Genetic Programming for Multiple Feature Construction on High-Dimensional Classification

Binh Tran[a,*], Mengjie Zhang[a], Bing Xue[a]

[a]*School of Engineering and Computer Science, Victoria University of Wellington, PO Box 600, Wellington 6140, New Zealand*

Abstract

Data representation is one of the most important factors deciding the performance of machine learning algorithms including classification. Feature construction can combine original features to form high-level ones that can help classification algorithms achieve better performance. Genetic programming (GP) has shown promise in feature construction due to its flexible representation. Most GP methods construct a single feature, which may not scale well to high dimensional data. Multiple feature construction is required to provide a better representation. This paper aims at investigating different approaches to constructing multiple high-level features, and analyse their underlying behaviours to reveal the insight of multiple feature construction using GP in high-dimensional data in terms of both the effectiveness and efficiency. The results show that the class-dependent method, which focuses on discriminating instances of a certain class from the others, achieves better performance than the class-independent counterpart. A combination of appropriate filter measures helps GP construct highly discriminating features in a shorter time than a combination of both filter and wrapper approaches. Furthermore, using multiple-tree GP representation is more effective than using the single-tree GP in multiple-feature construction thanks to the ability of determining the interaction of the newly constructed features during the construction process.

*Corresponding author

Email address: {Binh.Tran, Mengjie.Zhang, Bing.Xue}@ecs.vuw.ac.nz (Binh Tran)

Keywords: Feature construction, genetic programming, classification, class dependent, high-dimensional data,

1. Introduction

In machine learning, data representation is one of the most important factors deciding the performance of any learning algorithm including classification [1]. With the advances in data collection technologies, more and more high-dimensional data are collected with thousands or tens of thousands of features. Inducing patterns from these datasets is challenging for many common learning algorithms due to the curse of dimensionality. Furthermore, there may exist a large number of irrelevant and redundant features that are not actually useful in learning the target concept. The presence of these features may obscure the effect of the relevant features on showing the hidden pattern of the data, and thereby reduce the representativeness of the whole feature set [2].Therefore, dimensionality reduction is essential in the datasets. Feature selection has shown to be effective in improving classification performance of learning algorithms on high-dimensional data by removing irrelevant and redundant features. However, its performance may still be limited if the original features do not provide enough discriminative information for learning algorithms to learn a good classifier. There may exist certain combinations of features that may provide better discriminating ability [3]. In this case, feature construction can be used to construct new high-level features that better represent the problem.

Genetic programming (GP) is an evolutionary computation technique that has shown promise in feature construction. As a population-based algorithm, GP evolves a population of individuals, each of which represents a single constructed feature (i.e. single-tree representation) or a set of constructed features (i.e. multi-tree representation). Most feature construction methods are proposed to construct a single feature, which may not be enough to represent the complex feature space of the original thousands of features. Therefore, it is necessary to construct more features for these high-dimensional datasets to obtain

better representation.

Multiple feature construction methods have been proposed using both single-tree [4, 5, 6] and multi-tree [7, 8, 9] representation. When using single-tree representation, GP can construct multiple features by using all possible subtrees [4] or some subtrees under predefined special nodes [5]. Another approach to multiple feature construction using single-tree GP is to run GP multiple times, each time constructs a new feature [6]. Multiple-tree GP was also proposed and shown effective in constructing multiple features on datasets with about tens of features [7, 8, 9] as well as thousands of features [10].

In addition to the choice between single or multi-tree representation, constructing class-dependent or class-independent features is another option in designing a feature construction method. Most of the proposed GP-based feature construction methods are class-independent, where each high-level feature is constructed without focusing on any specific class of the problem. On the other hand, each class-dependent constructed feature aims at distinguishing instances of one class from other classes [6]. However, the method is limited to construct one feature for each class, which may not scale well with high-dimensional data. Recently, a multiple class-dependent feature construction method for high-dimensional data [11] was proposed and shown to achieve better performance than the class-independent constructed features.

Furthermore, during the feature construction process, different approaches can be chosen to evaluate the constructed features. They are filter, wrapper, embedded or combination of these approaches [12]. While filter methods use some measure such as information gain or impurity to evaluate the constructed features [6], wrapper methods have a classification algorithm to evaluate them [5]. While wrapper methods are usually computationally more expensive than filter methods, they usually obtain better classification accuracy. On the other hand, filter are usually more general than wrappers. Combination of the two measures was also proposed to better evaluate the constructed feature set [11]. Since a GP tree (or the constructed feature) can be used as a binary classifier, its classification performance can be used to evaluate itself. This scenario is

referred to as embedded feature construction method [13].

Although feature construction using GP has been studied for decades, most of the methods are applied on datasets with tens of features. Compare to feature selection, the search space of feature construction is larger since it requires to choose not only a good feature subsets but also an appropriate set of operators to combine them for a better discriminating features. Recently, two multiple feature construction methods were proposed for high-dimensional data, one was class-independent [10] and one was class-independent [11]. The latter was found to be better than the former. However, the two methods are different in a number of aspects (to be described in Section 3.2), any of which can lead to the difference in their performance. Therefore, it is necessary to further investigate them in the same context in order to confirm the effectiveness of class-dependent constructed features. Further comparison with other feature construction methods and analysis are needed to have a better insight into the performance of the above mentioned approaches to multiple feature construction on high-dimensional data.

1.1. Goals

This study aims at investigating GP-based multiple feature construction approaches including using single-tree versus multi-tree representation, class-dependent versus class-independent feature construction, and different approaches to evaluating the constructed features during the evolutionary process. Three multiple feature construction methods will be investigated including two methods using multi-tree representation, namely the class-independent (MCIFC) [10] and the class-dependent (CDFC) [11], and one method using single-tree representation to construct class-dependent features (1TGPFC) [6]. Performance of the constructed features by the three methods will be compared based on the classification performance of common learning algorithms including k-Nearest Neighbour (KNN), Naive Bayes (NB) and Decision Tree (DT). Specifically, the following research questions will be investigated:

- Whether the constructed features help learning algorithms achieve better classification accuracy than the original full feature set;
- Whether class-dependent constructed features achieve better classification performance than class-independent ones;
- Whether using multi-tree representation, GP constructs new features with higher discriminating ability than using single-tree representation; and
- Whether using filter method to evaluate constructed features is more effective and efficient than combining wrapper and filter measures.

2. Related Work

2.1. Single-Tree GP-Based Feature Construction

One of the first GP based feature construction methods using single-tree representation was proposed by Raymer et al. [14]. It aimed to improve their previous method that used a GA to evolve weights that can transform each original features into a new one. The GA was replaced by a GP to enable non-linear transformation for each feature. Each GP individual was a single-tree with n subtrees, where n was the number of original features. KNN was used to evaluate the constructed feature set. Results on a water displacement problem with four features showed that the proposed GP based feature construction method obtained better KNN accuracy than a GA-based one. However, dimensionality reduction is not the aim of this method.

Cooperative coevolution was also proposed to combine many single-tree GP populations to construct multiple features. In [15], one GP population was used to evolve a new feature for each original feature. Another GA population was used for feature selection. KNN was used to evaluate the new feature set. Similarly, in [16], n concurrent populations of single-tree individuals was used to evolve n new features where n is the desired number of constructed features. The entire new feature subset was evaluated using DT. Experiments on a dataset with nine features showed that the proposed method constructed better features

than the standard multi-tree GP method. However, since the number of fitness function calls was n time greater than that of standard GP, the computational time of cooperative coevolution approach was significantly high.

Constructing multiple features from all possible subtrees in a single-tree, Ahmed et al. [4] proposed two GP based wrapper feature construction methods, namely GPWFC1 and GPWFC2 for high dimensional data. While GPWFC1 used classification accuracy of random forest (RF) classifier as the fitness value, GPWFC2 used entropy gain of RF and the p-value of an ANOVA test on the selected features. Results showed that GPWFC2 achieved better generalisation ability and smaller numbers of features than GPWFC1. However, the computational cost was quite high when adopting RF as the learning algorithm for fitness evaluation. This is a common drawback of GP based wrapper approaches, which can be avoided with filter methods.

Using information gain ratio to evaluate the constructed feature, Otero et al. [17] proposed a GP based filter feature construction method to construct a new continuous or boolean feature using arithmetic and relation operators. The constructed feature was then augmented to the original feature set. Results on four datasets with 4 to 21 features showed that the constructed feature can improve the performance of C4.5. However, since both C4.5 and the proposed method are based on information gain, it is unknown whether the constructed features will also be beneficial for other learning algorithms. This question has been investigated in [18, 19] where the performance of C5, which is a DT algorithm, was compared with two other DT algorithms, CART and CHAID, and multilayer perceptron. Results showed that these classifiers generally benefited from the inclusion of the constructed feature. However, more datasets with larger number of features should be used to illustrate the performance and generalisation of this method.

Running single-tree GP program multiple times, Neshatian et al. [20] proposed a new approach to multiple feature construction. The number of constructed features was equal to the number of classes. Each GP run focused on one class. Constructed features were evaluated based on the impurity (using

Shannon entropy) of the intervals which were formed by applying class dispersion to the transformed datasets. The results indicated that constructed features helped DT achieved smaller error rates with much smaller sizes than using original features in most cases. However, when combining constructed features with original features, the performance of learnt DTs was worse than using constructed features only in most cases. The method was improved in [21, 6] by adding class-wise orthogonal transformed features to GP terminal sets. Results indicated that constructed features increased the classification accuracy of the learned DTs. However, by adding more features, this strategy significantly increased the GP search space. Therefore, it may not be effective and efficient for high-dimensional data.

Combining GA and GP in a single representation, Nguyen et al. [22] proposed to construct and select features simultaneously in a single evolutionary process. Each individual contained an $n - bit$ string for feature selection and a single tree to construct one feature. The whole set of selected and constructed features were evaluated by KNN. Results on 10 UCI datasets showed that the proposed method either obtained similar or significantly better accuracies than the compared methods in almost all cases. However, the performance of this method can be degraded when applying to high-dimensional data because the effect of the single constructed feature may not be recognised when using KNN to evaluate the whole set of selected and constructed features.

By considering the single-tree as a classifier which can evaluate the performance of the constructed feature, an embedded feature construction method (GPFC) [13] was proposed for high-dimensional gene expression data. Different ways of using the constructed and selected features in the single-tree were compared. The results showed that the combination of constructed and selected features had the best performance among the five combinations. However, GP performance might be limited when confronted with a large number of features which may be irrelevant or redundant. An improvement was proposed in a cluster-based feature construction method (CGPFC) [23] where feature clustering was used to narrow the GP search space in high-dimensional data. The

results on 8 gene expression datasets with thousands to tens of thousands of features showed that CGPFC selected a much smaller number of features to construct a better feature than GPFC. However using GP as a classifier to evaluate the constructed feature, both GPDC and CGPFC can only be applied to binary-class problems.

2.2. Multi-Tree GP-Based Feature Construction

Multi-tree representation has been used in multiple feature construction methods. Krawiec [8] used a multi-tree GP to construct a predefined number of features (n). Each GP individual comprised of n new features and n hidden features. Hidden features were also constructed features but kept out of the evolutionary process to avoid losing good constructed features during the evolutionary process. DT was used to evaluate each individual. Results on six datasets with less than ten features showed that constructed features improved the classification performance of DT on most datasets. Using hidden features also obtained close to or sometimes better results than standard approach where all features are involved in the evolutionary process.

Smith and Bull [24, 25] combined GP for multiple feature construction and GA for selection simultaneously which used DT to evaluate features. For an n-dimension problem, a GP individual had n trees, each associated with a boolean variable showing if it is selected or not, and a flag to choose among DT, KNN and NB for evaluation. The method has shown to significantly improve classification performance on 2 out of 10 datasets with 5 to 60 features. In [26], GA was used to select a subset of features which was then used by GP to construct new features. The result on an image dataset with 8 features were encouraging when GA chose 4 features, 2 of which were then used by GP to construct 3 features. The reverse order was also investigated in the GAP method [9] where GP was used to construct n new features, where n is the number of the original features. GA was applied to the augmented feature set to reduce the number of features. Experiments on 10 UCI datasets showed that GAP improved the performance of C4.5 on 8 datasets. Although C4.5 was used in

its fitness function, GAP results were robust to other classifiers including KNN and NB. The results from these work have demonstrated that combination of EC techniques is a promising approach to feature manipulation. However, on top of the high computational cost, these representations are not suitable for problems with thousands of features.

3. Multiple Feature Construction Methods

To make this paper easy to follow, this section will briefly describe the three methods that will be investigated as they were proposed in the original papers.

3.1. MCIFC: A Multiple Class-Independent Feature Construction Method

MCIFC [10] uses a multi-tree GP to construct a small number of features that is set proportional to the number of classes of the problem. This design is based on a hypothesis that a problem with a higher number of classes is more complex and may require more features to represent the data. The number of new features is calculated using Eq. 1.

$$nbr\_cf = r \times c \tag{1}$$

where r is a user-defined ratio and c is the number of classes. For example, if $r = 2$, MCIFC will construct 4 features for a binary-class problem. Fig. 1 shows the representation of MCIFC.

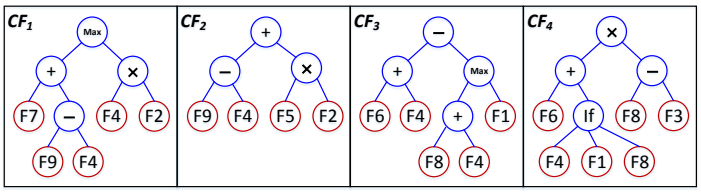


Figure 1: MCIFC representation.

Using this example individual, 4 features are generated as follows.

- $CF_1 = Max(F_7 + (F_9 - F_4), F_4 \times F_2)$
- $CF_2 = (F_9 - F_4) + (F_5 \times F_2)$

- $CF_3 = (F_6 + F_4) - Max(F_8 + F_4, F_1)$
- $CF_4 = (F_6 + If(F_4, F_1, F_8)) \times (F_8 - F_3)$

MCIFC uses a weight (α) to combine the DT classification accuracy and a distance measure in the fitness function as shown in Equation (2).

$$Fitness = \alpha \cdot Bal\_Accuracy + (1 - \alpha) \cdot Distance \tag{2}$$

Bal_Accuracy is the average of the balanced accuracies obtained from the K-fold (K=3) cross-validation (CV) on the transformed training set. In addition, the K-fold CV is repeated L times (L=3) with different data splitting. Therefore, $K \times L$, i.e. 9, models were built to evaluate each individual. This evaluation scheme is used to avoid overfitting even though it is a little bit more expensive but affordable because the number of constructed features are small. The balanced accuracy [27] is calculated based on Equation (3).

$$Bal\_Accuracy = \frac{1}{c} \sum_{i=1}^{c} \frac{TP_i}{|S_i|} \tag{3}$$

where c is the number of classes, TP_i and S_i are the number of correctly identified instances and the number of total instances of class i, respectively.

The *Distance* measure [28] calculated based on Equation (4) is used to maximise the distance of instances *between* class (D_b) and minimise the distance of instances *within* the same class (D_w). Let S be the training set, D_b and D_w are approximated based on Equations (5) and (6).

$$Distance = \frac{1}{1 + e^{-5(D_b - D_w)}} \tag{4}$$

$$D_b = \frac{1}{|S|} \sum_{i=1}^{|S|} \min_{\{j | j \neq i, class(V_i) \neq class(V_j)\}} Dis(V_i, V_j) \tag{5}$$

$$D_w = \frac{1}{|S|} \sum_{i=1}^{|S|} \max_{\{j | j \neq i, class(V_i) = class(V_j)\}} Dis(V_i, V_j) \tag{6}$$

$$Czekanowski(V_i, V_j) = 1 - \frac{2 \sum_{d=1}^{n} \min(V_{id}, V_{jd})}{\sum_{d=1}^{n} (V_{id} + V_{jd})} \tag{7}$$

where $Dis(V_i, V_j)$ is the distance between two vectors V_i and V_j, which was approximated by the Czekanowski measure [29] as shown in Equation (7).

3.2. CDFC: A Multiple Class-Dependent Feature Construction Method

Similar to MCIFC, the number of constructed features by CDFC is also calculated based on Eq. (1). However, different from MCIFC, each feature constructed by CDFC is class-dependent. It aims at discriminating instances of one class to other classes. In other words, each feature is associated with one class. Fig. 2 shows an example of an individual of CDFC with $r = 1$ for a three-class problem. In this case, while CF_1 is evolved towards a high-level feature that can distinguish instances of $Class_1$ to other classes, CF_2 and CF_3 focus on $Class_2$ and $Class_3$, respectively.

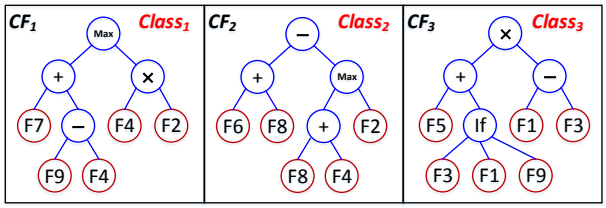


Figure 2: Representation of a class-dependent GP individual with construction ratio 1.

In CDFC, a new feature cf is constructed from a subset of original features that are relevant to the class that cf associates with. t-Test was used to measure how relevant a feature f is to class c. Values of f are first divided into two groups, one comprises values belonging to class c and one from other classes. Then, Equation (8) is used to measure its relevance to class c, $Rel_{f,c}$, which considers not only the difference between the means of two groups (t-value) but also the confidence of this difference (p-value). It is set to 0 if the two groups are not significantly different (i.e. p-value ≥ 0.05), and to the absolute of t-value divided by p-value, otherwise. Therefore, the larger the value of $Rel_{f,c}$, the more relevant the feature f to class c. For each class c, features are ranked by its $Rel_{f,c}$ values. Then half of the top-ranked features will be used to form

the terminal set of class c. This strategy not only eliminates irrelevant features but also narrows the search space so that the searching process will be more efficient.

$$Rel_{f,c} = \begin{cases} 0, & \text{if p-value} \geq 0.05 \\ \frac{|\textbf{t-value}(f_{class=c}, f_{class \neq c})|}{\textbf{p-value}}, & \text{otherwise} \end{cases} \quad (8)$$

To speed up the evaluation process, CDFC replaces the DT classification accuracy ($Bal\_Accuracy$) in Eq. (2) by information gain ($AvgIG$), which is the base measure of DT, as shown in Equation (9). The size of the GP individual (indSize) is also used here as a pressure for small-tree preference. Its weight is set to a very small value (10^{-7}) in order to limit its effect on cases when two individuals have the same information gain and distance.

$$Fitness = \alpha \cdot AvgIG + (1 - \alpha) \cdot Distance - 10^{-7} \cdot indSize \quad (9)$$

AvgIG is calculated based on Equation (10) where f_{max} is the best feature with the highest IG among m constructed features. The f_{max}'s IG is added to bias toward those candidates that have the better f_{max}. IG of feature f is calculated based on unconditional and conditional entropy H as in Equation (11), i.e. mutual information.

$$AvgIG = \frac{\sum_{i=1}^{m} IG(f_i, class) + IG(f_{max}, class)}{m + 1} \quad (10)$$

$$IG(f, class) = H(class) - H(class|f) \quad (11)$$

Fig. 3 shows the overall system of CDFC. Based on the training set, CDFC constructs c terminal sets, each of which is corresponding to one class. Note that they can be overlap since one feature can be relevant to more than one class. The tree associated with class i will use the ith terminal set.

It can be seen that MCIFC and CDFC are different in a number of aspects. Firstly, while MCIFC's terminal set includes all the original features, the terminal sets of CDFC are smaller than MCIFC and generally comprised of good features. Therefore, the search space of MCIFC is larger than CDFC. Secondly,

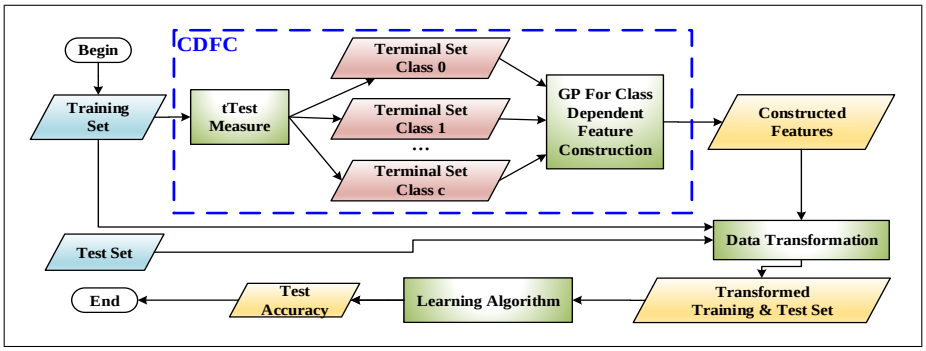


Figure 3: CDFC overall system.

they use different fitness functions, which will lead to different performance. Therefore, the difference in their performance may not necessarily contributed only by the class-dependency of the constructed features. Therefore, this study conducts further experiments where their terminal sets are equally set.

4. Experiment Design

Table 1: Datasets

Dataset	#F	#Inst.	#Class	Class Distribution	Ref.
Colon	2,000	62	2	35%, 65%	[30]
DLBCL	5,469	77	2	25%, 75%	[31]
Leukemia	7,129	72	2	35%, 65%	[30]
CNS	7,129	60	2	35%, 65%	[30]
Prostate	10,509	102	2	49%, 51%	[31]
Ovarian	15,154	253	2	36%, 64%	[30]
Leukemia1	5,327	72	3	12%, 35%, 53%	[31]
SRBCT	2,308	83	4	13%, 22%, 30%, 35%	[31]

4.1. Datasets

Eight gene expression datasets with thousands of features are used to examine the performance of the proposed method on high-dimensional data. Details about these datasets are shown in Table 1. Among the eight datasets, six are

binary-class problems, one has three classes (Leukemia1), and the last one has four classes (SRBCT).

Data is pre-processed to reduce noise generated during the data collection in laboratories as suggested in [32]. First, features are standardised (based on the training set) to have zero mean and unit variance, then discretised into -1, 0 and 1 representing three states which are the under-expression, the baseline and the over-expression of gene. If the feature values are in the range of $[-0.5, 0.5]$, they will be set to 0. Otherwise, they will be set to -1 or 1 depending on whether they are smaller than -0.5 or larger than 0.5, respectively.

4.2. Experiment Configuration

Due to the small number of instances in each dataset, 10-fold CV is used to generate training and test set, where one is used for testing, and the other 9 for training. Only the training data is used to during the feature construction process. As GP is a stochastic algorithm, 50 independent runs of each method with 50 different random seeds are executed on each training set. Average of 500 results (50×10) are used in comparisons.

Experiments were runs on PC with Intel Core i7-4770 CPU @ 3.4GHz, running Ubuntu 4.6 and Java 1.7 with a total memory of 8GB. The results of 50 runs from each method were compared using the statistical significance test [33], with a significance level of 0.05.

4.3. Parameter Settings

Table 2 describes the parameter settings of all GP based methods used in the experiments. The function set comprises of 3 arithmetic operators ($+$, $-$, $\times$), a max function which returns the maximum values from the two inputs and an if function which returns the second argument if the first argument is positive and returns the third argument otherwise. No constant values are used in the terminal set for simplicity. The population size is set proportional to the dimensionality of the problem using a coefficient β which is set to 3 for the Colon dataset and to 2 for all the others due to memory limit. The construction

Table 2: Parameter settings

Function set	$+$, $-$, $\times$, max, if
Maximum Tree Depth	8
Population Size	#features x β
Initial Population	Ramped Half-and Half
Generations	50
Crossover Rate	0.8
Mutation Rate	0.2
Elitism Size	1
Selection Method	Tournament Method
Tournament Size	7
Construction ratio r	2
Fitness weighting α	0.8

ratio r used to determine the number of constructed features is experimentally chosen as 2. The fitness weight α in both MCIFC and CDFC is set to 0.8 in order to bias fitness values towards the accuracy (in MCIFC) and information gain measure (in CDFC).

5. Results and Discussions

5.1. Filter Class-Dependent Versus Hybrid Class-Independent Methods

In this experiment, CDFC is compared with MCIFC whose terminal set is also filtered as in CDFC. MCIFC's terminal set is constructed by combining all the class-dependent terminal sets of CDFC. However, MCIFC still uses the hybrid evaluation method which combine DT accuracy and distance as in Eq. (2), which is called as hMCIFC to distinguish it from MCIFC proposed in [10].

Table 3 shows the average test accuracy of KNN, NB and DT using the constructed features obtained from 50 independent runs of CDFC compared with "Full" (i.e. using the original feature set) and hMCIFC. The number of features and instances of each dataset is displayed in parenthesis under the dataset name for reading convenience. For each learning algorithm, the best (B), the mean and standard deviation (M±Std) results are displayed. The best result among the three compared methods on each dataset is highlighted. In addition,

the Wilcoxon significance test is applied to the results with a 5% significance level. Its results for KNN, NB and DT are displayed in column S_1, S_2, and S_3, respectively. "+" or "-" indicates that the corresponding method is significantly better or worse than CDFC. "=" means they have similar performance. In other worse, the more "–", the better the class-dependent feature construction method.

Table 3: Results of the constructed features

Dataset	**Method**	**B-KNN**	**M±Std-KNN**	S_1	**B-NB**	**M±Std-NB**	S_2	**B-DT**	**M±Std-DT**	S_3
Colon	Full	74.29		–	72.62		–	74.29		–
(2000)	hMCIFC	85.71	75.74 ± 4.45	–	87.14	75.60 ± 4.36	–	84.05	74.85 ± 4.70	–
(62)	CDFC	88.81	**81.87 ± 3.08**		90.47	**83.52 ± 3.11**		87.38	**78.03 ± 4.00**	
DLBCL	Full	84.46		–	81.96		–	80.89		–
(5469)	hMCIFC	97.32	91.63 ± 2.41	–	96.25	91.19 ± 2.38	–	96.25	89.71 ± 3.13	=
(77)	CDFC	98.75	**96.03 ± 1.96**		98.75	**95.25 ± 2.19**		95.00	**90.76 ± 3.01**	
Leukemia	Full	88.57		–	91.96		–	91.61		=
(7129)	hMCIFC	97.50	92.60 ± 2.68	–	97.32	91.87 ± 2.70	–	95.89	90.23 ± 2.47	=
(72)	CDFC	98.57	**94.83 ± 1.71**		97.32	**94.07 ± 1.60**		94.82	90.72 ± 2.47	
CNS	Full	56.67		–	58.33		–	50.00		–
(7129)	hMCIFC	78.33	61.97 ± 5.02	–	78.33	63.07 ± 4.99	–	75.00	60.90 ± 4.84	=
(60)	CDFC	73.33	**65.10 ± 4.20**		73.33	**66.17 ± 3.75**		68.34	**61.03 ± 4.87**	
Prostate	Full	81.55		–	60.55		–	86.18		–
(10509)	hMCIFC	91.27	87.26 ± 2.70	–	93.18	87.44 ± 2.73	–	90.36	85.26 ± 2.24	–
(102)	CDFC	95.18	**92.81 ± 1.59**		96.09	**92.82 ± 1.50**		94.09	**88.04 ± 2.52**	
Ovarian	Full	91.28		–	90.05		–	98.41		–
(15154)	hMCIFC	100.00	99.41 ± 0.49	–	100.00	99.48 ± 0.45	=	100.00	**98.93 ± 0.65**	=
(253)	CDFC	100.00	**99.73 ± 0.31**		100.00	**99.55 ± 0.34**		100.00	98.78 ± 0.69	
Leukemia1	Full	88.57		–	88.75		–	94.46		+
(5327)	hMCIFC	94.46	90.97 ± 2.51	–	94.64	91.03 ± 2.21	–	94.46	90.53 ± 2.55	=
(72)	CDFC	97.32	**93.47 ± 1.82**		97.32	**93.07 ± 2.22**		95.89	89.72 ± 2.70	
SRBCT	Full	80.83		–	97.50		+	72.36		–
(2308)	hMCIFC	95.28	89.00 ± 3.14	–	93.89	88.87 ± 2.71	–	91.25	83.85 ± 3.78	–
(83)	CDFC	100.00	**95.88 ± 1.94**		100.00	95.08 ± 2.39		94.17	**88.01 ± 3.48**	

5.1.1. CDFC versus Full

All the "–"s appeared in column S_1 of Table 3 showed that the constructed features helped KNN achieve significantly higher accuracy than using full feature sets on all the eight datasets. The highest improvement was on the SRBCT dataset with 15% on average and 20% in the best case, reaching 100% accuracy.

The modest improvement was still 5% on average and 9% in the best case on Leukemia1. The results showed that the discriminating ability of the constructed features was much higher than the original all features although the number of constructed features was negligible to the original dimensionality.

For NB, the features constructed by CDFC also obtained better performance than Full on almost all datasets. For example, using the 4 features constructed on Prostate, NB achieved 32% higher accuracy than using the whole 10,509 features. Similarly, the improvement on Colon and DLBCL was 11% and 14% on average with 18% and 17% in the best case, respectively. Only on SRBCT, CDFC had about 2.4% lower accuracy than Full. However, the best accuracy achieved by CDFC was 100% which was 2.5% higher than the Full accuracy.

Compared with using Full, DT using features constructed by CDFC also had significantly better performance on six datasets, similar on one and worse on the remaining one. An improvement of at least 10% on average accuracy was achieved on three datasets, namely SRBCT, CNS and DLBCL, with the best accuracy improved from 15% to 22%. Only on Leukemia1, CDFC obtained about 5% lower average accuracy than Full. However, the best result was still better than Full.

In general, over the 24 comparisons with Full using the three learning algorithms on eight datasets, CDFC won 21, drew 1 and lost 2 in terms of average accuracy. However, in term of the best accuracy, CDFC outperformed Full in all 24 cases except for the tie result of DT on Leukemia1. The results showed that CDFC could construct a very small number of features with a high discriminating ability which can generalise well to the three learning algorithms in most cases.

5.1.2. CDFC versus hMCIFC

As shown in Table 3, although both methods constructed the same number of features for each dataset (since the construction ratio was set as 2 for both methods), KNN using features constructed by CDFC achieved significantly better performance than using those of hMCIFC on all datasets. The highest

improvement of 10% on average was found on Colon, where hMCIFC failed to maintain the Full accuracy. The results showed that using terminal sets comprising of features that are relevant to a specific class, CDFC achieved much better results than hMCIFC, allowing it to obtain the best KNN accuracy on all datasets.

Similarly, using features constructed by CDFC, NB achieved significantly better accuracy on 7 datasets than using those constructed by hMCIFC. Among these datasets, CDFC further improved the performance of hMCIFC from 6% to 12% on 5 datasets. Only on Ovarian, CDFC obtained a similar accuracy as hMCIFC.

For DT, features constructed by CDFC obtained significantly better performance than those of hMCIFC on five datasets, namely Colon, DLBCL, CNS, Prostate and SRBCT with a further improvement of 3% to 7%. For the remaining three datasets, CDFC had similar performance as hMCIFC on two and worse on one.

In general, compared with hMCIFC using the three learning algorithms, CDFC won 20, drew 3 and lost 1 out of the 24 comparisons.

Comparisons between the three learning algorithms revealed an interesting phenomenon. Recall that the features constructed by both hMCIFC and CDFC were optimised towards DT performance by using either DT accuracy (hMCIFC) or information gain (CDFC) in their fitness functions. However, the results showed that the improvement of DT performance was not as much as KNN and NB, making its accuracy usually lower than the other two learning algorithms. This may relate to the characteristics of DT where classification decision highly depends on the split values inside the internal nodes of the DT trees. These values are learned based on the training data, which may not be generalised well to the unseen test data of these datasets due to their skew distributions with many outliers as discussed in [13].

Another interesting observation from the results of CDFC and hMCIFC was that although DT used IG as its base measure, it had less power than the average IG measure in evaluating GP individual with multiple constructed features.

This finding is contradictory with common practice where wrapper approaches are preferred to filter ones because they usually produce better classification performance. However, the results showed that CDFC obtained significantly higher accuracy than hMCIFC in almost all cases. A closer investigation on the evolutionary process of hMCIFC showed that in a set of multiple constructed features that gave very good DT accuracy, there might exist very bad or even constant-value features. The reason is that DT does not use all features in building the DT tree. Therefore, the returned classification accuracy does not reflect the goodness of all constructed features. On the other hand, the average IG shows the average relevance level of all the constructed features of the individual. Therefore, using average IG as a fitness measure, GP can better evaluate individuals and choose those that comprise more good features to create better offspring in its evolutionary process.

In summary, features constructed by CDFC had significantly better performance than those constructed by hMCIFC on 20 cases, similar on 3 and worse on 1. Note that results of CDFC had smaller standard deviation than hMCIFC in almost all cases. This indicated that by constraining the terminal sets to class relevant features, the performance of CDFC was better and more stable than hMCIFC.

5.1.3. Computation Time

Fig. 4 shows the average running time to complete a single run for hMCIFC and CDFC. Results in the figure showed that the CDFC running time was less than half of hMCIFC on five out of the eight datasets, and less than 65% on the remaining three.

Note that both methods used the same population size and the maximum number of generations. In other words, they had the same number of evaluations. Therefore, the running time difference was mainly contributed by the size of GP individuals and the computation time of the fitness evaluations. Fig. 5 shows the average size of 500 GP individuals returned by both methods. The results showed that CDFC had larger individuals than hMCIFC on 6 out of

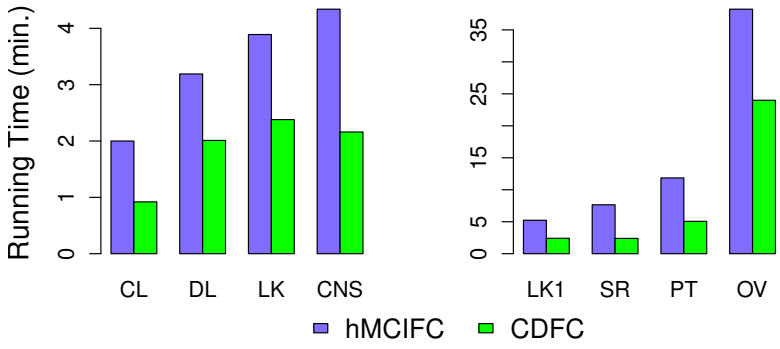


Figure 4: Computation time of CDFC versus hMCIFC.

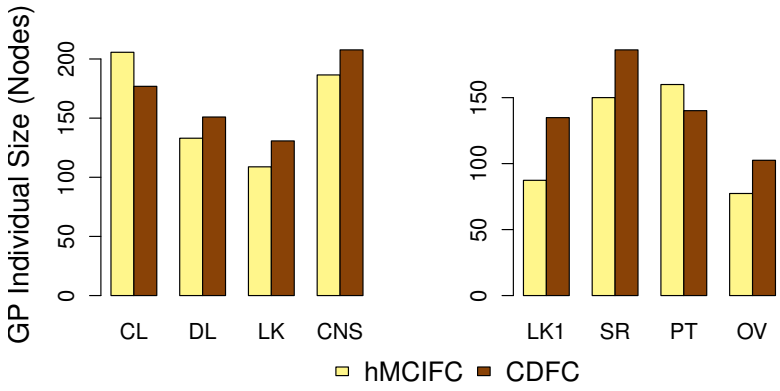
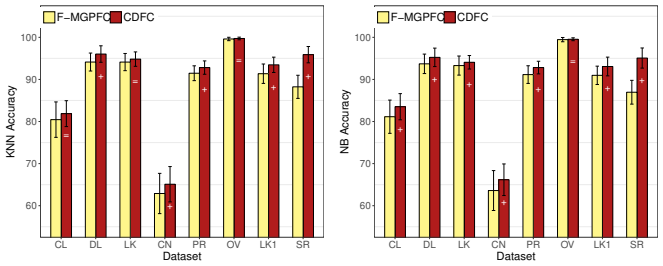


Figure 5: The average size of GP individuals of CDFC versus hMCIFC.

8 datasets. Therefore, CDFC would required longer time to process its individuals. However, as shown in Fig. 4, CDFC running time was much shorter than hMCIFC on all datasets. This means that the time increased in CDFC for processing GP trees was much smaller than the time increased in hMCIC for fitness evaluation. The fitness function of CDFC comprises of two filter measures, distance and IG, while hMCIFC combines distance with a wrapper measure which is an average accuracy of 9 DT models built on the training set. Although IG is the base measure of DT, computation time of the average IG



(a) KNN (b) NB

(c) DT (d) Average accuracy Improvement

Figure 6: Results of class-dependent (CDFC) versus class-independent (fMCIFC) feature construction.

of each constructed feature is still much lower. Therefore, the running time of CDFC was much faster than hMCIFC. This again confirmed the efficiency of filter measures.

5.2. Class-Dependent versus Class-Independent FC

Since CDFC differs from hMCIFC not only in the class-dependent feature construction strategy but also in the fitness function, the superior performance of CDFC over hMCIFC might be contributed by either components or both of them. Therefore, the effectiveness of the proposed class-dependent feature construction strategy should be confirmed by another set of experiments where CDFC is compared with with *fMCIFC*, which is the same method as hMCIFC except that its fitness function uses Eq. (9).

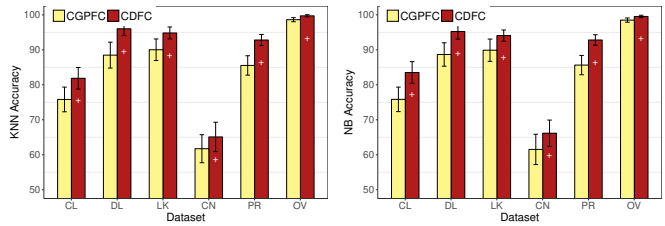
Fig. 6 shows the average results over the 50 runs of fMCIFC and CDFC.

The first three sub-figures present the average test accuracy of KNN, NB and DT of each method. In these figures, each group of bars shows the results of the features constructed by fMCIFC and CDFC on each dataset. On the CDFC bars, results of the Wilcoxon significance test with a 5% significance level comparing CDFC against CGPFC are displayed. "+" and "–" mean that CDFC is significantly better or worse than CGPFC. "=" means that they are similar. The last subfigure shows the average accuracy improvement of each learning algorithm on each dataset.

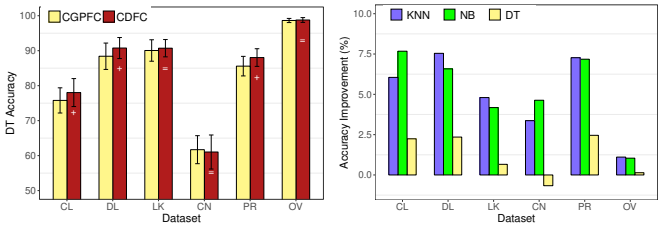
As can be seen in Fig. 6(a), using CDFC constructed features, KNN obtained significantly higher average accuracies than using fMCIFC constructed features on all datasets with the biggest gap of 7.9% on SRBCT which is a four-class problem. A similar pattern was seen for NB in Fig. 6(b) with a significant improvement made on seven out of the eight datasets. SRBCT was also the one that NB obtained the highest average improvement of more than 8% on average. For DT, significantly higher accuracies were found on three datasets and the remaining five had similar performance as fMCIFC. The compared results of different learning algorithms in Fig. 6(d) showed that among the three algorithms, KNN had the highest improvements on five datasets and NB on the remaining three. In general, features constructed by CDFC helped the three learning algorithms either obtained a significantly better or similar classification performance on all datasets. This indicated that by constructing features from relevant features to one class, thus narrowing the GP search space, and evaluating each constructed feature against the corresponding class only, CDFC could construct features with better discriminating ability.

5.3. Multiple versus Single Feature Construction

This section will check the hypothesis that multiple constructed features may better represent the original problem than a single constructed feature. Since the clustering-based single feature construction method (CGPFC) proposed in [23] has shown to have better performance than standard GP, the performance of the features constructed by CDFC will be compared with the single feature



(a) KNN (b) NB



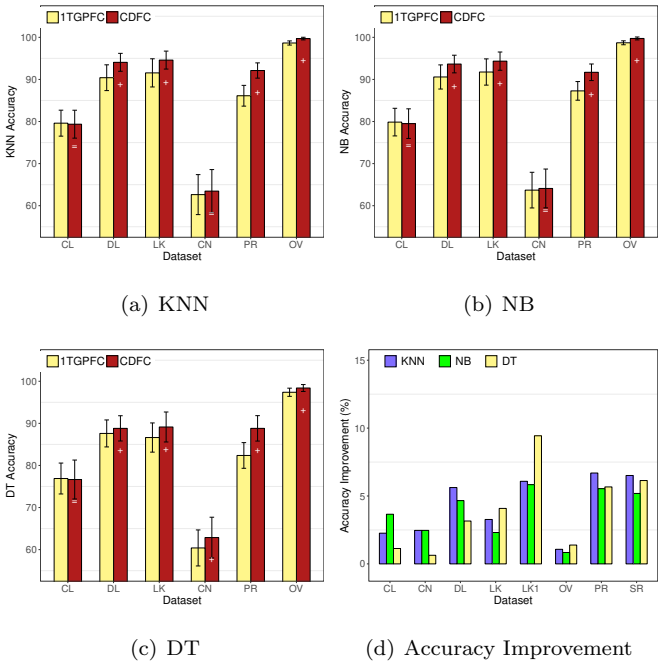
(c) DT (d) Accuracy Improvement

Figure 7: Results of multiple feature construction (CDFC) versus single feature construction (CGPFC)

constructed by CGPFC.

Fig. 7 shows the compared results of CDFC and CGPFC over the 50 runs on six binary-class datasets. Since CGPFC was designed to construct features for binary-class problems only, the two multiple-class datasets, namely Leukemia1 and SRBCT, are left out in this comparison.

Results in Fig. 7(a) showed that the CDFC constructed features obtained a significantly higher KNN accuracy than CGPFC on all datasets with the largest gap of 7.5% on DLBCL. Similar pattern was observed in Fig. 7(b) for NB with an improvement made on all the datasets. For DT, significantly higher accuracies were found on 3 datasets and similar accuracies were found on the remaining three. In general, compared with the single feature constructed by CGPFC, the four features constructed by CDFC helped the three learning algorithms either obtained a significantly better or similar classification performance on all datasets. The compared results of different learning algorithms in Fig. 7(d) showed that among the three algorithms, KNN had the highest improvements



(a) KNN (b) NB

(c) DT (d) Accuracy Improvement

Figure 8: Results of multiple-tree GP (CDFC) versus single-tree GP (1TGPFC) for multiple feature construction.

on four out of the six datasets. Furthermore, as can be seen in the first three subfigures of Fig. 7, the error bars of CDFC were smaller than the corresponding error bars of CGPFC in almost all cases. This indicated that using the CDFC constructed features, the three learning algorithms obtained more stable results than using the one constructed by CGPFC.

5.4. Multi-Tree GP versus Single-Tree GP for Multiple Feature Construction

This section will compare the performance of CDFC with the method proposed by Neshatian et al. [6] where single-tree GP was used to construct multiple features. It is named 1TGPFC here for presentation convenience. In this method, GP was run multiple times, and each time constructed one feature focusing on discriminating instances from one class to other classes. Therefore, two features will be constructed for a binary-class problem. 1TGPFC also follows a filter approach that uses the impurity of the class interval to evaluate

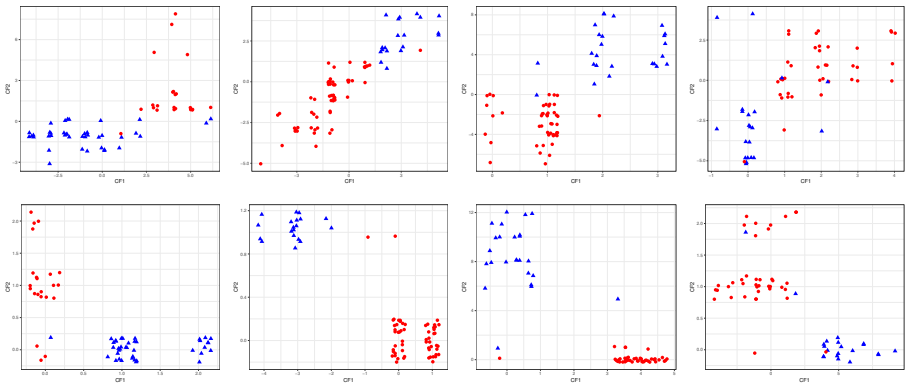
the constructed feature. The class interval is determined based on the range of the constructed feature values but excluding 5% of the data points at either end of the intervals. Then the total number of data points which appeared in the wrong class interval is used as the fitness value that needs to be minimised.

For a fair comparison with CDFC, the terminal set of 1TGPFC is also filtered as in hMCIFC and fMCIFC. 1TGPFC is also run with the same settings for GP parameters as in CDFC. In this set of experiments, CDFC was run with the construction ratio 1 to construct the same number of features as 1TGPFC. The average results over the 50 runs of both methods are displayed in Fig. 8.

Results from Fig. 8 showed that using the CDFC constructed features, all the three learning algorithms obtained significantly higher accuracies than using the ones created by 1TGPFC on all datasets. KNN obtained the largest increase of 8.1% average accuracy on Prostate, NB had 7.2% biggest gap on CNS, and DT achieved 13% increase on SRBCT. Comparisons between the three learning algorithms in Fig. 8(d) showed that KNN had the highest improvement on four datasets. On the other four datasets, DT had the largest difference of 10.5% and 13% on Leukemia1 and SRBCT, which are the multiple-class problems.

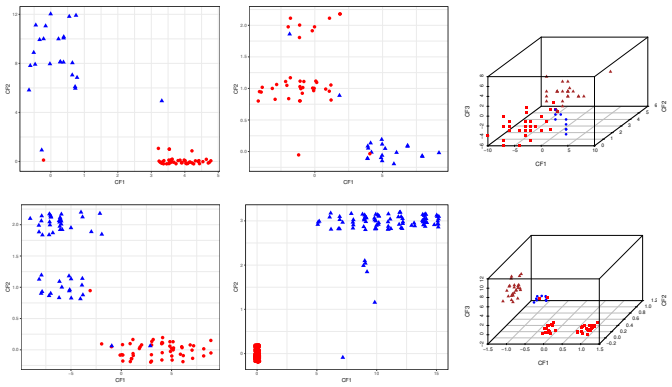
Although 1TGPFC used an impurity measure in its fitness function to minimise the impurity of the interval including constructed feature values of the focused class, its created features did not perform as well as expected, leading to significantly worse results than CDFC. This result may be related to the fact that by using a multi-tree representation to construct multiple features simultaneously, CDFC can evaluate the constructed features as a whole during the evolutionary process. This enabled it to optimise the discriminating power of the whole feature set, taking into account possible interactions between the constructed features. This ability is obviously impossible when running GP separately to construct one feature at a time as in 1TGPFC.

A confirmation of this hypothesis was made by having a closer look at the features constructed by both the CDFC and 1TGPFC methods. A fair comparison is made by choosing the set of constructed features that had the worst performance among 50 results from 50 runs obtained by each method on the first



(a) Colon (b) DLBCL (c) Leukemia (d) CNS

Figure 9: Data transformed by 1TGPFC (the first row) and CDFC (the second row) on Colon, DLBCL, Leukemia, and CNS.



(a) Prostate (b) Ovarian (c) Leukemia1

Figure 10: Data transformed by 1TGPFC (the first row) and CDFC (the second row) on Prostate, Ovarian, and Leukemia1.

pair of training and test folds. The reason for comparing the worst results of both method is that the difference between the best results is not large enough to be visually distinguished. The transformed data of seven datasets (excluding SRBCT with more than 3 constructed features) using the constructed features by both methods are plotted in Figs. 9 and 10.

As can be seen from these figures, the constructed features by both methods

were quite good in grouping instances of different classes into separate clouds. However, as can be seen from the second row of each figure, the data clouds produced by CDFC were more compact and separated from each other than those created by 1TGPFC, which are shown in the first row of the figures. Therefore, with the new representation obtained from CDFC, it would be much easier for the three classification algorithms to find a model that achieved higher accuracies.

Since both methods constructed features focusing on discriminating instances of one class from the others, contributions to the superior results of CDFC mainly came from its fitness function. While both methods used an entropy-based measure to minimise the impurity of the constructed feature values within each class, CDFC incorporated an additional distance measure to evaluate the whole set of constructed features. The aim of the distance measure is to maximise the distance between instances of different classes and minimise the distance between instances of the same class. This is the reason why the data clouds produced by CDFC are more compact and scattered far away from each other.

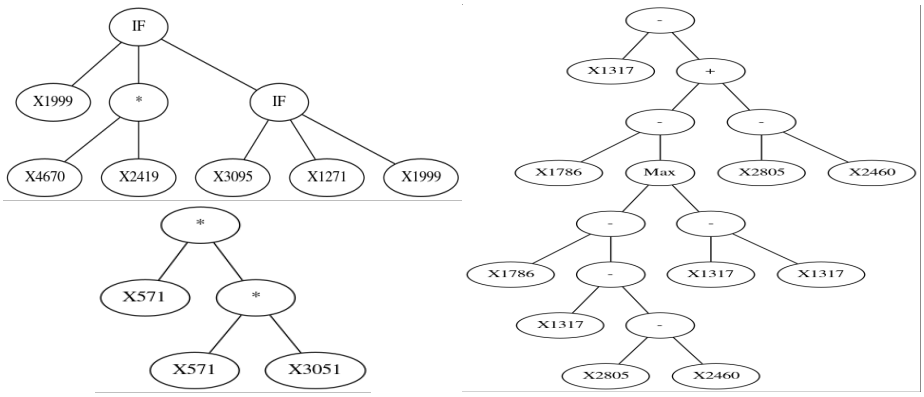


Figure 11: Three constructed features on Leukemia1: CF1 (upper left), CF2 (lower left) and CF3 (right).

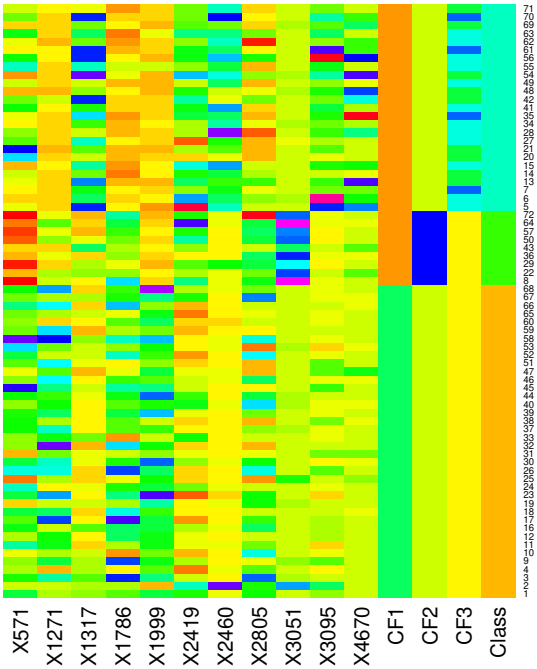


Figure 12: Heatmap showing the data of the 11 selected and 3 constructed features by the individual shown in Fig. 11 on Leukemia1.

5.5. Constructed Features

An investigation on the constructed features was conducted to further explain the effectiveness of CDFC. Among the 500 returned sets of three features constructed by CDFC for Leukemia1, a dataset with three classes, we chose one set that had a relatively small size to analyse.

Fig. 11 shows one of the returned GP individuals by CDFC on the Leukemia1 dataset. This individual has three trees representing 3 features constructed for 3 classes. It has totally 33 nodes including 11 features. Using this set of three constructed features, all the three learning algorithms which are KNN, NB and DT obtained 100% test accuracy.

A comparison between the original selected features and the constructed ones was conducted to show the effect of the feature construction. Fig. 12 shows the data of the 11 selected and 3 constructed features on Leukemia1 using the

individual shown in Fig. 11. All the instances (or rows) are sorted based on the class label (shown in the last column of Fig. 12). Class 1 (in orange) is the largest class with 25 instances located at the bottom of the heatmap. As can be seen from Fig. 12 that the values of feature X1999 can distinguish between class 1 and the other two classes. This confirms its importance and explains why CDFC chose it as the first feature in the tree CF1. Similarly, the two features appeared in CF2, X571 and X3051, have special ranges of values for instances belonging to class 2. Combining these features, CDFC constructed much purer features that are more relevant to the class label than the original ones as shown in the three columns CF1, CF2 and CF3 of Fig. 12. The constructed feature CF1 has only two values, one for class 1 and one for the other 2 classes. Similarly, CF2 and CF3 also have a distinct value for their associating class. Therefore, it is much easier for learning algorithms to generate a good model from these features.

6. Conclusion

The goal of this study was to investigate different approaches to multiple feature construction using GP which can produce a very small number of high-level features to improve the performance of common learning algorithms on high-dimensional data. Two methods using multi-tree GP representation were investigated, one constructs features that are independent to the class (MCIFC) and one constructs features that are class-dependent (CDFC). The two methods were compared in the same context of using t-Test to filter out irrelevant features as a means of narrowing GP search space. Two variants of MCIFC were compared with CDFC, one uses a hybrid evaluation method (hMCIFC) to synthesise the strength of wrapper and filter approach and the other uses the same filter measure as CDFC (fMCIFC).

Comparisons made on the eight high-dimensional datasets showed that the class-dependent constructed features achieved better classification performance than the class-independent ones, regardless of using the same filter measure or

the combined wrapper and filter one to evaluate the constructed features. Comparison of CDFC and the single feature construction method CGPFC proposed in [23]. has also shown that multiple constructed features better represent the original high-dimensional data than a single one.

CDFC also achieved better classification accuracy than the class-dependent feature construction method using the single tree GP representation (1TGPFC). This is due to the fact that CDFC uses a multi-tree representation, which enables it to construct and evaluate the whole set of constructed features in a single run. Therefore, it can determine the interaction of the newly constructed features during the construction process. Further analysis on the constructed features showed the effectiveness of CDFC's fitness function which combines two simple filter measures to guide the search for better discriminating features.

References

[1] P. N. Stuart J. Russell, Artificial intelligence: A modern approach (second edition), Pearson Education.

[2] H. Zhao, A. P. Sinha, W. Ge, Effects of feature construction on classification performance: An empirical study in bank failure prediction, Expert Systems with Applications 36 (2009) 2633–2644. doi:10.1016/j.eswa.2008.01.053.

[3] K. Neshatian, Feature manipulation with genetic programming, Ph.D. thesis, Victoria University of Wellington, Wellington, New Zealand (2010).

[4] S. Ahmed, M. Zhang, L. Peng, A new gp-based wrapper feature construction approach to classification and biomarker identification, in: IEEE Congress on Evolutionary Computation, 2014, pp. 2756–2763.

[5] L. Guo, D. Rivero, J. Dorado, C. R. Munteanu, A. Pazos, Automatic feature extraction using genetic programming: An application to epileptic eeg classification, Expert Systems with Applications 38 (8) (2011) 10425–10436.

[6] K. Neshatian, M. Zhang, P. Andreae, A filter approach to multiple feature construction for symbolic learning classifiers using genetic programming, IEEE Transactions on Evolutionary Computation 16 (5) (2012) 645–661.

[7] M. Garcia-Limon, H. J. Escalante, E. Morales, A. Morales-Reyes, Simultaneous generation of prototypes and features through genetic programming, in: Proceedings of the Annual Conference on Genetic and Evolutionary Computation, ACM, 2014, pp. 517–524.

[8] K. Krawiec, Genetic programming-based construction of features for machine learning and knowledge discovery tasks, Genetic Programming and Evolvable Machines 3 (2002) 329–343.

[9] M. Smith, L. Bull, Genetic Programming with a Genetic Algorithm for Feature Construction and Selection, Genetic Programming and Evolvable Machines 6 (2005) 265–281.

[10] B. Tran, M. Zhang, B. Xue, Multiple feature construction in classification on high-dimensional data using gp, in: 2016 IEEE Symposium Series on Computational Intelligence (SSCI), 2016, pp. 1–8.

[11] B. Tran, B. Xue, M. Zhang, Class dependent multiple feature construction using genetic programming forhigh-dimensional data, in: AI 2017: Advances in Artificial Intelligence, Vol. 10400 of Lecture Notes in Computer Science, Springer International Publishing, 2017, pp. 182–194.

[12] G. Chandrashekar, F. Sahin, A survey on feature selection methods, Computers & Electrical Engineering 40 (2014) 16–28.

[13] B. Tran, B. Xue, M. Zhang, Genetic programming for feature construction and selection in classification on high-dimensional data, Memetic Computing 8 (1) (2015) 3–15.

[14] M. L. Raymer, W. F. Punch, E. D. Goodman, L. A. Kuhn, Genetic programming for improved data mining: Application to the biochemistry of

protein interactions, in: Proceedings of the 1st Annual Conference on Genetic Programming, MIT Press, Cambridge, MA, USA, 1996, pp. 375–380.

[15] M. Ahluwalia, L. Bull, Coevolving functions in genetic programming: classification using k-nearest-neighbour, in: Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation-Volume 2, Morgan Kaufmann Publishers Inc., 1999, pp. 947–952.

[16] B. Bhanu, K. Krawiec, Coevolutionary construction of features for transformation of representation in machine learning, in: Proceedings of Genetic and Evolutionary Computation Conference, Press, 2002, pp. 249–254.

[17] F. Otero, M. Silva, A. Freitas, J. Nievola, Genetic programming for attribute construction in data mining, in: Genetic Programming, Vol. 2610, Springer Berlin Heidelberg, 2003, pp. 384–393. doi:10.1007/3-540-36599-0_36.

[18] M. Muharram, G. Smith, The Effect of Evolved Attributes on Classification Algorithms, in: AI 2003: Advances in Artificial Intelligence, Vol. 2903, Springer Berlin Heidelberg, 2003, pp. 933–941.

[19] M. Muharram, G. Smith, Evolutionary constructive induction, IEEE Transactions on Knowledge and Data Engineering 17 (2005) 1518–1528.

[20] K. Neshatian, M. Zhang, M. Johnston, Feature Construction and Dimension Reduction Using Genetic Programming, in: Advances in Artificial Intelligence, Vol. 4830, Springer Berlin Heidelberg, Berlin, Heidelberg, 2007, pp. 160–170.

[21] K. Neshatian, M. Zhang, Genetic programming for performance improvement and dimensionality reduction of classification problems, in: IEEE Congress on Computational Intelligence, 2008, pp. 2811–2818.

[22] H. B. Nguyen, B. Xue, P. Andreae, A Hybrid GA-GP Method for Feature Reduction in Classification, Springer International Publishing, 2017, pp. 591–604. doi:10.1007/978-3-319-68759-9_48.

[23] B. Tran, B. Xue, M. Zhang, Using feature clustering for gp-based feature construction on high-dimensional data, in: Proceedings of 20th European Conference on Genetic Programming, Springer International Publishing, Cham, 2017, pp. 210–226.

[24] M. G. Smith, L. Bull, Using Genetic Programming for Feature Creation with a Genetic Algorithm Feature Selector, Springer Berlin Heidelberg, Berlin, Heidelberg, 2004, pp. 1163–1171. doi:10.1007/978-3-540-30217-9_117.

[25] M. Smith, L. Bull, Improving the human readability of features constructed by genetic programming, in: Proceedings of the 9th annual conference on Genetic and evolutionary computation, ACM, 2007, pp. 1694–1701.

[26] H. Vafaie, K. De Jong, Genetic algorithms as a tool for restructuring feature space representations, in: Proceedings of the Seventh International Conference on Tools with Artificial Intelligence, 1995, pp. 8–11. doi:10.1109/TAI.1995.479372.

[27] G. Patterson, M. Zhang, Fitness functions in genetic programming for classification with unbalanced data, in: Proceedings of the 20th Australian Joint Conference on Artificial Intelligence (AI), Springer Berlin Heidelberg, 2007, pp. 769–775.

[28] H. Al-Sahaf, A. Al-Sahaf, B. Xue, M. Johnston, M. Zhang, Automatically evolving rotation-invariant texture image descriptors by genetic programming, IEEE Transactions on Evolutionary Computation 21 (1) (2017) 83–101. doi:10.1109/TEVC.2016.2577548.

[29] S.-H. Cha, Comprehensive survey on distance/similarity measures between probability density functions, International Journal of Mathematical Models and Methods in Applied Sciences 1 (2007) 300.

[30] Z. Zhu, Y.-S. Ong, M. Dash, Markov blanket-embedded genetic algorithm for gene selection, Pattern Recognition 40 (11) (2007) 3236–3248.

[31] A. Statnikov, C. F. Aliferis, I. Tsamardinos, D. Hardin, S. Levy, A comprehensive evaluation of multicategory classification methods for microarray gene expression cancer diagnosis, Bioinformatics 21 (2005) 631–643.

[32] C. Ding, H. Peng, Minimum redundancy feature selection from microarray gene expression data, Journal of bioinformatics and computational biology 3 (02) (2005) 185–205.

[33] F. Wilcoxon, Individual comparisons by ranking methods, Biometrics Bulletin 1 (6) (1945) 80–83.