Improving Genetic Programming Classification For Binary And Multiclass Datasets

Nailah Al-Madi and Simone A. Ludwig
Department of Computer Science
North Dakota State University
Fargo, ND, USA
nailah.almadi@my.ndsu.edu, simone.ludwig@ndsu.edu

Abstract — **Genetic Programming (GP) is one of the evolutionary computation techniques that is used for the classification process. GP has shown that good accuracy values especially for binary classifications can be achieved, however, for multiclass classification unfortunately GP does not obtain high accuracy results. In this paper, we propose two approaches in order to improve the GP classification task. One approach (GP-K) uses the K-means clustering technique in order to transform the produced value of GP into class labels. The second approach (GP-D) uses a discretization technique to perform the transformation. A comparison of the original GP, GP-K and GP-D was conducted using binary and multiclass datasets. In addition, a comparison with other state-of-the-art classifiers was performed. The results reveal that GP-K shows good improvement in terms of accuracy compared to the original GP, however, it has a slightly longer execution time. GP-D also achieves higher accuracy values than the original GP as well as GP-K, and the comparison with the state-of-the-art classifiers reveal competitive accuracy values.**

Keywords-Evolutionary Computation, Genetic Programming, Classification, Multiclass, Binary Classification.

I. INTRODUCTION

Evolutionary algorithms are one category of optimization techniques that are inspired by processes of biological evolution. The term optimization describes a process whereby the search for the optimum solution from a set of candidate solutions is sought; therefore, its goal is to find the best value for each variable in order to achieve satisfactory performance. In practical terms, this means to accomplish a task in the most efficient way to produce maximum yields given limited resources.

Evolutionary computation is also applied to the domain of data mining. Data mining is a relatively broad field that deals with the automatic knowledge discovery from databases and it is one of the most developed fields in the area of artificial intelligence. Given the rapid growth of data collected in various realms of human activity and their potential usefulness requires efficient tools to extract and make use of the potentially gathered knowledge. Evolutionary algorithms are very powerful tools that can be utilized to make use of knowledge hidden in the data collected [1].

One of the important data mining tasks is classification, which is an effective method that is used in many different fields. The main idea behind the classification technique is to build a model (classifier) that assigns items in a collection into target classes with the goal to accurately predict the target class for each item in the data [2]. There are many techniques that can be used to do a classification process such as decision trees, Bayes networks, genetic algorithms, genetic programming and many others. Genetic programming (GP) was found to be successful for classification problems and has emerged as a powerful tool for classifier evolution [3].

GP is an effective technique as it automatically solves problems without requiring the user to know or specify the form or structure of the solution in advance. Thus, it can be used anywhere when neither the solution nor its structure is known to the user. GP is distinguished from other evolutionary algorithms by the use of a tree representation of variable size rather than of a linear string of fixed length representation as in genetic algorithms. This flexible representation helps to automatically discover the underlying structure of the data. Genetic programming has proven to be a very powerful optimization technique in many application areas, such as evolving a computer program that plays a board game, where the parsing of trees is necessary, and thus the genetic programming approach seems to be the best option [4].

When GP is applied to a classification task, it basically derives a classifier that distinguishes two (binary classification) or more classes (multiclass classification) in order to apply the classifier to new data that determines the class of the data instance. GP has been used for classification problems in many research investigations. For example, GP was used to investigate the prognosis of breast cancer and classify them into two classes resulting in an accuracy of GP that outperforms the linear programming approach [5]. GP solves the problems by generating programs, and these programs consist of mathematical and logical operators and functions that result in numeric values. For classification problems, this value represents the class label. The goal of GP as an optimization process is to maximize the number of correctly classified records, which is referred to as the accuracy of the classifier. For binary classification, this process is simple since GP can be configured to return a boolean value that represents the class label. For multiclass classification on the other hand, the transformation of the resulting numeric values into the target class labels is not so trivial.

This paper discusses in particular the techniques of how to transform the GP output value into class labels in order to improve the classification accuracy. The first proposed approach (GP-K) uses a K-Means clustering technique that creates clusters from the GP output value and then uses the majority voting technique to decide the class label for the clusters. After that, the predicted class labels are used to calculate the accuracy. The second approach (GP-D) uses the

978-1-4673-5895-8/13/$31.00

discretization method that transforms continuous values into nominal values, thus transforming the GP's output numerical values to categorical class labels.

This paper is structured as follows. Section II describes related work in the area of classification and some proposed improvement techniques, in particular focusing on multiclass problems. In Section III, GP, K-Means clustering, and the discretization techniques are outlined followed by our proposed approaches. Section IV presents the experiments as well as the results obtained, and Section V concludes this paper.

II. RELATED WORK

Several techniques have been proposed to tackle classification using GP for both binary and multiclass problems [3]. Some research investigations applied GP in different areas, and others focused on the theoretical side to improve the accuracy of GP.

For binary classification, three main approaches are used. One approach uses zero as a boundary between classes, such that negative values are considered to belong to one class and positive values belong to the other class [6, 7]. The second approach is to make GP execute and produce a boolean value and then use this boolean value as a class label, such as in [8] where logical operators were used to produce "if then else" classification rules. The third approach uses a separate component to determine the class labels for the values of the GP output, while [9] used a linear perceptron. This approach can also be used for multiclass classification problems.

Communal Binary Decomposition (CBD) uses a probabilistic method to model the outputs of programs; where the program's fitness depends on its performance at separating only one pair of classes for a particular binary classification sub-problem. To test CBD, the authors used GP with 5 functions (+,-,*, /, if), and applied it on three image datasets with different class labels (3, 4, and 5 classes). They compared the CBD results with the Program Classification Map (PCM) [10, 11] and Probabilistic Model (PM) [12], and showed that CBD can achieve better accuracy values, however, with a much longer running time.

For the multiclass classification process, many approaches were proposed. One approach is to decompose the problem into group of binary sub-problems such as in [13], where the authors first construct a method to decompose a multiclass classification problem into a number of binary sub-problems, solve all these sub-problems in a single GP run, and then combine the binary sub-problems to solve the whole multiclass problem. Other approaches were done using a separate component, such as in [14], where the authors discuss the texture classification problem, which involves extracting texture features from two or more classes of texture images to train a classifier. They used GP to discover useful texture feature extraction algorithms, where a feature extraction algorithm is considered useful if the feature values have a high classification accuracy. They integrated the K-Means clustering algorithm into the GP to compute the fitness of the feature extraction algorithm, according to their intuition that "the better the separation, the better the fitness". They did this by computing the overlap between the clusters generated by K-Means, where they computed the average of the feature values for each class given the cluster centroid, then they calculated the midpoint of the two centroids as a cluster boundary, so the points above this boundary belong to one cluster and below this boundary belong to the other cluster. The number of points in the wrong clusters is considered the error. The grey level histograms of different textures were used as input to the evolved programs. The GP function set contains only the + function. They tested their approach using Brodatz as a learning set and the Vistex set for training and testing the classifier. The resulting accuracy was competitive with other feature sets especially when combined with the Haralick feature set [15]. The paper was concluded by stating that the algorithm has captured some texture regularities, but mentioned that it is very difficult to determine what they are. Also, the performance was limited by the fact that there is no spatial information in the histograms.

Static Class Boundary Determination (SCBD) was proposed in [10], in which two or more static pre-defined thresholds/boundaries are applied to the numeric output value of the genetic program, and the ranges/regions between these boundaries are linearly translated into different classes. These regions are set by fixed boundaries at the beginning of the evolution, and remain constant during the run. If there are n classes for a classification task, then these classes are sequentially assigned to n regions along the numeric output value space ranging from negative numbers to positive numbers with n-1 thresholds/boundaries. The first class is assigned to the region with all numbers less than the first boundary; the next class is allocated to all numbers that lie between the first and the second boundaries, and so on.

Slotted Dynamic Class Boundary Determination (SDCBD) and Centered Dynamic Class Boundary Determination (CDCBD) were proposed in [16], where in CDRS the class boundaries are dynamically determined by calculating the center of the program output value for each class. The algorithm starts by initializing the class boundaries as in SCBD, then during the evolution process of each class it calculates the center of the class based on an equation, then it calculates the boundary between every two classes by taking the midpoint of the two adjacent class centers, and then performs the classification based on the new boundaries. Modifying the class boundaries was done to the training examples after every five generations to balance between evolution and class boundary determination. In SDCBD, the output value of a program is split into certain slots. When a large number of slots are used, a large amount of computation is required. In their experiment, they used 100 slots derived from the range of [-25, 25] with steps of 0.50. Since the input features (terminals) are scaled into [-1, 1], each slot is assigned to a value for each class. First, it evaluates each genetic program based on the SCBD method. Then, it calculates the slot values for each class based on the program output value. After that, it dynamically determines which class each slot belongs to by simply taking the class with the largest value at the slot. However, in case a slot does not hold any positive value, that is, no programs produce any output at that slot for any training examples, then this slot is assigned to the class of the nearest neighboring slot. This method is applied after every 5 generations, similarly to the CDCBD method.

Another multiclass classification technique is presented in [17], where two populations are evolved simultaneously, one

contains fuzzy rule sets and the other contains membership functions; both work together to effectively adapt to each other. Moreover, in [18], the authors proposed an Evolved Class Boundary (ECB) approach that draws a boundary between classes by calculating the mean of the individual output for each class on the training data, and then calculates the midpoint of these two means. They tested their approach using 3 binary datasets, where they balanced the data to have equal classes, by removing some records. They compared their results with the static class boundary [10] and the centered dynamic class boundary approach [17]. The results showed that ECB obtained a good testing accuracy in 2 out of 3 experiments. Also, they compared the ECB accuracy results with other works, where their rank was between 4th and 8th (out of 27).

For the binary classification, configuring the GP to return a boolean value may lead to a loss in quality of the equation created by the GP program since less information is returned. In addition, using zero as a boundary is not very effective since the GP program may produce positive or negative values for both classes. Therefore, a technique is needed to save the GP's program numerical output and transform it into a class label. However, for the multiclass classification problems, related work proposed a technique that finds the boundaries between the classes, also referred to as discretization, by finding specific boundaries for the overall dataset. In our work, we want each program to be an independent classifier that produces different output, and therefore, must have its own boundaries. Moreover, the related approach that decomposes the multiclass classification problem into groups of binary classification problems results in a higher overhead since it has to divide and execute more runs in order for the results to be combined at the end.

Therefore, this paper proposes two techniques to solve the classification task for both binary and multiclass data by preserving the GP program structure (the equations), and transforming the GP output into class labels for each program (classifier) in order for the GP evolution to be trained to find the best classifier with the best accuracy. In our proposed approaches, K-Means and discretization techniques are used. In the first approach, K-Means is used in a different way than [14], where the authors used K-Means and find a midpoint between clusters to use as a boundary, however, we used K-Means to create clusters and then use these clusters to calculate the accuracy by applying majority voting.

III. PROPOSED APPROACHES

Given that our proposed approaches are based on GP as well as K-Means clustering and discretization, we first briefly introduce GP, K-Means, and the discretization method used before outlining the details of our algorithms.

A. Genetic Programming

GP [19] is an evolutionary computation technique that automatically solves problems without requiring the user to know or specify the form or structure of the solution in advance [20]. It offers a solution through the evolution of computer programs by methods of natural selection. Each program is composed of functions (+, -, ×, ÷, etc.) and terminals (variables like x, y or constants like 2.2, 7, 11, etc.), and is represented as a tree.

After the selection of the GP-specific settings such as chosen terminals, functions, population size, fitness function and termination criterion, the GP process can start and proceed by the following steps:

Step 1: Randomly generate an initial population of computer programs.

Step 2: Execute each computer program in the population and calculate its fitness using the fitness function.

Step 3: Use the selection operation and choose two programs (parents).

Step 4: With the assigned probabilities, select a genetic operator to perform crossover and mutation.

Step 5: Repeat Step 3 and 4 until the size of the new population becomes equal to the size of the initial population. Then replace the current population with the new population. Then Go to Step 2 and repeat the process until the termination criterion (maximum number of generation is reached, or a pre-specified accuracy) is satisfied.

The result of the run is usually the program with the best fitness value found during the whole evolution. All these steps are shown in Figure 1.

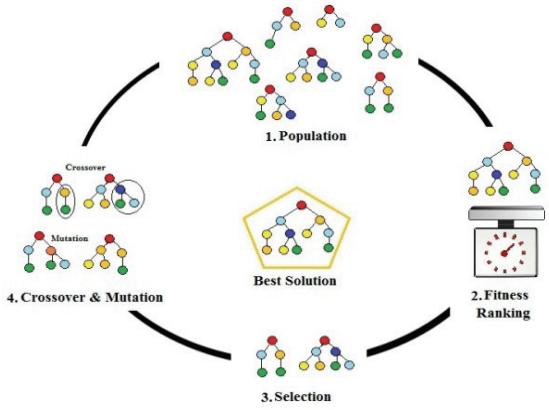

Fig. 1. GP Process

B. K-Means Clustering

K-Means was proposed in [21]. It is one of the efficient clustering algorithms and is used in many applications since it is simple to implement and has shown good performance. K-Means is categorized under the partitioning clustering algorithms, where the goal is to maximize the intra-cluster similarity, and minimize the inter-cluster similarity. K-Means is an efficient method as it runs in linear time. K-Means starts with specifying the number of clusters needed, K, and continues by following these steps (cluster example is shown in Figure 2):

Step 1: Choose K initial random centroids, C1, C2, C3...CK.

Step 2: Compute the distance from each point to every centroid, and then assign the point to the cluster with the minimum distance (closest one).

Step 3: Compute the new means (centroids) of the generated clusters, and repeat this step until no changes occur on the clusters, or until a predefined number of iterations is reached.

The distance between points and centroids can be computed using the Manhattan distance as follows:

$$\text{Distance}(P, Ci) = \sum | Cxi - Pxi | \quad (1)$$

To compute the new centroid the following equation is used:

$$C = \sum \frac{\text{ClusterPoints}}{\#\text{ClusterPoints}} \quad (2)$$

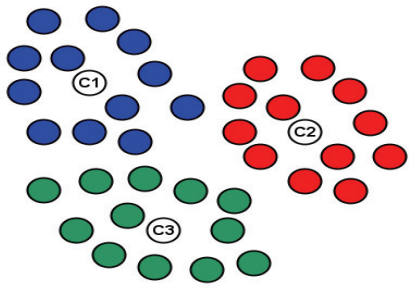

Fig. 2. K-Means Clustering Example

C. Discretization

Discretization [22] is the process that discretizes and converts numeric attributes in the dataset into nominal attributes. There are two types of discretization, supervised and non-supervised. One of the supervised versions takes the class label into account when discretizing the numbers by calculating the entropy on the basis of the class label. It first sorts the attributes, and then finds the split points using the entropy [23]. This technique is used by our approaches. It finds the best split such that the segments are as pure as possible; thus, the majority of the values in a segment correspond to having the same class label. Formally, the goal is to find the split with the maximal information gain (Info (S)), which is obtained when the information value (Info(S, T)) is minimal. Therefore, given a set of samples S, partitioned into two intervals S_1 and S_2 using boundary T, the entropy after partitioning is calculated as in Equation 3:

$$\text{Info}(S, T) = \frac{|S_1|}{S} \text{Info}(S_1) + \frac{|S_2|}{S} \text{Info}(S_2) \quad (3)$$

where |•| denotes the cardinality. The boundary T is chosen from the midpoints of the attributes values, and the information content (Info) is calculated as follows:

$$\text{Info}(S) = P_1 * \log_2(P_1) - P_2 * \log_2(P_2) \quad (4)$$

where P_1 is the fraction of pairs within the first class, and P_2 is the fraction of pairs within the second class.

The best splits are found by examining all possible splits and then selecting the optimal split points. The boundary that minimizes the entropy function over all possible boundaries is selected for the binary discretization. The process is recursively applied to partitions obtained until some stopping criterion is met, such as what is proposed by the Minimum Description Length (MDL):

$$\text{Gain} > \frac{\log_2(N-1)}{N} + \frac{\log_2(3^k - 2) - [KE - K_1E_1 - K_2E_2]}{N} \quad (5)$$

where K is the number of classes in S, K_1 is the number of classes in S_1, and K_2 is the number of classes in S_2, N is the number of instances, and E is the entropy of the instances, E_1 entropy of the instances in S_1, and E_2 is entropy of instances in S_2.

An example of the process is shown in Figure 3, where there are continuous numbers in the range from 24 to 60, and each number is related to a class label (in the second row). The discretization process uses the information gain by taking the class labels to find split points that have splits with pure class labels. After defining the splits, majority voting is used to define the class label of the split, thus, in our example, we have class labels C2, C1, C2, C1, C2, C1, C2 respectively, for the splits shown in Figure 3.

24	25	38	39	40	51	54	58	60	71	73	77
C2	C2	C1	C1	C1	C2	C2	C1	C2	C1	C1	C2

Fig. 3. Discretization Example

D. GP-K and GP-D

We propose two methods to achieve higher accuracy values for the classification of GP. The first method is based on using K-Means in combination with GP (GP-K), and the other is using GP together with the discretization process (GP-D). These techniques are used to transform the produced numeric output of the GP to a class label by storing the produced values from a program (classifier) in a data vector, and then applying the transformation process on it.

In GP-K, the K-Means clustering algorithm is applied on the GP program output, then the accuracy or fitness of this program is computed using the number of records that are in the correct cluster. The process to compute the fitness of each program and to apply the program (equation) on the data records works as follows. The result of GP for each record (a number) is recorded and stored in a data vector. After that, K-Means is applied on this resulting data vector, and the number of K clusters is defined (that is the same as the number of classes in the original dataset). Majority voting is used on the resulting clusters to define the label of the cluster (class label). Then, after labeling the clusters, the records that are incorrectly appearing in the cluster are counted and represent the misclassified records by calculating the accuracy using Equation 6.

$$\text{Accuracy} = \frac{(\#\text{AllRecords} - \#\text{MisclassifiedRecords})}{\#\text{AllRecords}} * 100\% \quad (6)$$

For GP-K, we found that the best number of iterations to run K-Means is 500, as it gives good accuracy values and does not affect the execution time too much (more iterations implies a larger execution time). However, K-Means may stop before reaching 500 iterations in case no changes occurred during consecutive iterations.

GP-D completes the same process by first applying the GP program on the data records, and then recording the output number by storing it into a data vector. Afterwards, the supervised discretization is applied on this data vector. The discretization was applied using the one implemented by the WEKA data mining software [23], where a vector of the outputs and its class labels are sent to the discretization component. The result from the discretization is a range value instead of a numeric value. After that, all the records in the same range are collected, and then based on the class label for the maximum number of records in the range the class label for that range is defined. The accuracy is also calculated using Equation 6.

IV. EXPERIMENTS AND RESULTS

To evaluate the proposed techniques and see how effective they are in improving the accuracy of the GP, experiments were performed using the Java Genetic Algorithms Package (JGAP) [24] on two types of datasets, one with binary classes (true and false), and the other with multiple classes (multiclass).

A. Datasets

The experiments are applied on two types of datasets [25]. The binary datasets are the Wisconsin Diagnostic Breast Cancer (WDBC) dataset that predicts the two breast cancer diagnoses of benign and malignant, the Hepatitis dataset contains data of whether a person lived or died, the Heart datasets refers to the presence of heart disease in patients, and the Diabetes dataset is the Pima Indians Diabetes Database and the diagnostic investigated is whether the patient shows signs of diabetes or not. The details of these datasets including the number of features, and the number of records contained are shown in Table I.

The second type of datasets that were used are multiclass datasets that include the Dermatology dataset, which determines the type of Eryhemato-Squamous disease, the Lymph dataset, which is about the Lymphography disease, the Splice dataset contains Primate splice-junction gene sequences (DNA) with associated imperfect domain theory, and the CTG dataset consists of fetal Cardiotocograms (CTGs) records. The CTG dataset has two outcome categories, the first one classifies a morphologic pattern (CTG-Class), and the second is related to the fetal state (CTG-NSP). All details including the number of features and records of these datasets are shown in Table II.

B. Experimental Settings

The experiments were conducted as follows. First of all, feature selection was performed on the datasets using the WEKA software [23], which reduced the number of features as shown in the brackets in Table I for the binary datasets, and in Table II for the multiclass datasets, respectively. The attribute selection method in WEKA is a supervised attribute filter that allows various search and evaluation methods to be combined [23]. Secondly, our GP algorithm was run using the following settings: population size = 500, number of generations = 1000, crossover probability = 0.5, mutation probability = 0.1, maximum initial depth = 5, maximum crossover depth = 8, function probability = 8, dynamize arity probability = 0.05, new chromosome percentage = 0.2. 37 functions are used, and 14 of them are variables and constants. The experiments are applied on the standard GP (GP), the proposed GP-K and GP-D using 66% of the datasets for training and the remaining 34% for testing.

TABLE I. BINARY DATASETS

Dataset	Classes	Features (Filtered)	Records
Breast	2	31 (11)	568
Diabetes	2	8 (4)	768
Heart	2	13 (9)	269
Hepatitis	2	19 (10)	155

TABLE II. MULTICLASS DATASETS

Dataset	Classes	Features (Filtered)	Records
Lymph	4	19 (10)	148
Splice	3	61 (22)	3190
Dermatology	6	33 (19)	366
CTG_NSP	3	22 (7)	2126
CTG_Class	10	22 (11)	2126

C. Results

To evaluate the results of the proposed GP algorithms, they are compared with standard GP measuring the accuracy and execution time. Moreover, a comparison with state-of-the-art classifiers was done to measure how well our algorithms compared to the ones known in the literature.

Table III shows the average with standard deviation for 30 runs, as well as the best accuracy results of the standard GP, and the proposed approaches GP-K and GP-D applied to the binary datasets for the training and testing phases.

We can infer from Table III that GP-K and GP-D show improvements compared to the accuracy of GP. For the breast cancer dataset, the GP obtained a testing accuracy of 92.85%, whereas GP-K and GP-D achieved 95.39% and 97.54%, respectively. Similarly, for the Diabetes, Heart and Hepatitis datasets, GP-K showed improvements over the original GP, whereby GP-D scored best.

TABLE III. ACCURACY FOR BINARY DATASETS

Datasets		Breast		Diabetes		Heart		Hepatitis	
		Training	**Testing**	**Training**	**Testing**	**Training**	**Testing**	**Training**	**Testing**
GP	**Avg.**	95.85	92.85	76.02	75.84	86.65	73.04	87.81	82.52
	Std.	±1.07	±1.90	±0.99	±1.25	±1.91	±4.19	±1.72	±2.88
	Best	97.86	95.88	77.87	78.63	89.27	81.52	91.18	86.79
GP-K	**Avg.**	97.76	95.39	77.52	78.34	88.93	76.34	88.20	86.03
	Std.	±0.58	±0.86	±0.29	±0.82	±0.51	±3.38	±0.83	±2.35
	Best	98.66	96.91	78.26	80.15	90.40	81.52	90.20	90.57
GP-D	**Avg.**	98.66	**97.54**	78.46	**80.07**	89.94	**80.50**	94.05	**88.55**
	Std.	±0.16	±0.48	±0.62	±1.00	±0.43	±2.56	±0.88	±1.10
	Best	99.20	98.45	79.84	82.44	90.96	85.87	96.08	90.57

TABLE IV. ACCURACY FOR MULTICLASS DATASETS

Datasets		Lymph		Splice		Dermatology		CTG-NSP		CTG-Class	
		Train.	Testing	Train.	Testing	Train.	Testing	Train.	Testing	Train.	Testing
GP	Avg.	83.71	73.07	71.59	70.98	74.43	72.32	84.38	83.53	47.56	44.74
	Std.	±3.13	±4.45	±4.10	±4.19	±5.13	±5.29	±0.99	±1.83	±3.40	±4.81
	Best	90.72	82.35	82.30	82.90	87.14	82.40	87.24	87.83	54.10	52.97
GP-K	Avg.	86.84	76.01	75.60	74.21	91.58	79.46	85.03	83.49	65.59	61.03
	Std.	±1.10	±2.94	±3.48	±4.59	±2.11	±4.05	±0.72	±1.19	±1.41	±2.79
	Best	88.66	80.39	87.45	86.91	95.44	86.40	86.60	86.86	69.43	65.98
GP-D	Avg.	89.97	**78.36**	91.24	**86.22**	97.28	**86.24**	90.12	**88.33**	73.50	**64.09**
	Std.	±1.43	±2.15	±0.78	±4.35	±1.05	±4.64	±0.65	±1.05	±1.40	±5.06
	Best.	91.75	84.31	92.35	90.69	99.17	96.00	91.17	90.32	76.41	73.17

Looking at the training accuracy, for the breast cancer dataset, GP obtained an accuracy of 95.85%, with the best value (from the 30 runs) of 97.86%. However, GP-K's best value is 98.66%, GP-D's average is 98.66%, and its best value is 99.20%. The same trend is found for the other binary datasets, where GP-K slightly improves GP's accuracy (3%-4%), and GP-D outperforms both GP and GP-K by 5%-8% compared to GP. It can also be seen that both GP-K and GP-D have a smaller standard deviation compared to GP, which indicates that they are more robust.

The accuracy results for the multiclass datasets are shown in Table IV. We can see that GP-D achieves the highest accuracy for all datasets, and GP-K improves the accuracy of standard GP. For example, for the Splice dataset, GP's accuracy is 70.98%, GP-K's is 74.21% and GP-D's is 86.22%. For the Dermatology dataset, GP's accuracy is 72.32%, GP-K's is 79.46% and GP-D's is 86.24%. The maximum improvement that was achieved was for the CTG-class dataset, where GP's accuracy is 44.74%, GP-K's is 61.03% (increased by 16%), and GP-D's is 64.09% (increased by 19% compared to GP's accuracy).

Looking at the best results accomplished while testing the datasets, the accuracy values obtained on the Lymph dataset are 82.35%, 80.39%, 84.31% for GP, GP-K, and GP-D, respectively. However, the difference for the Splice dataset is larger where 82.90%, 86.91% and 90.69% were measured. The difference between the best accuracy values of the GP, GP-K and GP-D for the Dermatology and the CTG-NSP datasets are smaller (close to what is shown for Lymph), while for the CTG-Class dataset, GP's best value is 52.97%, GP-K is 65.98%, GP-D is 73.17%. Moreover, commenting on the standard deviation of the testing process, the values for the Lymph dataset are ±4.45, ±2.94, and ±2.15 for GP, GP-D and GP-K, respectively. For the other datasets, the values for GP, GP-K and GP-D are in the same range, besides for the binary datasets, where the values are either smaller or no change is observed.

For the training phase, the results show the same trend as for the testing phase comparing GP, GP-K and GP-D. However, they have higher values compared to the testing phase. GP achieved average accuracy results of 90.72%, 82.90%, 87.14%, 87.24%, 54.10% on the Lymph, Splice, Dermatology, CTG-NSP, and CTG-Class datasets, respectively. However, GP-K obtained 88.66%, 87.45%, 95.44%, 86.60% and 69.43%. GP-D on the other hand achieved scores of 89.97%, 91.24%, 97.28%, 90.12% and 73.50%. In addition, the best values are much higher, where for the same order of datasets GP-D obtained 91.75%, 92.35%, 99.17%, 91.17%, and 76.41%.

The running time results are shown in Figure 4 (drawn using the logarithmic scale to adapt to the large difference between small and large datasets), where we can see that GP-K has a slightly longer running time than GP, while GP-D's running time is much longer compared to GP. For example, for the breast cancer dataset GP took 55.70 seconds, GP-K took 60.44 seconds, and GP-D took 367.80 seconds. For the binary datasets, and the Lymph and Dermatology, the running times are similar to the breast cancer dataset, whereas for Splice, CTG-NSP and CTG-Class the difference of GP-D is much larger. The difference in running time increases when the number of records gets larger (compare the number of records of Splice: 3190 with CTG: 2126).

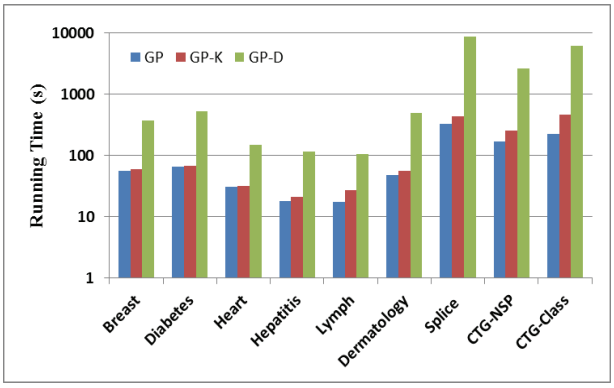

Fig. 4. Running time results for GP, GP-K and GP-D

The reason for this longer running time of GP-D, is that GP-D uses the discretization method as stated in [16] by calculating the information gain for each possible split point. If the dataset has for example 2126 records, then there are 2126 possible split points. In addition, this discretization process is done for every program (classifier) when its fitness is calculated, therefore, for a population size of 500, the discretization portion is executed 500 times per generation. Although GP-D has a longer running time compared to GP and

GP-K, however, a substantial improvement in accuracy is achieved, as well as the robustness of the algorithm is increased. However, if time is of essence, GP-K can be used since it has a higher accuracy than standard GP with a comparable running time.

A comparison of the accuracy of GP, GP-K and GP-D with other well-known classification methods on the same datasets is shown in Tables V and VI, where the numbers in bold are the highest values, and the values in bold and italic are the second highest values. The following classifiers were used and are described below:

- **Bayes Network (BN):** learning algorithm using various search algorithms and quality measures.
- **Naive Bayes (NB):** class for a classifier using estimator classes.
- **J48:** contains a class for generating a pruned or un-pruned C4.5 decision tree.
- **IBK:** K-nearest neighbors classifier, which can select an appropriate value of K based on cross-validation and also performs distance weighting.
- **MP:** multilayer perceptron is a classifier that uses back propagation to classify instances.
- **K*:** is an instance-based classifier, that is, the class of a test instance is based upon the class of those training instances similar to it, as determined by some similarity function.
- **JRip:** implements a propositional rule learner, Repeated Incremental Pruning to Produce Error Reduction (RIPPER).
- **SMO:** implements John Platt's sequential minimal optimization algorithm for training a support vector classifier.

All these algorithms were applied using the WEKA data mining software [16] with 10 fold cross-validation.

The binary dataset results are shown in Table V. GP-D outperforms all other classifiers with an average accuracy of 97.54% for the Breast dataset. However, for the Diabetes dataset, NB has the highest accuracy of 84.38%, whereas standard GP has 75.84%, GP-K and GP-D have average accuracy values of 78.34% and 80.07%, respectively, outperforming the K* and JRip results. For the Heart dataset, GP-D achieves the highest average accuracy of 80.50%, and the best accuracy of 85.87%. GP-K obtains an accuracy of 76.34% and outperforms a number of other classifiers such as J48, BN, IBK, K*, JRip, and MP. For the Hepatitis dataset, GP-D also achieves the highest accuracy of 88.55% with the best value of 90.57%, again outperforming all other classifiers. Moreover, GP-K also outperforms all other classifiers with an accuracy of 86.03% (best value of 90.57%). In summary, GP-D obtains the highest accuracy values for the binary datasets in 3 out of 4 cases, whereby GP-K has the highest average accuracy value on the Hepatitis dataset, as well as outperforming some of the other classifiers.

For the multiclass datasets, GP-D shows competitive values compared to the other classifiers, such as the Lymph dataset, where GP's best accuracy is 84.31%. Also, for the Dermatology and CTG-NSP datasets, GP-D's best value is 96.00%, which is somewhat close to the best result of 97.81% for Dermatology, and 90.32% (best 92.94%) for CTG-NSP. GP-K did not achieve close scores as GP-D did; however, it definitely improved the standard GP.

A statistical two-tailed Z-Test was applied on the average results of the 30 runs for GP-K compared to standard GP as well as for GP-D compared with standard GP using a significance level of 5%. The Z-Test results show that GP-K and GP-D achieve significant accuracy improvements on all datasets except for GP-K when applied on CTG-NSP.

V.CONCLUSION

This paper proposed two approaches in order to improve the accuracy of GP for the classification task, in particular for multiclass classification problems. The first proposed approach, GP-K, makes use of the K-Means clustering algorithm to transform the GP output from a numeric value to a class label. The second approach, GP-D, performs the same transformation process by applying the discretization method on every classifier.

TABLE V. ACCURACY RESULTS FOR BINARY DATASET COMPARED WITH OTHER CLASSIFIERS

Datasets	GP	GP-K	GP-D	J48	BN	NB	IBK	K*	JRip	SMO	MP
Breast	92.85 (95.88)	95.39 (96.91)	**97.54 (98.45)**	93.48	95.95	94.54	95.24	95.59	95.07	96.65	97.00
Diabetes	75.84 (78.63)	78.34 (80.15)	80.07 (82.44)	81.41	83.27	**84.38**	80.66	78.43	78.81	84.01	81.41
Heart	73.04 (81.52)	76.34 ***(81.52)***	**80.50 (85.87)**	74.86	75.52	77.47	68.35	69.92	74.73	76.82	75.52
Hepatitis	82.52 (86.79)	***86.03 (90.57)***	**88.55 (90.57)**	81.93	83.22	85.16	81.29	83.87	78.70	83.22	81.29

TABLE VI. ACCURACY RESULTS FOR MULTICLASS DATASET COMPARED WITH OTHER CLASSIFIERS

Datasets	GP	GP-K	GP-D	J48	BN	NB	IBK	K*	JRip	SMO	MP
Lymph	73.07 (82.35)	76.01 (80.39)	78.36 ***(84.31)***	78.37	79.72	80.40	77.70	81.75	77.02	**81.08**	79.72
Splice	70.98 (82.90)	74.21 (86.91)	86.22 (90.69)	93.54	**96.14**	92.06	68.24	77.86	94.07	85.20	89.78
Dermatology	72.32 (82.40)	79.46 (86.40)	86.24 ***(96.00)***	95.35	**97.81**	**97.81**	96.72	97.54	94.26	**97.81**	96.72
CTG-NSP	83.53 (87.83)	83.49 (86.86)	88.33 ***(90.32)***	**92.94**	89.46	84.43	92.09	91.90	91.95	88.19	89.51
CTG-Class	44.74 (52.97)	61.03 (65.98)	64.09 (73.17)	**82.83**	78.50	71.49	80.66	81.93	82.07	76.81	79.82

A comparison of the standard GP with the proposed approaches was conducted using binary and multiclass datasets measuring the accuracy and running time. Overall, it was shown that both, GP-K and GP-D, improve the accuracy of standard GP, however, this comes at the cost of a longer running time. GP-K only has a slightly longer running time, but GP-D takes a very long time to run. In addition, for the binary datasets, GP-K and GP-D were found to be more robust compared to standard GP as shown by the standard deviation. Comparing the proposed approaches with other known classifiers, it was found that GP-D outperforms 3 of 4 classifiers for the binary datasets, and for the multiclass higher accuracy values than GP were achieved with some competitive values compared to the know classifiers.

Given that both GP-K and GP-D have longer running times than GP, preliminary tests were run keeping the running time for standard GP the same as GP-D in order to see the effect on the accuracy. The tests were applied on the Breast dataset with 30 runs. GP was run for 6,000 iterations instead of 1,000 (running time on average was 373.02 seconds close to 367.8 seconds for GP-D). The average accuracy of GP improved from 92.85% to 94.36%. GP-D has an average accuracy of 97.54%, which is still much better than standard GP. However, more experiments are necessary to confirm the same trend on all other datasets.

Future work involves finding a way to keep the high accuracy and good robustness of GP-D, while at the same time shorten the running time. This might be done by parallelizing the proposed approach, or by limiting the discretization process to be run only on every other classifier in the population.

REFERENCES

[1] L.C. Jain and A. Ghosh, "Evolutionary Computation in Data Mining (Studies in Fuzziness and Soft Computing)", Springer-Verlag New York, Inc., Secaucus, NJ, 2005

[2] P. Tan, M. Steinbach, and V. Kumar, "Introduction to Data Mining", Addison-Wesley, May 2005. (ISBN:0-321-32136-7)

[3] H. Jabeen and A.R. Baig. "Review of Classification Using Genetic Programming". International Journal of Engineering Science and Technology, vol. 2. no. 2. pp.94-103, 2010

[4] A.E. Eiben, and J.E. Smith, "Introduction To Evolutionary Computing", Springer, Natural Computing Series, 1st edition, 2003.

[5] S.A. Ludwig and S. Roos, "Prognosis of Breast Cancer using Genetic Programming", Proceedings of 14th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems (KES), Cardiff, Wales, UK, September 2010.

[6] A. Song, T. Loveard, and V. Ciesielski. "Towards genetic programming for texture classification". In M. Stumptner, D. Corbett, and M. Brooks, editors, Proceedings of the 14th International Joint Conference on Artificial Intelligence AI 2001: Advances in Artificial Intelligence, vol. 2256 of Lecture Notes in Computer Science, pp. 461-472, Springer-Verlag. Adelaide, Australia, Dec. 10-14, 2001.

[7] W. A. Tackett. "Genetic programming for feature discovery and image discrimination", In S. Forrest, editor, Proceedings of the 5th International Conference on Genetic Algorithms, ICGA-93, pp. 303-309, Morgan Kaufmann , University of Illinois at Urbana-Champaign, 17-21 July, 1993.

[8] C. C. Bojarczuk, H. S. Lopes, and A. A. Freitas. "Discovering comprehensible classification rules by using genetic programming: a case study in a medical domain". In W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, editors, Proceedings of the Genetic and Evolutionary Computation Conference, vol. 2, pp. 953-958, Morgan Kaufmann, Orlando, Florida, USA, 13-17 July 1999.

[9] C. Estebanez, R. Aler, and J. M. Valls. A method based on genetic programming for improving the quality of datasets in classification problems. International Journal of Computer Science and Applications, vol. 4. no.1. pp.69-80, 2007.

[10] M. Zhang and V. Ciesielski. "Genetic programming for multiple class object detection". In N. Foo, editor, 12th Australian Joint Conference on Artificial Intelligence, vol. 1747 of LNAI, pp. 180-192, Springer-Verlag Sydney ,Australia, 6-10 Dec. 1999.

[11] M. Zhang, V. Ciesielski, and P. Andreae. "A domain independent window-approach to multiclass object detection using genetic programming", EURASIP Journal on Signal Processing, Special Issue on Genetic and Evolutionary Computation for Signal Processing and Image Analysis, vol. 8. pp.841-859, 2003.

[12] W. Smart and M. Zhang. "Probability based genetic programming for multiclass object classification". In Proceedings PRICAI 2004, LNAI vol. 3157, pp. 251-261, Springer-Verlag, 2004.

[13] W.D. Smart, M. Zhang: "Using Genetic Programming for Multiclass Classification by Simultaneously Solving Component Binary Classification Problems". pp. 227-239, EuroGP 2005.

[14] B. Lam and V. Ciesielski. "Discovery of human competitive image texture feature extraction programs using genetic programming". In Kalyanmoy Deb et al., editor, Proceedings of the Conference on Genetic and Evolutionary Computation (GECCO2004), vol. 2. pp. 1114-1125. Springer, June 2004.

[15] M. Haralick, K. Shanmugam, and I. Distein, "Texture Features for Image Classification", IEEE Transactions on Systems, Man, and Cybernetics, SMC vol. 3. no. 6. pp. 610-621, 1973

[16] M. Zhang and W. Smart, "Multiclass object classification using genetic programming". Technical Report CS-TR-04-2, Computer Science, Victoria University of Wellington, New Zealand, 2004.

[17] R.R.F. Mendes, F.B. Voznika, A.A. Freitas and J.C. Nievola. "Discovering fuzzy classification rules with genetic programming and co-evolution". Principles of Data Mining and Knowledge Discovery (Proceeding of the 5th European Conference, PKDD 2001) - Lecture Notes in Artificial Intelligence 2168, pp. 314-325. Springer-Verlag

[18] J. Fitzgerald , C. Ryan, "Drawing boundaries: using individual evolved class boundaries for binary classification problems", Proceedings of the 13th annual conference on Genetic and evolutionary computation, July 12-16, 2011, Dublin, Ireland.

[19] A.E. Eiben, and J.E. Smith, "Introduction to Evolutionary Computing", Springer, Natural Computing Series, 1st edition, 2003.

[20] R. Poli and W.B. Langdon and N.F. McPhee, "A Field Guide to Genetic Programming", with contributions by J.R. Koza, Published via lulu.com and freely available at http://www.gpeld-guide.org.uk, 2008.

[21] J.B. MacQueen, "Some methods for classification and analysis of multivariate observations", Proceeding of the 5th Berkeley Symp. Probability Statistics, University of California Press, Berkeley, pp. 281–2971967,.

[22] U.M. Fayyad and K.B. Irani, "Multi-interval discretization of continuousvalued attributes for classification learning". In: Thirteenth International Joint Conference on Articial Intelligence, pp. 1022-1027, 1993.

[23] I.H. Witten; E. Frank, M. A. Hall. "Data Mining: Practical Machine Learning Tools And Techniques", 3rd Edition . Morgan Kaufmann, San Francisco. 2011.

[24] K. Meffert et al.: JGAP - Java Genetic Algorithms and Genetic Programming Package. Retrieved from http://jgap.sf.net, January 2012.

[25] A. Asuncion and D. Newman, UCI Machine Learning Repository. (URL). University of California, Irvine, School of Information and ComputerSciences, 2007.