Pattern Recognition 93 (2019) 404–417

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

Genetic programming for multiple-feature construction on high-dimensional classification

Binh Tran*, Bing Xue, Mengjie Zhang

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

ARTICLE INFO

Article history:
Received 26 August 2018
Revised 4 April 2019
Accepted 1 May 2019
Available online 4 May 2019

Keywords:
Feature construction
Genetic programming
Classification
Class dependence
High-dimensional data

ABSTRACT

Data representation is an important factor in deciding the performance of machine learning algorithms including classification. Feature construction (FC) can combine original features to form high-level ones that can help classification algorithms achieve better performance. Genetic programming (GP) has shown promise in FC due to its flexible representation. Most GP methods construct a single feature, which may not scale well to high-dimensional data. This paper aims at investigating different approaches to constructing multiple features and analysing their effectiveness, efficiency, and underlying behaviours to reveal the insight of multiple-feature construction using GP on high-dimensional data. The results show that multiple-feature construction achieves significantly better performance than single-feature construction. In multiple-feature construction, using multi-tree GP representation is shown to be more effective than using the single-tree GP thanks to the ability to consider the interaction of the newly constructed features during the construction process. Class-dependent constructed features achieve better performance than the class-independent ones. A visualisation of the constructed features also demonstrates the interpretability of the GP-based FC approach, which is important to many real-world applications.

1. Introduction

In machine learning, data representation is a critical factor contributing to the performance of machine learning and pattern recognition methods. With the advances in data collection technologies, more and more high-dimensional data are collected. These datasets can have thousands of features or more. They may or may not interact with each other under unknown rules to determine the class label. 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 representation of the whole feature set [1]. Therefore, automatic data transformation to obtain a smaller and more discriminating feature set becomes an important process for effective machine learning and pattern recognition [2]. Feature learning has shown to be effective in speech recognition [3], face recognition [4], robotics [5], disease detection [6], etc.

A popular approach to automatic feature learning is the neural network (NN) based deep learning where high-level features are generated in hidden layers of neurons from the input images [7], video [8], and text [9]. However, how to design an appropriate deep NN architecture for a specific problem typically still requires a lot of trial and error or expert knowledge of the field. Furthermore, effectively training a deep NN usually requires a significant amount of data, which may not be available in many applications. These issues make feature learning less popular in problems that can not meet these requirements. This is where *Feature Construction (FC), which is one type of feature learning*, can be used to automatically learn more discriminating features from the data.

Genetic programming (GP) is an evolutionary computation technique that evolves a population of solutions or individuals based on the idea of the survival of the fittest. Using genetic operators such as crossover and mutation, GP can evolve better offspring from fittest parents evaluated based on some objective set in the fitness function. This evolutionary principle is the same as in genetic algorithms (GAs). However, while GAs work only on vector-based representation, GP can work on more flexible representations such as trees or graphs. The tree-based representation provides a natural representation for FC where a constructed feature can be represented as a tree with features or constants in the leaf nodes serving as arguments of the internal nodes which can be arithmetic operators. A GP individual can represent a single feature (i.e. single-tree representation) or multiple features (i.e multi-tree representation).

* Corresponding author.
E-mail addresses: binh.tran@ecs.vuw.ac.nz, tnbinh@cit.ctu.edu.vn (B. Tran).

https://doi.org/10.1016/j.patcog.2019.05.006
0031-3203/

With a flexible tree-based representation and a population-based search, GP has shown to be effective in automatically constructing a more discriminating feature without requiring a predefined model and a huge amount of training data [10]. Furthermore, the built-in feature selection process allows GP to select more relevant features to form the new feature. This ability is especially beneficial for big data, where a large number of features are collected before a specific task is created. This means that many features can be irrelevant to the task. Feeding all of them into the learning algorithms may unnecessarily increase the running time and degrade their performance. FC is also used for dimensionality reduction, especially for high-dimensional data. The number of constructed features can be very small compared to the original number of features, which significantly reduces the data size and helps machine learning methods improve their performance. In addition, the features constructed by GP have better interpretability than those constructed in the hidden layers of NNs. With these advantages, FC using GP could be an alternative choice for problems that deep learning approaches cannot provide a good solution.

Many GP-based FC methods have been proposed to construct a single feature as an augmentation or multiple features as a replacement of the original feature set. In the case of high-dimensional data, augmenting a single constructed feature cannot obtain a different performance [10]. On the other hand, how to create a small set of constructed features that can significantly improve the discriminating ability of the data is still challenging. An investigation of different approaches to multiple-feature construction using GP is needed to get an insight into this task. There are different types of multiple-feature construction methods, which can be categorised based on the *representation*, the *evaluation* method, and whether a constructed feature is *class-dependent* (i.e. the feature is constructed particularly for a particular class) or *class-independent*.

Representation: Multiple-feature construction methods have been proposed using both single-tree [11,12] and multi-tree [13,14] representation. When using single-tree representation, GP can construct multiple features by using all possible subtrees [11] or some subtrees under predefined special nodes [12]. Another approach is to run single-tree GP multiple times, each time constructs a new feature [15]. Multi-tree GP was also proposed and shown to be effective in constructing multiple features on datasets with about tens of features [13,14] as well as thousands of features [16].

Evaluation: During the FC process, different types of methods can be used to evaluate the constructed features. They are filter, wrapper, embedded or combination of these approaches [17]. While filter methods use some measure such as information gain or correlation to evaluate the constructed features [15], wrapper methods use a classification algorithm to evaluate them [12]. Although wrapper methods are usually computationally more expensive than filter methods, they usually obtain better classification accuracy. On the other hand, filters are usually more general than wrappers. Combination of the two measures was also proposed to better evaluate the constructed feature set [18]. Since a GP tree (or the constructed feature) can be used as a binary classifier, its classification performance can be used to evaluate its performance. This scenario is referred to as embedded FC [10].

Class-dependency: In addition to the choice between single or multi-tree representation, constructing class-dependent or class-independent features is another option in designing FC methods. Most of the proposed GP-based FC methods are class-independent [11,13], where a high-level feature is constructed without focusing on any class of the problem. In contrast, each class-dependent constructed feature in [15] aims at distinguishing instances of one class from the other classes. However, the method is limited to construct one feature for each class, which may not scale well to high-dimensional data. Recently, a multiple class-dependent FC method for high-dimensional data [18] was proposed and shown to achieve better performance than the class-independent FC methods.

Although FC 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 FC is larger since it requires to choose not only a good feature subset but also an appropriate set of operators to combine them for a more discriminating feature. This makes FC on high-dimensional data a challenging task. There have been many studies investigating GP operators and GP itself, but not much on GP for FC, especially multiple-feature construction on high-dimensional data. It is necessary to investigate what are the key factors and how they influence the performance of different approaches to GP for FC.

In this study, three multiple-feature construction methods are investigated including two methods using multi-tree representation, namely the class-independent (MCIFC) [16] and the class-dependent (CDFC) [18], and one method using single-tree representation (1TGPFC) proposed by Neshatian et al. [15] to construct class-dependent features. Performance of the constructed features by the three methods will be compared using the classification performance of common learning algorithms including k-Nearest Neighbour (KNN), Naive Bayes (NB) and Decision Tree (DT). Although this study is based on the two previously published conference papers, this paper substantially extends the two small conference papers by providing more comparisons and analysis between different approaches in designing key components of GP-based feature construction methods including: (1) Using *single-tree (1TGPFC)* versus *multi-tree (CDFC) representation* for multiple-feature construction; (2) Using different *evaluation methods* to evaluate constructed features during the evolutionary process; (3) Constructing *class-dependent* versus *class-independent features* within the same setting. More analysis is also provided to reveal the insights of the proposed approach by (4) visualising the *constructed features* and (5) comparing with *more baseline methods*.

2. Background and related work

2.1. Genetic programming algorithm

Algorithm 1 shows the pseudo code of a standard GP algorithm for multiple-feature construction using multi-tree representation to construct m new features.

Algorithm 1: GP-based multiple-feature construction.

```
   Input  : train_set, m
   Output: The best set of m constructed features
1  begin
2      Initialize a population of GP individuals. Each individual is an
       array of m trees;
3      best_ind ← the first individual;
4      while Maximum generation is not reached do
5          for i = 1 to Population Size do
6              transf_train ← Calculate constructed features of individual
               i on train_set;
7              fitness ← Apply fitness function on transf_train ;
8              Update best_ind if individual i is better than best_ind;
9          end
10         Select parent individuals using tournament selection for
           breeding;
11         Create new individuals from selected parents using crossover
           or mutation;
12         Place new individuals into the population of the next
           generation;
13     end
14     Return best_ind;
15 end
```

2.2. Related work

2.2.1. Single-Tree GP-Based feature construction

One of the early GP based FC methods using single-tree representation was proposed by Raymer et al. [19]. It aimed to improve a previously proposed GA to evolve weights that can transform each original features into a new one. GP was proposed to enable non-linear transformation for each feature. Results on a water displacement problem with four features showed that the proposed 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 m single-tree GP populations to construct m features. In [20], one GP population was used to evolve a new feature for each original feature. Another GA population was used for feature selection. Similarly, in [21], m concurrent populations of single-tree individuals was used to evolve m desired new features. 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 is m times greater than that of standard GP, the computational time of cooperative coevolution approach is significantly high.

Constructing multiple features from all possible subtrees of a single-tree, Ahmed et al. [11] proposed two GP based wrapper FC methods based on random forest (RF) for high-dimensional data, one used accuracy, one used entropy gain of RF and the p-value of an ANOVA test applied on the selected features as the fitness measure. Results showed that the latter achieved better generalisation ability and smaller numbers of features than the former. However, the computational cost was quite high when adopting RF in the fitness function.

Using information gain ratio to evaluate the constructed feature, filter FC methods [22,23] were proposed to construct continuous or boolean features using arithmetic and relational operators. Results showed that DT obtained better performance when the feature set was augmented by the constructed feature. Neshatian et al. [24] proposed a class-dependent multiple-feature construction by running single-tree GP program multiple times (called as 1TGPFC in this paper), each constructed one feature that distinguished instances of one class from the others. The number of constructed features is equal to the number of classes. Constructed features were evaluated based on the impurity of the intervals which were formed by applying class dispersion to the transformed data. A faster fitness function for 1TGPFC was proposed in [15]. The constructed features helped DT achieve smaller error rates with much smaller sizes than using original features in most cases. 1TGPFC [15] will be used as a baseline method in this paper. 1TGPFC's performance was also improved by adding one new class-wise orthogonal transformed feature for each original feature [25]. However, this strategy is not suitable for high-dimensional data.

By considering the single-tree as a classifier which can evaluate the performance of the constructed feature, an embedded FC method (GPFC) [10] 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 FC method (CGPFC) [26] where feature clustering was used to narrow the GP search space. The results 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 GPFC and CGPFC can only be applied to binary-class problems.

Combining GA and GP in a single representation, Nguyen et al. [27] proposed to construct and select features simultaneously in a single process. For an n-dimension problem, each individual contained an n-bit string for feature selection and a single tree to construct one new feature. Results on UCI datasets showed that the proposed method obtained either similar or significantly better accuracies than the compared methods. 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.

2.2.2. Multi-tree GP-Based feature construction

Multi-tree representation has been used in multiple-feature construction methods. Krawiec [28] used a multi-tree GP to construct m predefined number of features. Each GP individual has m new features and m hidden features and was evaluated by DT. Hidden features were kept out of the evolutionary process to avoid losing good features. Results on datasets with less than ten features showed that constructed features improved the classification performance of DT on most datasets.

A combined GP+GA method was proposed in [29] where each individual had n trees for multiple-feature construction and n flags for feature selection. The method has shown to significantly improve the classification performance on 2 out of 10 datasets with tens of features. The reverse order GA+GP was investigated in [30] which showed an encouraging result an image dataset. The results from these works have demonstrated that combinations of evolutionary computation techniques are promising approaches to feature learning. However, on top of the high computational cost, these representations may not scale well with datasets having thousands of features.

3. Multiple-feature construction methods

To make this paper easy to follow, this section will briefly describe the class-independent and class-dependent multiple-feature construction methods that will be investigated as they were proposed in the original papers.

3.1. MCIFC: a multiple class-independent feature construction method

3.1.1. MCIFC representation

MCIFC [16] 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 larger number of classes is more complex and may require more features to represent the data. Let r be the construction ratio (e.g. 2 or 3), the number of total constructed features m is calculated using Eq. (1) given c is the number of classes. For example, if $r = 2$, MCIFC constructs 4 features for a binary-class problem as shown in Fig. 1.

$$m = r \times c \tag{1}$$

3.1.2. MCIFC crossover and mutation

To create new individuals from current ones, MCIFC uses the subtree crossover and mutation operators as described in Algorithm 2. Each genetic operator is designed to mutate or crossover only one constructed feature in the parents.

Changing more features in one operator can increase exploration; however, with the price of lower convergence. Note that there are 2^N possible feature subsets where N is the number of features. The GP search space is larger than 2^N because GP has to choose not only a feature subset but also a subset of operators to

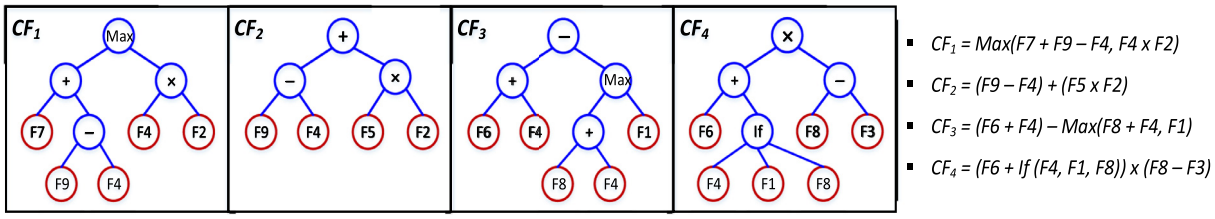


Fig. 1. MCIFC representation for a binary-class problem given $r = 2$.

Algorithm 2: MCIFC Crossover and Mutation.

```
prob ← randomly generated probability;
DoMutation ← (prob < mutation_rate);
if (DoMutation) then
    p ← Randomly select an individual using tournament selection;
    f ← Randomly select a feature/tree from m trees of individual p ;
    s ← Randomly select a subtree in tree f ;
    Replace s with a newly generated subtree;
    Return one new individuals;
else
    p1, p2 ← Randomly select 2 individuals using tournament selection;
    f1, f2 ← Randomly select a feature/tree from m trees of p1 and p2, respectively ;
    s1, s2 ← Randomly select a subtree in f1 and f2, respectively ;
    Swap s1 and s2;
    Return two new individuals;
end
```

combine them. Since the datasets used in this study have thousands of features, slightly changing parents to increase exploitation is a better choice.

3.1.3. MCIFC fitness function

MCIFC uses a weight (α) to combine the DT classification accuracy and a distance measure in the fitness function as shown in Eq. (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 are 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 is small. The balanced accuracy [31] is calculated based on Eq. (3) given c as the number of classes, TP_i and S_i as the number of correctly identified instances and the number of total instances of class i, respectively.

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

The *Distance* measure [32] calculated based on Eq. (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 Eqs. (5) and (6).

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

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

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

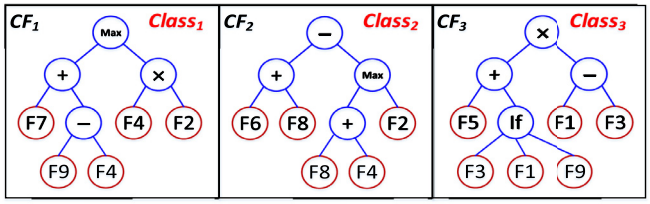


Fig. 2. Representation of a class-dependent GP individual with construction ratio 1.

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

where $Dis(V_i, V_j)$ is the distance between two vectors V_i and V_j, which is approximated by the Czekanowski measure [33] as shown in Eq. (7).

3.2. CDFC: a multiple class-dependent feature construction method

3.2.1. CDFC representation

CDFC [18] representation is similar to MCIFC with the number of constructed features 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 for a three-class problem given the construction ratio $r = 1$. 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.

In CDFC, a new feature *cf* is constructed from a subset of the original features that is relevant to the class that *cf* associates with. t-Test is used to measure how relevant a feature *f* is to class *c*. Values of *f* are first divided into two groups, one belongs to class *c* and one does not. Then, Eq. (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 $\geq$ 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.

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

3.2.2. CDFC crossover and mutation

CDFC uses the same genetic operators as MCIFC except for one variant: the crossover operator can only swap subtrees of features

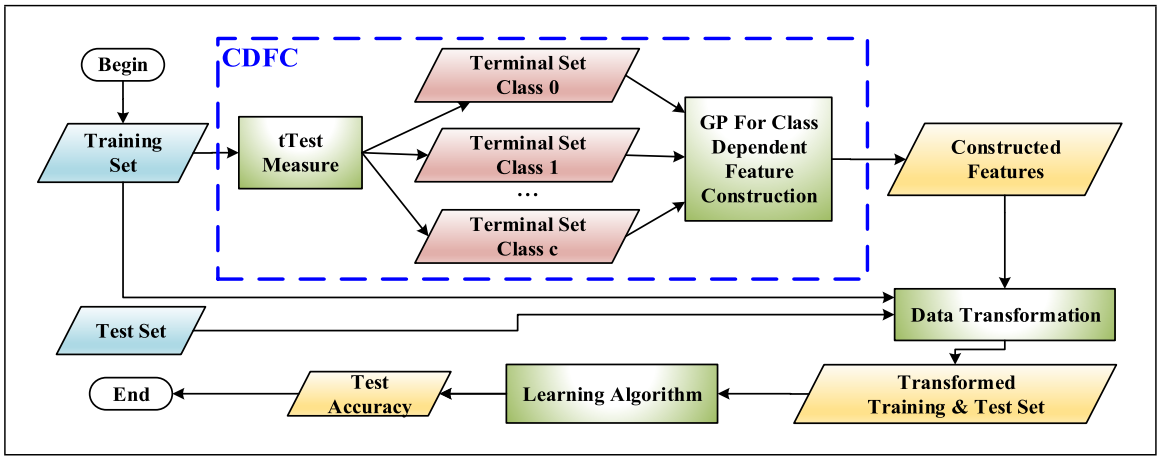


Fig. 3. CDFC overall system.

that associate with the same class since a constructed feature in CDFC is class-dependent. Specifically, Line 11 of Algorithm 2 selects the same array index among m features of two parents $p1$ and $p2$ for features $f1$ and $f2$.

3.2.3. CDFC fitness function

To speed up the evaluation process, CDFC replaces the DT classification accuracy (*Bal_Accuracy*) in Eq. (2) by the information gain (IG), which is the base measure of DT, as shown in Eq. (9). *AvgIG* is the average IG of all constructed features. The size of the GP individual *indSize* is also added for small-tree preference using a very small weight (10^{-7}) in order to limit its effect on cases when two individuals have the same information gain and distance.

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

AvgIG is calculated based on Eq. (10) where $f_{\max}$ is the best feature with the highest IG among m constructed features. The IG of $f_{\max}$ is added to distinguish two candidates with the same IG average. IG of feature f is calculated based on unconditional and conditional entropy H as in Eq. (11).

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

$$IG(f, class) = H(class) - H(class|f) \tag{11}$$

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 overlapped since one feature can be relevant to more than one class. The tree associated with class i is constructed using the ith terminal set.

MCIFC and CDFC are different in a number of aspects. Firstly, their aims are different, constructing class-independent versus class-dependent features. Secondly, they use different fitness functions, which will lead to different performance. Finally, while MCIFC selects features from the whole set of original features to create new ones, CDFC selects features from a smaller and generally better set than MCIFC. Therefore, CDFC's search space is smaller than MCIFC. Results in [18] have shown that CDFC achieves significantly better performance than MCIFC. However, which of the above aspects contributing to the superior performance of CDFC is unclear. Therefore, this study also conducts further experiments to better answer this question.

Table 1
Datasets.

Dataset	#F	#Inst.	#Class	Class Distribution	Ref.
Colon	2000	62	2	35%, 65%	[36]
DLBCL	5469	77	2	25%, 75%	[37]
Leukemia	7129	72	2	35%, 65%	[36]
CNS	7129	60	2	35%, 65%	[36]
Prostate	10,509	102	2	49%, 51%	[37]
Ovarian	15,154	253	2	36%, 64%	[36]
Leukemia1	5327	72	3	12%, 35%, 53%	[37]
SRBCT	2308	83	4	13%, 22%, 30%, 35%	[37]

4. Experiment design

4.1. Datasets

In order to investigate different GP-based FC approaches that generally work on high-dimensional data, our experiments use eight gene expression datasets with thousands of features to tens of thousands of features. These datasets are popularly used in studies addressing high dimensionality reduction [34,35]. Details about these datasets are shown in Table 1. The small number of instances of these datasets is due to the nature of gene expression data in which the cost of collecting one sample is very high.

Data is pre-processed to reduce noise generated during the data collection in laboratories [10]. Features are standardised (based on the training set) to have zero mean and unit variance, then discretised into -1, 0 and 1 representing the under-expression, the baseline and the over-expression states of a gene. The features are set to 0 if their values are in the $[-0.5, 0.5]$ interval, and to -1 or 1 if they are on the left and right of the interval, respectively.

4.2. Experiment configuration

Due to the small number of instances in each dataset, 10-fold CV is used to generate the training and test sets, where one fold is used for testing, and the other 9 folds for training. Only the training data is used during the FC process. As GP is a stochastic algorithm, 50 independent runs of each method with different random seeds are executed on each training set, resulting in 500 results (50×10). The results of 50 runs from each method are compared using the Wilcoxon statistical significance test [38], with a significance level of 0.05.

Table 2
Parameter settings.

Function set	+, −, ×, 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

4.3. Program and 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 (+ , − , ×), 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 value is 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 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

This section has five sections. Section 5.1 compares the multiple constructed features with the single constructed feature. Section 5.2 discusses the results of two approaches to multiple-feature construction using single-tree and multi-tree GP. Then comparisons between MCIFC and CDFC are presented in Sections 5.3 and 5.4 to reveal the effectiveness of different fitness evaluation functions and the class-dependent versus class-independent multiple-feature construction approaches. Finally, Section 5.5 compares GP-based FC approach with non-GP FC approaches.

5.1. Multiple versus single feature construction

This section checks the hypothesis that multiple constructed features may better represent the original problem than a single constructed feature. Since the clustering-based single FC method (CGPFC) [26] has shown to have better performance than standard GP, CDFC will be compared with CGPFC. Since CGPFC can be applied to binary-class problems only, the two multiple-class datasets, namely Leukemia1 and SRBCT, are left out in this comparison.

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 the single feature constructed by CGPFC. The number of 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. Column S_1, S_2, and S_3 display the Wilcoxon test results for KNN, NB, and DT, respectively. "+" or "−" indicates that the corresponding method is significantly better or worse than CDFC. "=" means they have similar performance. In other words, the more "−", the better the multiple-feature construction method.

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 DLBCL dataset with 12% on average and 14% in the best case, reaching 100% accuracy. The modest improvement was still 6% on average and 10% in the best case on Leukemia. The results showed that the discriminating ability of the four constructed features was much higher than the original thousands of features.

For NB, the features constructed by CDFC also obtained better performance than Full on all datasets. For example, using the 4 features constructed on Prostate, NB achieved 32% higher accuracy on average than using the whole 10,509 features. Similarly, the improvement on Colon and DLBCL was 11% and 13% on average with 18% and 17% in the best case, respectively.

Compared with using Full, DT using CDFC constructed features also had significantly better performance on five datasets, and similar on the remaining ones. An improvement of at least 10% on average accuracy was achieved on DLBCL and CNS with the best accuracy improved by 14% and 18%, respectively.

In general, over the 18 comparisons with Full using the three learning algorithms on six datasets, CDFC won 17 and drew 1 in terms of the average accuracy. 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 CGPFC

As can be seen from Table 3, the CDFC constructed features obtained a significantly higher KNN accuracy than CGPFC on all datasets with the largest gap of 7.5% on DLBCL. A similar pattern was observed for NB with an improvement of up to 7.9% on all the datasets. For DT, three datasets had significantly higher accuracies and three had similar results. In general, compared with the single feature constructed by CGPFC, the four features constructed by CDFC helped the three learning algorithms obtain either a significantly better or similar classification performance on all datasets. The results also showed that CDFC's constructed features could improve the performance of all the three learning algorithms with more stable results shown by smaller standard deviations in most cases.

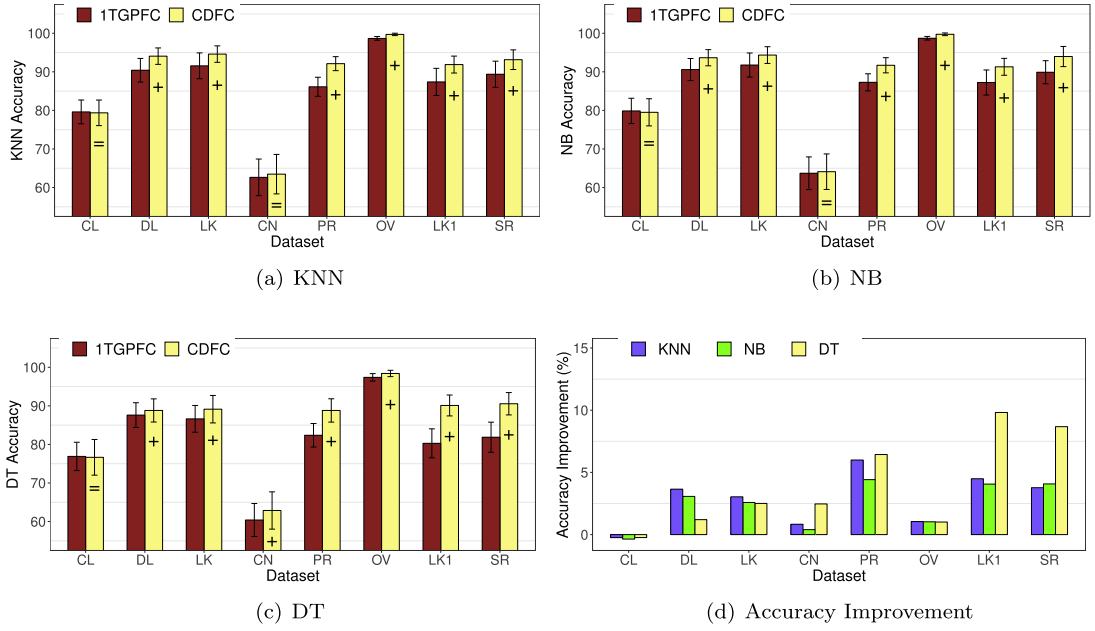
5.2. Multiple-feature construction: multi-tree GP versus single-tree GP

This section investigates the use of single-tree versus multi-tree representation for multiple-feature construction. CDFC is compared with 1TGPFC [15] where single-tree GP is used to construct multiple features. Please refer to Section 2.2.1 for details of 1TGPFC. In this set of experiments, CDFC was run with the construction ratio 1 to construct the same number of features as 1TGPFC. The same terminal set and GP parameter settings were used for both methods.

Fig. 4 shows the average results over the 50 runs of 1TGPFC and CDFC. The first three figures present the average test accuracy of KNN, NB, and DT of each method. The CDFC bars of these figures also display the results of the Wilcoxon significance test comparing CDFC against 1TGPFC. "+" and "−" mean that CDFC is significantly better or worse than CGPFC. "=" means that they are similar. The last figure shows the average accuracy improvement of each learning algorithm on each dataset. The results showed that using the CDFC constructed features, DT performed significantly better than using those constructed by 1TGPFC on all datasets except for Colon where both have a similar performance. Similarly, KNN and NB had better results on six datasets. DT obtained the largest increase of 9.8% average accuracy on Leukemia1 while KNN and NB had 6%

Table 3
Results of multiple-feature construction versus single feature construction.

Dataset	Method	#F	B-KNN	M ± Std-KNN	S_1	B-NB	M ± Std-NB	S_2	B-DT	M ± Std-DT	S_3
Colon (62)	Full	2000	74.29		–	72.62		–	74.29		–
	CGPFC	1	86.90	75.82 ± 3.53	–	85.48	75.84 ± 3.50	–	86.90	75.79 ± 3.60	–
	CDFC	4	88.81	81.87 ± 3.08		90.47	83.52 ± 3.11		87.38	78.03 ± 4.00	
DLBCL (77)	Full	5469	84.46		–	81.96		–	80.89		–
	CGPFC	1	97.50	88.49 ± 3.70	–	97.50	88.67 ± 3.35	–	97.50	88.41 ± 3.77	–
	CDFC	4	98.75	96.03 ± 1.96		98.75	95.25 ± 2.19		95.00	90.76 ± 3.01	
Leukemia (72)	Full	7129	88.57		–	91.96		–	91.61		=
	CGPFC	1	94.64	90.03 ± 3.08	–	94.64	89.90 ± 3.18	–	94.64	90.06 ± 3.07	=
	CDFC	4	98.57	94.83 ± 1.71		97.32	94.07 ± 1.60		94.82	90.72 ± 2.47	
CNS (60)	Full	7129	56.67		–	58.33		–	50.00		–
	CGPFC	1	70.00	61.73 ± 4.01	–	70.00	61.53 ± 4.34	–	70.00	61.70 ± 4.01	=
	CDFC	4	73.33	65.10 ± 4.20		73.33	66.17 ± 3.75		68.34	61.03 ± 4.87	
Prostate (102)	Full	10,509	81.55		–	60.55		–	86.18		–
	CGPFC	1	92.27	85.54 ± 2.77	–	92.27	85.64 ± 2.76	–	92.27	85.58 ± 2.80	–
	CDFC	4	95.18	92.81 ± 1.59		96.09	92.82 ± 1.50		94.09	88.04 ± 2.52	
Ovarian (253)	Full	15,154	91.28		–	90.05		–	98.41		–
	CGPFC	1	100.00	98.62 ± 0.59	–	99.62	98.50 ± 0.60	–	100.00	98.64 ± 0.59	=
	CDFC	4	100.00	99.73 ± 0.31		100.00	99.55 ± 0.34		100.00	98.78 ± 0.69	



(a) KNN (b) NB (c) DT (d) Accuracy Improvement

Fig. 4. Results of multi-tree GP versus single-tree GP for multiple-feature construction.

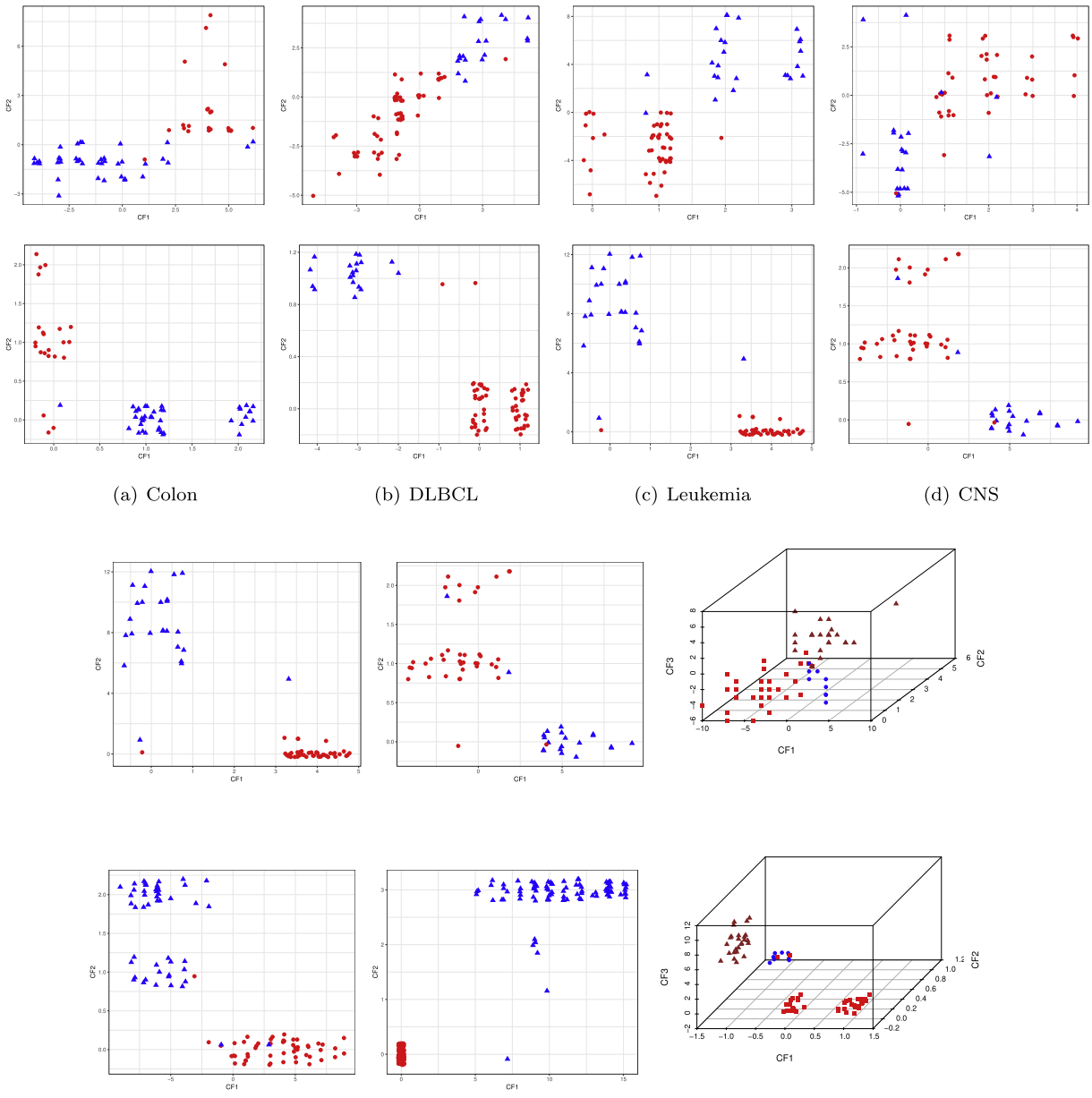
and 4.4% largest increase on Prostate, respectively. Comparisons between the three learning algorithms in Fig. 4(d) showed that DT had the largest improvement on four datasets and KNN on three.

Although 1TGPFC used an impurity measure in its fitness function to evaluate the constructed class-dependent features, 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 enables 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. To confirm this hypothesis, the following section will visualise the data transformed based on the features constructed by both methods.

Transformed Data by CDFC and 1TGPFC: Since GP is a stochastic algorithm, 50 independent runs with different random seeds were conducted for each method on each training set, resulting in 50 final constructed feature sets. Due to the page limit, we can only visualise one set of constructed features among the 50 final sets returned by GP. Because the difference between the best constructed feature sets of two methods is not large enough to be visually distinguished, we chose the worst set from each method running on the first pair of training and test fold for comparisons. The chosen constructed feature set from each method is then used to transform the datasets. The transformed data of seven datasets (excluding SRBCT with four constructed features) based on both methods are plotted in Fig. 5.

As can be seen from Fig. 5, the constructed features by both methods were quite good in grouping instances of different classes into separate clouds. However, the data clouds produced by CDFC in the second row of each figure were more compact and separated from each other than those created by 1TGPFC in the first row. 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.



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

(e) Prostate (f) Ovarian (g) Leukemia1

Fig. 5. Data transformed by 1TGPFC (the first row) and CDFC (the second row).

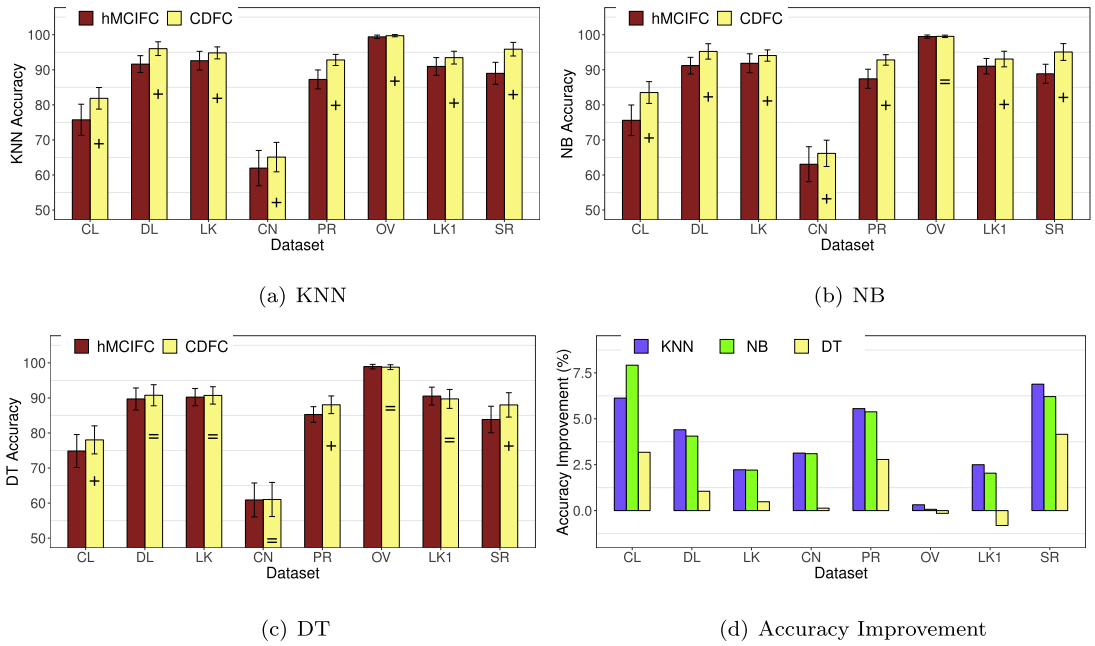
While both methods used an entropy-based measure to minimise the impurity of the constructed feature values within each class, CDFC incorporated an additional distance measure to evaluate the whole set of constructed features. The aim of the distance measure is to maximise the distance between instances of different classes and minimise the distance between instances of the same class. It is this measure that helps CDFC produce more compact clouds that are scattered far away from each other.

5.3. Multiple-feature construction: filter versus hybrid method

In order to see which evaluation method has a better effect on GP performance, both CDFC and MCIFC should construct features based on the same set of features. Therefore, instead of using the whole feature set as in [16], in this set of experiments, MCIFC uses the combined set of all the class-dependent features generated by CDFC. However, MCIFC still uses the hybrid evaluation method described in Eq. (2). Therefore, we called this variant as hMCIFC to distinguish it from MCIFC proposed in [16]. Since the evaluation method highly affects the running time of GP-based methods, the two methods are compared not only in classification accuracy but also in computation time.

5.3.1. Classification accuracy

Fig. 6 shows the average results over the 50 runs of hMCIFC and CDFC with the construction ratio of 2. As shown in Fig. 6, although both methods constructed the same number of features, KNN using features constructed by CDFC achieved significantly better performance than using those of hMCIFC on all datasets. An improvement of more than 6% on average was found on Colon and SRBCT. 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 CDFC constructed



(a) KNN (b) NB

(c) DT (d) Accuracy Improvement

Fig. 6. Results of filter versus hybrid of filter and wrapper feature construction.

features, NB achieved significantly better results on 7 datasets with up to 7.9% higher accuracy than using hMCIFC ones. For DT, features constructed by CDFC obtained significantly better performance than those of hMCIFC on three datasets, namely Colon, Prostate and SRBCT, and similar on the remaining five. In general, compared with hMCIFC using the three learning algorithms, CDFC won 18 and drew 6 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 in hMCIFC or information gain in CDFC. 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 algorithms. This may relate to the characteristics of DT where classification decision highly depends on the split values in the internal nodes of the 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 [10].

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 the 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 of the evolutionary process of hMCIFC showed that in a set of constructed features that gave very good DT accuracy, there might exist bad or even constant-value features. The reason is that DT does not use all features in building the DT tree. Therefore, the returned accuracy does not reflect the goodness of all constructed features. In contrast, the average IG measures the 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. The smaller standard deviation in CDFC results also indicated that the performance of CDFC was more stable than hMCIFC.

5.3.2. Computation time

Fig. 7 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 50% 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. 8 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 8 datasets, which would require a longer processing time in CDFC. However, as shown in Fig. 7, 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, the computation time of the average IG of each constructed feature is still much shorter, leading to much faster running time in CDFC. This again confirmed the efficiency of filter measures.

5.4. Multiple-feature construction: class-dependent versus class-independent

Since CDFC differs from hMCIFC not only in the class-dependent FC 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 class-dependent FC strategy should be confirmed by another set of experiments where CDFC is compared with *fMCIFC*, which is the same method as hMCIFC except that both use the same fitness function as shown in Eq. (9).

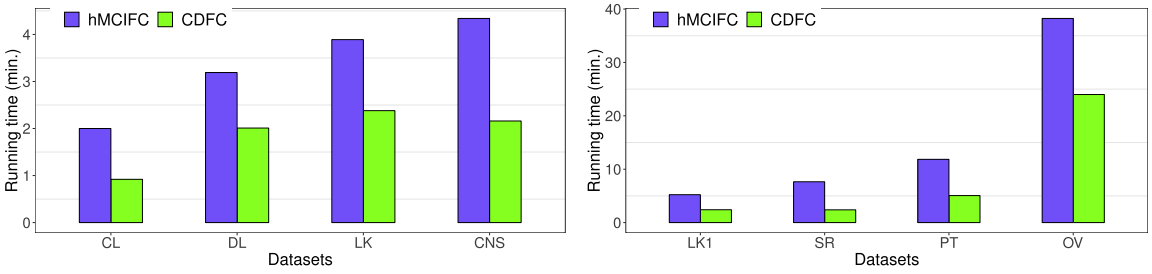


Fig. 7. Computation time of CDFC versus hMCIFC.

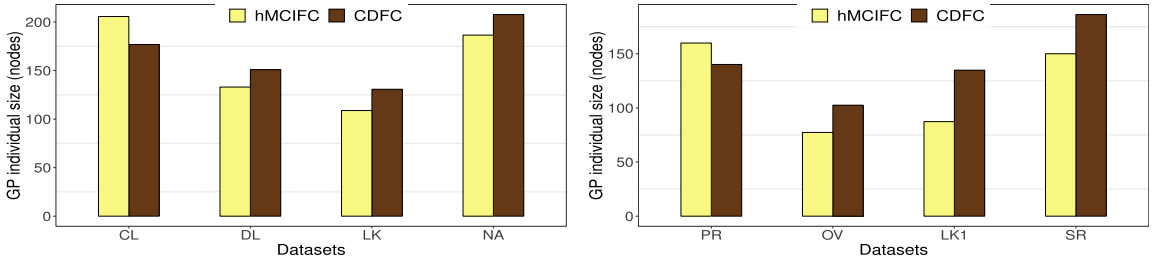


Fig. 8. The average size of GP individuals of CDFC versus hMCIFC.

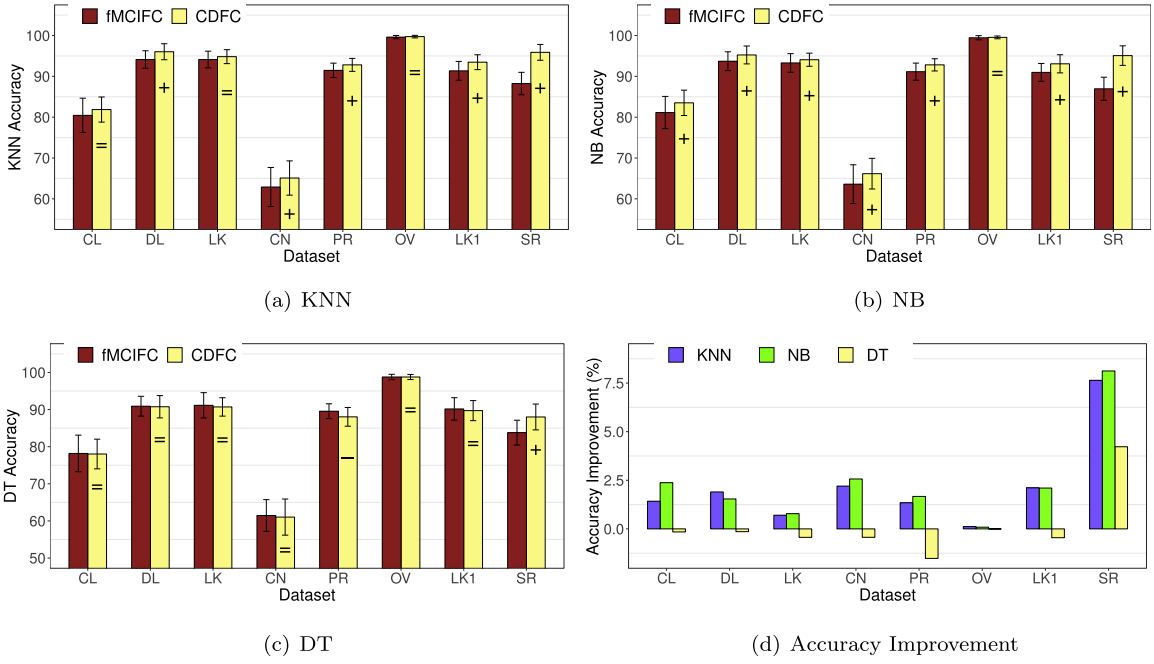


Fig. 9. Results of class-dependent versus class-independent feature construction.

Fig. 9 shows the average results over the 50 runs of fMCIFC and CDFC. As can be seen in Fig. 9(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. 9(b) with a significant improvement made on seven out of the eight datasets. SRBCT was also the dataset 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. 9(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 obtain either a significantly better or a similar classification performance to fMCIFC on all datasets. This indicated that by evaluating each constructed feature against the corresponding class only, CDFC could construct features that have more discriminating power.

5.4.1. Constructed features. An investigation of the constructed features was conducted to further explain the effectiveness of CDFC. Leukemia1 was chosen to demonstrate in this section because its best GP individual had a smaller size than the other datasets.

Fig. 10 shows a relatively small GP individual evolved by CDFC on the Leukemia1 dataset. This individual has three trees representing three features constructed from 11 features. Using this set of three constructed features, all the three learning algorithms obtained 100% test accuracy.

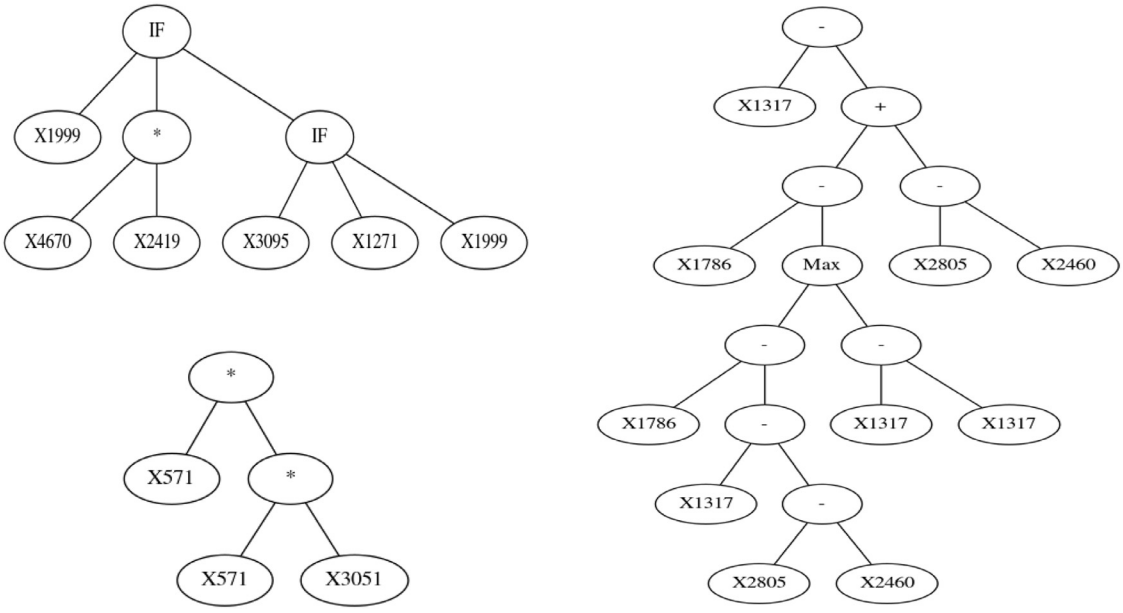


Fig. 10. Three constructed features on Leukemia1 CF1 (upper left), CF2 (lower left) and CF3 (right) corresponding to the three classes, respectively.

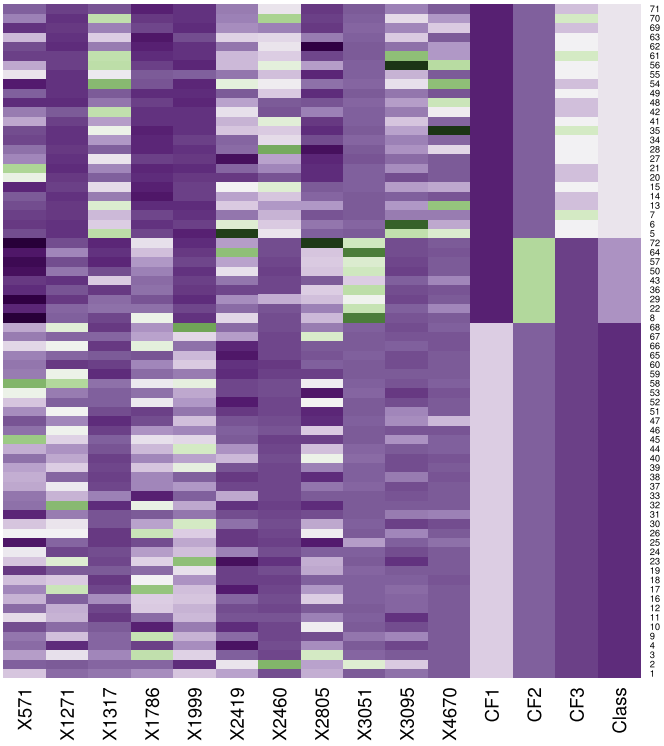


Fig. 11. Heatmap showing the data of the 11 selected and 3 constructed features by the individual shown in Fig. 10 on Leukemia1.

Fig. 11 shows the data of the 11 selected features and 3 constructed features on Leukemia1 using this individual. Note that there is no relationship in terms of values between colours used in different columns. We are only interested in the difference between colours in the same column. All the instances (or rows in this map) are sorted based on the class label shown in the last column of Fig. 11. Class 1 (in dark purple) is the largest class with 25 instances located at the bottom of the heatmap. Class 2 (in light purple) and class 3 (in the lightest purple) are lying on top of class 1.

As can be seen from Fig. 11 that feature X1999 has two major ranges of values shown in a darker and a lighter colour corresponding to instances of class 3 and the other two classes, respectively. This confirms its importance and explains why CDFC chose it as the first feature in the tree CF1. This also shows that X1999 can only distinguish instances of class 3 to the other classes, but cannot distinguish instances between class 1 and 2. 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 than the original ones as shown in the three columns CF1, CF2 and CF3 of Fig. 11. The constructed feature CF1 has only two values, one for class 1 and one for the other two 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.

5.5. Non-GP versus GP-based feature construction

This section compares CDFC with some popular non-GP approaches to FC including the principal component analysis method (PCA), linear discriminant analysis (LDA), and autoencoder (AE). While PCA transforms features without considering class labels, LDA returns $c-1$ features that maximise the difference between the c classes. The four constructed features by CDFC are compared with the first four PCA components and those returned by LDA. As a typical NN based feature learning method, AE is also applied to learn 4 features. AE with 4 layers for each encoder/decoder was found to give the best test accuracy on Colon among different numbers of layers ranging from 2 to 10 for each encoder/decoder. The results of KNN, NB, and DT on the constructed features by these FC methods are also compared with well-known classification methods that have a built-in FC process including support vector machine (SVM) and deep NNs (DNN).

Popular DNN approaches are convolutional NNs (CNNs), recurrent NNs (RNNs), and multi-layer feedforward NNs. While CNNs are good for data with structures such as images and videos, RNNs are good for sequence data where the order of information is important such as time series and text. Since the data addressed in this study is not suitable for CNNs and RNNs, multi-layer feedforward NNs is chosen for comparisons. DNNs with 10 to

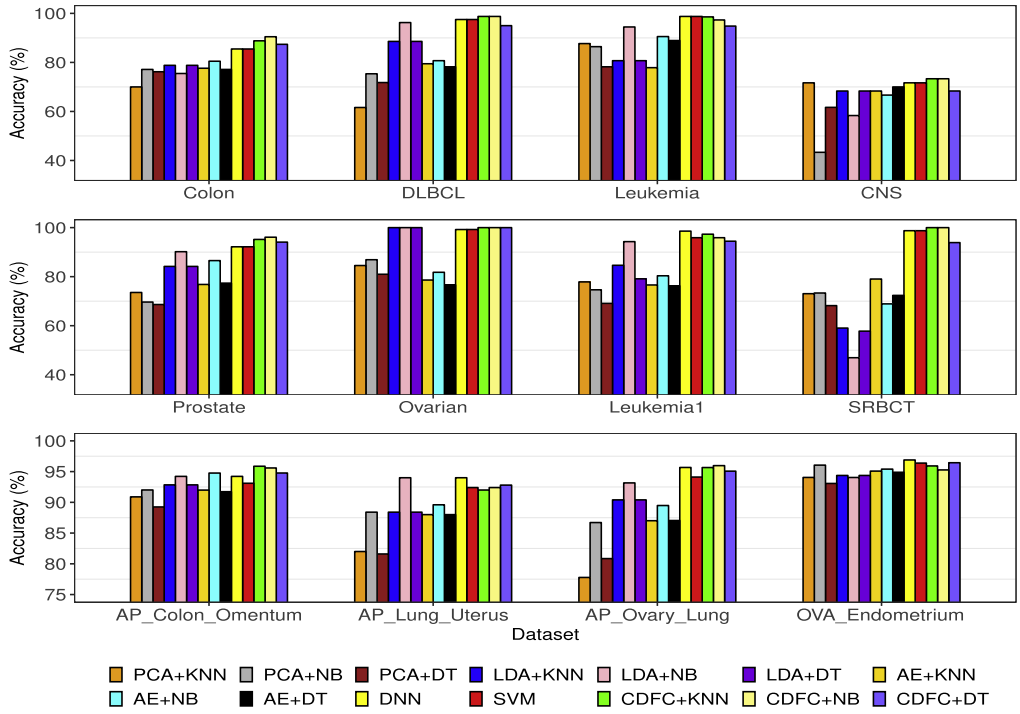


Fig. 12. Results of PCA, LDA, SVM, DNN versus CDFC.

50 layers are tried on the smallest dataset (Colon) to choose the best number of layers that gives the highest test accuracy. Ten layers was found to give the best result of 85.48%, which is better than the result of a recently proposed CNN-based method called SE1DCNN in [39] (84.9%) on the same dataset. With two convolutional layers, two max pooling layers and one fully-connected layer, SE1DCNN outperformed the six DNN-based methods compared in [39]. A DDN with 10 layers, therefore, is applied to the remaining 11 datasets. The stochastic gradient descent algorithm and the cross entropy loss function are chosen for training with 500 epochs, and other parameters are default in the Keras package (https://keras.io).

To better investigate the performance of CDFC, this experiment added four datasets originated from Stiglic and Kokol [40], which have been made available on https://www.openml.org since 2014. The four binary-class datasets are AP_Colon_Omentum, AP_Lung_Uterus, AP_Ovary_Lung, and OVA_Endometrium, all of which have 10,937 features and 363, 250, 324 and 1545 instances, respectively.

Fig. 12 plots the best results of KNN, NB, and DT using features constructed by CDFC (the last three columns) versus the results of PCA, LDA, and AE (the first nine columns). The 10th and 11th columns are the results of DNN and SVM. The results showed that CDFC led to better results than PCA, LDA, and AE on all datasets except for Ovarian where both LDA and CDFC obtained the highest accuracy of 100%, and AP_Lung_Uterus where CDFC reached 92.8% and LDA reached 94%. Note that the three learning algorithms produced very different results with the biggest difference of 28% on CNS and 15% on Leukemia1 when using the constructed features by PCA and LDA, respectively. In contrast, the difference in the results of these learning algorithms was much smaller when using CDFC constructed features. The largest difference was 6% on SRBCT. This indicates that the constructed features by CDFC are well generalised to more classification algorithms.

As can be seen from the 9th and 10th columns of Fig. 12, DNN obtained slightly higher accuracy than SVM on five datasets and similar on the remaining seven. Compared to the best results of these two algorithms, the best results obtained by the three learning algorithms when using the constructed features by CDFC were higher on eight out of 12 datasets, and lower on the remaining four, namely Leukemia, Leukemia1, AP_Lung_Uterus, and OVA_Edometrium.

In summary, this section has compared and analysed the results of the constructed features by five GP-based and three non-GP based FC methods as well as two learning algorithms that have a built-in FC process. The constructed features are also visualised to explain the effectiveness of the CDFC algorithm.

6. Conclusions

The goal of this study was to investigate the performance of different approaches to multiple-feature construction on high-dimensional data using GP. Different methods were compared in the same context to reveal the important factors. Multiple-feature construction has shown to be more effective than single-feature construction. Using multi-tree GP representation achieved better results than single-tree GP thanks to the ability to consider the interaction of the newly constructed features during the construction process. Class-dependent constructed features provided better discriminating ability than class-independent constructed features. Compared with PCA, LDA, and autoencoder, CDFC constructed features had a better discriminating ability and worked equally well for more learning algorithms in most cases. Using CDFC constructed features, KNN, NB, and DT also obtained better accuracy than DNN and SVM in many cases. This confirms the previously mentioned assumption about the limitation of DNN on datasets with small sample size. The visualisation of CDFC constructed features also demonstrated another advantage of the GP-

based method over the non-GP FC approaches. This interpretability enables experts to have a better insight of the learnt model, which is important to many real-world applications.

Evolving a fixed and predefined number of constructed features may limit CDFC in creating an optimal set of constructed features for a particular problem. More dynamic representations that allow CDFC to increase or decrease the number of constructed features during the evolutionary learning process may help obtain better performance. Currently, CDFC can only work with numeric data. Our future work also includes an extension to enable CDFC work on other types of data such as categorical, image or text data.

Acknowledgment

This work was supported in part by the Marsden Fund of New Zealand Government under Contracts VUW1509 and VUW1615, Huawei Industry Fund E2880/3663, and the University Research Fund at Victoria University of Wellington 209862/3580, and 213150/3662.

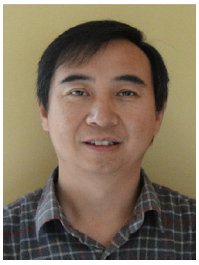
References

[1] P. Zhou, X. Hu, P. Li, X. Wu, OFS-Density: a novel online streaming feature selection method, Pattern Recognit. 86 (2019) 48–61.
[2] W. Gao, L. Hu, P. Zhang, Class-specific mutual information variation for feature selection, Pattern Recognit. 79 (2018) 328–339.
[3] T. Afouras, J.S. Chung, A. Senior, O. Vinyals, A. Zisserman, Deep audio-visual speech recognition, IEEE Trans. Pattern Anal. Mach. Intell. (2018), doi:10.1109/TPAMI.2018.2889052.
[4] M.S. Aslan, Z. Hailat, T.K. Alafif, X.W. Chen, Multi-channel multi-model feature learning for face recognition, Pattern Recognit. Lett. 85 (2017) 79–83.
[5] P. Loncomilla, J.R. del Solar, L. Martinez, Object recognition using local invariant features for robotic applications: a survey, Pattern Recognit. 60 (2016) 499–514.
[6] R. Rastghalam, H. Pourghassem, Breast cancer detection using MRF-based probable texture feature and decision-level fusion-based classification using HMM on thermography images, Pattern Recognit. 51 (2016) 176–186.
[7] D. Cheng, Y. Gong, X. Chang, W. Shi, A. Hauptmann, N. Zheng, Deep feature learning via structured graph Laplacian embedding for person re-identification, Pattern Recognit. 82 (2018) 94–104.
[8] O. Gupta, D. Raviv, R. Raskar, Illumination invariants in deep video expression recognition, Pattern Recognit. 76 (2018) 25–35.
[9] Y.N. Dauphin, A. Fan, M. Auli, D. Grangier, Language modeling with gated convolutional networks, in: Proceedings of the International Conference on Machine Learning (ICML), 2017, pp. 933–941.
[10] B. Tran, B. Xue, M. Zhang, Genetic programming for feature construction and selection in classification on high-dimensional data, Memetic Comput. 8 (1) (2015) 3–15.
[11] S. Ahmed, M. Zhang, L. Peng, A New GP-Based wrapper feature construction approach to classification and Biomarker identification, in: Proceedings of the IEEE Congress on Evolutionary Computation, 2014, pp. 2756–2763.
[12] L. Guo, D. Rivero, J. Dorado, C.R. Munteanu, A. Pazos, Automatic feature extraction using genetic programming: an application to epileptic eeg classification, Expert Syst. Appl. 38 (8) (2011) 10425–10436.
[13] 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.
[14] M. Smith, L. Bull, Genetic programming with a genetic algorithm for feature construction and selection, Genet. Program. Evol. Mach. 6 (2005) 265–281.
[15] K. Neshatian, M. Zhang, P. Andreae, A filter approach to multiple feature construction for symbolic learning classifiers using genetic programming, IEEE Trans. Evol. Comput. 16 (5) (2012) 645–661.
[16] B. Tran, M. Zhang, B. Xue, Multiple Feature Construction in Classification on High-dimensional Data Using Gp, in: Proceedings of the IEEE Symposium Series on Computational Intelligence (SSCI), 2016, pp. 1–8.
[17] G. Chandrashekar, F. Sahin, A survey on feature selection methods, Comput. Electrical Eng. 40 (2014) 16–28.
[18] B. Tran, B. Xue, M. Zhang, Class dependent multiple feature construction using genetic programming for high-dimensional data, in: Proceedings of the AI Advances in Artificial Intelligence, Vol. 10400 of Lecture Notes in Computer Science, Springer International Publishing, 2017, pp. 182–194.
[19] 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.
[20] 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.
[21] 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.
[22] F. Otero, M. Silva, A. Freitas, J. Nievola, Genetic programming for attribute construction in data mining, in: Genetic Programming, 2610, Springer Berlin Heidelberg, 2003, pp. 384–393.
[23] M. Muharram, G. Smith, Evolutionary constructive induction, IEEE Trans. Knowl. Data Eng. 17 (2005) 1518–1528.
[24] K. Neshatian, M. Zhang, M. Johnston, Feature construction and dimension reduction using genetic programming, in: M.A. Orgun, J. Thornton (Eds.), Proceedings of the AI Advances in Artificial Intelligence, Springer Berlin Heidelberg, Berlin, Heidelberg, 2007, pp. 160–170.
[25] K. Neshatian, M. Zhang, Genetic programming for performance improvement and dimensionality reduction of classification problems, in: Proceedings of the IEEE Congress on Computational Intelligence, 2008, pp. 2811–2818.
[26] 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.
[27] H.B. Nguyen, B. Xue, P. Andreae, A Hybrid Ga-gp method for feature reduction in classification, in: Proceedings of the 11th International Conference Simulated Evolution and Learning, SEAL, Springer International Publishing, Shenzhen, China, 2017, pp. 591–604.
[28] K. Krawiec, Genetic programming-based construction of features for machine learning and knowledge discovery tasks, Genet. Program. Evol. Mach. 3 (2002) 329–343.
[29] 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.
[30] 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.
[31] 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.
[32] H. Al-Sahaf, A. Al-Sahaf, B. Xue, M. Johnston, M. Zhang, Automatically evolving rotation-invariant texture image descriptors by genetic programming, IEEE Trans. Evol. Comput. 21 (1) (2017) 83–101.
[33] S.H. Cha, Comprehensive survey on distance/similarity measures between probability density functions, Int. J. Math. Models Methods Appl. Sci. 1 (2007) 300.
[34] F. Han, C. Yang, Y. Wu, J. Zhu, Q. Ling, Y. Song, D. Huang, A gene selection method for microarray data based on binary PSO encoding gene-to-class sensitivity information, IEEE/ACM Trans. Comput. Biol. Bioinf. 14 (1) (2017) 85–96.
[35] A.K. Shukla, P. Singh, M. Vardhan, A two-stage gene selection method for Biomarker discovery from microarray data for cancer classification, Chemomet. Intell. Laboratory Syst. 183 (2018) 47–58.
[36] Z. Zhu, Y.S. Ong, M. Dash, Markov blanket-embedded genetic algorithm for gene selection, Pattern Recognit. 40 (11) (2007) 3236–3248.
[37] 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.
[38] F. Wilcoxon, Individual comparisons by ranking methods, Biometr. Bull. 1 (6) (1945) 80–83.
[39] J. Liu, X. Wang, Y. Cheng, L. Zhang, Tumor gene expression data classification via sample expansion-based deep learning, Oncotarget 8 (65) (2017) 109646–109660.
[40] G. Stiglic, P. Kokol, Stability of Ranked Gene Lists in Large Microarray Analysis Studies, BioMed Research International, 2010.

Binh Tran (S'14) received her B.E. in Computer Science from Cantho University, Vietnam, in 1998, the M.Sc. degree in Applied Computer Science from Free University of Brussels, Belgium, in 2002, the Ph.D degree in computer science in 2018 at Victoria University of Wellington, New Zealand. She is currently a Post Doctoral Research Fellow in the School of Engineering and Computer Science at Victoria University ofWellington. Her research interests are in evolutionary computation, feature manipulation including feature selection and construction, high dimensional data, and machine learning.
Ms. Tran is a member of the IEEE Computational Intelligence Society (CIS). She has been serving as a reviewer for over 10 international journals and conferences in the field.

Bing Xue (M'10) received the B.Sc. degree from the Henan University of Economics and Law, Zhengzhou, China, in 2007, the M.Sc. degree in management from Shenzhen University, Shenzhen, China, in 2010, and the Ph.D. degree in computer science in 2014 at Victoria University of Wellington, New Zealand. She is currently a Senior Lecturer in School of Engineering and Computer Science at Victoria University of Wellington. Her research focuses mainly on evolutionary computation, feature selection, feature construction, multi-objective optimisation, image analysis, transfer learning, data mining, and machine learning. She has over 100 papers published in fully refereed international journals and conferences and most of them are on evolutionary feature selection and construction.

Dr Xue is currently the Chair of the IEEE Task Force on Evolutionary Feature Selection and Construction, IEEE Computational Intelligence Society (CIS), Vice-Chair of the IEEE CIS Data Mining and Big Data Analytics Technical Committee, and Vice-Chair of IEEE CIS Task Force on Transfer Learning and Transfer Optimisation. She is also an Associate Editor/member of Editorial Board for five international journals and a reviewer of over 50 international journals. Dr Xue is the Finance Chair of IEEE Congress on Evolutionary Computation (CEC) 2019, a Program Co-Chair of the 31th Australasian AI 2018, ACALCI 2018, and the 7th International Conference on SoCPaR2015, and she is also a tutorial chair, special session chair, or publicity chair for many other international conferences.



Mengjie Zhang (M'04-SM'10) received the B.E. and M.E. degrees from Artificial Intelligence Research Center, Agricultural University of Hebei, Hebei, China, and the Ph.D. degree in computer science from RMIT University, Melbourne, VIC, Australia, in 1989, 1992, and 2000, respectively. He is currently Professor of Computer Science, Head of the Evolutionary Computation Research Group, and the Associate Dean (Research and Innovation) in the Faculty of Engineering. His current research interests include evolutionary computation, particularly genetic programming, particle swarm optimization, and learning classifier systems with application areas of image analysis, multi-objective optimization, feature selection and reduction, job shop scheduling, and transfer learning. He has published over 350 research papers in refereed international journals and conferences.

Prof. Zhang is a Fellow of Royal Society of New Zealand and have been a Panel member of the Marsden Fund (New Zealand Government Funding). He is also a senior member of IEEE and a member of ACM. He is currently chairing the IEEE CIS Intelligent Systems and Applications Technical Committee, and the immediate Past Chair for the IEEE CIS Emergent Technologies Technical Committee and the Evolutionary Computation Technical Committee, and a member of the IEEE CIS Award Committee. He is a vice-chair of the IEEE CIS Task Force on Evolutionary Feature Selection and Construction, a vice-chair of the Task Force on Evolutionary Computer Vision and Image Processing, and the founding chair of the IEEE Computational Intelligence Chapter in New Zealand. He is also a committee member of the IEEE NZ Central Section.