SizeReduc denotes the reduction on the averaged tree size achieved by FGE-L against FGE. Note that an FPT containing just a leaf node would have depth of 0.

Table 3: Classification performance comparison of FGE versions against previous related work results, showing the classification error on the test data for the best solution found. Bold indicates the best performance.

Dataset	FGE	FGE-L	FCGP	SVM-L	RF	SVM-R	PTTDE
Lupus	0.73	0.73	0.74	0.74	0.62	0.73	0.77
Haber	0.74	0.72	0.73	0.72	0.65	0.71	0.74
Law	0.96	0.94	0.93	0.99	0.97	0.96	0.94
Transf	0.76	0.77	0.76	0.76	0.70	0.73	0.77
Pima	0.74	0.74	0.72	0.77	0.77	0.71	0.76
Austr	0.86	0.86	0.85	0.86	0.79	0.85	0.85
Iris	0.96	0.96	0.95	0.96	0.95	0.95	0.95
Wine	0.83	0.83	0.90	0.98	0.98	0.98	0.98

Across all problems tested remarkably smaller trees were found by FGE-L. In particular, the Haber, Pima and Australian problems were all seen to reduce in size by over 80%. Parsimony pressure does not appear to affect the performance, however, with only a decrease in accuracy seen in two problems: Haber and Lawsuit. Strikingly, there was an increase in the performance on the Transfusion problem by 1%. The major reduction seen in the size of the final solutions found in every experiment may hint at bloat being a problem in FGE. A pressure of 1% of size was applied in these experiments but tuning the pressure applied is an avenue for future research. FGE-L was the best performing classifier on the Transfusion, Iris and Australian problems. On the problems studied there appears to be very little, or sometimes none, trade off in performance associated with evolving smaller trees. It is possibly the case that the global optimums for these problems were smaller trees, but this requires further study, on larger, more complicated benchmarks. These results do seem to strongly suggest that bloat may be an issue in FGE.

The trees found using CGP are much smaller than those found using FGE but larger than those using FGE-L. When the search is biased towards smaller sized individuals FGE-L finds smaller solutions in 7 problems. Due to these small sizes, FPTs found using FGE-L should lead to very interpretable results.

Overall the best performing method was SVM-L, achieving best performance on 5 of the the benchmark problems. SVM-L does not allow any interpretability of its solutions. FGE was best performing on 3 problems, FGE-L was best performing on 3 problems and FCGP was not best on any. FGE beat or equalled FCGP in 6/8 problems studied and FGE-L evolved the smallest trees in all but one problem, Lupus, and was able to beat FCGP in 5/8 problems.

The mean size of the final trees found by FGE, FGE-L with parsimony pressure and CGP are shown in Table 4, best results are in bold.

Table 4: Average size comparison in terms of the tree depth between fuzzy pattern trees approaches; FCGP, FGE, and FGE-L. Best results are in bold. SizeReduce is the average size reduction seen in FGE-L vs FGE.

Dataset	FCGP	FGE	FGE-L	SizeReduce
Lupus	1.65	7.84	2.38	70%
Haber	1.85	9.42	0.2	98%
Law	1.05	5.02	0.98	79%
Transf	2	6.76	1.54	77%
Pima	1	6.7	1	85%
Austr	1.5	5.12	0.92	82%
Iris	1.24	1.8	0.64	64%
Wine	1	2.47	0.68	72%

6 CONCLUSIONS

This paper proposes a new way of inducing Fuzzy Pattern Trees using Grammatical Evolution as the learning algorithm, FGE. FPT is a viable alternative to the classic rules-based fuzzy models since their hierarchical structure allows a more compact representation and a compromise between the accuracy and the simplicity of the model. The experimental results showed that FGE has a competitive performance in the task of classification with respect to some of the best classifiers available. Crucially it also provides an interpretable model, that is, the knowledge obtained in the learning process can be extracted from the model and presented to a user in comprehensible terms.

A promising aspect of the present work is that several future lines of research can be explored. The proposed algorithms should be evaluated on other machine learning problems, such as unsupervised clustering.

One interesting possibility is to try different approaches to reduce the tree size, including regularization or encapsulation (Murphy and Ryan, 2020). Furthermore, the use of different sets of grammar for GE could be explored.

A final avenue for future research is to empirically

examine if smaller tree size does offer more user interpretability. It is possible that another metric, such as the number of variables used or the presence of particular subtrees, may grant better interpretability and must be investigated.

ACKNOWLEDGEMENTS

The authors thank the anonymous reviewers for their time, comments and helpful suggestions. The authors are supported by Research Grants 13/RC/2094 and 16/IA/4605 from the Science Foundation Ireland and by Lero, the Irish Software Engineering Research Centre (www.lero.ie). The third and fourth authors are partially financed by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

REFERENCES

- Adadi, A. and Berrada, M. (2018). Peeking inside the black-box: A survey on explainable artificial intelligence (xai). *IEEE Access*, 6:52138–52160.
- Ain, Q. U., Al-Sahaf, H., Xue, B., and Zhang, M. (2018). A multi-tree genetic programming representation for melanoma detection using local and global features. In Mitrovic, T., Xue, B., and Li, X., editors, *AI 2018: Advances in Artificial Intelligence*, pages 111–123, Cham. Springer International Publishing.
- Arrieta, A. B., Díaz-Rodríguez, N., Ser, J. D., Bennetot, A., Tabik, S., Barbado, A., Garcia, S., Gil-Lopez, S., Molina, D., Benjamins, R., Chatila, R., and Herrera, F. (2020). Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information Fusion*, 58:82–115.
- Azad, R. M. A. and Ryan, C. (2014). The best things don't always come in small packages: Constant creation in grammatical evolution. In Nicolau, M., Krawiec, K., Heywood, M. I., Castelli, M., García-Sánchez, P., Merelo, J. J., Rivas Santos, V. M., and Sim, K., editors, *Genetic Programming*, pages 186–197, Berlin, Heidelberg. Springer Berlin Heidelberg.
- Carvalho, D. V., Pereira, E. M., and Cardoso, J. S. (2019). Machine Learning Interpretability: A Survey on Methods and Metrics. *Electronics*, 8(8):832.
- Cordón, O. (2011). A historical review of evolutionary learning methods for Mamdani-type fuzzy rule-based systems: Designing interpretable genetic fuzzy systems. *International journal of approximate reasoning*, 52(6):894–913. Publisher: Elsevier.
- Došilović, F. K., Brčić, M., and Hlupić, N. (2018). Explainable artificial intelligence: A survey. In *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 0210–0215.
- dos Santos, A. R. (2014). *Síntese de Árvores de Padrões Fuzzy através de Programação Genética Cartesiana*. PhD thesis, Dissertação de mestrado, Universidade do Estado do Rio de Janeiro.
- dos Santos, A. R. and do Amaral, J. L. M. (2015). Synthesis of Fuzzy Pattern Trees by Cartesian Genetic Programming. *Mathware & soft computing*, 22(1):52–56.
- dos Santos, A. R., do Amaral, J. L. M., Soares, C. A. R., and de Barros, A. V. (2018). Multi-objective fuzzy pattern trees. In *2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–6.
- Dua, D. and Graff, C. (2017). UCI machine learning repository.
- Espejo, P. G., Ventura, S., and Herrera, F. (2009). A survey on the application of genetic programming to classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 40(2):121–144.
- Fitzgerald, J. and Ryan, C. (2012). Exploring boundaries: optimising individual class boundaries for binary classification problem. In *Proceedings of the 14th annual conference on Genetic and evolutionary computation*, pages 743–750.
- Goldman, B. W. and Punch, W. F. (2014). Analysis of cartesian genetic programming's evolutionary mechanisms. *IEEE Transactions on Evolutionary Computation*, 19(3):359–373. Publisher: IEEE.
- He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. *arXiv:1502.01852 [cs]*. arXiv: 1502.01852.
- Herrera, F. (2008). Genetic fuzzy systems: taxonomy, current research trends and prospects. *Evolutionary Intelligence*, 1(1):27–46. Publisher: Springer.
- Hüllermeier, E. (2005). Fuzzy methods in machine learning and data mining: Status and prospects. *Fuzzy Sets and Systems*, 156(3):387–406.
- Huang, Z., Gedeon, T. D., and Nikraves, M. (2008). Pattern trees induction: A new machine learning method. *Trans. Fuz Sys.*, 16(4):958–970.
- Karim, M. R. and Ryan, C. (2011a). Degeneracy reduction or duplicate elimination? an analysis on the performance of attributed grammatical evolution with lookahead to solve the multiple knapsack problem. In *Nature Inspired Cooperative Strategies for Optimization, NICSO 2011, Cluj-Napoca, Romania, October 20-22, 2011*, pages 247–266.
- Karim, M. R. and Ryan, C. (2011b). A new approach to solving 0-1 multiconstraint knapsack problems using attribute grammar with lookahead. In *Genetic Programming - 14th European Conference, EuroGP 2011, Torino, Italy, April 27-29, 2011. Proceedings*, pages 250–261.
- Karim, M. R. and Ryan, C. (2014). On improving grammatical evolution performance in symbolic regression with attribute grammar. In *Genetic and Evolutionary Computation Conference, GECCO '14, Vancouver, BC, Canada, July 12-16, 2014, Companion Material Proceedings*, pages 139–140.

- Koza, J. R. (1992). *Genetic Programming - On the programming of Computers by Means of Natural Selection*. Complex adaptive systems. MIT Press.
- Lee, J.-H., Anaraki, J. R., Ahn, C. W., and An, J. (2015). Efficient classification system based on fuzzy-rough feature selection and multitree genetic programming for intension pattern recognition using brain signal. *Expert Systems with Applications*, 42(3):1644 – 1651.
- Luke, S. and Panait, L. (2002). Lexicographic parsimony pressure. In *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation, GECCO'02*, page 829–836, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.
- Miller, J. F. (1999). An empirical study of the efficiency of learning boolean functions using a cartesian genetic programming approach. In *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation - Volume 2, GECCO'99*, page 1135–1142, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.
- Murphy, A. and Ryan, C. (2020). Improving module identification and use in grammatical evolution. In Jin, Y., editor, *2020 IEEE Congress on Evolutionary Computation, CEC 2020*. IEEE Computational Intelligence Society, IEEE Press.
- Nyathi, T. and 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.
- O'Neill, M. and Ryan, C. (2001). Grammatical evolution. *IEEE Trans. Evolutionary Computation*, 5(4):349–358.
- Patten, J. V. and Ryan, C. (2015). Attributed grammatical evolution using shared memory spaces and dynamically typed semantic function specification. In *Genetic Programming - 18th European Conference, EuroGP 2015, Copenhagen, Denmark, April 8-10, 2015, Proceedings*, pages 105–112.
- Ryan, C. and Azad, R. M. A. (2003). Sensible initialisation in grammatical evolution. In *GECCO*, pages 142–145. AAAI.
- Ryan, C., Collins, J. J., and O'Neill, M. (1998). Grammatical evolution: Evolving programs for an arbitrary language. In Banzhaf, W., Poli, R., Schoenauer, M., and Fogarty, T. C., editors, *EuroGP*, volume 1391 of *Lecture Notes in Computer Science*, pages 83–96. Springer.
- Senge, R. and Hüllermeier, E. (2011). Top-down induction of fuzzy pattern trees. *IEEE Transactions on Fuzzy Systems*, 19(2):241–252.
- StatLib (2020). Statlib – datasets archive.
- Turner, A. J. and Miller, J. F. (2014). Cartesian genetic programming: Why no bloat? In Nicolau, M., Krawiec, K., Heywood, M. I., Castelli, M., García-Sánchez, P., Merelo, J. J., Rivas Santos, V. M., and Sim, K., editors, *Genetic Programming*, pages 222–233, Berlin, Heidelberg. Springer Berlin Heidelberg.
- Wilson, D. and Kaur, D. (2006). Fuzzy classification using grammatical evolution for structure identification. pages 80 – 84.
- Wilson, G. and Banzhaf, W. (2008). A comparison of cartesian genetic programming and linear genetic programming. pages 182–193.
- Yi, Y., Fober, T., and Hüllermeier, E. (2009). Fuzzy operator trees for modeling rating functions. *International Journal of Computational Intelligence and Applications*, 8:413–428.
- Zadeh, L. (1965). Fuzzy sets. *Information and Control*, 8(3):338 – 353.
- Zhang, M. and Smart, W. (2004). Multiclass object classification using genetic programming. In *Workshops on Applications of Evolutionary Computation*, pages 369–378. Springer.