A Threshold-free Classification Mechanism in Genetic Programming for High-dimensional Unbalanced Classification

Wenbin Pei[1] Bing Xue[1] Lin Shang[2] and Mengjie Zhang[1]
1. School of Engineering and Computer Science, Victoria University of Wellington,
PO Box 600, Wellington 6140, New Zealand
2. State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China
Email: {Wenbin.Pei, Bing.Xue, Mengjie.Zhang}@ecs.vuw.ac.nz, shanglin@nju.edu.cn

***Abstract*—Class imbalance is an unavoidable issue in many real-world applications. Learning from unbalanced data, classifiers are often biased toward the majority class, while the minority class is important as well (even more important in many cases). How the issue of class imbalance is addressed becomes more challenging if a classification task further encounters the high dimensionality issue. This paper proposes a new genetic programming (GP) approach to high-dimensional unbalanced classification. A new classification mechanism is proposed for GP to improve its classification performance. This new classification mechanism is independent of a classification threshold to separate the majority class and the minority class. The effectiveness of the proposed method is examined on seven high-dimensional unbalanced datasets. Experimental results indicate that the proposed GP method often performs better than other GP methods that use a fitness function to solve the issue of class imbalance, in terms of classification performance and training time.**

Keywords: Unbalanced classification, High dimensionality, Genetic programming

I. Introduction

Unbalanced classification has many real-world applications, such as fraud detection [1] and medical diagnosis [2]. In these tasks, the minority class is often more important than the majority class. However, class imbalance often leads to classifiers biased toward the majority class. This is because the majority class typically outnumbers the minority class, but the standard classification algorithms assume all instances (from different classes) being equally important (with the same misclassification cost).

To address the issue of class imbalance, sampling methods (mainly including undersampling [3] and oversamping methods [4]–[6]) are commonly used to re-balance an unbalanced dataset before using classification algorithms. However, sampling methods have to change the original information. Moreover, it is often hard for undersampling methods to avoid information loss when judging which instances are to be excluded from the majority class. In oversampling methods, some instances from the minority class are learned repeatedly, or some new instances are generated for the minority class, which may increase an additional computation cost. Cost-sensitive learning [7] is a new learning paradigm that takes the costs (mainly misclassification costs) into consideration, so that different classification mistakes are treated differently. Cost-sensitive learning has been widely applied to unbalanced classification. However, for most existing cost-sensitive algorithms, the cost matrices required by cost-sensitive algorithms are often manually designed.

Recently, the increasing number of unbalance datasets further encounter the issue of high dimensionality. High dimensionality often makes it more challenging for classifiers to correctly discriminate the boundary between the majority class and the minority class [8]. Genetic programming (GP) [9] is a population-based evolutionary algorithm, which automatically evolves solutions to a problem. When GP is used to evolve classifiers, a GP tree (also called a program or individual) could be seen as a classifier. It is noteworthy that GP is able to automatically select the informative features when classifiers are constructed.

However, because of the uneven class distribution, the classification performance of GP is degraded. The constructed classifiers often achieve a high accuracy on the majority class but a low accuracy on the minority class. This is because, for many GP methods, the overall classification accuracy is often used as a fitness function, which fails to take class imbalance into consideration. As a result, GP is biased toward the majority class because the majority class contributes more to the overall classification accuracy than the minority class.

In GP, development of a fitness function is an effective solution to address the problem of class imbalance [10], [11]. When an accuracy measure, such as the weighted average classification accuracy, is used as a fitness function to evolve classifiers, a classification threshold TH is required to separate the output of a program for predictions in binary classification. By taking an instance as an input, if an output value of a program is greater than or equal to TH, this instance is classified to the minority class, otherwise it is classified to the majority class. Usually, TH is set to be 0 for separating the original program outputs or 0.5 if the outputs are normalized to a range of [0, 1].

In unbalanced classification, this classification threshold setting is important [10]. In cost-sensitive learning, the threshold-moving methods are often effective, based on the idea that

978-1-7281-6929-3/20/$31.00 ©2020 IEEE

the classification threshold is moving toward the inexpensive instances (i.e. instances with the lower misclassification cost), so that those expensive instances with the higher misclassification cost are easier to be correctly classified [12], [13]. However, a new moved classification threshold is calculated by misclassification cost values that are often provided by domain experts.

In this paper, we propose an easy classification mechanism that is independent of a predefined classification threshold, working with a fitness function for GP in high-dimensional unbalanced classification.

Goals: The overall goal of this paper is to investigate a classification mechanism for GP to improve its performance for high-dimensional unbalanced classification. This goal is composed of the following sub-goals:

- Develop a new classification mechanism,
- Develop a new fitness function working with the new classification mechanism,
- Investigate whether the proposed method enhances the classification performance of GP for unbalanced classification,
- Investigate whether the proposed method is time-efficient and achieves at least similar performance as other existing classification methods.

The remainder of this paper is organised as follows. Section II introduces important background knowledge. In Section III, the proposed method is introduced. Section IV is devoted to introducing experiment design. Results are reported and discussed in Section V. In Section VI, the two evolved trees are shown for further analysis. Finally, the conclusion of this paper is drawn in section VII.

II. Background

A. Methods to Solve the Issue of Class Imbalance

a) At the data level: Sampling methods are used to rebalance an unbalanced dataset before classification algorithms are applied. Generally speaking, sampling methods have three groups, including undersampling methods [3], oversampling methods [4], [6], and hybrid sampling methods [14]. Hybrid sampling methods combine oversampling and undersampling. New instances are created by an oversampling method for the minority class, and then some of them, which are less useful, are removed.

b) At the algorithmic level: Cost-sensitive learning is the most popular method in this group, which incorporates misclassification costs into classification paradigms to construct cost-sensitive classifiers [15], [16]. For support vector machines (SVMs) or other kernel-based classification algorithms, a kernel function is modified to address the problem of class imbalance [17].

B. Methods to Solve the High Dimensionality Issue

a) Provision of more instances: In high-dimensional classification, when features typically outnumber instances, the data space is sparse. Accordingly, it is often very challenging to discover the useful pattern or rules. In addition, because of the lack of the training instances, classifiers are often overfitting to training data. An easy solution is to provide more instances [18]. However, in many situations, it is often difficult to collect new useful instances.

b) Feature selection: Feature selection is a popular solution to solve high dimensionality issue. The limited number of instances are described by too many features, while many of them are irrelevant or redundant features. The goal of feature selection is to select the smallest subset of features that are necessary and sufficient to describe the target labels [19]. The selected features, instead of all features, are used for classification. However, in high-dimensional classification, feature selection is a challenging task because the search space is large (the total number of possible solutions is 2^n, n is a number of original features). Moreover, there might be two-way, three-way or complex multi-way interactions among features [20]. Therefore, a feature, which is weakly relevant to target labels, might become important when using it together with other features. Oppositely, an important feature may become redundant when using it together with other features.

Feature selection with unbalanced data is more challenging. This is because the selected features should benefit the majority class as well as the minority class [21]. If the issue of class imbalance is not addressed, the selected features are possibly biased toward the majority class. As a result, the classifiers using these biased features are more likely to be biased toward the majority class.

C. GP for Classification

In GP, for each tree, its internal nodes are taken from a function set, while the leaf nodes are taken from a terminal set. Usually, a GP tree is seen as a classifier, which could be translated into a mathematical expression [22]. The output of this mathematical expression is used to classify instances. Usually, all features are fed to GP as terminals to evolve classifiers. However, not all features are used by each GP tree. Informative features are automatically selected to improve the classification performance of a GP tree.

However, in unbalanced classification, classifiers evolved by GP are often biased toward the majority class, resulting in a performance bias issue. In [23], GP is used for learning fuzzy rule bases, where the linguistic variables are used in a hierarchical way, and synthetic minority over-sampling technique (SMOTE) is adopted to address the issue of class imbalance. Hunt et al. [24] investigate the use of sampling methods in GP to solve the issue of class imbalance. In [25], [26], cost-sensitive learning is used with GP for unbalanced classification. In [10], [11], [27], [28], new fitness functions are proposed for GP in unbalanced classification. Bhowan et al. [29]–[32] develop multi-objective GP (MOGP) methods for unbalanced classification, where the accuracies of the majority class and the minority class are used as two potentially conflicting objectives to be evolved together.

Sampling methods often need to change the original data distribution and require some additional computation. Cost-sensitive learning methods require the cost matrices that are

often manually designed. For a single-objective GP method, a fitness function can be used to address the issue of class imbalance. Generally speaking, a single-objective GP method is often time-efficient than MOGP methods. This is because MOGP methods have more components than a single-objective GP method. Moreover, it is often time-consuming to obtain the complete Pareto front in MOGP methods. Therefore, this paper focus on single-objective GP, where a new classification mechanism is developed, working with a fitness function, to address the issue of class imbalance for high-dimensional unbalanced classification.

III. The Proposed Method

This section introduces the proposed method, called **G**enetic **P**rogramming with a **T**hreshold-**f**ree **C**lassification **M**echanism (**GPTFCM**).

A. The Threshold-free Classification Mechanism

As introduced previously, when an accuracy measure is used as a fitness function, a classification threshold TH is required for separating the output of a program to predict the majority class and the minority class. Usually, TH is set to be 0. However, in unbalanced classification, the main limitation of using a predefined threshold is that the classification performance of a program (as a classifier) is only evaluated at this predefined threshold, but the classification performance of this program may be different by using other thresholds to separate its output [10]. Therefore, for unbalanced classification, area under a curve (AUC) is often used as a fitness function, which evaluates true positive rate and false positive rate many times by varying thresholds to provide an accurate rendition of the curve. However, GP using AUC as a fitness function is very time-consuming [10].

Accordingly, this paper investigates a threshold-free classification mechanism for GP. The idea is explained by Figure 1. In binary classification, for each GP tree, its left sub-tree and right sub-tree are able to learn input data independently. When learning from the minority class (denoted as Min), the output value of the left sub-tree (denoted as $output1$) is expected to be greater than or equal to the output value of the right sub-tree (denoted as $output2$). Oppositely, when learning from the majority class (denoted as Maj), $output1$ is expected to be smaller than $output2$.

The proposed method (GPTFCM) is based on strongly-typed genetic programming (STGP), because STGP makes it possible to evolve GP programs based on a predefined program representation. In STGP, every terminal has a type, and correspondingly, each function has types for its arguments and its returned values [9].

The Function Set and The Terminal Set

In Table I, we report the terminal set and the function set in the proposed method (GPTFCM) for binary classification. The function set has eight functions in total, including four basic arithmetic functions (i.e. +, −, × and protected division ÷), and four functions (i.e. if, $Left$, $Right$ and $Root$). The protected division ÷ returns zero when dividing by zero.

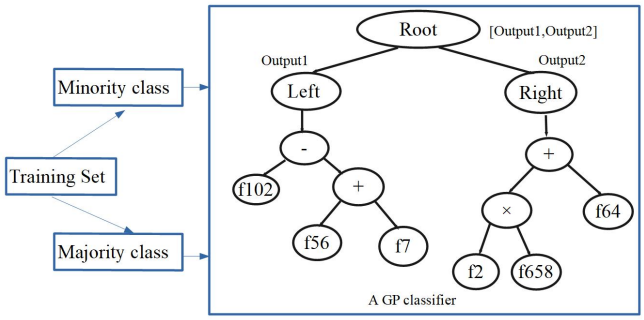

Figure 1: Classification mechanism in the training process

A conditional function if takes three arguments (when the first argument is negative, the second argument is returned, otherwise it returns the third argument). $Left$ function takes one argument, and it directly returns the value of this argument (data type of the returned value is $Oput1$). The role of $Left$ function is to transfer an output value of the left sub-tree to the root node of a tree. Similar to $Left$ function, $Right$ function also has one argument, and it directly returns the value of its argument (data type of the returned value is $Oput2$). $Right$ function transfers an output value of the right sub-tree to the root node of a tree. $Root$ function has two arguments (i.e. $argu1$ with the data type $Oput1$ and $argu2$ with the data type $Oput2$). $Root$ function directly returns $[argu1, argu2]$ (i.e. a two-value vector). $Root$ function accepts outputs of the left sub-tree and right sub-tree, and combines them together as an output of a tree. Therefore, for a tree, when taking instance i as an input, its output is a two-value vector $[Output_i1,\ Output_i2]$.

Classification Predictions in a Training Set

- For instance i from Min, if $Output_i1 \geq Output_i2$, then this instance is correctly classified;
- For instance j from Maj, if $Output_j1 < Output_j2$, then this instance is correctly classified.

B. Fitness Function

In a population, every individual is evaluated by a fitness function to measure its classification performance. In the proposed method, the fitness function consists of two parts. The first part is to measure the classification performance by using the squared root of the product of the accuracy on the minority class and the accuracy on the majority class. The second part is used to measure the mean values of outputs of the two sub-trees when Min is inputted into the left sub-tree and Maj is inputted into the right sub-tree, respectively. The fitness function is defined as:

$$R\_L = \sqrt{\frac{C_{Min}}{C_{Min}+M_{Min}} * \frac{C_{Maj}}{C_{Maj}+M_{Maj}}} + I(Min, Maj) \qquad (1)$$

Table I: The function set and terminal set

Terminal Set		Function Set		
Name	Type	Name	Type (Input)	Type (Output)
		• $Root$	$[Oput1, Oput2]$	$Oput$ (two-value vector)
		• $Left$	$[float]$	$Oput1$ (float)
		• $Right$	$[float]$	$Oput2$ (float)
• Features of a dataset	$float$	• +	$[float, float]$	$float$
• A random constant	$float$	• $-$	$[float, float]$	$float$
		• $\times$	$[float, float]$	$float$
		• $\div$ (protected)	$[float, float]$	$float$
		• if	$[float, float, float]$	$float$

Note: $Oput1$ and $Oput2$ are different types in an evolved tree, even though they are actually the same type (float, but it is redefined as different types in the tree representation).

where C_{Min} (or C_{Maj}) indicates how many instances from Min (or Maj) are correctly classified; M_{Min} (or M_{Maj}) indicates how many instances from Min (or Maj) are incorrectly classified. In *R_L*, $I(Min, Maj)$ is:

$$I(Min, Maj) = \begin{cases} 1, if \frac{\Sigma_{i=1}^{|Min|} Output_i 1}{|Min|} > 0 \text{ and } \frac{\Sigma_{j=1}^{|Maj|} Output_j 2}{|Maj|} > 0 \\ 0, \ otherwise \end{cases}$$

The main steps in the fitness evaluation are summarized as follows:

1) $C_{Min} = C_{Maj} = M_{Min} = M_{Maj} = 0$.

2) Classification process:

   For instance i from Min, if $Output_i 1 \geq Output_i 2$, then $C_{Min} = C_{Min} + 1$, else $M_{Min} = M_{Min} + 1$; For instance j from Maj, if $Output_j 1 < Output_j 2$, then $C_{Maj} = C_{Maj} + 1$, else $M_{Maj} = M_{Maj} + 1$.

3) Calculate $\mu_{Min} = \frac{\Sigma_{i=1}^{|Min|} Output_i 1}{|Min|}$.

4) Calculate $\mu_{Maj} = \frac{\Sigma_{j=1}^{|Maj|} Output_j 2}{|Maj|}$.

5) If $\mu_{Min} > 0$ and $\mu_{Maj} > 0$, then $I(Min, Maj) = 1$, else $I(Min, Maj) = 0$.

6) Calculate the fitness value of a tree according to Eq. (1).

C. The Overall Design of the Proposed Method

The overall design is shown in Figure 2. The training set is split into the majority class and the minority class. After initializing a population in terms of trees, the fitness of every tree (as a classifier) is calculated by Eq. (1). Based on the fitness values, the better trees are selected by tournament selection. Genetic operators, i.e. mutation, crossover and elitism, are used to create a new population. The evolutionary process is stopped until a termination criterion is satisfied. After the training process, the best tree from the final generation is chosen as a classifier to predict class labels of unseen instances in a test set. For each instance, if the output value of the left sub-tree is greater than or equal to that of the right sub-tree, this instance is predicted into Min, otherwise it is predicted into Maj.

It is straightforward to generalize this method to multi-class classification. For n-class classification, a tree is constituted by n sub-trees and a root node. For example, for a three-class classification task, a GP tree consists of three sub-trees and a root node. Therefore, when taking instance i as an input, the output of a tree is a three-value vector, i.e. $[Output_i 1, Output_i 2, Output_i 3]$. On a test set, for instance i, if $Output_i 1 \geq Output_i 2$ and $Output_i 1 \geq Output_i 3$, then this instance is predicted into $Class$ 1; if $Output_i 2 > Output_i 1$ and $Output_i 2 \geq Output_i 3$, then this instance is predicted into $Class$ 2; if $Output_i 3 > Output_i 1$ and $Output_i 3 > Output_i 2$, then this instance is predicted into $Class$ 3.

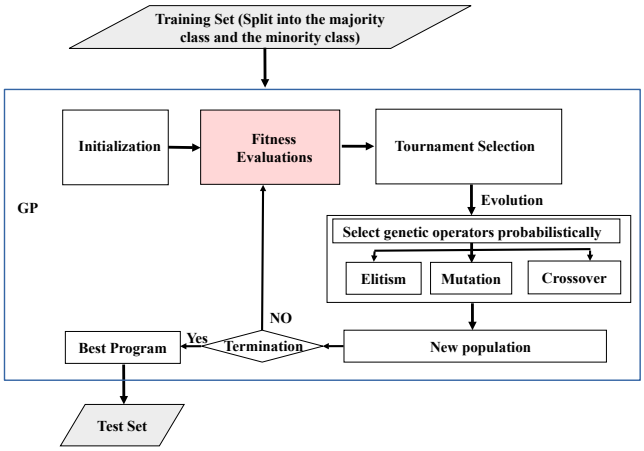

Figure 2: Overall design of GPTFCM

IV. Experiment Design

A. Datasets

In the experiments, gene expression datasets are used to examine the classification performance of the proposed method. This is because many gene expression datasets have a large number of features (thousands), but a few number of instances are available. Therefore, these datasets may have the high dimensionality issue. Moreover, many of these datasets are unbalanced. These datasets can be downloaded at *https://schlieplab.org/Static/Supplements/CompCancer/datasets.htm*.

More details of these datasets are reported in Table II, including the number of features and instances, and class imbalance ratio ($IR = \frac{|Maj|}{|Min|}$, where $|Maj|$ is the number

Table II: Dataset description

Dataset	#Features	#Instances	IR (Approximately)
Armstrong-2002-v1	1,081	72	2
Golub_1990	1,868	72	2
Colon	2,000	62	2
Leukemia	7,129	72	2
DLBCL	5,469	77	3
Yeoh-2002-v1	2,526	248	5
Lung	12,600	158	8

instances in Maj and $|Min|$ is the number of instances in Min).

The stratified sampling is employed to split a dataset into a training set and a test set (70% as a training set and 30% as a test set), to ensure the same class imbalance ratio in the training set and the test set (also the same as that in the original whole dataset). To examine the classification performance of the proposed method on a highly unbalanced dataset, Lung is changed to a binary dataset ($IR = 8$) by using label 1 as the majority class and label 2 as the minority class.

B. Baseline Methods

The proposed method is compared with GP methods and non-GP methods from machine learning.

In single-objective GP, a fitness function is often adopted to solve the problem of class imbalance. The proposed method is compared with GP methods with different fitness functions, including weighted-average classification accuracy (Ave), the geometric mean (G_Mean), average mean squared error ($Amse$) [10], Correlation ratio ($Corr$) [10], $Dist$ [10], and an AUC measure WMW (Auc_w). In GP, Ave and G_Mean are often used to replace the overall classification accuracy as a fitness function for unbalanced classification. $Amse$ [10] has been proposed as a fitness function to improve the classification performance of GP in unbalanced classification. Because it is very time-consuming for GP to use a full AUC measure as a fitness function, $Corr$ and $Dist$ [10] have been proposed to approximate AUC for saving training time. Auc_w is an AUC measure, and GP using it often achieves a good performance in unbalanced classification. The definitions of these fitness functions are listed as following:

- $Ave = 0.5 * \frac{TP}{TP+FN} + 0.5 * \frac{TN}{TN+FP}$

  where TP is true positive, FP is false positive, TN is true negative and FN is false negative.

- $G\_Mean = \sqrt{\frac{TP}{TP+FN} * \frac{TN}{TN+FP}}$

- $Amse = \frac{1}{K}\Sigma_{c=1}^{K}(1 - \frac{\Sigma_{i=1}^{N_c}(sig(P_{c_i})-T_c)^2}{N_c*2})$

  where $sig(x) = \frac{2}{1+e^{-x}} - 1$, T_c are -0.5 and 0.5 for Maj and Min, respectively. K is the number of classes ($K = 2$ for binary classification), N_c is the number of instances in class c, and P_{ci} is an output value of a genetic program taking instance i from class c.

- $Corr = \frac{1}{K}(r + I_{zt}(1, \mu_{min}, \mu_{maj}))$

  where $r = \sqrt{\frac{\sum_{c=1}^{K} N_c(\mu_c-\overline{\mu})^2}{\sum_{c=1}^{K}\sum_{i=1}^{N_c}(P_{ci}-\overline{\mu})^2}}$, $\mu_c = \frac{\sum_{i=1}^{N_c} P_{ci}}{N_c}$, and $\overline{\mu} = \frac{\sum_{c=1}^{K} N_c\mu_c}{\sum_{c=1}^{K} N_c}$. Indicator function I_{zt} enforces a zero class threshold, which returns 1 if $\mu_{min} > 0$ and $\mu_{maj} < 0$, otherwise it returns 0.

- $Dist = \frac{|\mu_{min}-\mu_{maj}|}{\sigma_{min}+\sigma_{maj}} * I_{zt}(2, \mu_{min}, \mu_{maj})$

  where $\mu_c = \frac{\sum_{i=1}^{N_c} P_{ci}}{N_c}$, $\sigma_c = \sqrt{\frac{1}{N_c} * \sum_{i=1}^{N_c}(P_{ci} - \mu_c)^2}$, and the distance value of a program is doubled if I_{zt} returns 1.

- $Auc_w = \frac{\sum_{i\in Min}\sum_{j\in Maj} I_{wmw}(P_i,P_j)}{|Min|*|Maj|}$

  where $I_{wmw}(P_i, P_j) = \begin{cases} 1, & P_i > P_j \ and \ P_i \geq 0 \\ 0, & otherwise \end{cases}$.

In machine learning, classification algorithms often use sampling methods to solve the problem of class imbalance at the data level. The proposed method is compared with non-GP classification algorithms, including 1-nearest neighbours (1NN), decision trees (DT), random forests (RF), gradient boosting decision tree (GBDT) and naive bayes (NB). The oversampling methods, i.e. SMOTE [4] and adaptive synthetic sampling approach (ADASYN) [6], are used to solve the issue of class imbalance for these classification algorithms.

C. Parameter Settings

The parameter settings of GP methods are reported in Table III. Note that the compared GP methods are standard tree-based GP (not STGP). For other GP methods, their function set includes four arithmetic functions and a conditional operator if; the terminal set includes all features and a random constant. The function set of compared GP methods is slightly different from that of the proposed method (reported in Table I). However, it should be fair to compare the proposed method with other GP methods. This is because, in the function set in GPTFCM, $Left$, $Right$ and $Root$ do not directly manipulate on input data.

Table III: Parameter settings

Parameters	Values
Population size	1024
Generations	50
Initialization	Ramped half-and-half
Mutation rate (Subtree mutation)	0.2
Crossover rate (Subtree crossover)	0.8
Elitism	1
Selection method	Tournament selection (size=6)
Maximum tree depth	10

V. Results and Discussions

Each GP method has been independently run 30 times with different random seeds, and results (i.e. AUC) of different GP methods are reported in Table IV. AUC has been widely used as a metric to show the classification performance of a

Table IV: GPTFCM Versus other GP methods on the test sets

		AUC (%)			Training Time (seconds)
Datasets	Methods	Best	Mean±Std	ST	Mean
Armstrong-2002-v1 ($IR = 2$)	GP$_{Ave}$	100	94.48 ± 8.4	+	114.86
	GP$_{G_Mean}$	100	92.13 ± 8.01	+	114.88
	GP$_{Amse}$	100	90.17 ± 7.65	+	146.24
	GP$_{Corr}$	100	94.67 ± 7.56	+	142.55
	GP$_{Dist}$	100	95.84 ± 3.93	=	141.33
	GP$_{Auc_w}$	100	94.46 ± 4.93	=	1917.99
	GPTFCM	100	**97.22 ± 4.21**		123.89
Golub_1990 ($IR = 2$)	GP$_{Ave}$	100	91.93 ± 10.09	+	158.77
	GP$_{G_Mean}$	100	88.99 ± 11.89	+	158.31
	GP$_{Amse}$	100	82.78 ± 11.62	+	226.35
	GP$_{Corr}$	100	96.06 ± 6.32	=	225.06
	GP$_{Dist}$	100	96.9 ± 5.23	=	229.09
	GP$_{Auc_w}$	100	**98.42 ± 3.38**	=	3089.78
	GPTFCM	100	97.47 ± 4.01		155.92
Colon ($IR = 2$)	GP$_{Ave}$	91.67	75.52 ± 10.11	+	177.08
	GP$_{G_Mean}$	92.86	71.51 ± 12.95	+	174.21
	GP$_{Amse}$	95.24	74.8 ± 10.76	+	203.33
	GP$_{Corr}$	96.43	75.28 ± 10.1	+	201.08
	GP$_{Dist}$	92.86	76.59 ± 9.63	=	203.64
	GP$_{Auc_w}$	91.67	78.97 ± 7.3	=	2348.72
	GPTFCM	94.05	**79.33 ± 10.78**		168.81
Leukemia ($IR = 2$)	GP$_{Ave}$	98.21	88.79 ± 7.74	=	975.7
	GP$_{G_Mean}$	100	81.79 ± 15.38	+	979.29
	GP$_{Amse}$	100	81.73 ± 11.84	+	793.34
	GP$_{Corr}$	100	86.16 ± 10.84	+	785.98
	GP$_{Dist}$	97.32	86.32 ± 8.95	+	788.22
	GP$_{Auc_w}$	100	86.28 ± 9.68	+	10396.7
	GPTFCM	100	**91.93 ± 6.0**		858.06
DLBCL ($IR = 3$)	GP$_{Ave}$	98.15	75.4 ± 15.67	+	740.03
	GP$_{G_Mean}$	100	77.01 ± 15.75	+	731.45
	GP$_{Amse}$	100	77.19 ± 13.18	+	638.3
	GP$_{Corr}$	98.15	81.02± 11.42	+	643.18
	GP$_{Dist}$	99.07	84.35 ± 9.96	=	633.55
	GP$_{Auc_w}$	100	85.54 ± 10.83	=	7845.03
	GPTFCM	100	**86.1 ± 12.42**		642.44
Yeoh-2002-v1 ($IR = 5$)	GP$_{Ave}$	100	83.97 ± 11.91	+	773.26
	GP$_{G_Mean}$	95.78	66.33 ± 16.09	+	767.4
	GP$_{Amse}$	92.06	63.79 ± 12.06	+	717.74
	GP$_{Corr}$	100	93.29 ± 7.77	=	685.35
	GP$_{Dist}$	100	91.1 ± 8.22	=	694.73
	GP$_{Auc_w}$	100	**98.95 ± 2.32**	−	24174.23
	GPTFCM	100	90.57± 6.78		893.64
Lung ($IR = 8$)	GP$_{Ave}$	100	83.46 ± 14.73	+	3048.62
	GP$_{G_Mean}$	99.05	80.89 ± 18.41	+	3038.89
	GP$_{Amse}$	100	81.78 ± 16.55	+	2503.15
	GP$_{Corr}$	100	80.71 ±17.21	+	2490.26
	GP$_{Dist}$	100	84.27 ± 14.8	+	2493.37
	GP$_{Auc_w}$	100	92.35 ± 13.23	=	45375.33
	GPTFCM	100	**93.59 ± 6.57**		2861.7

1: Std means standard deviation.
2: ST means significance test results.

classifier in unbalanced classification. This is because AUC is invariant to data distributions [10]. The Wilcoxon statistical significance test is conducted to further compare the proposed method with different GP methods, with a significance level of 0.05. The results of the significance test are also reported in Table IV, where "+", "="and "−" indicate that the proposed method (i.e. GPTFCM) is significantly better, similar, and significantly worse than a compared method.

A. GPTFCM Versus Other GP Methods

(1) Analysis on Classification Performance

Generally, GPTFCM achieves similar or significantly better performance in 41 out of the 42 cases (significantly better performance in 28 cases and similar performance in 13 cases, respectively). On five datasets (seven datasets in total), GPTFCM achieves the best AUC result than other GP methods.

GP$_{Ave}$, GP$_{G_Mean}$ and GP$_{Amse}$ use different accuracy-based fitness functions (i.e Ave, G_Mean and $Amse$), respectively. The three GP methods use the threshold-based classification mechanism (threshold $TH = 0$). Comparing the proposed method with the three GP methods, GPTFCM achieves significantly better performance in 20 out of the 21 cases, based on results in Table IV.

GP$_{Corr}$, GP$_{Dist}$ and GP$_{Auc_w}$ adopt AUC approximation measures (i.e. $Corr$ and $Dist$) or AUC (i.e. Auc_w) as a fitness function, respectively. The three GP methods are independent of a classification threshold in the training process [10]. GP$_{Auc_w}$ often achieves better performance than other GP methods in unbalanced classification. Compared to GP$_{Auc_w}$, GPTFCM achieves similar or significantly better performance in 6 out of the 7 cases.

On the slightly unbalanced datasets (i.e $1 < IR \leq 2$), such as Armstrong-2002-v1 and Golub_1990, GPTFCM achieves at least similar classification performance than other GP methods in all cases. On the highly unbalanced datasets (i.e $IR > 2$), the proposed method also performs better than other GP methods in almost all cases.

In addition, based on results reported in Table IV, the best AUC result (among 30 GP runs) on each dataset, is often at least similar to other GP methods.

(2) Analysis on Training Time

Training time of different GP methods is reported in Table IV, which is further summarized in Table V. According to Table V, in 27 out of the 42 cases, GPTFCM consumes less training time than other GP methods. More importantly, GPTFCM consumes much less training time than GP$_{Auc_w}$, but GPTFCM achieves similar classification performance on 5 datasets and significantly better performance on one dataset than GP$_{Auc_w}$, based on Table IV.

Table V: Summary on training time

Datasets	GPTFCM consuming less training time than GP methods (times)
Armstrong-2002-v1	4
Golub_1990	6
Colon	6
Leukemia	3
DLBCL	4
Yeoh-2002-v1	1
Lung	3

B. GPTFCM Versus Other Classification Methods Using Sampling Methods

SMOTE and ADASYN are used to solve the issue of class imbalance for 1NN, DT, RF, GBDT and NB. The AUC results are reported in Table VI. Note that deterministic algorithms (i.e. 1NN and NB) are run once, while DT, RF and GBDT are conducted 30 times. The Wilcoxon statistical significance test is also conducted (a significance level is 0.05). Similarly, "+", "="and "−" indicate that the proposed method is significantly better, similar, and significantly worse than a compared method.

On three datasets, i.e. Armstrong-2002-v1, Golub_1990 and colon, the proposed method achieves the best classification performance, compared to other methods. In general, GPTFCM achieves significantly better or similar performance in

Table VI: GPTFCM Versus other traditional classification methods using sampling methods (AUC %)

	1NN w.t. SMOTE	1NN w.t. ADASYN	DT w.t. SMOTE	DT w.t. ADASYN	RF w.t. SMOTE	RF w.t. ADASYN	GBDT w.t. SMOTE	GBDT w.t. ADASYN	NB w.t. SMOTE	NB w.t. ADASYN	GPTFCM Best	GPTFCM Mean	GPTFCM Significantly better (Times)
Armstrong-2002-v1	96.67 =	93.33 =	88.46 +	89.59 +	90.97 +	91.97 +	89.52 +	89.52 +	85.71 +	92.86 +	100	**97.22**	8
Golub_1990	93.75 =	93.75 =	89.91 +	90.51 +	89.80 +	89.35 +	92.86 +	92.86 +	68.75 +	68.75 +	100	**97.47**	8
Colon	74.40 =	74.40 =	64.04 +	66.69 +	64.46 +	66.19 +	65.83 +	62.20 +	47.62 +	47.62 +	94.05	**79.33**	8
Leukemia	90.18 =	83.04 +	86.61 +	86.61+	81.52 +	80.47 +	86.61 +	86.61+	**100** −	**100** −	100	91.93	7
DLBCL	69.44 +	77.78+	67.31 +	67.31 +	77.13 +	74.35 +	72.22 +	72.22 +	80.86 +	**88.89** =	100	86.1	9
Yeoh-2002-v1	86.29 =	91.13 =	96.03 −	95.64 −	72.66 +	72.66 +	**96.15** −	**96.15** −	80.58 +	86.04 =	100	90.57	3
Lung	85.24 +	76.90 +	95.21 =	97.83 =	82.23 +	79.74+	**100** −	99.67 −	80.00 +	74.60 +	100	93.59	6

62 out of the 70 cases. By GPTFCM, the best result on each dataset is at least similar to other classification methods.

On slightly unbalanced datasets, including Armstrong-2002-v1, Golub_1990, colon and Leukemia, GPTFCM generally performs well, achieving similar or significantly better performance than other GP methods in 38 out of the 40 cases (significantly better performance in 31 cases and similar performance in 7 cases, respectively). On highly unbalanced datasets, including DLBCL, Yeoh-2002-v1 and Lung, the new method achieves significantly better or similar performance in 24 out of the 30 cases (significantly better performance in 18 cases and similar performance in 6 cases, respectively).

C. Summary

In summary, based on results in Table IV and Table VI, the new method often achieves at least similar classification performance than other classification methods. However, according to results, it is noticed that the proposed method is more suitable for sightly unbalanced datasets than highly unbalanced datasets.

VI. Further Analysis

In this section, we will take two examples of evolved programs (from two datasets, i.e. Armstrong-2002-v1 and Lung) to further analyse the proposed method.

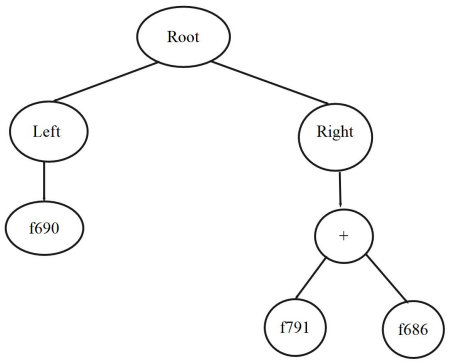

Figure 3: An evolved tree on Armstrong-2002-v1

An evolved tree on Armstrong (1081 features, 72 instances, $IR = 2$) is shown in Figure 3, which is the best tree from a final generation in a GP run. Using this tree to classify instances in the test set, the AUC score is 100%, and the accuracies of the majority class and the minority class are also 100%. Training time of this GP run is 79.69s, which is faster than the averaged training time on this dataset (i.e. 123.89s). This tree has 7 nodes in total, where only three features are used as terminals to achieve 100% AUC.

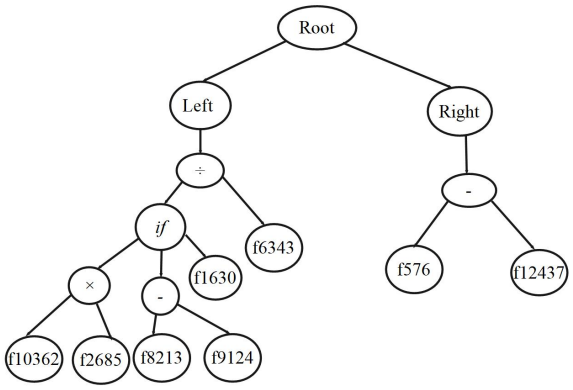

Figure 4: An evolved tree on Lung

In Figure 4, we show another program that is evolved in a GP run on Lung (12600 features, 158 instances, and $IR = 8$). On the test set, AUC result of this tree is 100%, and its accuracies of the majority class and the minority class are also 100%. The training time of this GP run takes 3060.26s, which is longer than the averaged training time on this dataset (i.e. 2861.7s). In addition, the tree (on Lung) in Figure 4, having 16 nodes in total, is more complicated than the evolved tree (on Armstrong-2002-v1) in Figure 3. This is mainly because the number of features (12600 features) in Lung is much more than that of Armstrong-2002-v1 (1081 features).

VII. Conclusions

This paper develops a new threshold-free classification mechanism for GP in high-dimensional unbalanced classification. In binary classification, for a tree, when an instance from the minority class is inputted, the output value of its left sub-tree is expected to be greater than or equal to that of its right sub-tree. Oppositely, when an instance from the majority class is inputted, the output value of its left sub-tree is expected to be smaller than that of its right sub-tree. To achieve this, in the new method (i.e. GPTFCM), the tree representation is predefined, working with a terminal set and a function set, to obtain two output values from the left sub-tree and right sub-tree.

In the experiments, the proposed method was examined on seven high-dimensional unbalanced datasets. Compared to other GP methods, the classification performance of the new method is often better than other methods. More specifically, compared to GP methods based on threshold-based classification mechanism (i.e. GP_{Ave}, GP_{G_Mean} and GP_{Amse}), GPTFCM often achieves significantly better performance. In addition, GPTFCM often achieves similar classification performance as GP_{Auc_w}, but the new method consumes much less training time. Compared to other non-GP classification methods, GPTFCM also achieves at least similar classification performance in most cases.

However, based on results, the proposed method seems to be more suitable for classification with sightly unbalanced data than highly unbalanced data. In the future, we will further investigate how classification performance could be improved on highly unbalanced datasets.

ACKNOWLEDGE

This work was supported in part by the Marsden Fund of New Zealand Government under Contracts VUW1509 and VUW1615, the Science for Technological Innovation Challenge (SfTI) fund under grant E3603/2903, the University Research Fund at Victoria University of Wellington grant number 216378/3764 and 223805/3986, and MBIE Data Science SSIF Fund under the contract RTVU1914. National Natural Science Foundation of China (NSFC), under Grant 61876169.

REFERENCES

[1] T. M. Padmaja, N. Dhulipalla, R. S. Bapi, and P. R. Krishna, "Unbalanced data classification using extreme outlier elimination and sampling techniques for fraud detection," in *15th International Conference on Advanced Computing and Communications (ADCOM 2007)*, pp. 511–516, IEEE, 2007.

[2] F. Yang, H.-z. Wang, H. Mi, W.-w. Cai, *et al.*, "Using random forest for reliable classification and cost-sensitive learning for medical diagnosis," *BMC bioinformatics*, vol. 10, no. 1, p. S22, 2009.

[3] X.-Y. Liu, J. Wu, and Z.-H. Zhou, "Exploratory undersampling for class-imbalance learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 2, pp. 539–550, 2008.

[4] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[5] H. Han, W.-Y. Wang, and B.-H. Mao, "Borderline-smote: a new over-sampling method in imbalanced data sets learning," in *International conference on intelligent computing*, pp. 878–887, Springer, 2005.

[6] H. He, Y. Bai, E. A. Garcia, and S. Li, "Adasyn: Adaptive synthetic sampling approach for imbalanced learning," in *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pp. 1322–1328, IEEE, 2008.

[7] C. Elkan, "The foundations of cost-sensitive learning," in *International joint conference on artificial intelligence*, vol. 17, pp. 973–978, Lawrence Erlbaum Associates Ltd, 2001.

[8] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on knowledge and data engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[9] R. Poli, W. Langdon, and N. McPhee, *A field guide to genetic programming*. 2008.

[10] U. Bhowan, M. Johnston, and M. Zhang, "Developing new fitness functions in genetic programming for classification with unbalanced data," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 2, pp. 406–421, 2012.

[11] G. Patterson and M. Zhang, "Fitness functions in genetic programming for classification with unbalanced data," in *Australasian Joint Conference on Artificial Intelligence*, pp. 769–775, Springer, 2007.

[12] X.-Y. Liu and Z.-H. Zhou, "Towards cost-sensitive learning for real-world applications," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 494–505, Springer, 2011.

[13] M. Sahare and H. Gupta, "A review of multi-class classification for imbalanced data," *International Journal of Advanced Computer Research*, vol. 2, no. 3, p. 160, 2012.

[14] E. Ramentol, Y. Caballero, R. Bello, and F. Herrera, "Smote-rsb*: a hybrid preprocessing approach based on oversampling and undersampling for high imbalanced data-sets using smote and rough sets theory," *Knowledge and information systems*, vol. 33, no. 2, pp. 245–265, 2012.

[15] Y. Zhang and Z.-H. Zhou, "Cost-sensitive face recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 10, pp. 1758–1769, 2009.

[16] F. Li, X. Zhang, X. Zhang, C. Du, Y. Xu, and Y.-C. Tian, "Cost-sensitive and hybrid-attribute measure multi-decision tree over imbalanced data sets," *Information Sciences*, vol. 422, pp. 242–256, 2018.

[17] R. Batuwita and V. Palade, "Class imbalance learning methods for support vector machines," 2013.

[18] G. Guo and C. R. Dyer, "Learning from examples in the small sample case: face expression recognition," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 35, no. 3, pp. 477–488, 2005.

[19] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of machine learning research*, vol. 3, no. Mar, pp. 1157–1182, 2003.

[20] B. Xue, M. Zhang, W. N. Browne, and X. Yao, "A survey on evolutionary computation approaches to feature selection," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 4, pp. 606–626, 2015.

[21] L. Yin, Y. Ge, K. Xiao, X. Wang, and X. Quan, "Feature selection for high-dimensional imbalanced data," *Neurocomputing*, vol. 105, pp. 3–11, 2013.

[22] P. G. Espejo, S. Ventura, and F. Herrera, "A survey on the application of genetic programming to classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 40, no. 2, pp. 121–144, 2010.

[23] V. LóPez, A. FernáNdez, M. J. Del Jesus, and F. Herrera, "A hierarchical genetic fuzzy system based on genetic programming for addressing classification with highly imbalanced and borderline data-sets," *Knowledge-Based Systems*, vol. 38, pp. 85–104, 2013.

[24] R. Hunt, M. Johnston, W. Browne, and M. Zhang, "Sampling methods in genetic programming for classification with unbalanced data," in *Australasian Joint Conference on Artificial Intelligence*, pp. 273–282, Springer, 2010.

[25] J. Li, X. Li, and X. Yao, "Cost-sensitive classification with genetic programming," in *2005 IEEE congress on evolutionary computation*, vol. 3, pp. 2114–2121, IEEE, 2005.

[26] W. Pei, B. Xue, M. Zhang, and L. Shang, "A cost-sensitive genetic programming approach for high-dimensional unbalanced classification," in *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1770–1777, IEEE, 2019.

[27] W. Pei, B. Xue, L. Shang, and M. Zhang, "New fitness functions in genetic programming for classification with high-dimensional unbalanced data," in *2019 IEEE Congress on Evolutionary Computation (CEC)*, pp. 2779–2786, IEEE, 2019.

[28] W. Pei, B. Xue, L. Shang, and M. Zhang, "Reuse of program trees in genetic programming with a new fitness function in high-dimensional unbalanced classification," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pp. 187–188, 2019.

[29] U. Bhowan, M. Johnston, and M. Zhang, "Ensemble learning and pruning in multi-objective genetic programming for classification with unbalanced data," in *Australasian Joint Conference on Artificial Intelligence*, pp. 192–202, Springer, 2011.

[30] U. Bhowan, M. Johnston, and M. Zhang, "Evolving ensembles in multi-objective genetic programming for classification with unbalanced data," in *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pp. 1331–1338, ACM, 2011.

[31] U. Bhowan, M. Johnston, M. Zhang, and X. Yao, "Reusing genetic programming for ensemble selection in classification of unbalanced data," *IEEE Transaction on Evolutionary Computation*, vol. 18, no. 6, pp. 893–908, 2014.

[32] U. Bhowan, M. Johnston, M. Zhang, and X. Yao, "Evolving diverse ensembles using genetic programming for classification with unbalanced data," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 3, pp. 368–386, 2013.