A Multi-dimensional Genetic Programming Approach for Multi-class Classification Problems

Vijay Ingalalli[1,2,3], Sara Silva[1,4,5], Mauro Castelli[6], and Leonardo Vanneschi[6]

[1] INESC-ID, Lisbon, Portugal
[2] LIRMM, Montpellier, France
[3] IRSTEA, Montpellier, France
[4] LabMAg, FCUL, University of Lisbon, 1749-016 Lisbon, Portugal
[5] CISUC, Universidade de Coimbra, 3030-290 Coimbra, Portugal
[6] ISEGI, Universidade Nova de Lisboa, 1070-312 Lisbon, Portugal
{vijay.ingalalli,castelli.mauro}@gmail.com, sara@fc.ul.pt, lvannesc@gmail.com

Abstract. Classification problems are of profound interest for the machine learning community as well as to an array of application fields. However, multi-class classification problems can be very complex, in particular when the number of classes is high. Although very successful in so many applications, GP was never regarded as a good method to perform multi-class classification. In this work, we present a novel algorithm for tree based GP, that incorporates some ideas on the representation of the solution space in higher dimensions. This idea lays some foundations on addressing multi-class classification problems using GP, which may lead to further research in this direction. We test the new approach on a large set of benchmark problems from several different sources, and observe its competitiveness against the most successful state-of-the-art classifiers.

1 Introduction

In the last two decades, Genetic Programming (GP) [1] has established itself as a solid research field, not only because of the numerous practical successes that have been reported in many different application domains [2], but also due to the strengthening of the theoretical foundations [3], and the several attempts to bridge theory and practice [4]. Nevertheless, various references report on the poor performance of GP in multi-class classification (intended here, as opposite to binary classification, as the supervised learning task of partitioning data into a number of classes larger than two) when compared to other state-of-the-art classifiers (see for instance [5]).

With the objective of attenuating this possible weakness of GP, in this paper we propose a new GP framework called Multi-dimensional Multi-class Genetic Programming (M_2GP). Several ideas have inspired the definition of this framework. First of all, we hypothesize that single expressions, being represented as trees or any other existing flavors of GP, are not an informative enough representation to effectively solve multi-class classification tasks. For this reason M_2GP uses a multi-expression representation of individuals. Although the idea is not new [6], we present it in a different light by integrating multiple expressions into a single tree. This makes the representation more

M. Nicolau et al. (Eds.): EuroGP 2014, LNCS 8599, pp. 48–60, 2014.

compact and allows us to implement M_2GP with very few modifications to standard GP. Secondly, we do not prefix any explicit connection between the different expressions represented in an individual and the classes. This relationship is free to evolve and allows a higher effectiveness of the algorithm. Thirdly, we present a new algorithm that tends to cluster the numeric values returned by the expressions belonging to a class, minimizing their dispersion. Also this not being a new idea [7,8], M_2GP presents it under a new light by introducing a new fitness function. Finally, M_2GP returns an enriched data model, including not only the set of expressions that compose the best individual, but also other information that is useful for the evaluation of new data.

The paper is structured as follows: Section 2 discusses some previous work in multi-class classification with GP. Section 3 introduces M_2GP, motivating the choices that led us to the definition of the proposed algorithm. Section 4 presents the M_2GP algorithm. Section 5 contains our experimental study, where the test problems are presented, the experimental settings are specified and the results are shown and discussed. Finally, Section 6 concludes the paper and suggests possible future research.

2 Related Work

In this section, we outline several methods that have been proposed in order to tackle multi-class classification problems using GP. The section only presents a restricted subset of the most important and recent contributions in this area. For a more complete survey on this topic the reader is referred to [9].

Several works [10,11,12,13,14] in this area are based on a common and straightforward approach that consists in evolving a single rule in each GP run. In particular, c runs are performed for a c-class classification problem. In this way, the final classifier has a single rule for each class. All these works evolve multiple comprehensible IF-THEN classification rules.

However, the focus of this short literature review is on another common approach, which consists in evolving a discriminant function. In this case the two main approaches are (1) range selection methods and (2) binary decomposition methods. Range selection methods are applicable to GP classifiers that output numerical values. The method works by declaring $c - 1$ thresholds for c-class classification problems. To select optimal thresholds, several mechanisms have been proposed, including static thresholds selection [15,7], dynamic thresholds [8,16] and slotted thresholds [8]. In binary decomposition methods, one classifier is trained to recognize samples belonging to a particular class and reject all other samples. This results in c classifiers for a c-class classification problem. A well-known drawback of this approach is related to the fact that the multiple classifiers may result in conflicts, whose number usually grows up proportionally to the number of classes. Hence, this approach produces an increased classification error as the number of classes gets larger. Binary decomposition methods have been explored in [17,18,19]. The two approaches for multi-class classification, constructing a single classification function or c binary classifiers, are compared in [20], by considering a hand-written digit recognition problem. As reported in [9], when a single function is evolved, able to discriminate all the classes, the function directly outputs the numeric value of the predicted class, since each class is an integer digit. In both cases, the fitness function is based on classification accuracy.

In [6] the authors proposed a GP-based approach to multi-class classification in which each individual is a multi-tree structure made of c trees, where c is the number of classes. Each of these c trees $(T_1, \cdots, T_c)$ encodes a threshold function for a particular class. The system considers that a data instance x belonging to class i is correctly classified if $T_i(x) \geq 0$ and $T_j(x) < 0$, for all $j \neq i$. The fitness function is computed as the classification accuracy. A similar system evolving a multiple-threshold discriminant function is described in [21], where a fitness function based on the sum of squared errors is employed.

One of the most recent contributions of GP for multi-class classification is found in [22]. In this work, the authors propose a two-stage strategy for multi-class classification problems, which is an improvement of a traditional binary decomposition method.

3 Formulation of Multi-dimensional GP

This section introduces the novel ideas we have explored and adopted to tackle the problem of efficient classification of multi-class data sets. Although this work focuses on tree based GP [1], it can be generalized for other types of GP.

Solution Tree. For classification tasks, tree based GP generally uses parse trees for representing the individuals, where the root node and all other non-terminal nodes belong to a set $\mathcal{O}$ of predefined operations (e.g., $\mathcal{O} = \{+, -, *, /\}$), and the terminal nodes/leaves belong to the given attribute set $\mathcal{A}$ for the given data set. At the end of the search process, the solution is available at the root of the best tree, which is a readily interpretable function that is used for the task of classification.

In our approach, we slightly modify the representation of each parse tree by adding a root node ⓡ of arity d ($d \geq 1$), as shown in Figure 1. Thus, the root node ⓡ shall have d branches, $T_1, \ldots, T_d$, each one of them being a normal GP tree, created according to the regular settings discussed in Section 5.2. Once the evolution terminates, GP individuals can still be evaluated at the root node ⓡ, as before; but instead of one solution, we obtain d different solutions, which we use/explore later to perform the task of classification. For any data set, the value of d is independent of any parameters (e.g., the number of classes/attributes) and we will discuss its choice in Section 5.2. The genetic operators (e.g., mutation, crossover) are used normally, with the only restriction that the mutation and crossover points must be chosen below the root node ⓡ. In this way, all the GP individuals will be rooted in ⓡ during the whole evolution.

Solution Space. Let $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ be the training data with n number of samples and $|\mathcal{A}|$ number of attributes. Each sample $\mathbf{x}_i$ corresponds to any class value

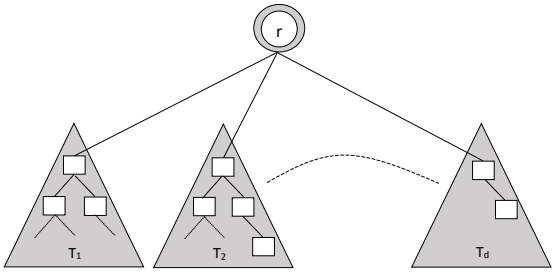

Fig. 1. Representation of a solution tree

$E_i \in C$, where C is the unique set of all the classes. In the usual single tree representation, where we obtain a single function f at the root node, solutions can be mapped in a space (that we call solution space from now on) of 1-dimension, because the output is a scalar numeric value. It is normally in this mono-dimensional solution space that GP is required to perform the task of classification. However, now that individuals are represented by d different functions $\mathcal{F} = \{f_1, f_2, \ldots, f_d\}$, they can be mapped in a d-dimensional solution space. Thus, any individual represented by a tree with a d-arity root node (r), can be represented in a d-dimensional space.

4 Algorithm

With the formulation of multi-dimensional GP, we now propose the algorithm called Multi-dimensional Multi-class GP (M_2GP).

Algorithm 1. M_2GP - Training Module

1 INPUT: $\mathbf{X}$, $\mathbf{E}$, d
2 **for** $g \in 1 \ldots \mathcal{G}$ **do**
3 GENERATE: $\mathcal{F} = \{f_1, f_2, \ldots, f_d\}$ - set of d solutions
4 EVALUATE: $\mathbf{Z_s} = Eval(f_s(\mathbf{X}))$ for all $s \in 1, \ldots, d$
5 CLUSTER: $\mathbf{Z^k} \in \mathbf{Z}$ for all $k \in 1, \ldots, |C|$
6 **for** $k \in 1 \ldots |C|$ **do**
7 $\mathcal{C}^k = covar(\mathbf{Z}^k)$, a $d \times d$ co-variance matrix *% $\mathcal{C}^k$ must be non-singular %*
8 $\mathcal{M}^k = centroid(\mathbf{Z}^k)$, a $1 \times d$ centroid vector
9 $\mathcal{D}_i^k = \sqrt{(Z_i - \mathcal{M}^k) \cdot (C^k)^{-1} \cdot (Z_i - \mathcal{M}^k)^T}, \forall i = 1, 2, ..., n$ *% Mahalanobis distance %*
10 *% where n is the number of samples in the training set %*
11 $\forall i = 1, 2, ..., n: \; Pred_i = h$ such that $\mathcal{D}_i^h = min(\mathcal{D}_i^1, \mathcal{D}_i^2, ..., \mathcal{D}_i^{|C|})$
12 $\forall i = 1, 2, ..., n: \; Matched_i = 1$ if $Pred_i = E_i$
13 MAXIMIZE: $\frac{1}{n} \sum_{i \in n} Matched_i$; *% fitness function %*
14 RETURN: $\mathcal{F}, \mathcal{C}, \mathcal{M}$

Training Phase. The training module is run for $\mathcal{G}$ generations as described by Algorithm 1. The goal of M_2GP is to maximize the percentage of correctly classified samples, i.e., the classification accuracy (line 13). As M_2GP iterates over the generations, solutions with better training accuracy should be generated. In every generation, a population of new sets of functions $\mathcal{F}$ is generated. The training data is evaluated by every function solution to obtain the mapped data $\mathbf{Z} = \mathcal{F}(\mathbf{X})$, where $\mathbf{Z}$ represents the mapped data in the d-dimensional solution space. Since we are using the training data, and we know the class value for each mapped sample, we cluster/group the mapped samples $\mathbf{Z}$ according to their corresponding class values (line 5). Then we calculate the distance $\mathcal{D}_i$ between every mapped training sample Z_i and the centroid of the clustered mapped data $\mathbf{Z}^k$ for each class $k \in |C|$ (line 6-9).[1] Any mapped sample is predicted to belong to class k if it has the minimum Mahalanobis distance measured against the centroid of

[1] To calculate the Mahalanobis distance, one needs to calculate the inverse of co-variance matrix $\mathcal{C}^k$, and hence $\mathcal{C}^k$ should be non-singular.

Algorithm 2. M_2GP - Classification Module

1 INPUT: $\mathbf{U}, \mathcal{F}, \mathcal{C}, \mathcal{M}, d$
2 OUTPUT: $\mathbf{Pred}$ - Predicted classes
3 EVALUATE: $\mathbf{Z_s} = Eval(f_s(\mathbf{U})), \forall s = 1, 2, ..., d$
4 **for** $k \in 1 \ldots |C|$ **do**
5 $\quad \mathcal{D}_i^k = \sqrt{(Z_i - \mathcal{M}^k) \cdot (C^k)^{-1} \cdot (Z_i - \mathcal{M}^k)^T}, \forall i = 1, 2, ..., n$ *% Mahalanobis distance %*
6 $\quad$ *% where n is the number of samples in the test set %*
7 RETURN: $Pred_i = h$ such that $\mathcal{D}_i^h = min(\mathcal{D}_i^1, \mathcal{D}_i^2, ..., \mathcal{D}_i^{|C|}), \forall i = 1, 2, ..., n;$

the k^{th} $|C|$-clustered mapped data $\mathbf{Z}$ (line 11), which is then used to maximize the fitness function. The importance/usefulness of the Mahalanobis distance, and the choice of dimension d - an important input for M_2GP - will be discussed in Section 5.2. After training for $\mathcal{G}$ generations, we return a classification model $\mathcal{S} = \{\mathcal{F}, \mathcal{C}, \mathcal{M}\}$, comprising of 3 components: $\mathcal{F}$ - a set of d solutions, $\mathcal{C}$ - a set of $|C|$ co-variance matrices of size $d \times d$, and $\mathcal{M}$ - a set of $|C|$ centroid vectors of size $1 \times d$.

Testing phase. To verify the performance of the classification model and to check its generalization ability, we use the evolved model $\mathcal{S}$ to predict the classes for the test data set $\mathbf{U}$. Algorithm 2 describes the procedure to classify the test data set $\mathbf{U}$.

5 Experimental Analysis

5.1 Data Sets

We have used a variety of data sets to test the performance of M_2GP. Table 1 lists the pool of data sets that encompass both real world and synthetic data, having integer and real data types, with varying number of attributes, classes and samples. The 'Heart' (HRT), 'Segment' (SEG), 'Vowel' (VOW), 'Yeast' (YST) and 'movement-libras' (M-L) data sets can be found at KEEL[2] [23], whereas the 'Waveform' (WAV) data set is available at [24]. 'IM-3' and 'IM-10' are the landsat satellite data sets that were used in [25]. All the data sets we have used have no missing values. We have partitioned each data set with the training and test data ratio of 70:30. In order to maintain consistency in the results and perform fair comparisons, we have generated 30 different random partitions for each data set, to be used henceforth.

Table 1. Data sets used for the experimental analysis

Data Set	HRT	IM-3	WAV	SEG	IM-10	YST	VOW	M-L
No. of classes	2	3	3	7	10	10	11	15
No. of attributes	13	6	40	19	6	8	13	90
No. of samples	270	322	5,000	2,310	6,798	1484	990	360

[2] KEEL - dataset repository, `http://keel.es/datasets.php`

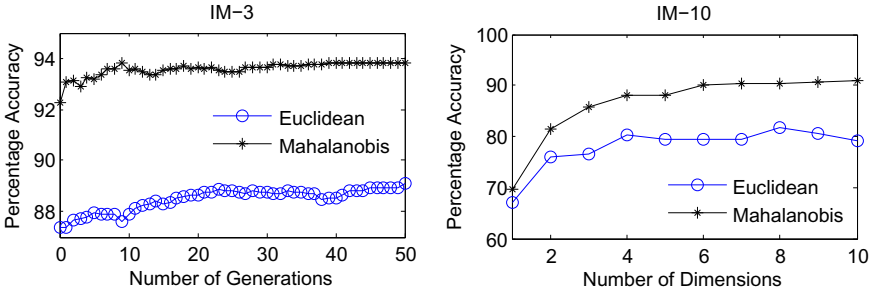

Fig. 2. Percentage accuracy for Euclidean and Mahalanobis distance. For IM-3, dimension d=1; for IM-10, number of generations g=50.

5.2 Experiments with GP Classifiers

Tools and Settings. We have used a modified version of GPLAB[3] [26], version 3 (latest) to conduct all the GP experiments. The GPLAB settings, subject to minimum tuning, are listed next. The remaining GPLAB settings were the ones set by default on the software. Generations = 50 (for analysis) and 100 (for final results - Table 4); Population size = 500 individuals; Crossover / Mutation Rate = 0.9 / 0.1; Function set ($\mathcal{O}$) = {+,-,*,/ (division protected as in [1])}; Terminal set = one variable for each attribute in the data set, plus ephemeral random constants (as in [1]), randomly generated with uniform probability from the interval $[0, 1]$; Tree Initialization: Ramped Half-and-Half with the ratio of 75:25 between the Full and Grow methods.

Importance of Distance Measure. Since one of the crucial steps in M_2GP is to calculate the distance between the sample and the centroid of each class clusters, we have compared two distance measures - Euclidean and Mahalanobis. Figure 2 shows examples of the performance of both distance measures for the data sets IM-3 and IM-10. For IM-3, we plot the mean of 30 different runs for $d = 1$ (which is equivalent to standard GP in terms of solution representation), in order to prevent the effect of using higher dimensions. For IM-10, we plot the mean of 3 different runs for each number of dimensions from 1 to $|C|$. The results were highly consistent, with very low dispersion among the different runs. From these plots we claim that the distance measure indeed plays a significant role in the performance of M_2GP, especially in the higher dimensional solution spaces. Unlike the Euclidean distance, the Mahalanobis distance not only is able to capture the physical distance between the test sample and the class clustered data sets, but also considers the statistical correlation between them, thereby reasserting the work of [27].

Effect of Augmenting Dimensions. As already explained in Algorithm 1, we are representing the mapped input sample $\mathbf{Z}$ (evaluated with the help of d-function solutions) in a d-dimensional solution space. Now, with the help of various data sets, we analyze

[3] GPLAB – A Genetic Programming Toolbox for MATLAB, `http://gplab.sourceforge.net/`

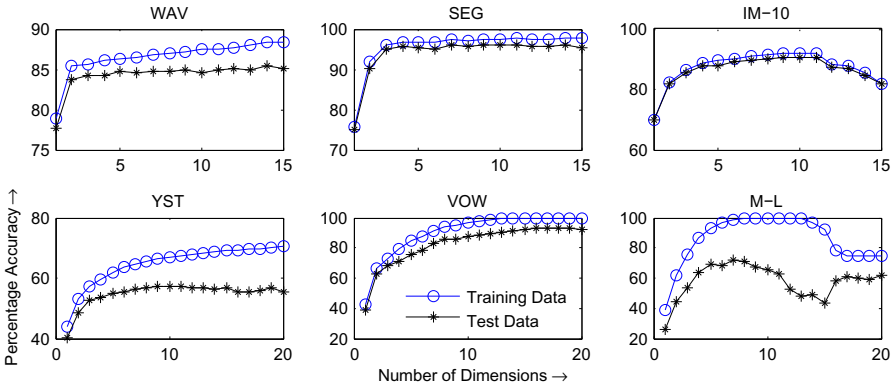

Fig. 3. Variation of accuracy values with increasing number of dimensions for various data sets

the effect of increasing the number of dimensions d. Figure 3 shows plots of accuracy (final values after 50 generations) against the increasing number of dimensions. The plots show the mean values obtained in 10 different runs, randomly chosen from the set of 30 different partitions. Looking at the training lines, we observe two distinct phases as we add dimensions in M_2GP, that we call progressive learning and regressive learning. In the progressive learning phase, the training accuracy improves with increasing dimensions. Progressive learning can be observed in the lower dimensions for all the data sets. If we observe the test lines in the progressive learning phase, we can infer that the classifiers perform the best during this phase, also in terms of generalization. WAV, SEG, YST and VOW enjoy this progressive phase almost until the end of their respective plots. For other data sets, this phase is followed by a regressive learning phase where accuracy degrades with increasing dimensions, as clearly visible in IM-10 and M-L. We hypothesize that a strong regressive learning would be observed for all data sets at much higher dimensions, depending on the complexity of the data set - which in turn depends on n, $\mathcal{A}$ and C. Additionally, it is interesting to observe that the beginning of the regressive phase is always greater than the number of classes for that particular data set, except in the case of M-L. We are regarding M-L as a special category of data set, since it is the only one where $n << |\mathcal{A}| \cdot |C|$, owing to its small sample size. It is also the only data set where the test accuracy decreases substantially during the progressive phase, suggesting strong overfitting. We regard M-L as a challenging data set, in particular when it comes to choosing the best dimension to use.

Choice of Dimension *d*. In Figure 4, we can observe the evolution of the accuracy on both training and test data, for a selected number of dimensions (d = 1, 2, 12, 13) on the IM-10 data set. Looking at the training plot we observe that the accuracy curves of different dimensions remain almost parallel to each other, from the initial generation to the final generation, even when the switch between progressive and regressive phase happens (from d=12 to d=13). This pattern suggests that, if we look at the accuracy values obtained in the initial generation for various dimensions, we will be able to predict which value of d will achieve the best accuracy in the final generation. Indeed, there

Table 2. Automatically chosen dimension d

	IM-3 C=3	WAV C=3	SEG C=7	IM-10 C=10	YST C=10	VOW C=11	M-L C=15
Mean	3.47	5.50	4.53	7.23	6.10	9.27	10.37
Std.Dev.	1.07	2.26	1.17	1.52	2.98	2.90	1.30

exists a very strong correlation of 0.95 ± 0.04 between the initial and the final accuracy values on the training data, when measured for all the data sets from Figure 3, considering their respective set of d values. This helps us choose an appropriate dimension for a particular data set, just by looking at the accuracy values obtained in the initial generation. If we look at the test plots, similar patterns are observed, provided that M_2GP is good with generalization for all d. However this may not be the case, as already observed for M-L in Figure 3. Nevertheless, leaving room for improvement in the future (see Section 6) we adopt a simple procedure to automatically choose the dimension d, described next.

For any given data set, we adopt the best d of the progressive phase. To do this, we keep increasing the number of dimensions as long as the accuracy obtained with the initial generation keeps improving, and adopt as d the value used right before the first degradation occurs. For each problem, the choice of d may be different for different runs. We remark that M_2GP performs best when $d = 1$ for the binary classification data set, and for the remaining data sets we have recorded the chosen d for the 30 runs and obtained the values in Table 2. To validate the appropriateness of these choices, we present in Figure 5 the results on the test set of 30 runs using the automatically chosen d (a-d in the figure) side by side with the results obtained with $d = 1$ and with $d = |C|$ (respectively 1-d and c-d in the figure).

Looking at Figure 5, firstly we observe that in most data sets M_2GP largely improves its performance from 1-d to c-d or a-d. However, deciding the winner between c-d and a-d is not trivial, since their relative performance varies depending on the data set. Choosing $d = |C|$ usually results in a d higher than choosing d automatically (see Table 2), however this does not necessarily translate into a better accuracy. The M-L data

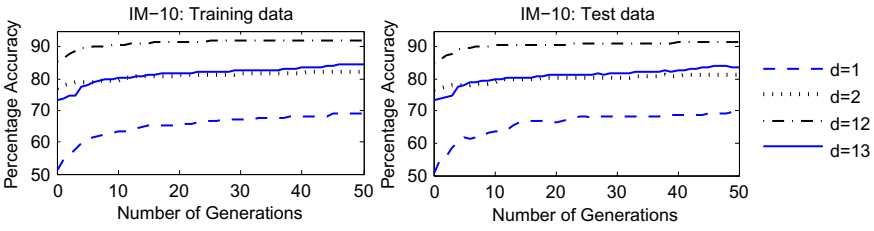

Fig. 4. Witnessing the correlation between the accuracy of the initial solutions and the accuracy of the evolved solutions

Table 3. Mean solution size for 1-d, c-d and a-d

	IM-3	WAV	SEG	IM-10	YST	VOW	M-L
1-d	29.37	107.66	70.57	84.10	56.93	46.23	28.53
c-d	37.03	80.67	39.60	90.27	189.23	40.57	291.06
a-d	24.30	131.97	42.87	123.30	152.06	47.66	42.83

set is the most obvious case where $d = |C|$ is not a good choice when compared to the automatically chosen d. We recall from Figure 3 that on this data set the procedure to automatically choose d does not even do a good job, since the test accuracy starts degrading early in the progressive phase. Still, it is much better than using $d = |C|$. All in all, at least in terms of accuracy, it seems fairly reliable to always use the automatically chosen d.

Size of the Individuals. Here we observe the effect of increasing the number of dimensions on the size of the individuals, i.e. the number of nodes of its tree, including any possible amount of redundant code. Table 3 reports the mean size of 30 different final solutions for each data set and each choice of d (excluding HRT that always uses d=1). We can observe that the increase in solution size from 1-d to c-d or a-d is not so prominent, and in many cases the size is indeed reduced. Therefore, also in terms of solution size the automatically chosen d seems to be a fair choice.

Comparison among GP Classifiers. With the goal of comparing the performance of M_2GP (using the automatically chosen d) with the performance of other GP systems, we chose the 'range selection method with static threshold selection' mentioned in Section 2 [8,16] as the benchmark for comparison, since it is a fairly standard way of performing multi-class classification with GP. However, in data sets with a higher number of classes we immediately observed the often reported inadequacy of this standard GP method to perform multi-class classification. It was losing the race too quickly, so we abandoned any further comparison. Just to provide some numbers, on the WAV and SEG data sets M_2GP improved the accuracy upon the standard method in approximately 25 and 55 percentual points, respectively.

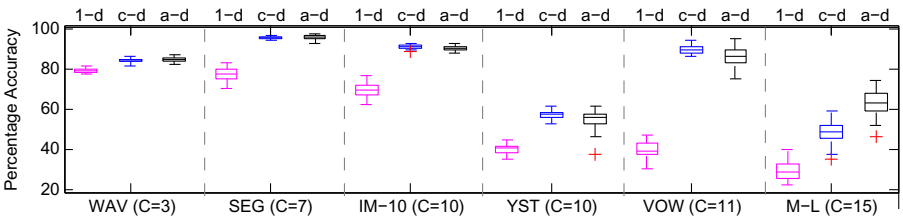

Fig. 5. Percentage accuracy of M_2GP for 3 settings of choosing d, for various data sets

5.3 Comparison with Various Classifiers

We now compare M_2GP with a number of classifiers available in Weka[4], version 3.6.10 (latest). Random Forests (RF) and Decision Trees (J48) are tree based classifiers; Random Subspace (RS) and Multi-Class Classifier (MCC) are meta classifiers; Multi-Layer Perceptron (MLP) and Support Vector Machines (SVM) are function based classifiers. For M_2GP, we set the number of generations to 100; we set dimension $d = 1$ for the binary class data set (HRT), and for the multi-class data sets we choose d automatically during the process of initialization, as already explained. For the rest of the classifiers, we use default settings from Weka. SVM uses the "one-against-one" approach to multi-

Table 4. Comparison among various classifiers. Median accuracy value and Best accuracy value on the test data set for 30 runs are reported. For each problem, the best values among the classifiers are in bold (if more than one, it means there is no statistically significant difference between their medians) and the worst values are in italics (the same). For each problem, a highlighted value means the classifier is significantly better than M_2GP values, while an underlined value means the classifier is significantly worse than M_2GP.

→ Data Set		HRT	IM-3	WAV	SEG	IM-10	YST	VOW	M-L
↓ Classifiers		C=2	C=3	C=3	C=7	C=10	C=10	C=11	C=15
SVM	Median	*55.556*	93.814	**86.3**	*55.844*	90.363	*41.124*	81.818	*14.352*
	Best	*65.432*	*97.938*	88.067	*61.616*	92.055	*46.067*	85.859	*24.074*
J48	Median	79.630	93.814	*74.800*	96.104	94.654	55.169	75.926	63.426
	Best	85.185	**98.969**	*78*	97.691	95.537	57.977	83.838	75.000
RF	Median	80.247	94.845	81.500	**97.258**	**96.861**	**57.528**	**89.394**	71.759
	Best	87.654	**98.969**	83.067	**98.557**	**97.744**	61.124	93.266	76.852
RS	Median	**81.481**	92.784	82.200	95.960	93.919	56.629	82.828	65.741
	Best	90.124	*97.938*	84.400	97.403	95.096	60.674	88.216	74.074
MLP	Median	80.247	95.876	83.333	96.320	90.216	**57.977**	82.492	**75.926**
	Best	87.654	*97.938*	85.200	97.403	91.319	**62.921**	87.542	**84.259**
MCC	Median	**83.951**	95.361	**86.800**	92.424	*81.829*	**57.977**	*57.576*	60.648
	Best	**90.124**	*97.938*	**88.267**	94.228	*83.865*	62.247	*65.657*	72.222
M_2GP	Median	**82.099**	94.845	84.867	95.599	90.191	53.82	85.859	62.963
	Best	88.889	**98.969**	86.467	97.403	92.545	60.225	**94.613**	74.074

class classification, which has comparable performance to "one-against-all" while requiring less training time [28]. Table 4 contains the median and the best accuracy values of the 30 different runs for the test data sets. We have used the same set of 30 different partitions to perform 30 different runs with all the classifiers listed in Table 4. To test for statistical significance of the results, the non-parametric Kruskal-Wallis with Bonferroni correction has been used under the alternative hypothesis that the accuracy values of the different classifiers do not have equal medians.

Table 4 has many things to reveal. First of all, on the IM-3 data set all the classifiers obtained median accuracy values that are not statistically different from each other.

[4] Weka – Waikato Environment for Knowledge Analysis, `http://www.cs.waikato.ac.nz/ml/weka/`

In terms of best accuracy, on this data set M_2GP was one of the classifiers achieving the best value (in bold). Also in the VOW data set M_2GP achieved the best accuracy. Regarding the median accuracy values, M_2GP was one of the best classifiers on HRT (in bold), and never one of the worst classifiers on any of the data sets (in italics). On data sets WAV, YST and VOW, only the best classifiers were able to outperform M_2GP (highlighted values), whereas M_2GP was able to outperform many other classifiers (underlined values), at least one on each data set except IM-3. Recall that on the M-L data set M_2GP was not able to choose the ideal d, otherwise it would probably outperform more classifiers. Regarding the comparison with the other function based classifiers (MLP and SVM), M_2GP was clearly superior to SVM in almost all problems, and fairly competitive with MLP, which together with MCC was one of the best classifiers. RF was, however, the clear winner, in particular on the data sets with a higher number of classes.

6 Conclusions and Future Directions

We have proposed a novel approach for representing solutions, both in terms of solution tree and solution space, to address multi-class classification problems with GP. At the core, we have experimentally analyzed the effect of increasing the number of dimensions used to represent the solution space, and we have proposed a simple yet effective and relatively cheap way of choosing an appropriate dimension.

From the idea of multi-dimensional solution representation, we have proposed an efficient classification algorithm - M_2GP - and compared its performance with many of the best state-of-the-art methods for multi-class classification. We have shown that M_2GP offers competitive results on a large variety of data sets. We have also realized that being able to improve the choice of the number of dimensions would allow us to improve the competitiveness of M_2GP even further.

Naturally, we will focus our future work on better choosing the number of dimensions of the solution space. In fact, we will explore the idea that the number of dimensions does not have to be a parameter of M_2GP, and instead can be implicitly evolved together with the solution itself. The issue of overfitting should be taken into account in these future studies, and we also want to perform detailed analyses of diversity and bloat, with a strong focus on the interpretability of the M_2GP solutions.

Although still in its infancy, this new approach has already been able to elevate GP to a competitive method for multi-class classification, and we believe it represents the first step towards a general framework for multi-class classification with GP.

Acknowledgments. The authors acknowledge projects EnviGP (PTDC/EIA-CCO/ 103363/ 2008) and MaSSGP (PTDC/EEI-CTP/2975/2012), FCT, Portugal.

References

1. Koza, J.R.: Genetic Programming: On the programming of computers by means of natural selection, vol. 1. MIT Press (1992)
2. Poli, R., Langdon, W.B., Mcphee, N.F.: A field guide to genetic programming (March 2008)
3. Langdon, W., Poli, R.: Foundations of Genetic Programming. Springer (2002)

4. Special issue on bridging the gap between theory and practice in evolutionary algorithms research. Evolutionary Computation 15(4) (2007)
5. Castelli, M., Silva, S., Vanneschi, L., Cabral, A., Vasconcelos, M.J., Catarino, L., Carreiras, J.M.B.: Land cover/Land use multiclass classification using GP with geometric semantic operators. In: Esparcia-Alcázar, A.I. (ed.) EvoApplications 2013. LNCS, vol. 7835, pp. 334–343. Springer, Heidelberg (2013)
6. Muni, D., Pal, N., Das, J.: A novel approach to design classifiers using genetic programming. IEEE Transactions on Evolutionary Computation 8(2), 183–196 (2004)
7. Zhang, M., Ciesielski, V.: Genetic programming for multiple class object detection. In: Foo, N.Y. (ed.) AI 1999. LNCS (LNAI), vol. 1747, pp. 180–192. Springer, Heidelberg (1999)
8. Zhang, M., Smart, W.: Multiclass object classification using genetic programming. In: Raidl, G.R., et al. (eds.) EvoWorkshops 2004. LNCS, vol. 3005, pp. 369–378. Springer, Heidelberg (2004)
9. Espejo, P., Ventura, S., Herrera, F.: A survey on the application of genetic programming to classification. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 40(2), 121–144 (2010)
10. Bojarczuk, C.C., Lopes, H.S., Freitas, A.A.: Genetic programming for knowledge discovery in chest-pain diagnosis. IEEE Engineering in Medicine and Biology Magazine 19(4), 38–44 (2000)
11. Sakprasat, S., Sinclair, M.: Classification rule mining for automatic credit approval using genetic programming. In: IEEE Congress on Evolutionary Computation, CEC 2007, pp. 548–555 (2007)
12. Shen, S., Sandham, W., Granat, M., Dempsey, M.F., Patterson, J.: A new approach to brain tumour diagnosis using fuzzy logic based genetic programming. In: Proceedings of the 25th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, vol. 1, pp. 870–873 (2003)
13. Falco, I.D., Cioppa, A.D., Tarantino, E.: Discovering interesting classification rules with genetic programming. Applied Soft Computing 1(4), 257–269 (2002)
14. Tan, K.C., Tay, A., Lee, T., Heng, C.M.: Mining multiple comprehensible classification rules using genetic programming. In: Proceedings of the 2002 Congress on Evolutionary Computation, CEC 2002, vol. 2, pp. 1302–1307 (2002)
15. Tackett, W.A.: Genetic programming for feature discovery and image discrimination. In: Proceedings of the 5th International Conference on Genetic Algorithms, pp. 303–311. Morgan Kaufmann Publishers Inc., San Francisco (1993)
16. Li, X.M., Wang, M., Cui, L.J., Huang, D.M.: A new classification arithmetic for multi-image classification in genetic programming. In: 2007 International Conference on Machine Learning and Cybernetics, vol. 3, pp. 1683–1687 (2007)
17. Kishore, J.K., Patnaik, L., Mani, V., Agrawal, V.K.: Application of genetic programming for multicategory pattern classification. IEEE Transactions on Evolutionary Computation 4(3), 242–258 (2000)
18. Silva, S., Tseng, Y.-T.: Classification of seafloor habitats using genetic programming. In: Giacobini, M., et al. (eds.) EvoWorkshops 2008. LNCS, vol. 4974, pp. 315–324. Springer, Heidelberg (2008)
19. Lin, J.Y., Ke, H.R., Chien, B.C., Yang, W.P.: Classifier design with feature selection and feature extraction using layered genetic programming. Expert Systems With Applications 34(2), 1384–1393 (2008)
20. Teredesai, A., Govindaraju, V.: Issues in evolving gp based classifiers for a pattern recognition task. In: Congress on Evolutionary Computation, CEC 2004, vol. 1, pp. 509–515 (2004)
21. Winkler, S., Affenzeller, M., Wagner, S.: Advanced genetic programming based machine learning. Journal of Mathematical Modelling and Algorithms 6(3), 455–480 (2007)

22. Jabeen, H., Baig, A.R.: Two-stage learning for multi-class classification using genetic programming. Neurocomputing 116, 311–316 (2013)
23. Alcala-Fdez, J., Fernandez, A., Luengo, J., Derrac, J., Garcia, S., Sanchez, L., Herrera, F.: Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. Journal of Multiple-Valued Logic and Soft Computing 17(2-3), 255–287 (2011)
24. Bache, K., Lichman, M.: (uci) machine learning repository, university of California, Irvine, school of information and computer sciences (2013), http://archive.ics.uci.edu/ml
25. U.S. geological survey (usgs) earth resources observation systems (eros) data center (edc), http://glovis.usgs.gov/
26. Silva, S., Almeida, J.: GPLAB - A Genetic Programming Toolbox for MATLAB. In: Proc. of the Nordic MATLAB Conference, NMC 2003, pp. 273–278 (2005)
27. Xiang, S., Nie, F., Zhang, C.: Learning a mahalanobis distance metric for data clustering and classification. Pattern Recognition 41(2), 3600–3612 (2008)
28. Hsu, C.W., Lin, C.J.: A comparison of methods for multi-class support vector machines. IEEE Transactions on Neural Networks 13(2), 415–425 (2002)