2013 IEEE Congress on Evolutionary Computation
June 20-23, Cancún, México

Hybridisation of Genetic Programming and Nearest Neighbour for Classification

Harith Al-Sahaf
School of Engineering and CS
Victoria Uni. of Wellington
New Zealand
harith.al-sahaf@ecs.vuw.ac.nz

Andy Song
School of CS and IT
RMIT University
Australia
andy.song@rmit.edu.au

Mengjie Zhang
School of Engineering and CS
Victoria Uni. of Wellington
New Zealand
mengjie.zhang@ecs.vuw.ac.nz

***Abstract*—In this paper, we propose a novel hybrid classification method which is based on two distinct approaches, namely Genetic Programming (GP) and Nearest Neighbour (kNN). The method relies on a memory list which contains some correctly labelled instances and is formed by classifiers evolved by GP. The class label of a new instance will be determined by combining its most similar instances in the memory list and the output of GP classifier on this instance. The results show that this proposed method can outperform conventional GP-based classification approach. Compared with conventional classification methods such as Naïve Bayes, SVM, Decision Trees, and conventional kNN, this method can also achieve better or comparable accuracies on a set of binary problems. The evaluation cost of this hybrid method is much lower than that of conventional kNN.**

I. Introduction

Classification is of great importance both practically and theoretically. Numerous methods have been proposed in the last few decades. One of the most popular algorithms is Nearest Neighbour (kNN) which is simple in nature. Unlike many peer methods, it does not have a training phase to build classifiers. Instead, it directly takes an unseen instance to compare with existing labelled instances to determine the class [12], [23]. The hypothesis of kNN often works well. That is, the class of an instance is likely to be identical to that of its surrounding instances (or its k number of neighbours), "Birds of same feature flock together". However, the computational cost of this algorithm in application is rather high, because each new instance to be classified requires a process of measuring distances with the entire set of training examples. Although there are works addressing this issue, kNN remains a computationally expensive method [11], [15], [25].

On the other hand, evolutionary methods such as Genetic Programming (GP) are much more time consuming and more complicated in term of the training process. However the classifiers generated by such process often have high accuracy and fast in execution [7], [14], [16], [22], as the classification of new instances is in isolation of the training examples and the evolved classifiers are often of low complexity (usually no loops are involved hence the complexity is somewhat linear). Therefore we hypothesize that the combination of kNN and GP can enhance the performance of GP without inheriting the high cost problems of kNN. A hybrid method is therefore proposed in this paper to take advantages of both methods. At this stage, the main target is binary classification where only two classes exist. Upon this investigation, a multi-class method can be built either through binary decomposition or direct separation of multiple classes.

The basic principle of this hybrid method is still to evolve a classifier by GP. The classifier cumulatively builds up a knowledge base which we called it the *memory-list*. The memory list contains some of the correctly classified examples. Based on these examples, new instances can then be classified. So previous success does have impact on subsequent classification. This is very different to the conventional GP classification approach, where an instance is labelled independently from other cases. We expect this kind of combination of evolved GP classifiers and the kNN approach can result in an improved classification process. More specially we aim to answer the following questions in this study:

1) What is the suitable methodology of combining GP and kNN so the hybridisation can inherit the advantages of both approaches?
2) How well is this method comparing to the conventional GP classification in some typical binary classification problems?
3) How well is this method comparing to those widely used classification methods including Naïve Bayes, SVM, J48 decision trees, and kNN itself?

The rest of this paper is organized in the following structure. Section II gives a brief discussion on the background of this work and the related existing studies. Section III presents the proposed hybrid method in detail. Section IV describes the settings of our experiments and the five data sets used in this study. The corresponding results are presented and discussed in Section V. The conclusions and future works are presented in Section VI.

II. Background

Genetic Programming (GP) is a well-known member of Evolutionary Computing (EC) paradigm. GP have been extensively adopted to solve variety of complex problems. Classification is certainly one of focuses in this field since GP trees can represent discrimination functions [4], [29]. In particular handling binary classification is convenient using GP approaches, because the single output of an evolved program tree can be directly used to indicate the class label, either a positive case or a negative case [28]. The studies of GP on classification are extended to general classification or domain specific classification such as image classification, event classification from time series patterns and object classification from video streams [24].

In the field of classification, k-Nearest Neighbour (kNN) algorithm is one of the most popular method. This algorithm has been widely used due to implementation simplicity and multi-class classification nature [1] [19], [20]. However, conventional kNN algorithm has three drawbacks [13], [26]:

1) High computation complexity;
2) Equality of training instances; and
3) Heavily dependent on training instances.

For the first drawback (in terms of computation complexity), the algorithm calculates the distance between the instance being evaluated and each instance of the training set. The training instances then needs to be sorted in order of distance in a multi-dimensional space, then the class label is assigned based on the class label of the majority of the k nearest neighbours of the given instance. Hence, more time is required to perform the classification when the collection of data is large, and the cost is proportional to the number of instances of the training set and the number of attributes existing per instance.

For the second drawback of kNN algorithm, it is obvious that conventional kNN algorithm treats all training instances equally. It does not have a mechanism to value instances which may have more information for discriminating different classes. The only consideration for classification is the distance which in many cases is not the sole factor or not that reliable by itself.

The third drawback of conventional kNN algorithm is that it relies heavily on existing training instances and does not use or add newly arrived data. To avoid ignoring fresh instances which may be more relevant than obsolete instances, the algorithm has to recalculation every time whenever there is a change in the training data.

In order to achieve better performance, many combinations of different methods, including combinations of kNN and GP, have been proposed in the literature for various tasks. In [5] GP and kNN were combined for image classification. Their method utilizes GP to evolve some good functions from a set of global descriptors. The learnt function of each class of the data set was used as a kNN classifier. Then the voting approach is adopted to obtain the final classification result. In [27] GP was combined with kNN to solve multi-class classification problem. In this work, GP was used to generate features. Then kNN was used as a classifier on these generated features. Such combination resulted in good performance. However it is a two-stage approach where GP and kNN are independent from each other and responsible for different tasks. The work presented in [3] also combined GP and kNN for classification. It is similar to the work in [27] where a two-stage approach is adopted. GP is for generating new feature values and kNN is for classification. However at each generation of the evolutionary process, features generated by GP will be fed into kNN to assist the fitness evaluation. So these two methods intertwined coherently here.

Guo *et al.* [9] proposed a hybrid method that combines GP and kNN methods to perform automatic feature extraction. The proposed method utilizes multi-objective search to reduce the number of input features as a mean of dimensionality reduction. The performance of these evolved classifiers can be enhanced because of these newly selected or extracted features. Their method comprises three stages: 1) using discrete wavelet transform to transform the raw data values into a more usable form for a GP-based system; 2) applying GP and kNN methods to evolve a new set of features based on these features generated in the previous step; 3) verifying the performance of extracted features from Step 2 on classification by a kNN classifier. Two epileptic EEG detection problems were used for evaluation. The proposed method had been compared against the conventional kNN and achieved significantly higher accuracy by this complex combination of GP and kNN.

Suguna and Thanushkodi combined kNN and Genetic Algorithm (GA) in order to improve the classification accuracy and to overcome some of the limitations of kNN [21]. A multi-stage approach was adopted in their work, that is 1) reducing the number of original features by combining Bee Colony Optimization (BCO) and Rough Set; 2) evolving a solution by GP to select k-neighbours from the training set; and 3) calculating the distance between each of the sample point selected in Step 2 and the test instance, then assigning the majority class label to that instance as its class. Five different medical datasets were used for evaluation. This method significantly outperformed traditional methods including kNN, SVM and CART on all of the five datasets.

Majid *et al.* proposed a modified Nearest Neighborhood (ModNN) method which aims to use GP techniques to improve classification performance [18]. ModNN uses a voting approach to determine a good value of k. Then GP is used to find a better class mapping function that can effectively reduce outliers. Similar to that work in [9], the proposed method was compared against traditional kNN. The results show that ModNN can achieve better performance. Moreover, this study has shown another advantage of ModNN over traditional kNN, that is, its low false alarm rate.

Although the above approaches utilized kNN with GP/GA and achieved good performance, in some degree they all treated evolved individuals and kNN as separate components, responsible for different aspects of the classification task. On the contrary, what we propose is a way to allow kNN and GP individuals to participate in the same class labelling process, without multiple stages.

[1]The number of classes does not impact the distance measure process in kNN.

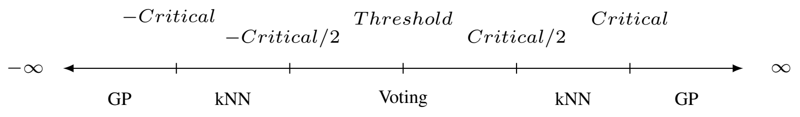

Fig. 1. Classification Based on the Output Value of a GP Classifier

III. Methodology

The methodology of our hybrid approach is presented below. The main part is the use of kNN and voting to compensate adversarial effect of difficult training examples. The rest of this section shows the GP representation for classification including the terminal set, the function set and the fitness.

A. Mapping GP Output to a Class Label

To establish the hybridisation of GP and kNN, firstly a list is created for each individual. It is called *memory-list*, as it "memorizes" some of the correctly classified instances of this individual. Instead of using a threshold value to separate between classes (e.g the positive class being output greater than the threshold and the negative class being smaller than the threshold), we propose the use of a set of other points to assist the classification. As shown in Figure 1, other than the $Threshold$ point, they are points of $Critical$, $Critical/2$, $-Critical/2$ and $-Critical$ along the output axis. In this figure, let us assume the area left to $Threshold$ denotes the negative class and the area on the righthand side is the positive.

In order to classify an instance, each individual GP classifier takes the attribute values of that instance as inputs and returns a value. The absolute difference between that value and the *threshold* is then used to determine the class label according to the following three scenarios.

In the first scenario, the difference is greater than a predefined *Critical* value, as

$$\mid Threshold - GP\ Output \mid > Critical$$

Then the instance is classified as "positive" if the returned value is greater than the threshold. Otherwise it is considered "negative". In other words, values outside $[-Critical, Critical]$ are used similar to that in normal GP classification. Correctly classified training examples will be added to the *memory-list* of that individual to be used later.

The second scenario is that the absolute difference between the output and the *Threshold* is smaller than *Critical* but greater than $Critical/2$. This means the output from this instance falls into the area of $[-Critical, -Critical/2]$ or $[Critical/2, Critical]$ in Figure 1. The possibility of misclassification in these areas should be higher than that in the first scenario. In this case the classification would rely on the instances stored in the *memory-list* of a GP individual. The kNN method is then used to give the class label. Similar to that in the first scenario, if the instance is from training data and it is corrected classified then it will be added to the *memory-list*.

There are two major differences between conventional kNN and the kNN used here. The first is that all attributes will participate in the distance measure in conventional kNN. However the distance measure here is only based on attributes presented on the GP tree. So that is a subset of attributes because GP often select a small proportion of attributes for classification. This reduction in dimensionality of attributes can result in less computation and more meaningful distance calculation because it is not affected by unselected attributes. For the very reason, each GP individual has its own memory-list which cannot be shared among individuals since different individuals would select different subset of attributes. The second difference is that only a subset of "good" instances are used here while conventional kNN uses the entire set. This difference can further reduce the computation cost and reduce the bias caused by misleading examples.

The third scenario is that the absolute difference between the returned value and the threshold is within the range between $[-Critical/2, Critical/2]$, the middle part in Figure 1. The possibility of misclassification is even higher here. Hence we introduce a *majority voting* mechanism to enhance the classification. The voting involves the class label based on comparing the GP output with the threshold and the class labels from kNN with different k values. The instance is then labeled as the class which wins the majority votes. Unlike two other scenarios, correctly classified instances in this scenario will not be added into the *memory-list* as they are considered as "weak" or "indecisive" instances.

There is no other significant difference between this approach and normal GP classification. The only distinction is how an output value from individual is mapped to a class label if the value is in the range of $[-Critical, Critical]$. The classification decision is a collaborative effort by GP and kNN. So this method is named as `GPkNN`. Figure 2 shows a flowchart on how the classification process is carried out in GPkNN.

B. Function Set

The function set is made of the four standard arithmetic operators:

$$Function\ Set = \{+,\ -,\ \times,\ /\}$$

They are simply addition, subtraction, multiplication and protected division proving the return value when the denominator is 0. Each of these operators takes two parameters which can be terminals or functions.

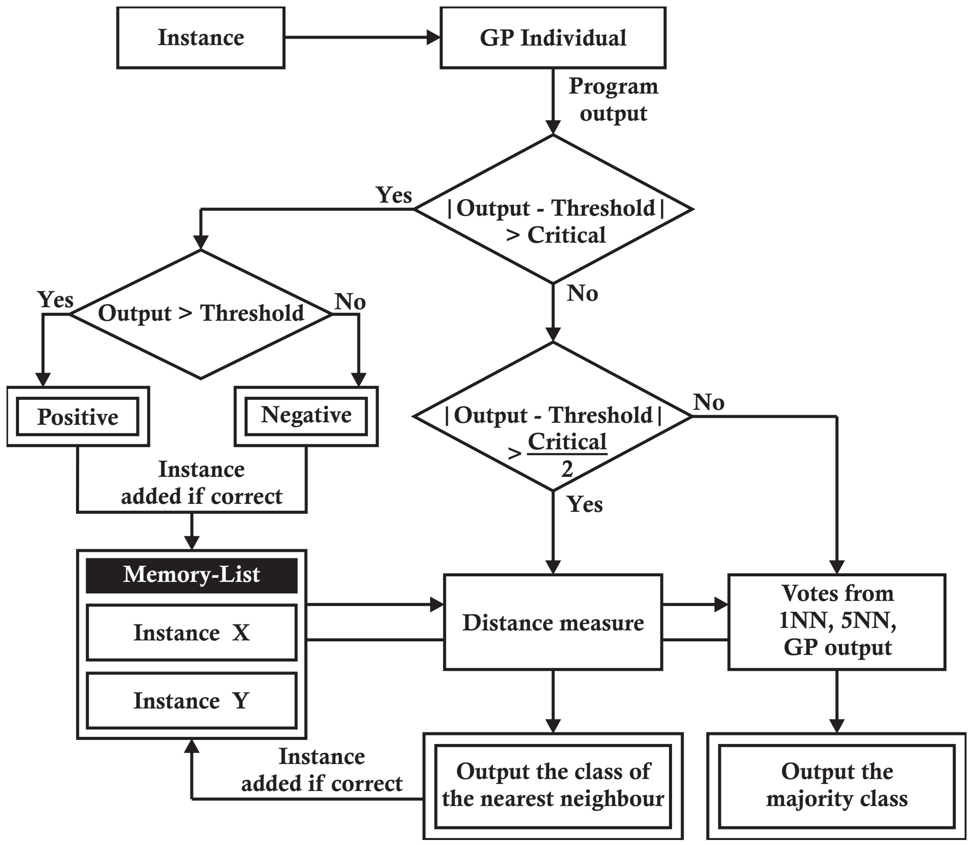
Instance
GP Individual
Program output
|Output - Threshold| > Critical
Yes
No
Output > Threshold
Yes
No
Positive
Negative
Instance added if correct
|Output - Threshold| > Critical/2
No
Yes
Memory-List
Instance X
Instance Y
Distance measure
Votes from 1NN, 5NN, GP output
Instance added if correct
Output the class of the nearest neighbour
Output the majority class

Fig. 2. The Flowchart of GPkNN: how to combine GP output with nearest neighbours

C. Terminal Set

In classification each data set is often made of an $N \times M$ matrix, where N represents the number of instances in the data set such that $\mathcal{D} = (I_1, I_2, \ldots, I_N)$. Each instance consists of M attributes such that $I_i = (r_{(i,1)}, r_{(i,2)}, \ldots, r_{(i,M)})$. So the terminals should be able to read in these attribute values as classifier inputs. In addition randomly generated constants should be provided to classifiers as coefficients. So the terminal set is:

$$Terminal\ Set = \{Attribute_{[m]},\ Constant\}$$

These constants are randomly generated floating-point values in the range of [-10, 10]. The $Attribute$ node has an index m, which is also randomly generated but in the range of [1, M], where M is the number of attributes available in the data set.

D. Fitness Measure

In the case of binary classification the fitness for evolving a classifier can be simply expressed in Equation 1.

$$F(x) = accuracy(x) = \frac{\text{TP + TN}}{\text{TP + FP + TN + FN}} \tag{1}$$

Where TP and TN represent the number of instances been correctly labelled as positive and negative respectively. While FN and FP are False Negatives and False Positives respectively. So the denominator of Equation 1 is the total number of instances, and the fitness is simply the classification accuracy.

IV. Experiments

The hybrid classification method was implemented based on the Evolutionary Computation Java-based package (ECJ) [17]. To compare the performance of our methods, different classifiers including Naïve Bayes, Support Vector Machine (SVM), Decision Trees (J48), and kNN implemented in Weka [10] were evaluated against a same set of data.

A. Data Sets

Five different data sets have been used in this study to measure the performance of the proposed method. The data sets are varying in number of instances and number of attributes as detailed below. However, these data sets are for binary classification where each instance is belonging to one of two categories. Except Leukemia [8] and Colon Cancer [2] data sets, all other data sets have been taken from the *UCI Machine Learning Repository* [6]. No big data set was used as the disadvantage of kNN would be more obvious on them since the number of instances directly affect the computational cost in kNN. Leukemia and Colon Cancer data sets were selected

TABLE I
Details of the Five Datasets Used in Experiments

	Training Set			Test Set			Overall	#Attributes
	Positive	*Negative*	Total	*Positive*	*Negative*	Total		
PIMA	250	134	384	250	134	384	768	8
Heart	73	62	135	77	58	135	270	13
WDBC	178	106	284	179	106	285	569	30
Colon	19	12	31	21	10	31	62	2000
Leukemia	27	11	38	20	14	34	72	7129

because of their large number of attributes. So the effect of GPkNN utilising GP's attribute selection may be observed.

The first data set is Pima Indians Diabetes Database. This data set is made of 768 instances, where each has 8 feature values. The data set has 500 instances of patients have tested negative for diabetes and 268 instances for patients have tested positive. The total number of instances of the two classes was equally divided between training and test sets that each consists of 250 and 134 instances of negative and positive respectively.

The Statlog Heart data set was the second data set that was used to evaluate the performance of GPkNN. This data set represents a slightly modified version of the original *Heart Disease databases*, where only 13 features have been extracted out of 75. In total, this data set has 270 instances that fall into either *absence* or *presence* of heart disease. Worth to mention, that the number of instances of each class was nearly equally divided into training and test samples.

Wisconsin Diagnostic Breast Cancer (WDBC) represents the third data set we used in this study. This data set consist of 569 instances in total where each of them represents either *malignant* or *benign* case. The number of instances of benign and malignant cases is 357 and 212 respectively that was equally split between training and test sets. Each instance is made of 30 real-valued input features, where 10 of them represent the different computations of the cell nucleus.

The fourth data set is Colon Cancer that overall has 62 gene expression patterns of colon instances that fall into two categories: 1) *tumours*; and 2) *normal* tissues. The number of samples that has tumour as class label is 40 and the other 22 samples has normal as class label. Furthermore, each instance of this data set consists of 2000 real-valued features.

The fifth and last data set is Leukemia which consists of 72 samples. The data set indicates two types of leukemia cancer: 1) *Acute myeloid leukemia* (AML); and 2) *Acute lymphoblastic leukemia* (ALL), where 25 samples have been labelled as AML, and 47 samples were labelled as ALL. Each sample of this data set has 7129 features that each represents an expression level of a gene.

Table I lists the number of instances, either positive or negative, in the training and test sets for each data. There is no severe imbalance in these data sets. Handling unbalanced data is beyond the scope of this paper, but will be investigated in future work.

B. Experimental Settings

Since the proposed method combines GP and kNN methods, parameters for both aspects need to be supplied. As mentioned in methodology, the kNN method is used for two of three scenarios when mapping the single program output to class labels, the k value is required. In the second scenario, k is set as 1, e.g. simply the nearest neighbour. In the third scenario, where voting is used, we used two kNN methods in the voting committee. Their k values are 1 and 5 respectively.

As for the GP parameters, we set *crossover*, *mutation* and *reproduction* rates to 0.8, 0.19 and 0.01 respectively. Standard crossover and mutation operators are used here. The reproduction is simply moving the selected individuals to the next generation. The size of the generated trees was set to minimum depth of 2 levels (including the root node) and maximum of depth 10 to allow variety range of solutions. Population size is set to 2000 individuals in order to alleviate the problem of early convergence. The maximum number of generation is set to 20, because convergence is usually reached before that .

Table II summarises these experimental parameters. Note the goal of this study is not to find the optimal setting for classification. The choice of this set of parameters is empirical but consistent with that in many studies on GP classification.

In order to obtain reliable performance measure, every experiment involving GP was repeated 30 times using different random seeds. However, the exact seed value that was used in a normal GP run was also used in the hybrid counterpart. This is to ensure an identical start point in both cases as seed values may significantly affect the final results.

V. Results

Two series of experiments were conducted in this study. The focus of the first series of experiments was on comparing the accuracies of normal GP classification and the hybrid method (GPkNN). That of the second series was on comparing the accuracies of GPkNN with traditional methods. Additionally their execution time were evaluated.

A. GPkNN vs GP

Table III shows the gathered experimental results from $30 \times 2 \times 5 = 300$ runs in total of both GP-based methods (GP and GPkNN). This table consists of five blocks from the top to the bottom, each presenting the results for one of

TABLE II
SUMMARY OF PARAMETER VALUES

Parameter	Value
k-Nearest Neighbour	
K1	1
K2	5
Genetic Programming	
Generations	20
Population Size	2000
Crossover Rate	0.80
Mutation Rate	0.19
Elitism Rate	0.01
Tree Depth	2-10
Selection Type	Tournament
Tournament Size	7
Builder Type	Ramped HALF-AND-HALF
Other parameters	
Threshold	0
Critical value	10

the five data sets. Both training and test accuracies are listed in a form of minimum (worst), maximum (best), mean and standard deviation. They are from the 30 runs for each task. For comparison (t-test) was perform on each pair of results from the baseline GP and the hybrid GPkNN. The significance is indicated by asterisks[2].

For all data sets, except colon cancer, the results show significant increase in training accuracies comparing GPkNN with standard GP. However in the case of colon cancer data, GPkNN actually showed significant difference on test. The superior performance of GPkNN compared to GP is consistent in test on all five data sets. The performance differences between the two methods are considered highly significant in the case of PIMA, Heart and WDBC data, and still statistically significant in the case of colon cancer and leukemia data.

The performance evaluation method presented in this paper is the two-fold training and test approach. Another common way is the n-fold cross validation which iteratively trains on $n - 1$ folds of data and tests on the remaining fold until all folds have been involved in both training and test. We performed 10-fold cross-validation on the same data. There was no statistically significant differences between these two evaluation approaches.

B. GPkNN vs Traditional Methods

The second set of experiments compare our GPkNN with conventional classification methods such as Naïve Bayes, SVM, J48 decision trees, and kNN with $k = 1$ and $k = 5$. Table IV presents this comparison on test accuracies. The programs achieved the best test accuracy on each data set from the above experiments were used. They actually are more accurate than their counterparts on every data set. It is true that GPkNN appears average, if we compare not the best but the mean test accuracies. Due to the stochatic nature of GP, the performance of evolved solutions fluctuate significantly. So the mean accuracy does not reflect the potential of GPkNN which could outperform these conventional methods or at least be comparable to them.

[2]$*$ Means GPkNN significantly better than GP, with a confidence level 95%
$**$ Means GPkNN is even better, with a confidence level 99%

C. Comparisons on Execution Speed

To verify the benefit of GPkNN in terms of computational cost, three methods were used to evaluate the five data sets on a machine with a 4-core 2.2 GHz Intel i7-2670QM CPU, running Ubuntu 4.6 and Java 6. These three methods are the conventional kNN ($k = 1$), the conventional GP (the best individuals evolved from experiments in Section V-A) and GPkNN (also the best from Section V-A). The process was repeated 30 times on each data set. The execution time are presented in Table V which include the minimum, the maximum, the mean and the standard deviation of each 30 runs. A“0” in the table means the time is less than recording precision. Clearly kNN took much longer than other two methods on every data set. GPkNN was slower than programs evolved by conventional GP. That is expected as there is no distance calculation in conventional GP. However the difference between GPkNN and GP is not that significant. For example they are almost the same on Leukemia while kNN took more than 250 times longer. Because Leukemia has 7129 attributes, the distance measure of kNN is very expensive. Both GPkNN and GP only use a handful of attributes. So their execution is very fast. GPkNN is more sensitive to the number of instances though. It took much longer on PIMA and WDBC which have the highest number of instances among the five sets (See Table I).

VI. CONCLUSIONS

This paper proposes a hybrid method of GP and kNN for binary classification. It uses a *memory-list* to store some good instances that have been correctly classified and then applies

TABLE III
COMPARISON BETWEEN CONVENTIONAL GP AND THE HYBRID METHOD GPKNN

		Training (%)			Test (%)		
		Min	Max	Mean±St.dev	Min	Max	Mean±St.dev
PIMA	GP	68.23	75.52	70.56 ±1.87	65.10	78.91	71.07 ±4.11
	GPkNN	71.61	76.30	73.51 ±1.12 **	70.31	79.95	76.45 ±2.21 **
Heart	GP	79.26	87.41	82.76 ±2.03	75.56	85.19	80.44 ±2.72
	GPkNN	81.48	88.15	85.24 ±1.42 **	76.30	85.44	82.24 ±2.16 **
WDBC	GP	89.82	96.49	93.22 ±1.75	84.15	95.77	91.51 ±3.01
	GPkNN	95.09	96.49	95.98 ±0.35 **	92.96	96.83	95.05 ±1.01 **
Colon	GP	87.10	100.0	95.27 ±3.86	67.74	87.10	75.59 ±5.06
	GPkNN	87.10	100.0	96.34 ±3.47	70.52	87.12	78.10 ±3.25 *
Leukemia	GP	89.47	100.0	95.97 ±3.64	50.00	91.18	72.80 ±5.86
	GPkNN	97.37	100.0	99.74 ±0.80 **	60.24	94.12	76.41 ±5.31 *

TABLE IV
COMPARISON BETWEEN CONVENTIONAL CLASSIFIERS AND THE HYBRID GPKNN

	GPkNN	Naïve Bayes	SVM	DT(J48)	kNN (k=1)	kNN (k=5)
PIMA	**79.95**	77.60	79.69	78.13	71.88	74.48
Heart	**85.44**	84.44	84.44	80.74	74.82	79.26
WDBC	**96.83**	95.78	96.48	91.20	95.42	96.13
Colon	**87.12**	70.97	87.01	77.42	80.65	77.42
Leukemia	**94.12**	85.29	88.24	88.24	70.58	70.59

TABLE V
COMPARISON OF EXECUTION TIME ON FIVE DATA SETS (IN MILLISECONDS)

	Conventional kNN			Conventional GP			GPkNN		
	Min	Max	Mean±Stdev	Min	Max	Mean±Stdev	Min	Max	Mean ±Stdev
PIMA	51	170	56.83 ±21.43	0	1	0.20 ±0.41	2	24	10.53 ±5.77
Heart	8	75	12.53 ±13.12	0	1	0.13 ±0.35	0	5	1.6 ±1.63
WDBC	71	227	79.00 ±28.04	0	1	0.40 ±0.50	0	42	15.8 ±7.08
Colon	51	67	53.13 ±3.08	0	1	0.03 ±0.18	0	1	0.20 ±0.41
Leukemia	251	274	255.23 ±5.48	0	1	0.04 ±0.20	0	1	0.27 ±0.45

kNN method on these stored instances to classify unseen instances. The output from a GP classifier and a predefined $Critical$ value will determine how to map the program return value into a class label.

The proposed method has been evaluated using five different data sets that vary in number of instances and number of attributes per instance. The results show that the proposed hybrid method can significantly outperform GP without this hybridised mechanism. Furthermore, the performance of the best evolved program has been compared to other widely used classifiers such as Naïve Bayes, SVM, Decision Trees, and kNN. This hybrid GP method can be better than or at least comparable to those classifiers. In addition, this method's computational cost on evaluating a data set is not vastly more than that of a normal GP classifier, but significantly lower than that of kNN method. In particular the large number of attributes has little impact on its evaluation cost because the GPkNN method only select a small subset of attributes to participate in the classification process.

In conclusion, the proposed method for combining GP and kNN is feasible. It can achieve good classification performance without causing high computational cost.

VII. FUTURE WORKS

Due to the limited scope of this study, the work presented here can not address all concerns and questions. In the near future we will continue to investigate and extend this hybrid methodology, for example what is the optimal 'Critical Value" and how to set this value dynamically to fit different problems; could it be possible that the selected "Critical" value divides one cluster of data points, hence be counterproductive.

Furthermore we will also extend this method to handle multi-class problems. In addition we will systematically investigate the impact of parameters like k and $Critical$ although they did not effect the results much in the above experiments. Issues like over-fitting and class imbalance will also be studied. Another extension is to combine GPkNN with Two-Tier GP [1] for image classification.

REFERENCES

[1] H. Al-Sahaf, A. Song, K. Neshatian, and M. Zhang. Two-tier genetic programming: towards raw pixel-based image classification. *Expert Systems with Applications*, 39(16):12291 – 12301, 2012.

[2] U. Alon, N. Barkai, D. A. Notterman, K. Gishdagger, S. Ybarradagger, D. Mackdagger, and A. J. Levine. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proceedings of the National Academy of Sciences of the United States of America*, 96(12):6745–6750, June 1999.

[3] M. W. Aslam, Z. Zhu, and A. K. Nandi. Automatic modulation classification using combination of genetic programming and knn. *IEEE Transactions on Wireless Communications*, 11(8):2742–2750, 2012.

[4] P. G. Espejo, S. Ventura, and F. Herrera. A survey on the application of genetic programming to classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 40(2):121–144, 2010.

[5] F. A. Faria, R. T. Calumby, and R. da Silva Torres. Recod at imageclef 2011: Medical modality classification using genetic programming. In V. Petras, P. Forner, and P. D. Clough, editors, *CLEF (Notebook Papers/Labs/Workshop)*, 2011.

[6] A. Frank and A. Asuncion. UCI machine learning repository, 2010.

[7] A. Friedlander, K. Neshatian, and M. Zhang. Meta-learning and feature ranking using genetic programming for classification: Variable terminal weighting. In *Evolutionary Computation (CEC), 2011 IEEE Congress on*, pages 941 –948, june 2011.

[8] T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, and C. D. Bloomfield. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, 286:531–537, 1999.

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

[10] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: an update. *SIGKDD Explor. Newsl.*, 11(1):10–18, Nov. 2009.

[11] W.-J. Hwang and K.-W. Wen. Fast knn classification algorithm based on partial distance search. *Electronics Letters*, 34(21):2062–2063, oct 1998.

[12] N. Ishii, T. Murai, T. Yamada, and Y. Bao. Text classification by combining grouping, lsa and knn. In *Proceedings of the 5th IEEE/ACIS International Conference on Computer and Information Science and 1st IEEE/ACIS International Workshop on Component-Based Software Engineering,Software Architecture and Reuse*, ICIS-COMSAR '06, pages 148–154, Washington, DC, USA, 2006. IEEE Computer Society.

[13] S. Jiang, G. Pang, M. Wu, and L. Kuang. An improved k-nearest-neighbor algorithm for text categorization. *Expert Syst. Appl.*, 39(1):1503–1509, Jan. 2012.

[14] K. Krawiec, D. Howard, and M. Zhang. Overview of object detection and image analysis by means of genetic programming techniques. In *Frontiers in the Convergence of Bioscience and Information Technologies, 2007. FBIT 2007*, pages 779 –784, oct. 2007.

[15] Y. Li and B. Cheng. An improved k-nearest neighbor algorithm and its application to high resolution remote sensing image classification. In *Geoinformatics, 2009 17th International Conference on*, pages 1 –4, aug. 2009.

[16] T. Liddle, M. Johnston, and M. Zhang. Multi-objective genetic programming for object detection. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1 –8, july 2010.

[17] S. Luke, L. Panait, G. Balan, S. Paus, Z. Skolicki, E. Popovici, K. Sullivan, J. Harrison, J. Bassett, R. Hubley, A. Chircop, J. Compton, W. Haddon, S. Donnelly, B. Jamil, J. Zelibor, E. Kangas, F. Abidi, H. Mooers, and J. O'Beirne. Ecj: A java-based evolutionary computation research system, March 2010. [Online]. Available: http://cs.gmu.edu/~eclab/projects/ecj/.

[18] A. Majid, A. Khan, and A. Mirza. Improving performance of nearest neighborhood classifier using genetic programming. In *Machine Learning and Applications, 2004. Proceedings. 2004 International Conference on*, pages 469–476, December.

[19] C. D. Martínez-Hinarejos, A. Juan, and F. Casacuberta. Median strings for k-nearest neighbour classification. *Pattern Recognition Letters*, 24(1-3):173–181, 2003.

[20] N. A. Samsudin and A. P. Bradley. Nearest neighbour group-based classification. *Pattern Recognition*, 43(10):3458–3467, 2010.

[21] N. Suguna and K. Thanushkodi. An improved k-nearest neighbor classification using genetic algorithm. *International Journal of Computer Science Issues (IJCSI)*, 7(2):18–21, 7 2010.

[22] A. Tamboli and M. Shah. A generic structure of object classification using genetic programming. In *Communication Systems and Network Technologies (CSNT), 2011 International Conference on*, pages 723 – 728, june 2011.

[23] H. A. Vrooman, C. A. Cocosco, R. Stokking, M. A. Ikram, M. W. Vernooij, M. M. Breteler, and W. J. Niessen. kNN-based multi-spectral MRI brain tissue classification: manual training versus automated atlas-based training. In J. M. Reinhardt and J. P. W. Pluim, editors, *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 6144 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, pages 1142–1150, Mar. 2006.

[24] F. Xie, A. Song, and V. Ciesielski. Event detection in time series by genetic programming. In X. Li, editor, *Proceedings of the 2012 IEEE Congress on Evolutionary Computation*, pages 2507–2514, Brisbane, Australia, 10-15 June 2012.

[25] C. Yang, Y. Li, C. Zhang, and Y. Hu. A fast knn algorithm based on simulated annealing. In R. Stahlbock, S. F. Crone, and S. Lessmann, editors, *DMIN*, pages 46–51. CSREA Press, 2007.

[26] W. Yu. and W. ZhengOu. A fast knn algorithm for text categorization. In *Proceedings of the 6th International Conference on Machine Learning and Cybernetics*, pages 3436–3441, Hong Kong, 2007.

[27] L. Zhang, L. B. Jack, and A. K. Nandi. Extending genetic programming for multi-class classification by combining k-nearest neighbor. In *Acoustics, Speech, and Signal Processing, 2005. Proceedings. (ICASSP '05). IEEE International Conference on*, volume 5, Mar. 2005.

[28] M. Zhang, V. Ciesielski, and P. Andreae. A domain-independent window approach to multiclass object detection using genetic programming. *EURASIP J. Adv. Sig. Proc.*, 2003(8):841–859, 2003.

[29] M. Zhang and P. Wong. Genetic programming for medical classification: a program simplification approach. *Genetic Programming and Evolvable Machines*, 9(3):229–255, Sept. 2008.