Genetic Programming based Feature Construction for Classification with Incomplete Data

Cao Truong Tran, Mengjie Zhang, Peter Andreae and Bing Xue
School of Engineering and Computer Science
Victoria University of Wellington, PO Box 600, Wellington 6140, New Zealand
{cao.truong.tran,mengjie.zhang,peter.andreae,bing.xue}@ecs.vuw.ac.nz

ABSTRACT

Missing values are an unavoidable problem in many real-world datasets. Dealing with incomplete data is an crucial requirement for classification because inadequate treatment of missing values often causes large classification error. Feature construction has been successfully applied to improve classification with complete data, but it has been seldom applied to incomplete data. Genetic programming-based multiple feature construction (GPMFC) is a current encouraging feature construction method which uses genetic programming to evolve new multiple features from original features for classification tasks. GPMFC can improve the accuracy, and reduce the complexity of many decision trees and rule-based classifiers; however, it cannot directly work with incomplete data. This paper proposes IGPMFC which is extended from GPMFC to tackle with incomplete data. IGPMFC uses genetic programming with interval functions to directly evolve multiple features for classification with incomplete data. Experimental results reveal that not only IGPMFC can substantially improve the accuracy, but also can reduce the complexity of learnt classifiers facing with incomplete data.

CCS CONCEPTS

•**Computing methodologies → Genetic programming;** *Supervised learning by regression;* Classification and regression trees;

KEYWORDS

incomplete data, feature construction, genetic programming, classification

ACM Reference format:
Cao Truong Tran, Mengjie Zhang, Peter Andreae and Bing Xue . 2017. Genetic Programming based Feature Construction for Classification with Incomplete Data. In *Proceedings of GECCO '17, Berlin, Germany, July 15-19, 2017,* 8 pages.
DOI: http://dx.doi.org/10.1145/3071178.3071183

1 INTRODUCTION

Classification is one of the most important tasks in machine learning and data mining [13]. The input space is of key importance in most classifiers. Many classifiers such as decision trees and rule-based classifiers cannot achieve adequate predictive performance facing difficult problems. One of the main reasons is that the classifiers cannot transform the input space to obtain good class separability. Feature construction is a data preprocessing process which transforms data from its original space to a new space having better separability between classes [25].

An incomplete dataset is a dataset which contains missing values in some features. Missing values are a common problem in many datasets [17]. For instance, in the UCI repository [1], one of the most popular data repositories for benchmarking machine learning tasks, 45% of the datasets contain missing values. Incomplete data causes a number of serious problems for classification. One of the most severe problems is non-applicability of classifiers since almost all existing classifiers require complete data. Consequently, these classifiers are not able to directly work with original incomplete data. Furthermore, incomplete data often leads to large classification error [8].

The problem of incomplete data has been addressed extensively in the statistical analysis field [9, 17] and also, but with less effort, in the classification literature. There are two main approaches to classification with incomplete data. One approach is to use imputation methods that fill plausible values into missing fields before using classifiers [8, 28, 30]. For example, mean imputation fills all missing fields in each feature with the mean of complete values in the same feature. Another approach is to use classifiers that can directly classify incomplete data without using imputation methods [22, 29]. For example, C4.5 [22] can directly classify incomplete data without using imputation methods. Despite the fact that the two approaches are able to deal with incomplete data to a certain level, they often lead to large classification error [7]. Therefore, further approaches to improving classification accuracy of incomplete data should be investigated.

Feature construction is a process of constructing better features from original features for classifiers. Genetic programming (GP) is a popular evolutionary technique inspired by biological evolution to search a solution for a problem in the form of a computer program [15]. GP is able to learn the definition of a function itself from example data, so GP is an excellent choice for feature construction, and has been widely applied [6, 27, 31].

Although there are many GP based feature construction methods, most of them construct only a single feature, which needs combine with the original feature set, but will lead to a higher dimensionality and a more complex classifier [6]. GP-based multiple feature construction (GPMFC) [21] is a recent promising filter approach using GP for feature construction. GPMFC is able to evolve multiple high-level features from the original features. The empirical results show that, in almost all cases, GPMFC can not only improve the classification performance, but can also reduce the complexity of many decision trees and rule-based classifiers.

However, GPMFC is not able to deal with incomplete data. As a result, to use GPMFC for incomplete data, imputation methods are required to transform incomplete data into complete data before using GPMFC [27]. In order to obtain good performance, GPMFC has to be combined with sophisticated imputation methods such as MICE imputation [32]. Unfortunately, sophisticated imputation methods such as MICE are often suitable for batch imputation, but computationally intensive for imputing missing values in a single instance in the unseen set for classification [30]. Therefore, the ability of GPMFC to directly deal with incomplete data should be investigated.

1.1 Research goals

The overall goal of this paper is to demonstrate a new method that uses GP to directly construct multiple features for classification with incomplete data without using imputation methods. To achieve this goal, we develop an extension of GPMFC, called IGPMFC, that uses GP with interval functions as the function set to directly construct multiple features. To evaluate the impact of IGPMFC on classification with incomplete data, the experiments will be conducted to answer the following questions:

(1) Whether IGPMFC can improve classification performance and reduce the complexity of classifiers with incomplete data compared to using original features; and
(2) Whether IGPMFC can improve classification performance and reduce the complexity of classifiers with incomplete data compared to using GPMFC combined with imputation methods.

1.2 Organisation

The rest of the paper is organised as follows. Section 2 discusses related work. Section 3 presents the IGPMFC algorithm which performs multiple feature construction for classification with incomplete data using GP with interval functions. Section 4 outlines experiment design. Section 5 presents empirical results and analysis. Section 6 draws conclusions and proposes future work.

2 RELATED WORK

This section presents related work including classification with incomplete data, GP for feature construction and GP-based multiple feature construction.

2.1 Classification with Incomplete data

The main approaches to classification with incomplete data include deletion approach, imputation approach, model-based approach and machine learning approach [8].

2.1.1 Deletion approach. This approach removes all incomplete instances before using classifiers. The main advantage of this approach is to provide complete data that can be then classified by any classifiers. However, incomplete instances are not participated in the classification process [8].

2.1.2 Imputation approach. This approach uses imputation methods to put plausible values into missing fields before using classifiers. For example, one simple imputation method is mean imputation that fills missing fields in each feature with the average of the complete values in the same feature. One of the most sophisticated imputation methods is multivariate imputation by chained equations (MICE) [32]. MICE uses a set of regression models to build regression functions that are then used to estimate missing values. Initially, missing values in each feature are randomly filled with complete values in the feature. After that, each feature containing missing values is regressed on other features to compute a better estimate for the feature. The process is repeated several times for all features containing missing values to generate one imputed dataset. The whole procedure is repeated N times to generate N imputed datasets. After that, the N imputed datasets are combined to provide the final imputed data.

The main advantage of using imputation approach is to provide complete data for classification. Therefore, both complete and incomplete instances are participated in the classification process. Consequently, using imputation methods is a major approach to classification with incomplete data. However, the quality of classification using imputed data strongly depends on the quality of imputation methods. Using simple imputation methods such as mean imputation often results in large classification error. In contrast, using more sophisticated imputation methods such as MICE imputation results in smaller classification error, but they are computationally intensive [30].

2.1.3 Model-based approach. This approach builds the data distribution model from input data. After that, the data distribution model is used to classify both complete and incomplete instances by using the Bayesian decision theory [2]. Although this approach is able to classify both complete and incomplete instances, it requires making assumptions about the joint distribution of all features in the model [8].

2.1.4 Machine learning approach. This approach builds classifiers that can directly classify incomplete data without using imputation methods. For example, C4.5 can handle missing values in both the training set and test set by using a probabilistic approach. In the training stage, each feature value is assigned a weight: if a feature value is known, then the weight is assigned one; otherwise, the weight of any other values for that feature is the frequency of that values. In the test stage, if a test case is unknown, from the current node, it finds all the available branches and decides the class label by using the most probable value [22]. Some other decision trees can deal with incomplete data including CART [4], REPTree [3] and BFTree [23].

2.2 GP for Feature Construction

Feature construction is a process of creating a set of new features which can provide a new input space for classification that is better than the original input space. Constructed features are typically mathematical expressions of the original features. The original purpose of GP is to evolve computer programs that perform a user-defined task. Therefore, GP is an ideal choice for evolving constructed features, and using GP for feature construction has been a research trend in recent years [6].

In a GP-based feature construction approach, a new constructed feature is often represented by a tree-like individuals, where internal nodes are arithmetic operators or functions, and leaf nodes are original features or constants. GP acts as a search technique that combines with a fitness function evaluation method that guides GP for searching new constructed features [6].

Two main ways to evaluate constructed features are the wrapper approach and the filter approach. In the wrapper approach, constructed features are evaluated by the performance of a classifier [16], [24], [26]. Every evaluation requires training a classifier and then testing its performance; hence, the search process of the wrapper approach is typically computationally intensive, but the classification accuracy is often better than filter approach. In the filter approach, constructed features are evaluated by a measure such as information gain, the gini index, chi-square [19] and Fisher criterion [10]. None of classifiers is included in the evaluation of constructed features; hence, the search of the filter approach is expected to be more efficient and the results are expected to be more general [24].

2.3 GP-based Multiple Feature Construction

GPMFC is a filter approach to feature construction that uses GP for constructing multiple features [21]. GPMFC uses GP to evolve new features, and the purity of class intervals as a measure to evaluate new features.

Algorithm 1 shows the main body of the GPMFC algorithm. The input data for GPMFC includes two parts. The first part is a matrix containing values of original features. The second part is an array containing class labels corresponding observations in the matrix. For each class label, GPMFC evolves the best constructed feature that maximise the purity of the corresponding class interval. As a result, the output of GPMFC is a set of constructed features equal to the number of classes in the problem. The detail of GPMFC can be seen in [21].

In order to apply GPMFC, firstly, GPMFC uses the training data to build a set of constructed features that then forms a transformation. Next, the training data is put into the transformation to generate transformed training data. After that, a classifier uses the transformed training data to build a classification model that is then used to classify the transformed unseen data.

The experimental results show that in almost all cases, GPMFC greatly improves classification accuracy of decision trees and rule-based classifiers. Furthermore, GPMFC helps to reduce the complexity of the learnt classifiers. However, GPMFC is not able to directly deal with incomplete data. Therefore, the ability of GPMFC to directly deal with incomplete data should be investigated.

Algorithm 1: GPMFC [21]

```
Input:
D: a matrix containing values of original features
C: an array containing class values corresponding
observations in D
Output: CF– a set of constructed features
1  $CF \leftarrow \{\}$
2  for $c \in C$ do
3      $P \leftarrow InitialPopulation$
4      $bestFitness \leftarrow +\infty$
5      while $\neg maxGenerations \wedge bestFitness \neq 0$ do
6          for $\phi \in P$ do
7              $\phi_{fitness} \leftarrow Fitness(D, \phi, c)$
8              if $\phi_{fitness} < bestFitness$ then
9                  $bestProgram \leftarrow \phi$
10                 $bestFitness \leftarrow \phi_{fitness}$
11             end
12         end
13         Perform selection
14         Perform genetic operators
15     end
16     $CF \longleftarrow CF \cup \{bestProgram\}$
17 end
18 return CF
```

3 GP WITH INTERVAL FUNCTIONS FOR MULTIPLE FEATURE CONSTRUCTION

Although GPMFC is a powerful feature construction method for complete data, it cannot directly work with datasets containing missing values. To tackle this problem, we proposes IGPMFC which is an extension of GPMFC. IGPMFC uses GP with interval functions to evolve new multiples features for incomplete data. The underlying idea of IGPMFC is that it uses interval functions as the function set of GP to deal with missing values. If a feature value is missing, it will be substituted by an interval associated with the feature. If a feature value is complete, it will still be substituted by an interval such that both the lower bound and upper bound are equal to the feature value. The purpose of using interval functions is that missing fields are unknown; hence replacing a missing field with an interval instead of a single value may reflect better the uncertainty of the missingness.

The interval associated with each feature must represent the range of possible values of the feature and needs to be estimated by the algorithm. Furthermore, the interval function set of GP also needs to be defined.

3.0.1 Finding the Interval of a Feature. A feature interval is the range which covers the most occurrence values of the feature. The interval of a feature should be estimated from the distribution of the feature values. A simple way to find the interval of a feature is to consider that the range between the minimum and maximum of the feature which is the interval of the feature. However, this kind of interval possibly contains outliers which are not desired [21].

If the values of feature f is normally distributed, the interval $(\mu_f - 3\sigma_f, \mu_f + 3\sigma_f)$ could cover 99% of the feature values, where μ_f and σ_f are the mean and the standard deviation of the feature f, respectively [21]. Unfortunately, the values of a feature are not necessarily normally distributed; therefore, the interval might include too many values, or too few values. Hence, it would be essential to find a method of estimating the interval of a feature which is able to apply to all kinds of distribution. One simple method to perform the task is to sort all feature values and then remove a few lowest and highest values from both ends of the data range. A more advanced method is to use the introselect algorithm proposed in [20]. We used the algorithm in [21] for estimating the interval of a feature.

3.0.2 Interval Functions. Assuming the interval of feature a is represented by the range between the lower bound a_l and the upper bound a_u, and the interval of feature b is represented by the range between the lower bound b_l and the upper bound b_u. In IGPMFC, the function set of GP uses four interval arithmetic operations defined as follows [14]:

$$a + b = \begin{cases} lower: & a_l + b_l \\ upper: & a_u + b_u \end{cases}$$

$$a - b = \begin{cases} lower: & a_l - b_u \\ upper: & a_u - b_l \end{cases}$$

$$a * b = \begin{cases} lower: & min(a_l * b_l, a_l * b_u, a_u * b_l, a_u * b_u) \\ upper: & max(a_l * b_l, a_l * b_u, a_u * b_l, a_u * b_u) \end{cases}$$

$$a/b = \begin{cases} lower: & min(a_l/b_l, a_l/b_u, a_u/b_l, a_u/b_u) \\ upper: & max(a_l/b_l, a_l/b_u, a_u/b_l, a_u/b_u) \end{cases}$$

It is important to notice that the division operation is not right if the lower and upper bounds of denominator have different signs. Therefore, it requires an assumption that the denominator lower bound has the same sign with the denominator upper bound. Nevertheless, we permit GP search to remove trees which break the assumption.

3.1 Estimating the Real Output of an Individual

The output of an individual evolved by GP with interval functions is an interval. Nevertheless, to use constructed features for classification, single values are required. Hence, in IGPMFC, the real output of an individual is calculated as the middle point of the interval. Assuming that $[out_l, out_u]$ is the output of an individual, the real output can be defined as follows:

$$out = \frac{out_l + out_u}{2}$$

4 EXPERIMENT DESIGN

This section shows detailed experiment design including the comparison methods, datasets, the imputation methods used in the experiments, GP settings and classification algorithms.

4.1 Comparison Method

The experiments are designed to evaluate the impact of IGPMFC to construct new features for classification with incomplete data. To achieve this, three experimental setups are designed, as shown in Fig.1, Fig.2 and Fig.3. The Fig.1 shows classification with incomplete data by using a classifier that is able to directly classify incomplete data. The Fig.2 shows classification with incomplete data by using an imputation method to transfer incomplete data into complete data that is then used by GPMFC to construct new features before using a classifier. The Fig.3 shows classification with incomplete data by using IGPMFC to construct new features from incomplete data before using a classifier.

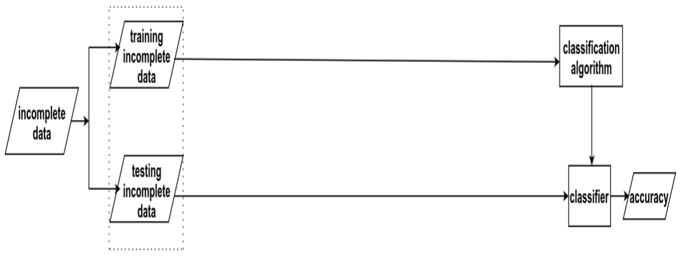

Figure 1: Classification with incomplete data by using a classifier able to classify incomplete data.

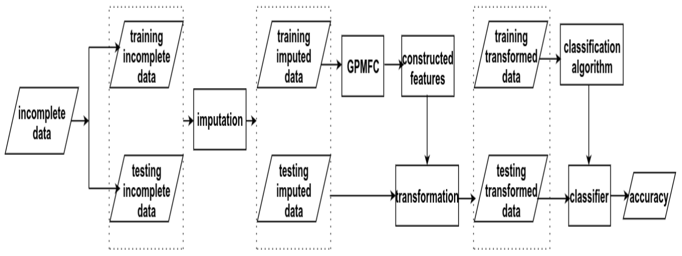

Figure 2: Classification with incomplete data by using an imputation method and GPMFC before using a classifier.

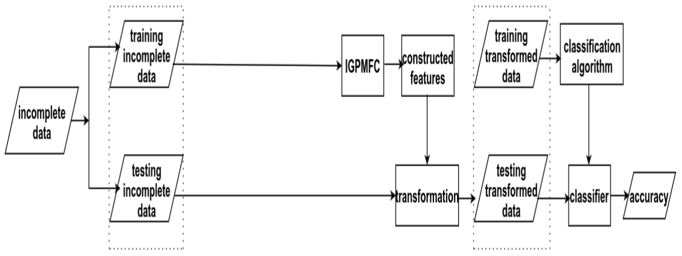

Figure 3: Classification with incomplete data by using IGPMFC before using a classifier.

In the three experimental setups, firstly, incomplete data is divided into training incomplete data and testing incomplete data. In the first setup, as shown in Fig.1, the training incomplete data is directly put into a classification algorithm to build a classifier that is then used to classify testing incomplete data. In the second setup, as shown in Fig.2, both training incomplete data and testing incomplete data are put into an imputation method to generate training imputed data and testing imputed data, and then, the training imputed data is put into GPMFC to build a data transformation. The data transformation is then used to transform the training imputed data and the testing imputed data into training transformed data and testing transformed data, respectively. In the third setup, as shown in Fig.3, IGPMFC directly uses training incomplete data to construct new features that is then used to build a data transformation. The data transformation is then used to transform the training incomplete data and the testing incomplete data into training transformed data and testing transformed data, respectively. After that, in both the second and third setups, the training transformed data is then put into a classification algorithm to build a classifier that is then used to classify the testing transformed data.

4.2 Datasets

The experiments used six benchmark datasets selected from the UCI machine learning repository [1]. Table 1 summarises the main characteristics of each dataset including the number of instances, the number of features, the number of classes and the percentage of incomplete instances.

Table 1: The Datasets Used in the Experiments

Dataset	#Instances	#Features	#Classes	Incomplete instances (%)
Bands	205	25	6	26.38
Breast	286	9	2	3.15
Hepatitis	155	19	2	48.39
Balance	625	4	3	0
Iris	150	4	3	0
Liver	345	6	2	0

The first three datasets suffer from missing values in a "natural" way. To evaluate more precisely the performance of IGPMFC on the incomplete data, "artificial" missing values were introduced into important features in the last three complete datasets. The important features were selected using the correlation-based feature selection method (CFS), where subsets of features that are highly correlated with the class while having low intercorrelation are preferred [12]. With each dataset, firstly, use CFS to choose the relevant features, and then perform 30 times: put randomly 20% missing values into the relevant features. Therefore, for each dataset, 30 incomplete datasets were generated, and a total of 90 (30 × 3) artificial incomplete datasets were used in the experiments.

None of the datasets in the experiments comes with a specific test set. Moreover, in some datasets, the number of instances is relatively small. Therefore, the ten-fold cross-validation method was used to measure the performance of the learned classifiers. With the first three incomplete datasets, the ten-fold cross-validation method was performed 30 times. With each of the last three complete datasets, the ten-fold cross-validation method was performed on the 30 incomplete datasets. Consequently, for each dataset, 300 pairs of training and testing sets were generated.

4.3 Imputation algorithms

The experiment used three imputation methods which are mean imputation, KNN-based imputation and MICE imputation [32]. Mean imputation and KNN-based imputation were in-house implementation. The experiment used MICE's implementation in [5] for multiple imputation using random forest for regression by setting their parameters as the default values.

4.4 GP settings

The experiments used the ECJ package [18] to implement GP. The parameters of GP were the same in all experiments and are shown in Table 2. For each pair of training set and test set, GPMFC combined with mean imputation, GPMFC combined with KNN-based imputation, GPMFC combined with MICE imputation and IGPMFC run a number of times, each constructing a new feature for a particular class. Therefore, on each dataset, GP runs 300×NumberOfClasses×4 times. Consequently, for the six datasets, the total number of experiments is 300×(6+2+2+3+3+2)×4 = 21600.

Table 2: GP parameters.

Parameter	Value
Function set	Interval functions, +, -, x, / (protected division)
Variable terminals	Interval of the original features $\{f_1, f_2, .., f_n\}$
Constant terminals	Random float values
Population size	1024
Initialization	Ramped half-and-half
Generations	50
Crossover probability	60%
Mutation probability	30%
Reproduction rate	10%
Selection type	Tournament(size=7)

4.5 Classification algorithms

The experiments used four decision trees that are able to classify incomplete data: C4.5 [22], CART [4], REPTree [3] and BFTree [23]. For all the classifiers, WEKA's implementation was used and all parameters set to WEKA's defaults [11].

5 RESULTS AND ANALYSIS

Table 3 and Table 4 present the average of classification accuracy along with standard deviation and the average of size of the decision trees (the number of nodes in the trees), respectively. In both the tables, the averages were calculated on 30 times performing ten-fold cross-validation on each dataset. Baseline column indicates results from the first experimental setup in Fig.1; MeGPMFC, KNNGPMFC and MiGPMFC columns indicate results from the second experimental setup in Fig.2 by using mean imputation, KNN-based imputation and MICE imputation combined with using GPMFC, respectively; IGPMFC column indicates results from the third experimental setup in Fig.3. In order to compare the classification accuracy of IGPMFC with the other methods, the Wilcoxon signed-ranks tests at 95% confidence level have been conducted. "T" columns in Table 3 indicate significant tests of the columns before them against IGPMFC, where "+" means IGPMFC was significantly more accuracy, "=" means not significantly different and "-" means significantly less accuracy.

Table 3: Average of Classification Accuracy on the Test Set

Dataset	Classifier	IGPMFC	Baseline	T	MeGPMFC	T	KNNGPMFC	T	MiGPMFC	T
Bands	C4.5	68.67±1.13	68.45±1.88	=	68.33±1.70	=	62.06±2.00	+	64.05±2.48	+
	CART	68.59±1.32	65.99±1.69	+	68.80±1.35	=	63.38±1.54	+	67.86±1.35	+
	REPTree	67.84±1.16	65.82±1.71	+	67.95±1.77	=	62.79±1.34	+	67.22±1.56	=
	BFTree	67.81±1.37	66.66±1.94	+	66.83±1.71	+	61.55±1.94	+	63.46±2.08	+
Breast	C4.5	95.68±0.51	94.83±0.46	+	95.76±0.50	=	95.79±0.52	=	95.65±0.50	=
	CART	96.07±0.49	94.42±0.44	+	96.12±0.44	=	96.20±0.42	=	96.10±0.48	=
	REPTree	96.04±0.50	94.35±0.65	+	95.99±0.51	=	96.15±0.46	=	96.12±0.43	=
	BFTree	95.32±0.86	94.49±0.54	+	95.59±0.86	=	95.44±0.93	=	95.63±0.86	=
Hepatitis	C4.5	80.52±2.41	79.21±1.75	+	80.20±2.62	=	80.50±2.65	=	80.80±2.08	=
	CART	80.94±2.59	77.47±1.45	+	80.59±2.05	=	80.97±2.22	=	81.10±2.30	=
	REPTree	80.37±2.39	79.32±2.17	=	80.50±2.31	=	80.10±2.41	=	81.03±2.38	=
	BFTree	79.93±2.36	78.55±1.80	+	79.51±2.71	=	79.85±2.24	=	80.51±2.45	=
Balance	C4.5	94.41±0.92	77.41±1.26	+	92.45±1.45	+	92.04±1.46	+	93.02±1.36	+
	CART	94.37±0.89	77.97±1.19	+	92.20±1.53	+	91.80±1.74	+	92.75±1.49	+
	REPTree	94.31±0.88	77.23±1.70	+	91.74±1.68	+	91.64±1.70	+	92.46±1.44	+
	BFTree	93.99±1.01	77.44±0.99	+	91.80±1.59	+	91.38±1.78	+	92.52±1.51	+
Iris	C4.5	91.77±1.97	89.35±2.10	+	89.93±2.38	+	88.06±2.67	+	93.53±1.66	-
	CART	92.33±2.00	89.62±1.76	+	90.13±2.50	+	88.42±2.57	+	94.02±2.26	-
	REPTree	92.24±1.85	86.15±2.26	+	89.42±2.88	+	88.19±2.43	+	94.22±1.43	-
	BFTree	92.33±1.84	89.73±1.80	+	89.95±2.37	+	88.48±2.41	+	94.33±1.45	-
Liver	C4.5	64.27±1.89	61.56±2.15	+	64.06±1.67	=	64.67±1.60	=	65.41±1.99	-
	CART	64.49±2.07	63.27±2.20	+	63.56±2.10	=	64.52±2.06	=	65.72±2.15	=
	REPTree	64.36±2.40	63.38±2.61	=	63.20±2.70	+	63.87±2.20	=	65.51±2.29	=
	BFTree	64.20±2.37	62.88±2.33	=	63.33±2.19	=	63.77±1.98	=	64.65±2.25	=

Table 4: Average of Size of classifiers

Dataset	Classifier	IGPMFC	Baseline	MeGPMFC	KNNGPMFC	MiGPMFC
Bands	C4.5	5.21±0.83	85.10±4.23	5.36±0.77	5.60±0.91	4.94±0.96
	CART	9.28±3.26	57.05±13.93	10.50±3.84	19.37±5.81	18.94±7.81
	REPTree	22.56±4.71	43.50±3.35	25.20±5.55	28.10±4.54	29.74±5.05
	BFTree	28.90±8.88	99.38±13.87	30.07±11.69	30.46±11.26	22.12±10.53
Breast	C4.5	6.28±0.95	23.20±1.38	6.91±1.25	7.04±1.43	6.84±1.01
	CART	5.01±1.17	16.57±2.69	4.73±1.03	5.10±1.38	4.83±0.95
	REPTree	5.25±1.39	13.14±1.96	5.27±1.10	5.07±1.23	5.46±1.04
	BFTree	11.70±1.94	29.82±4.15	11.34±1.34	11.56±1.64	11.58±1.94
Hepatitis	C4.5	7.07±0.83	17.04±1.03	7.09±1.04	7.24±1.06	7.54±0.87
	CART	6.78±1.30	7.38±2.65	7.29±1.60	7.01±1.20	6.68±1.18
	REPTree	6.89±1.11	7.08±1.28	6.78±0.92	6.90±1.18	6.58±1.01
	BFTree	11.16±2.14	22.23±4.74	11.30±1.57	11.33±1.84	11.19±1.39
Balance	C4.5	9.76±1.72	57.48±3.47	17.39±3.79	18.34±4.44	16.52±3.36
	CART	8.28±1.84	68.70±9.11	15.78±4.32	15.72±4.49	13.60±3.94
	REPTree	8.36±1.42	43.03±2.30	13.25±2.34	14.46±2.83	11.84±2.41
	BFTree	16.94±3.96	156.1±15.35	29.34±5.51	32.18±7.58	24.51±4.81
Iris	C4.5	7.04±0.75	14.01±1.33	8.49±1.08	8.20±0.96	5.11±0.27
	CART	5.62±0.51	17.10±1.81	7.46±0.93	7.26±1.01	4.99±0.13
	REPTree	5.48±0.45	9.51±0.95	6.82±0.77	6.74±0.68	5.03±0.09
	BFTree	7.28±0.75	23.56±2.67	9.18±1.18	9.21±1.14	5.72±0.54
Liver	C4.5	5.13±1.37	34.80±3.16	4.95±0.78	5.37±0.77	5.76±0.79
	CART	15.10±4.76	39.27±7.77	13.74±3.48	14.94±4.69	13.24±4.02
	REPTree	19.58±4.21	28.58±3.00	20.04±3.35	19.14±4.33	20.16±4.08
	BFTree	32.54±10.0	65.32±8.10	32.06±9.30	35.36±8.01	30.74±7.60

5.1 Effect of Constructed Features on Classification Accuracy

Fig.4 summarises classification accuracy improvement of the classifiers by using constructed features generated by IGPMFC compared to using original features. It is clear from Fig.4 that in almost all cases, IGPMFC substantially improves classification accuracy of the classifiers. However, the classification accuracy improvement is different among datasets. For example, on the Balance Scale, the improvement is much higher than in the other datasets. Moreover, the classification accuracy improvement in each classifier is also different on different datasets.

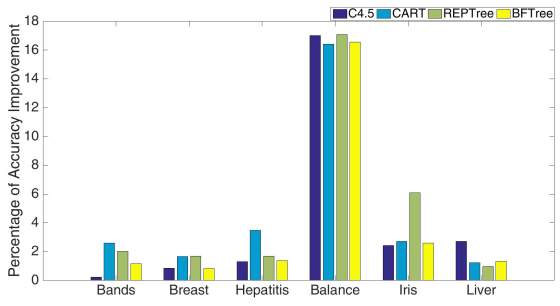

Figure 4: Classification improvement by using IGPMFC compared to Baseline

Fig. 5 summarises the accuracy comparison of IGPMFC with Baseline, MeanGPMFC, KNNGPMFC and MiceGPMFC. It is clear from Fig. 5 that IGPMFC achieves significantly better classification accuracy than Baseline in almost all cases. Moreover, in half cases, IGPMFC achieves significantly better classification accuracy than MeanGPMFC and KNNGPMFC and similar on the other half cases. Furthermore, in 24 cases, IGPMFC achieves significantly better classification accuracy than MiceGPMFC in seven cases, similar accuracy in 12 cases and significantly worse in five cases .

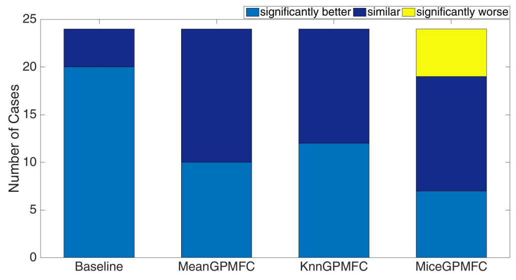

Figure 5: Accuracy comparison of IGPMFC with Baseline, MeanGPMFC and MiceGPMFC

In summary, in almost all cases, IGPMFC not only can achieve better classification accuracy compared to using original features, but also can achieve better classification accuracy than using GPMFC combined with mean imputation or KNN-based imputation in most cases. Moreover, IGPMFC is comparable with GPMFC combined with using Mice Imputation that is expensive for classification tasks.

5.2 Effect of Constructed Features on the Complexity of Classifiers

The complexity of the classifiers is evaluated by the average number of nodes in the decision trees. A decision tree with a small number of nodes is preferred because having too many nodes is often a symptom for poor generalisation, especially in nonrectangular decision spaces [21].

Fig.6 shows the percentage of size reduction by using the constructed features generated by IGPMFC over using the original features ($reduction = \frac{size_{Baseline} - size_{IGPMFC}}{size_{Baseline}}$). As can be seen from Fig.6 that using the constructed features generated by IGPMFC helps to decrease considerably the complexity of the learned classifiers. In all of the datasets, except for REPtree achieving about 30% size reduction in some datasets, the complexity of the other three classifiers reduces around 50%. Especially, on the Balance Scale dataset, the decrease in complexity is around 90%.

Fig. 7 shows the average of ratio tree size of the other methods over IGPMFC. On average, Baseline generates about 4.0 times bigger trees than those using IGPMFC, and all MeanGPMFC, KNNGPMFC and MiceGPMFC generate bigger trees than IGPMFC.

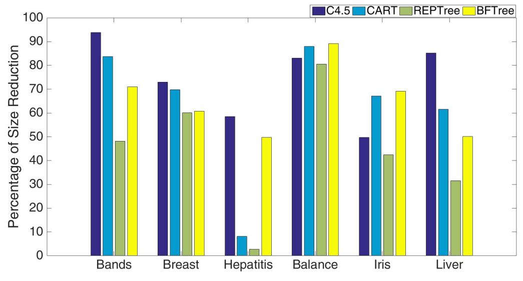

Figure 6: Size reduction by using IGPMFC compared to Baseline

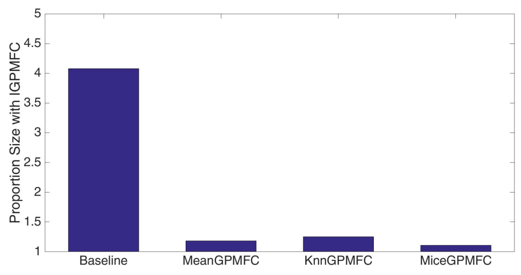

Figure 7: The average of ratio tree sizes of Baseline, MeanGPMFC and MiceGPMFC over IGPMFC

In summary, in all cases, IGPMFC can dramatically reduce the complexity of the classifiers by using original features. Furthermore, IGPMFC can better reduce the complexity of the classifiers than GPMFC with both simple and sophisticated imputations.

Table 5: Computation time of different methods for constructing multiple features (millisecond).

Dataset	IGPMFC	MeGPMFC	KNNGPMFC	MiGPMFC
Bands	5.3×10^{-6}	1.3×10^{-6}	1.7×10^{-1}	1.6×10^{4}
Breast	2.7×10^{-6}	9.1×10^{-7}	8.1×10^{-1}	2.3×10^{2}
Hepatitis	1.2×10^{-5}	1.8×10^{-6}	3.8×10^{-1}	1.5×10^{4}
Balance	5.3×10^{-6}	2.4×10^{-6}	7.2×10^{-2}	2.4×10^{3}
Iris	6.6×10^{-6}	2.2×10^{-6}	4.6×10^{-2}	3.3×10^{3}
Liver	2.3×10^{-6}	8.5×10^{-7}	1.9×10^{0}	1.1×10^{3}

5.3 Computation Time

Table 5 shows the average computation time of IGPMFC and the other methods for constructing new features in the testing process.

It is clear from Table 5 that the combination of GPMFC with mean imputation is the fastest method and following GPMFC with interval functions, KNN-based imputation and MICE imputation. Especially, GPMFC with MICE imputation is million times slower than the other methods. The main reason is that MICE requires rebuilding the regression functions using all the training data and the new instance each time when it needs to estimate missing values in a new instance.

In summary, IGPMFC is as quick as other simple imputations combined with GPMFC and can achieve better accuracy than the other methods.

6 CONCLUSIONS

This paper proposed IGPMFC which is a GP-based feature construction for classification with incomplete data. IGPMFC is extended from GPMFC which is a recent promising feature construction method, but it cannot directly work with incomplete data. IGPMFC uses interval functions as the GP function set to tackle with missing values by replacing each missing feature value by the feature interval. Three experimental setups are designed to evaluate the impact of IGPMFC on classification with incomplete data: classification with incomplete data by using classifiers able to deal with incomplete data; classification with incomplete data by using imputation methods combined with GPMFC before using classifiers, and classification with incomplete data by using IGPMFC to construct new features from incomplete data before using classifiers. Experimental results show that IGPMFC can achieve better accuracy than using original features or combining GPMFC with simple imputation methods. Moreover, the accuracy of IGPMFC is comparable with combining GPMFC with expensive imputation methods. Furthermore, IGPMFC can better reduce the complexity of learnt classifiers than all the other methods.

The majority of other GP-based feature construction methods cannot deal with incomplete data. Hence, future work could integrate interval functions into the methods to deal with incomplete data.

REFERENCES

[1] A. Asuncion and D. Newman. UCI machine learning repository, 2007.
[2] J. O. Berger. *Statistical decision theory and Bayesian analysis.* Springer Science & Business Media, 2013.
[3] A. Bifet, G. Holmes, B. Pfahringer, and E. Frank. Fast perceptron decision tree learning from evolving data streams. In *Advances in knowledge discovery and data mining*, pages 299–310. 2010.
[4] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and regression trees.* CRC press, 1984.
[5] S. Buuren and K. Groothuis-Oudshoorn. mice: Multivariate imputation by chained equations in R. *Journal of statistical software*, 45, 2011.
[6] P. G. Espejo, S. Ventura, and F. Herrera. A survey on the application of genetic programming to classification. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 40:121–144, 2010.
[7] A. Farhangfar, L. Kurgan, and J. Dy. Impact of imputation of missing values on classification error for discrete data. *Pattern Recognition*, 41:3692–3705, 2008.
[8] P. J. García-Laencina, J.-L. Sancho-Gómez, and A. R. Figueiras-Vidal. Pattern classification with missing data: a review. *Neural Computing and Applications*, 19:263–282, 2010.
[9] J. W. Graham. Missing data analysis: Making it work in the real world. *Annual review of psychology*, 60:549–576, 2009.
[10] H. Guo, Q. Zhang, and A. K. Nandi. Feature extraction and dimensionality reduction by genetic programming based on the fisher criterion. *Expert Systems*, 25:444–459, 2008.
[11] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11:10–18, 2009.
[12] M. A. Hall. *Correlation-based feature selection for machine learning.* PhD thesis, The University of Waikato, 1999.
[13] J. Han, M. Kamber, and J. Pei. *Data mining: concepts and techniques: concepts and techniques.* Elsevier, 2011.
[14] E. Hansen and G. W. Walster. *Global optimization using interval analysis: revised and expanded*, volume 264. CRC Press, 2003.
[15] J. R. Koza. *Genetic programming: on the programming of computers by means of natural selection*, volume 1. 1992.
[16] Y. Lin and B. Bhanu. Evolutionary feature synthesis for object recognition. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 35:156–171, 2005.
[17] R. J. Little and D. B. Rubin. *Statistical analysis with missing data.* John Wiley & Sons, 2014.
[18] S. Luke, L. Panait, G. Balan, S. Paus, Z. Skolicki, E. Popovici, K. Sullivan, J. Harrison, J. Bassett, R. Hubley, et al. A java-based evolutionary computation research system. *Online (March 2004) http://cs. gmu. edu/~ eclab/projects/ecj*, 2004.
[19] M. Muharram and G. D. Smith. Evolutionary constructive induction. *Knowledge and Data Engineering, IEEE Transactions on*, 17:1518–1528, 2005.
[20] D. R. Musser. Introspective sorting and selection algorithms. *Softw., Pract. Exper.*, 27:983–993, 1997.
[21] K. Neshatian, M. Zhang, and P. Andreae. A filter approach to multiple feature construction for symbolic learning classifiers using genetic programming. *Evolutionary Computation, IEEE Transactions on*, 16:645–661, 2012.
[22] J. R. Quinlan. *C4. 5: programs for machine learning.* Elsevier, 2014.
[23] H. Shi. Best-first decision tree learning. Master's thesis, University of Waikato, Hamilton, NZ, 2007. COMP594.
[24] M. G. Smith and L. Bull. Genetic programming with a genetic algorithm for feature construction and selection. *Genetic Programming and Evolvable Machines*, 6:265–281, 2005.
[25] A. Srinivasan and R. D. King. Feature construction with inductive logic programming: A study of quantitative predictions of biological activity aided by structural attributes. *Data Mining and Knowledge Discovery*, 3:37–57, 1999.
[26] X. Tan, B. Bhanu, and Y. Lin. Fingerprint classification based on learned features. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 35:287–300, 2005.
[27] C. T. Tran, P. Andreae, and M. Zhang. Impact of imputation of missing values on genetic programming based multiple feature construction for classification. In *Evolutionary Computation (CEC), 2015 IEEE Congress on*, pages 2398–2405, 2015.
[28] C. T. Tran, M. Zhang, and P. Andreae. Multiple imputation for missing data using genetic programming. In *Proceedings of the 2015 annual conference on genetic and evolutionary computation*, pages 583–590, 2015.
[29] C. T. Tran, M. Zhang, and P. Andreae. Directly evolving classifiers for missing data using genetic programming. In *Evolutionary Computation (CEC), 2016 IEEE Congress on*, pages 5278–5285, 2016.
[30] C. T. Tran, M. Zhang, and P. Andreae. A genetic programming-based imputation method for classification with missing data. In *European Conference on Genetic Programming*, pages 149–163, 2016.
[31] C. T. Tran, M. Zhang, P. Andreae, and B. Xue. Directly constructing multiple features for classification with missing data using genetic programming with interval functions. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion*, pages 69–70, 2016.
[32] I. R. White, P. Royston, and A. M. Wood. Multiple imputation using chained equations: Issues and guidance for practice. *Statistics in medicine*, 30:377–399, 2011.