Multiple Feature Construction in Classification on High-Dimensional Data Using GP

Binh Tran, Mengjie Zhang and Bing Xue
School of Engineering and Computer Science
Victoria University of Wellington, Wellington, New Zealand
Email: Binh.Tran,Mengjie.Zhang,Bing.Xue@ecs.vuw.ac.nz

***Abstract*—Feature construction and feature selection are common pre-processing techniques to obtain smaller but better discriminating feature sets than the original ones. These two techniques are essential in high-dimensional data with thousands or tens of thousands of features where there may exist many irrelevant and redundant features. Genetic programming (GP) is a powerful technique that has shown promising results in feature construction and feature selection. However, constructing multiple features for high-dimensional data is still challenging due to its large search space. In this paper, we propose a GP-based method that simultaneously performs multiple feature construction and feature selection to automatically transform high-dimensional datasets into much smaller ones. Experiment results on six datasets show that the size of the generated feature set is less than 4% of the original feature set size and it significantly improves the performance of K-Nearest Neighbour, Naive Bayes and Decision Tree algorithms on 15 out of 18 comparisons. Compared with the single feature construction method using GP, the proposed method has better performance on half cases and similar on the other half. Comparisons between the constructed features, the selected features and the combination of both constructed and selected features by the propose method reveal different preferences of the three learning algorithms on these feature sets.**

I. Introduction

Quality of the input data is a key factor influencing the performance of any machine learning method including classification [1]. Given a set of instances or examples described by a set of features and the class labels, a classification algorithm aims at learning a model that can correctly classify an unseen instance [2]. According to the *garbage in garbage out* principle, a good quality input data is a prerequisite for learning algorithms to achieve this goal. Therefore, preprocessing data is an important step in most machine learning applications especially on high-dimensional data with thousands or more features.

Feature selection (FS) and feature construction (FC) [3] are popular methods to enhance the discriminating ability of the feature set and at the same time keep the number of features as small as possible. While FS selects relevant features from the original feature set, FC selects informative features and combines them to construct new high-level features with better discriminating power. Therefore, FS can be considered a built in process in a FC method.

FS and FC have been proposed as wrapper, filter or embedded methods [4]. A FS or FC method is classified as wrapper or filter depending on whether it uses a classification algorithm to evaluate features or not. Wrappers usually achieve a high classification performance at the expense of long computation time. On the other hand, using the intrinsic characteristics of the training data to evaluate features, filters are usually faster than wrappers. However, the features selected or constructed by filter methods usually achieve lower classification accuracy than those generated by wrappers. Therefore, a hybrid approach that combines wrapper and filter is also proposed in FS methods [5], [6]. Different from the above approaches, embedded methods conduct FS or FC simultaneously within the process of learning a classifier. They are typically faster than wrappers.

Although FS and FC have been studied for decades, applying them to high-dimensional data is still challenging due to its large search space. With n features, FS methods search for good solutions from 2^n possible subsets. On top of searching for good features from this space, FC methods also need to choose appropriate operators and a good way to apply these operators on the selected features. Therefore, the search space of FC is even larger than FS. Because of this complexity, FC methods need a powerful search technique to construct better high-level features.

Genetic programming (GP) is an evolutionary computation technique that can automatically evolve solutions based on the idea of the survival of the fittest. With a population-based search and a flexible representation, GP can evolve any mathematical model without any assumption such as linear or non-linear. Therefore, GP has been used in feature construction methods for biomarker identification [7], for image classification [8], [9], etc. GP-based FC methods use either single-tree representation [7] or multiple-tree representation [10].

A common practice of using constructed features (CFs) is either forming a new feature set from the CFs [11] or adding them to the original feature set [12]. While both approaches are popularly used for datasets with tens of features, the latter is not usually applied to high-dimensional data. However, using only a small number of CFs may not be enough to represent the original large feature set. Combining CFs with some selected good features is expected to achieve better performance.

Recently, we proposed an embedded method using single-tree GP for both FS and FC in a single process [13]. For presentation convenience, in this paper we call it GPFC. From the best tree evolved by GPFC, we constructed one feature

from the entire tree (CF), multiple features from all possible subtrees ($subCFs$) and selected all the features in the terminal nodes. Six different combinations of the CF, $subCFs$ and the selected features were compared using four classification algorithms on datasets with thousands to tens of thousands of features. The results showed that among the six combinations produced by GPFC, the combined set of the constructed feature and selected features achieve the highest performance on most datasets. However, constructing only one CF may hinder GPFC from achieving better results. In this study, we proposed a multiple-feature construction method based on GP using multiple-tree representation called *MultGPFC*. Furthermore, using the embedded approach, GPFC may construct features that are too overfitting to the training data since the constructed feature is evaluated based on its performance as a classifier on the whole training data. This problem is even worse when the number of examples or instances given for training is small. Therefore, in MultGPFC, we propose to use both wrapper and filter measures to better evaluate the discriminating ability of the constructed features.

A. Goals

The main goal of this paper is to combine filter and wrapper measures in a multiple-tree GP-based method for feature construction and selection in classification. The created subsets include a set of CFs, a set of selected features and a combination of both constructed and selected features. The solutions evolved by the proposed method is expected to improve the classification performance of common learning algorithms including k-nearest neighbour (KNN), Naive Bayes (NB) and Decision Tree (DT) on high-dimensional classification problems. The proposed method (MultGPFC) will also be examined and compared with GPFC [13]. Specifically, we will investigate the following research objectives:

- Whether the constructed features by MultGPFC have higher discriminating power than the original feature set and the one constructed by GPFC.
- Whether the selected features by MultGPFC have a better classification performance than the original feature set and those selected by GPFC.
- Whether the combination of the MultGPFC constructed and selected features performs better than the original feature set and the one produced by GPFC.
- Whether the constructed, selected and the combination of both feature sets have the same effect on the performance of different classification algorithms.

II. Background and Related Work

A. Genetic Programming

As a population based algorithm, GP [14] maintains a pool of candidate solutions or individuals which are evaluated based on a predefined fitness function. Fittest solutions will be selected to create offspring solutions by applying genetic operators such as crossover or mutation. The process of evaluation-selection-evolution will be continued until a stopping criterion is met. It can be whether a predefined maximum number of generations is reached or if the optimal solution is found. When this criterion is met, GP will return the fittest individual found so far as the best solution.

When using GP for feature construction, a constructed feature is usually presented in a tree structure with various sizes or depths. The internal nodes of the tree are operators or functions with a different number of arguments. Leaf nodes or terminal nodes can be constant values or variables/features. A GP individual may contain a single or multiple trees to construct one or multiple features forming single-tree GP or multiple-tree GP methods, respectively. Since a GP tree does not take all variables/features to construct a new feature, there is an implicit selection process for relevant features in the construction process. A GP tree can also be seen as a program that applies functions on terminal values, and it can be executed to produce an output value. With this ability, a GP tree can be used as a classifier which returns a category or a class label for a given example or instance. Therefore, using GP for classification actually involves feature selection, feature construction and classification.

B. Genetic Programming for FC and FS

Although GP has been used to evolve classifiers in many different domains [15], we situate our work most closely with the prior work relating to GP approach to FC and/or FS. Broader and more comprehensive review can be seen in [15], [16].

GP has been proposed to construct multiple high-level features with different strategies. In [17], each GP individual comprises of predefined numbers of new features and hidden features. Hidden features are kept out of the evolutionary process to avoid loosing good constructed features. They were updated from features that had highest usage frequency in the decision tree learned in the fitness function. The results showed that constructed features improved the classification performance of DT on five out of six datasets. Cooperative coevolution GP is also proposed to construct m new features in [18] using m populations. Another approach is to run a single-tree GP FC method multiple times [19] to construct multiple features, each for one class. The constructed feature is evaluated based on the impurity (using Shannon entropy) of the intervals which are formed by applying class dispersion to the transformed datasets. The method was extended in [20] by adding class-wise orthogonal transformed features to GP terminal sets. Results of these methods showed that GP is a promising approach in feature construction. However, the datasets used in these studies are quite small with about tens of features.

For high-dimensional data, Ahmed et al. [7] proposed two GP-based wrapper FC methods, namely GPWFC1 and GPWFC2. Multiple features are constructed from the best GP tree. While GPWFC1 uses classification accuracy of random forest (RF) classifier as the fitness value, GPWFC2 uses entropy gain of RF and the p-value of an ANOVA test on the selected features. Results showed that GPWFC2 achieves better generalisation ability with a smaller number of features

than GPWFC1. However, the computational cost is quite high when using RF for fitness evaluation.

Based on the implicit FS happened during the feature construction process in GP, Neshatian et al. [21] proposed a GP-based filter feature selection method. A binary relevance measure was proposed to evaluate the relationship between the constructed feature value and the class label. Features in a GP tree that has its fitness better than a predefined threshold will be used to form a new subset. For each subset size, the best subset was archived over 50 independent runs. Then, a wrapper approach was used to select the best subset from these. Results on three datasets with tens of features showed that using the selected features improved the performance of common learning algorithms. However, the proposed method required a long running time. A GP-based embedded approach was also proposed for FS in [22]. GP was used to select features from a combination set of 50 top ranked features selected by Information Gain and ReliefF methods. Experiments on high-dimensional data showed it achieves better results than the based line methods. However, domain knowledge or substantial trials are needed to choose a good number of top ranked features. GP was also proposed as a FS method before evolving classifiers in [23], [24].

Instead of conducting either FC or FS, Smith et al. proposed a method for FC and FS in two separate stages [25]. First, GP is used to construct as many new features as the original feature set. Then, GA is used to select features from the augmented feature set which is a combination of the original and the constructed features. Experiments on datasets with tens of features showed that the proposed method improved the performance of DT, KNN and NB. However, on top of the high computational cost, the GP representation of this method is not suitable for problems with thousands of features. In a previous study [13], we proposed a GP-based embedded approach for both FS and FC in a single stage. Multiple features are constructed and selected from the best GP tree. Different combinations of feature sets are investigated. Results on high-dimensional data showed that the combination of the constructed feature and selected features gives the best performance among the compared combinations.

In summary, with a very flexible representation and a global search technique, GP has shown its high potential in FC and FS. However, most GP-based FC methods are applied to small datasets with tens of features. They may not scale well to high-dimensional data due to high computational cost or large memory demanding. Some other methods applied to high-dimensional data are single FC. In this study, we will propose one method that performs multiple FC.

III. The Proposed Method

The aim of this study is to propose a multiple feature construction and feature selection method that can form a much smaller number of features than the original set with thousands to tens of thousands of features. To achieve this goal, GP with multiple-tree representation is used to construct a number of new features which is proportional to the number of classes of the problem we hypothesise that the more classes the problem has, the more complex the solution space might be. This section will describe the proposed representation, the hybrid evaluation measure used to evaluate the constructed feature set, and the overall structure of the system.

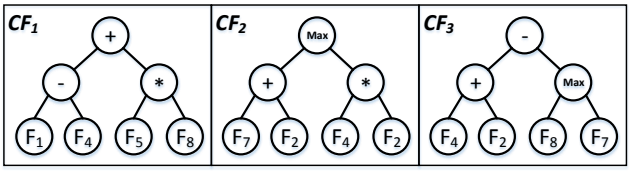

Fig. 1: MultGPFC Representation.

A. Representation

To construct m features, each individual in MultGPFC method has m trees. Fig. 1 shows an example of an individual which constructs three features:

- $CF_1 = (F_1 - F_4) + (F_5 * F_8)$
- $CF_2 = Max((F_7 + F_2), (F_4 * F_2))$
- $CF_3 = (F_4 + F_2) - Max(F_8, F_7)$

Based on these evolved features, we construct three new feature sets including the constructed and/or selected features. For example, the three new feature sets generated from the individual shown in Fig. 1 will be:

- The constructed feature set: $CF_1,\ CF_2,\ and\ CF_3$.
- The selected feature set: $F_1,\ F_2,\ F_4,\ F_5,\ F_7,\ and\ F_8$.
- The constructed and selected feature set: $CF_1,\ CF_2,\ CF_3,\ F_1,\ F_2,\ F_4,\ F_5,\ F_7,\ and\ F_8$.

B. A New Fitness Function

Selecting relevant features from thousands of features to build higher level features is not a trivial task especially when the number of training examples is much smaller than the dimensionality. Designing an objective or fitness function to guide the search in this scenario is very challenging. While a wrapper measure based on a learning algorithm can be a good indicator for searching good feature sets, the resulted feature set may not be general for other learning algorithms. On the other hand, a filter measure is based on the intrinsic characteristics of the data, its solutions may be effective for many learning algorithms, however, with the price of lower classification accuracy than wrapper approach. Therefore, we propose a hybrid approach that combines both wrapper and filter to synthesize their strengths. Decision Tree is used to evaluate the classification performance of the constructed feature set as it is a fast and efficient learning algorithm. For filter approach, distance measure is chosen because it is simple and multi-variate which means it can evaluate the discriminating ability of a set of features at a time.

To evaluate an individual, its constructed features are used to transform the training set into a new dataset with m features. The discriminating ability of the transformed training set will be used to determine the fitness of the individual. Equation (1) describes the fitness function which combines

the classification performance and a distance measure using a weighting coefficient α.

$$Fitness = \alpha \cdot Accuracy + (1 - \alpha) \cdot Distance \quad (1)$$

where $Accuracy$ is the average accuracy of a learning algorithm over K-fold (K=3) cross validation (CV) on the transformed training set. To avoid overfitting, this K-fold CV is repeated L times (L=3) with different data splitting similar to [17]. Totally, K x L models was built to evaluate the set of new constructed features. Because many high-dimensional datasets are unbalanced, we use the balanced accuracy [26] as shown in Equation (2) in which 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.

$$balanced\_accuracy = \frac{1}{c}\sum_{i=1}^{c}\frac{TP_i}{|S_i|} \quad (2)$$

The $Distance$ measure [8] is calculated based on Equation (3) which evaluates the discriminating power of the transformed training set. It is used to maximise the distance of instances *between* class (D_b) and minimise the distance of instances *within* the same class (D_w). D_b is appropriated by the average distance between an instance and its nearest miss which is the nearest instance of other classes. D_w is appropriated by the average distance between an instance and its farthest hit which is of the same class. Let S be the training set, D_b and D_w are calculated based on Equations (4) and (5).

$$distance = \frac{1}{1 + e^{-5(D_b - D_w)}} \quad (3)$$

$$D_b = \frac{1}{|S|}\sum_{i=1}^{|S|}\min_{\{j|j\neq i, class(V_i)\neq class(V_j)\}} Dis(V_i, V_j) \quad (4)$$

$$D_w = \frac{1}{|S|}\sum_{i=1}^{|S|}\max_{\{j|j\neq i, class(V_i)= class(V_j)\}} Dis(V_i, V_j) \quad (5)$$

where $Dis(V_i, V_j)$ can be any measure used to approximate the distance between two vectors V_i and V_j. In this method, we use Czekanowski distance [27] to evaluate the dissimilarity of two vectors because it is calculated based on the shared portion between two vectors as shown in Equation (6). Therefore, its value is bounded in the interval [0,1]. As a result, D_b and D_w values also fall in [0,1] interval and their difference ($D_b - D_w$) will fall in [-1,1]. To use this difference as the second component in the fitness function, we use a logistic function with coefficient -5 as shown in the right plot of Fig. 2 to transform the difference into a value of [0,1] interval. It is notice that the Czekanowski distance can only be used with non-negative values. Therefore, the constructed features are normalised before applying the formula.

$$Czekanowski(V_i, V_j) = 1 - \frac{2\sum_{d=1}^{n}\min(V_{id}, V_{jd})}{\sum_{d=1}^{n}(V_{id} + V_{jd})} \quad (6)$$

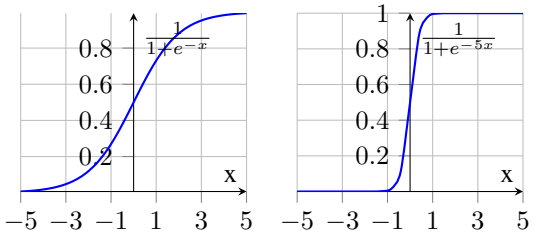

Fig. 2: Logistic function $f(x) = \frac{1}{1+e^{-x}}$, $f(x) = \frac{1}{1+e^{-5x}}$

C. The Overall Algorithm

Algorithm 1: The pseudo code of MultGPFC

```
   input : Training set, CF_Ratio
   output: The best constructed and selected features
1  begin
2      m ← CF_Ratio × Nbr_Classes ;
3      Randomly initialise individuals, each has m trees;
4      while Maximum generations is not reach do
5          for i = 1 to Population Size do
6              transf_train ← Transform training set based on the
                 constructed features in individual i;
7              Apply learning algorithm on transf_train to get
                 average accuracy;
8              Calculate distance on the normalised transf_train data
                 using Eq. (3);
9              Calculate fitness of individual i using Eq. (1);
10         end
11         Select parent individuals using tournament;
12         Create offspring individuals by applying crossover or mutation
             on the selected parents;
13         Place new individuals into the population of the next
             generation;
14     end
15     Return the constructed and selected features in the best individual;
16 end
```

The overall algorithm as shown in Algorithm 1 returns a set of constructed features and selected features for a given training set and a constructed feature ratio. GP starts by randomly initialising individuals having m trees using the defined function set and terminal set. Each individual corresponds to a new candidate solution which is m constructed features. These features are used to transform the training set where accuracy and distance will determine the fitness of the corresponding individual (lines 6 to 9). After evaluation, a normal selection and evolutionary process is conducted. This is repeated until the maximum number of generations is reach. Then the set of constructed and selected features in the best individual will be returned as the final solution.

IV. Experiment Design

This section describes the datasets used to test the performance of MultGPFC as well as the parameter settings used in the experiments.

A. Datasets

In the experiments, six high-dimensional datasets with thousands to tens of thousands of features are used. These datasets are gene expression data and publicly available at *http://www.gems-system.org*, and

TABLE I: Datasets

Dataset	#Features	#Ins.	#Class	Class-Distribution
Colon	2,000	62	2	35% - 65%
DLBCL	5,469	77	2	25% - 75%
Leukemia	7,129	72	2	35% - 65%
CNS	7,129	60	2	35% - 65%
Prostate	10,509	102	2	50% - 50%
Ovarian	15,154	253	2	36% - 64%

http://csse.szu.edu.cn/staff/zhuzx/Datasets.html. Details about these datasets are shown in Table I. It can be seen that these datasets have a small number of instances compared to their number of features. The last column shows the class distribution of the data which is the percentage of instances in each class. The big difference between these percentages shows that these datasets are unbalanced data.

Since gene expression data are generated in laboratory with substantial noise, discretisation is applied to reduce noise as suggestion in [28]. Each feature is first standardised to have zero mean and unit variance. Then its values are discretised into -1, 0 and 1 representing three states which are the under-expression, the baseline and the over-expression of gene. Values that fall in the interval $[\mu - \sigma/2, \mu + \sigma/2]$ are transformed to state 0. Values that are in the left or in the right of this interval will be transformed to state -1 or 1, respectively.

B. Experiment Configuration and Parameter Settings

To test the effectiveness of MultGPFC, we compared the classification accuracy of the constructed and/or selected features with those produced by GPFC as well as the original feature set. Since the number of instances in each dataset is small, all comparisons are made upon the average test accuracy of 10-fold CV [29] in which one fold is kept as the test set and the rest is used for training. During the FC process, 3-fold CV within the training set is used to evaluate the constructed features (see Section III-B). As GP is a stochastic algorithm, 30 independent GP runs with 30 different random seeds were conducted for each dataset to eliminate statistical variations.

TABLE II: GP Settings

Function set	$+$, $-$, $\times$, max, if
Terminal set	Features values
Population Size	#feature x β
Generations	50
Initial Population	Ramped Half-and Half
Maximum Tree Depth	8
Selection Method	Tournament Method
Tournament Size	7
Crossover Rate	0.8
Mutation Rate	0.2
Elitism Size	1
CF_Ratio	2
Fitness weighting μ	0.8

Table II describes the parameter settings of GP. Basically, MultGPFC uses the same settings as GPFC except that MultGPFC has a smaller function set, terminal set and maximum tree depth. The main purpose of these changes is to reduce the complexity of the search space. In MultGPFC, the function set comprises of 3 arithmetic operators ($+$, $-$, $\times$), max function which returns the maximum values from the two inputs and 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 terminal set for simplicity. The tree maximum depth is set to 8. The remaining parameter settings are the same as in [13]. The population size is set proportional to the dimensionality of the problem using a coefficient β which is set 3 for Colon dataset and to 2 for others due to memory limitation.

V. Results and Analysis

Results shown in this section are the average test results of the 30 independent GP runs on each dataset. To maintain a consistent presentation, all the result tables have the same format. Column "#F" shows the average size of each feature set. The following columns display the best (B), average accuracy and standard deviation (A$\pm$Std) obtained by KNN, NB and DT using the full feature set (Full), the feature set produced by GPFC and MultGPFC. The Wilcoxon significant test results (with significant level of 0.05) of the corresponding method over MultGPFC are also displayed in the average accuracy column. "+" or "-" means the result is significantly better or worse than MultGPFC and "=" means their results are similar. In other worse, the more "–", the better the proposed method.

The name of each dataset is displayed along with the number of instances (in the parenthesis) and the running time (in minutes) used to complete one GP run on a training set for feature construction. It can be seen from Table III that although the proposed method uses a hybrid approach of wrapper and filter, the running time on these problems is reasonable small, ranging from 2 minutes for the smallest dataset to 44 minutes for the largest. One of the main reasons is that the transformed dataset is small with a small number of constructed features and a small number of instances. Furthermore, a simple learning algorithm and a distance measure are used in the evaluation method.

In the remaining of this section, we will examine the performance of MultGPFC by comparing it with Full and with GPFC based on the three resulted feature sets: the constructed features, the selected features, and the combination of both constructed and selected features. Then we compare the effect of the three feature sets produced by MultGPFC on the three learning algorithms.

A. The Constructed Features

Table III shows the results of the MultGPFC constructed features versus Full and those constructed by GPFC. It can be seen that the number of features constructed by MultGPFC is obviously negligible compared with the original number of features. However, when using the constructed features, KNN obtains a significantly better accuracy with 2% to 8% higher than using Full on four datasets. Its performance on the other two datasets, namely Colon and CNS, are similar on average. However, the best accuracy obtained by KNN on the transformed Colon and CNS datasets are still 9% and 15% higher than the Full, respectively.

TABLE III: Results of constructed features

Dataset	Subset	#F	B-KNN	A±Std-KNN	B-NB	A±Std-NB	B-DT	A±Std-DT
Colon (62)	Full	2000	74.28	**74.28 ±0.00** =	72.62	72.62 ±0.00 =	74.29	**74.29 ±0.00** +
2.3(min.)	GPFC	1	79.28	71.40 ±4.46 =	78.81	69.64 ±4.17 =	79.28	72.25 ±4.07 =
	MultGPFC	4	85.47	72.48 ±5.95	85.48	**71.10 ±6.32**	85.48	71.42 ±5.82
DLBCL (77)	Full	5469	84.46	84.46 ±0.00 –	81.96	81.96 ±0.00 –	80.89	80.89 ±0.00 –
3.7(min.)	GPFC	1	96.07	86.65 ±3.76 –	92.32	86.27 ±4.28 –	94.64	86.51 ±4.08 =
	MultGPFC	4	97.32	**89.50 ±3.23**	97.32	**89.01 ±3.55**	94.82	**87.47 ±4.34**
Leukemia (72)	Full	7129	88.57	88.57 ±0.00 –	91.96	91.96 ±0.00 =	91.61	**91.61 ±0.00** =
4.5(min.)	GPFC	1	94.46	89.03 ±2.71 –	93.21	87.26 ±4.44 –	95.89	88.97 ±2.96 –
	MultGPFC	4	95.89	**92.71 ±1.89**	95.89	**92.45 ±1.88**	95.89	90.99 ±2.76
CNS (60)	Full	7129	56.67	56.67 ±0.00 =	58.33	58.33 ±0.00 =	50.00	50.00 ±0.00 –
6(min.)	GPFC	1	70.00	57.56 ±5.87 =	70.00	58.44 ±5.94 =	70.00	57.78 ±6.05 =
	MultGPFC	4	71.67	**58.00 ±8.27**	71.67	**58.67 ±7.88**	78.33	**58.00 ±8.93**
Prostate (102)	Full	10509	81.55	81.55 ±0.00 –	60.55	60.55 ±0.00 –	86.18	**86.18 ±0.00** =
13(min.)	GPFC	1	90.18	83.72 ±3.18 –	90.18	83.18 ±3.68 –	90.18	83.82 ±2.85 –
	MultGPFC	4	92.18	**86.44 ±3.08**	92.18	**85.89 ±2.73**	91.27	85.29 ±3.04
Ovarian (253)	Full	15154	91.28	91.28 ±0.00 –	90.05	90.05 ±0.00 –	98.41	98.41 ±0.00 –
44(min.)	GPFC	1	99.62	97.86 ±1.22 –	99.62	97.22 ±1.48 –	99.62	97.89 ±1.18 –
	MultGPFC	4	100.00	**99.23 ±0.40**	100.00	**99.23 ±0.52**	99.60	**98.82 ±0.50**

Similarly, NB and DT improve their performance when using the MultGPFC constructed features on three datasets. The largest improvement that NB achieves is on Prostate dataset with 25% on average and 32% in the best case. DT also has an impressive increase on CNS with 8% on average and 28% in the best solution. Both learning algorithms have similar results in the remaining datasets except for a degradation of DT on Colon with 2.9% drop. However, the best accuracy of DT on this dataset is still 11% higher than the Full.

When compared with the GPFC constructed feature, the MultGPFC constructed features help KNN and NB obtain a significantly better results on four datasets, and DT on three datasets. Similar results appear on the remaining datasets. The construct features by MultGPFC obtain a similar performance as GPFC on these datasets, but its best accuracy always higher than the best accuracy obtained by GPFC. For example, the best accuracy of DT on Colon and CNS increase 6% and 8%, respectively.

In summary, over the 36 pairs of comparisons of MultGPFC against Full and GPFC using three learning algorithms on six datasets, the constructed feature set by MultGPFC wins 18, draw 17 and lose 1. It obtains the best accuracy for NB on all datasets, KNN on five and DT on three. The results indicate that combining wrapper and filter measure in evaluation enable GP to construct a significant small number of new features with better discriminating ability than the original feature set and the one constructed by GPFC which uses an embedded approach.

B. The Selected Features

It can be seen from Table IV that the number of selected features by MultGPFC is about 3.5% of the original features in Colon, 1.2% in CNS and less than 1% in the other datasets. Using this aggressive smaller number of features, NB improves its performance on all datasets with the largest increase of 21% on Prostate dataset. KNN and DT achieve significantly better results on four out of the six datasets with up to 8% improvement.

Compared with GPFC, MultGPFC selects more features as expected because its individual includes more constructed features than GPFC's. However, using the larger subsets selected by MultGPFC, KNN and DT have significantly better results than using the features selected by GPFC on half of the datasets and maintain the same performance on the others. The average accuracy obtained by MultGPFC selected features is the best on five out of the six datasets with an improvement up to 3% and 6% by KNN and DT, respectively. NB also maintains or improves its performance over those obtained by GPFC on all datasets except the Prostate dataset.

In general, the MultGPFC selected features win 22, draw 13, and lose 1 out of 36 comparisons, reaching the best accuracy on at least four datasets.

C. Combining Constructed and Selected Features

Table V shows the average results obtained by using the combination of the constructed and selected features. The significant test results show that using this combination, KNN and NB achieves significantly better results than using full on all datasets. The highest improvement of these two learning algorithms are 8% on Ovarian and 27% on Prostate datasets, respectively. Compared with this combination produced by GPFC, KNN obtains up to 4% higher accuracy on four datasets and NB also has significantly better results on two datasets namely Leukemia and Ovarian. MultGPFC maintains a similar accuracy as GPFC on the remaining datasets.

In contrast to KNN and NB, DT does not gain much from this combination. It can be seen in Table V that the results of DT using the combination set is almost the same as its results when using only the constructed features. This means that the constructed features have the highest information gain among all features in this combination set; therefore, DT always use them to build classifiers, resulting in the observed results.

In summary, the combination set of the constructed and selected features by MultGPFC help the three learning algorithms achieve even higher improvement resulting in 24 wins and 11 draws and 1 lose in totally 36 comparisons.

D. Comparison Between Different MultGPFC Feature Sets

To see which subset among the created feature subsets is the best subset for each learning algorithm, we plot the difference

TABLE IV: Results of selected features

Dataset	Subset	#F	B-KNN	A±Std-KNN	B-NB	A±Std-NB	B-DT	A±Std-DT
Colon (62)	Full	2000	74.28	74.28 ±0.00 =	72.62	72.62 ±0.00 –	74.29	74.29 ±0.00 =
2.3(min.)	GPFC	22	85.47	**76.40 ±4.82** =	85.48	**75.81 ±3.95** =	80.95	73.32 ±4.18 =
	MultGPFC	69	84.28	75.44 ±5.29	79.05	74.64 ±3.43	82.86	**74.57 ±4.18**
DLBCL (77)	Full	5469	84.46	84.46 ±0.00 –	81.96	81.96 ±0.00 –	80.89	80.89 ±0.00 –
3.7(min.)	GPFC	15	95.00	86.36 ±4.13 –	96.25	88.49 ±3.49 =	98.75	85.04 ±5.45 =
	MultGPFC	29	98.57	**88.77 ±4.38**	93.57	**88.95 ±2.53**	93.57	**87.33 ±3.10**
Leukemia (72)	Full	7129	88.57	88.57 ±0.00 –	91.96	91.96 ±0.00 –	91.61	91.61 ±0.00 –
4.5(min.)	GPFC	12	95.89	89.39 ±3.53 –	96.07	92.24 ±2.69 –	98.75	89.85 ±4.32 –
	MultGPFC	17	98.75	**92.54 ±2.68**	97.50	**94.65 ±1.56**	97.32	**95.37 ±1.31**
CNS (60)	Full	7129	56.67	56.67 ±0.00 =	58.33	58.33 ±0.00 –	50.00	50.00 ±0.00 –
6(min.)	GPFC	30	70.00	57.56 ±6.09 =	70.00	59.89 ±3.86 =	73.33	**57.78 ±5.63** =
	MultGPFC	85	73.33	**58.39 ±5.40**	65.00	**60.56 ±2.78**	70.00	56.22 ±5.69
Prostate (102)	Full	10509	81.55	81.55 ±0.00 –	60.55	60.55 ±0.00 –	86.18	86.18 ±0.00 =
13(min.)	GPFC	22	90.36	83.05 ±3.77 =	90.27	**87.04 ±2.06** +	90.18	82.32 ±3.39 –
	MultGPFC	53	89.18	**83.69 ±3.03**	86.36	81.88 ±2.87	92.18	**86.38 ±2.88**
Ovarian (253)	Full	15154	91.28	91.28 ±0.00 –	90.05	90.05 ±0.00 –	98.41	98.41 ±0.00 –
44(min.)	GPFC	9	100.00	98.15 ±0.96 –	98.82	97.75 ±0.68 –	100.00	97.87 ±1.08 –
	MultGPFC	10	100.00	**99.70 ±0.39**	99.62	**98.69 ±0.52**	100.00	**99.56 ±0.4**1

TABLE V: Results of constructed and selected features

Dataset	Subset	#F	B-KNN	A±Std-KNN	B-NB	A±Std-NB	B-DT	A±Std-DT
Colon (62)	Full	2000	74.28	74.28 ±0.00 –	72.62	72.62 ±0.00 –	74.29	**74.29 ±0.00** +
2.3(min.)	GPFC	23	87.38	**76.90 ±5.21** =	87.14	**75.96 ±4.03** =	79.28	72.25 ±4.07 =
	MultGPFC	73	84.28	75.91 ±4.80	80.71	75.38 ±2.97	85.48	71.15 ±5.89
DLBCL (77)	Full	5469	84.46	84.46 ±0.00 –	81.96	81.96 ±0.00 –	80.89	80.89 ±0.00 –
3.7(min.)	GPFC	16	95.00	86.80 ±4.83 –	96.07	89.36 ±4.00 =	94.64	86.51 ±4.08 =
	MultGPFC	33	97.32	**90.47 ±4.33**	97.32	**89.81 ±3.42**	94.82	**87.47 ±4.34**
Leukemia (72)	Full	7129	88.57	88.57 ±0.00 –	91.96	91.96 ±0.00 –	91.61	**91.61 ±0.00** =
4.5(min.)	GPFC	13	97.32	90.28 ±3.58 –	97.32	91.46 ±2.91 –	95.89	88.97 ±2.96 –
	MultGPFC	21	95.89	**93.33 ±1.66**	96.07	**93.38 ±1.56**	95.89	90.99 ±2.76
CNS (60)	Full	7129	56.67	56.67 ±0.00 –	58.33	58.33 ±0.00 –	50.00	50.00 ±0.00 –
6(min.)	GPFC	31	73.33	57.33 ±6.25 =	70.00	60.22 ±4.85 =	70.00	57.78 ±6.05 =
	MultGPFC	89	71.67	**59.61 ±6.68**	73.33	**61.33 ±5.31**	76.67	**57.94 ±8.66**
Prostate (102)	Full	10509	81.55	81.55 ±0.00 –	60.55	60.55 ±0.00 –	86.18	**86.18 ±0.00** =
13(min.)	GPFC	23	90.36	84.09 ±3.71 –	90.36	87.07 ±2.52 =	90.18	83.82 ±2.85 –
	MultGPFC	57	91.27	**86.35 ±2.90**	92.18	**87.22 ±2.79**	91.27	85.36 ±3.04
Ovarian (253)	Full	15154	91.28	91.28 ±0.00 –	90.05	90.05 ±0.00 –	98.41	98.41 ±0.00 –
44(min.)	GPFC	10	100.00	98.42 ±0.99 –	99.62	98.20 ±0.89 –	99.62	97.89 ±1.18 –
	MultGPFC	14	100.00	**99.55 ±0.36**	100.00	**99.36 ±0.46**	99.60	**98.82 ±0.50**

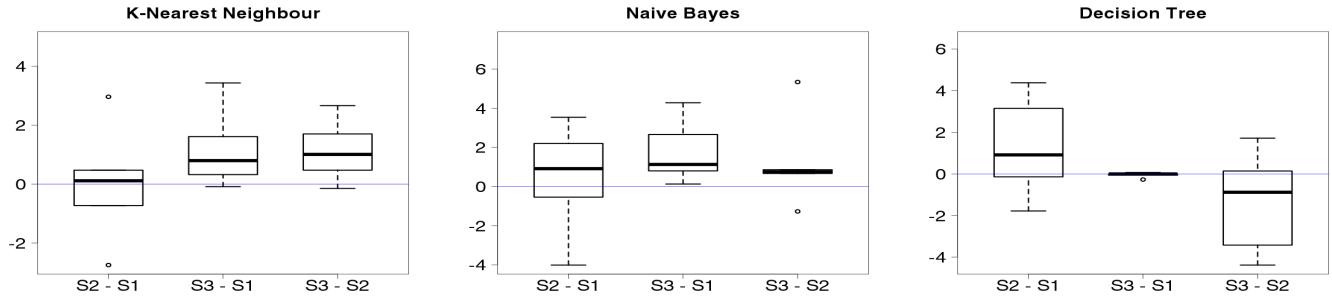

Fig. 3: Comparison between the constructed features (S1), selected features (S2), and the combination of both (S3).

in test accuracy between pairs of feature sets over all datasets. Fig. 3 shows the box plots of these differences using KNN, NB, and DT respectively. In this figure, $S1$ is used to represent the constructed feature set, $S2$ for the selected features, and $S3$ for the combination of both. Each sub-figure has three box plots in which "S2-S1" shows the subtraction result of $S1$ from $S2$. If this value is positive, the selected features would perform better than the constructed features and vice versa. Similarly, "S3-S1" and "S3-S2" reveal how good the combination set over the constructed and the selected features, respectively.

The different patterns shown in Fig. 3 reflect that the created feature sets have different effect on different learning algorithms. For KNN, the combination of constructed and selected features achieves better results than using either of them. For NB, although the effect of this combination set is not as high as in KNN, it still obtains slightly better results than either the constructed features or the selected features. On the other hand, using the selected features, DT obtains slightly better results than using the other two feature sets. To explain this phenomenon, we compared the training and test accuracies of DT. The comparisons (not shown here due to the page limit) reveal an overfitting problem since $S3$ performs better than $S2$ on training data. This indicates that the constructed features are too overfit to the training data. Therefore, the selected features used to construct these new features can be more general than

the constructed features.

It is noticed that the difference between $S3$ and $S1$ are positive in KNN and NB. This means that when combined with the constructed features, the selected features have some contribution to the improvement of KNN and NB. In contrast, this difference in DT is almost zero indicating that the selected features in $S3$ are hardly used in the learned classifiers, which is shown in the results. This phenomenon also shows that the information gain of the constructed features is higher than all the selected features.

VI. Conclusions

The goal of this study was to propose a feature construction and selection method that can produce much smaller feature sets to improve the performance of common learning algorithms on high-dimensional data. The goal was successfully achieved by proposing a new multiple feature construction method called MultGPFC that constructs a few high-level features which are evaluated by a hybrid method of wrapper and filter approach. MultGPFC is designed so that it can be applied to high-dimensional data within a reasonable time.

Experiment results on six high-dimensional datasets show that by combining the strengths of wrapper and filter approaches, MultGPFC can construct a few features that can significantly improve the performance of the three learning algorithms on the original feature sets. Compared with GPFC, the MultGPFC produced feature sets either obtain significantly better or similar results almost all datasets. Comparisons between different feature sets produced by MultGPFC show that using the combination of the selected and constructed features, KNN and NB obtain slightly better results than using either of them. On the other hand, the selected features are more effective for DT than the other two feature sets.

In this study, the combination of wrapper and filter measure is static, predefined and applied to all datasets. A dynamic combination of wrapper and filter approach that can be automatically tuned during the evolutionary process may boost the performance of the algorithm to a better result. Furthermore, testing the performance of MultGPFC on multiple-class problems is also included in our future work. Finally, applying local search to balance the exploration and exploitation in the evolutionary search is also a promising approach.

References

[1] P. N. S. J. Russell, "Artificial intelligence: A modern approach (second edition)," *Pearson Education*, 2003.

[2] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*. Wadsworth and Brooks, 1984.

[3] H. Liu and H. Motoda, *Feature Extraction, Construction and Selection: A Data Mining Perspective*. Norwell, MA, USA: Kluwer Academic Publishers, 1998.

[4] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," *Computers & Electrical Engineering*, vol. 40, pp. 16–28, 2014.

[5] H.-H. Hsu, C.-W. Hsieh, and M.-D. Lu, "Hybrid feature selection by combining filters and wrappers," *Expert Systems with Applications*, vol. 38, no. 7, pp. 8144–8150, 2011.

[6] P. Saengsiri, P. Meesad, S. N. Wichian, and U. Herwig, "Comparison of hybrid feature selection models on gene expression data," in *Conference on ICT and Knowledge Engineering*. IEEE, 2010, pp. 13–18.

[7] S. Ahmed, M. Zhang, and 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.

[8] H. Al-Sahaf, A. Al-Sahaf, B. Xue, M. Johnston, and M. Zhang, "Automatically evolving rotation-invariant texture image descriptors by genetic programming," *IEEE Transactions on Evolutionary Computation*, vol. PP, no. 99, pp. 1–1, 2016.

[9] L. Shao, L. Liu, and X. Li, "Feature learning for image classification via multiobjective genetic programming," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 7, pp. 1359–1371, 2014.

[10] K. Krawiec, "Evolutionary feature selection and construction," in *Encyclopedia of Machine Learning*, 2010, pp. 353–357.

[11] K. Neshatian, M. Zhang, and P. Andreae, "A filter approach to multiple feature construction for symbolic learning classifiers using genetic programming," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 5, pp. 645–661, 2012.

[12] M. Muharram and G. Smith, "Evolutionary constructive induction," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, pp. 1518–1528, 2005.

[13] B. Tran, B. Xue, and M. Zhang, "Genetic programming for feature construction and selection in classification on high-dimensional data," *Memetic Computing*, vol. 8, no. 1, pp. 3–15, 2015.

[14] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

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

[16] B. Xue, M. Zhang, W. N. Browne, and X. Yao, "A survey on evolutionary computation approaches to feature selection," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 4, pp. 606–626, 2016.

[17] K. Krawiec, "Genetic programming-based construction of features for machine learning and knowledge discovery tasks," *Genetic Programming and Evolvable Machines*, vol. 3, pp. 329–343, 2002.

[18] B. Bhanu and 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.

[19] K. Neshatian, M. Zhang, and M. Johnston, "Feature Construction and Dimension Reduction Using Genetic Programming," in *Advances in Artificial Intelligence*, 2007, vol. 4830, pp. 160–170.

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

[21] ——, "Pareto front feature selection: Using genetic programming to explore feature space," in *Conference on Genetic and Evolutionary Computation*, 2009, pp. 1027–1034.

[22] S. Ahmed, M. Zhang, and L. Peng, *Feature selection and classification of high dimensional mass spectrometry data: a genetic programming approach*. Springer, 2013.

[23] D. Muni, N. Pal, and J. Das, "Genetic programming for simultaneous feature selection and classifier design," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 36, no. 1, pp. 106–117, 2006.

[24] K. Nag and N. Pal, "A multiobjective genetic programming-based ensemble for simultaneous feature selection and classification," *Cybernetics, IEEE Transactions on*, vol. 46, no. 2, pp. 499–510, Feb 2016.

[25] M. Smith and L. Bull, "Genetic Programming with a Genetic Algorithm for Feature Construction and Selection," *Genetic Programming and Evolvable Machines*, vol. 6, pp. 265–281, 2005.

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

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

[28] C. Ding and H. Peng, "Minimum redundancy feature selection from microarray gene expression data," *Journal of bioinformatics and computational biology*, vol. 3, no. 02, pp. 185–205, 2005.

[29] A. Statnikov, C. F. Aliferis, I. Tsamardinos, D. Hardin, and S. Levy, "A comprehensive evaluation of multicategory classification methods for microarray gene expression cancer diagnosis," *Bioinformatics*, vol. 21, pp. 631–643, 2005.