M3GP – Multiclass Classification with GP

Conference Paper · April 2015

DOI: 10.1007/978-3-319-16501-1_7

3 authors:

Luis Muñoz
Tijuana Institute of Technology
14 PUBLICATIONS **68** CITATIONS
SEE PROFILE

Sara Silva
University of Lisbon
101 PUBLICATIONS **1,919** CITATIONS
SEE PROFILE

Leonardo Trujillo
Tijuana Institute of Technology
156 PUBLICATIONS **1,464** CITATIONS
SEE PROFILE

Metadata of the chapter that will be visualized in SpringerLink

Book Title	Genetic Programming	
Series Title		
Chapter Title	M3GP – Multiclass Classification with GP	
Copyright Year	2015	
Copyright HolderName	Springer International Publishing Switzerland	
Author	Family Name	**Muñoz**
	Particle	
	Given Name	**Luis**
	Prefix	
	Suffix	
	Division	Tree-Lab, Posgrado En Ciencias de la Ingeniería
	Organization	Instituto Tecnológico de Tijuana
	Address	Blvd. Industrial Y Av. ITR Tijuana S/N, Mesa Otay C.P., 22500 , Tijuana, BC, Mexico
	Email	lmunoz@tectijuana.edu.mx
Corresponding Author	Family Name	**Silva**
	Particle	
	Given Name	**Sara**
	Prefix	
	Suffix	
	Division	BioISI – Biosystems and Integrative Sciences Institute, Faculty of Sciences
	Organization	University of Lisbon
	Address	Lisbon, Portugal
	Division	NOVA IMS
	Organization	Universidade Nova de Lisboa
	Address	1070-312, Lisboa, Portugal
	Division	CISUC, Department of Informatics Engineering
	Organization	University of Coimbra
	Address	Coimbra, Portugal
	Email	sara@fc.ul.pt
Author	Family Name	**Trujillo**
	Particle	
	Given Name	**Leonardo**
	Prefix	
	Suffix	
	Division	Tree-Lab, Posgrado En Ciencias de la Ingeniería
	Organization	Instituto Tecnológico de Tijuana
	Address	Blvd. Industrial Y Av. ITR Tijuana S/N, Mesa Otay C.P., 22500 , Tijuana, BC, Mexico
	Email	leonardo.trujillo@tectijuana.edu.mx

Abstract	Data classification is one of the most ubiquitous machine learning tasks in science and engineering. However, Genetic Programming is still not a popular classification methodology, partially due to its poor performance in multiclass problems. The recently proposed M2GP - Multidimensional Multiclass Genetic Programming algorithm achieved promising results in this area, by evolving mappings of the p-dimensional data into a d-dimensional space, and applying a minimum Mahalanobis distance classifier. Despite good performance, M2GP employs a greedy strategy to set the number of dimensions d for the transformed data, and fixes it at the start of the search, an approach that is prone to locally optimal solutions. This work presents the M3GP algorithm, that stands for M2GP with multidimensional populations. M3GP extends M2GP by allowing the search process to progressively search for the optimal number of new dimensions d that maximize the classification accuracy. Experimental results show that M3GP can automatically determine a good value for d depending on the problem, and achieves excellent performance when compared to state-of-the-art-methods like Random Forests, Random Subspaces and Multilayer Perceptron on several benchmark and real-world problems.
Keywords (separated by '-')	Genetic programming - Classification - Multiple classes - Multidimensional clustering

M3GP – Multiclass Classification with GP

Luis Muñoz[1], Sara Silva[2,3,4](✉), and Leonardo Trujillo[1]

[1] Tree-Lab, Posgrado En Ciencias de la Ingeniería, Instituto Tecnológico de Tijuana, Blvd. Industrial Y Av. ITR Tijuana S/N, Mesa Otay C.P., 22500 Tijuana, BC, Mexico
{lmunoz,leonardo.trujillo}@tectijuana.edu.mx

[2] BioISI – Biosystems and Integrative Sciences Institute, Faculty of Sciences, University of Lisbon, Lisbon, Portugal

[3] NOVA IMS, Universidade Nova de Lisboa, 1070-312 Lisboa, Portugal

[4] CISUC, Department of Informatics Engineering, University of Coimbra, Coimbra, Portugal
sara@fc.ul.pt

Abstract. Data classification is one of the most ubiquitous machine learning tasks in science and engineering. However, Genetic Programming is still not a popular classification methodology, partially due to its poor performance in multiclass problems. The recently proposed M2GP - Multidimensional Multiclass Genetic Programming algorithm achieved promising results in this area, by evolving mappings of the p-dimensional data into a d-dimensional space, and applying a minimum Mahalanobis distance classifier. Despite good performance, M2GP employs a greedy strategy to set the number of dimensions d for the transformed data, and fixes it at the start of the search, an approach that is prone to locally optimal solutions. This work presents the M3GP algorithm, that stands for M2GP with multidimensional populations. M3GP extends M2GP by allowing the search process to progressively search for the optimal number of new dimensions d that maximize the classification accuracy. Experimental results show that M3GP can automatically determine a good value for d depending on the problem, and achieves excellent performance when compared to state-of-the-art-methods like Random Forests, Random Subspaces and Multilayer Perceptron on several benchmark and real-world problems. AQ1

Keywords: Genetic programming · Classification · Multiple classes · Multidimensional clustering

1 Introduction

Genetic programming (GP) [10] has been used to solve many difficult problems from various domains, an extensive list of noteworthy examples are reviewed in [7]. However, probably the most straightforward formulation for a GP search is to apply it in supervised machine learning problems, particularly symbolic regression and data classification. In particular, this paper is concerned with the

P. Machado et al. (Eds.): EuroGP 2015, LNCS 9025, pp. 1–14, 2015.
DOI: 10.1007/978-3-319-16501-1_7

latter, an area in which a variety of proposals have been developed [3]. Even though GP has been used to achieve state-of-the-art performance in several benchmark problems and real-world scenarios, it has been particularly difficult to use in multiclass problems [5].

In general, for a supervised classification problem some pattern $\mathbf{x} \in \mathbb{R}^p$ has to be classified in one of M classes $\omega_1, \ldots, \omega_M$ using a training set $\mathcal{X}$ of N p-dimensional patterns with a known class label. Then, the goal is to build a mapping $g(\mathbf{x}) : \mathbb{R}^p \rightarrow M$, that assigns each pattern $\mathbf{x}$ to a corresponding class ω_i, where g is derived based on the evidence provided by $\mathcal{X}$. In these problems fitness is usually assigned in two general ways. One approach is to use a *wrapper* method, where GP is used as a feature extraction method that performs the transformation $k(\mathbf{x}) : \mathbb{R}^p \rightarrow \mathbb{R}^d$, and then another classifier is used to measure the quality of the transformation based on accuracy or another performance measure. The second approach is to use GP to evolve g directly, performing the feature transformation step implicitly. However, current techniques have left room for improvement, such as automatically determining the proper value for d or dealing with multiclass problems (with $M > 2$).

This paper presents an extension of the recently proposed Multidimensional Multiclass Genetic Programming (M_2GP, from now on M2GP) algorithm [5], a wrapper-based GP classifier that effectively deals with multiclass problems by performing a multidimensional transformation of the input data. The M2GP algorithm uses a fixed number of new feature dimensions d, that must be chosen and fixed before the run starts. On the other hand, the algorithm proposed in this paper is able to heuristically determine an appropriate value for d during the run. To achieve this, the algorithm includes specialized search operators that can increase or decrease the number of feature dimensions produced by each tree, and that allow the search to maintain a population of different transformation functions k that construct a different number of new features dimensions. The proposed algorithm is named M3GP, which stands for M2GP with multidimensional populations.

The remainder of this paper is organized as follows. Section 2 briefly reviews previous works related to the present contribution. Section 3 describes the original M2GP algorithm, explaining how it works and referring to its strengths and its major weakness. Section 4 explains the new improved version of the algorithm, M3GP. Section 5 describes the experiments performed, while Sect. 6 reports and discusses all the results obtained. Finally, Sect. 7 concludes and describes some future work.

2 Related Work

Espejo et al. [3] present a comprehensive discussion on GP-based classification methods, while Ingalalli et al. [5] surveys work specifically focused on multiclass classification with GP. Here we briefly address previous works that present a similar goal as the one outlined for M3GP (besides M2GP), highlighting the main differences to the present contribution.

Lit et al. [8] proposed a layered multipopulation approach, where each layer has d populations, and each population produces a single transformation

$k(\mathbf{x}) : \mathbb{R}^p \rightarrow \mathbb{R}$, and classification is performed based on a threshold. While each population is evaluated independently, all of them are combined to generate new feature vectors of dimension d, which are given as input to a new layer, and only the final layer has a single population with $d = 1$. For multiclass problems a Euclidean distance classifier is used and results show the method improves the search efficiency and reduces training time. However, the approach does not improve upon the performance of a standard GP classifier, it is not tested on problems with many classes (highest is $M = 3$), and it requires an a priori setting for the number of layers and populations used in each layer.

Another, more closely related work, is the one presented by Zhang and Rockett [12], who propose a multidimensional feature extraction method that uses a similar solution representation to the one used in M2GP and M3GP. However, the authors set a fixed limit on the maximum number of feature dimensions, set to $d = 50$, and initialize the population with trees that use different number of features within this range. Other important difference is that the authors use a multiobjective search process considering class separation and solution size, and do not explicitly consider multiclass problems, instead relying on a hierarchical nesting of binary classifiers.

3 The M2GP Multiclass Classification Method

M2GP is a recent and innovative method of performing multiclass classification with GP [5]. It has shown to be competitive with the best state-of-the-art classifiers in a wide range of benchmark and real-world problems, something that GP had not achieved.

The algorithm and its strengths. The basic idea of M2GP is to find a transformation, such that the transformed data of each class can be grouped into unique clusters. In M2GP the number of dimensions in which the clustering is performed is completely independent from the number of classes, such that high dimensional datasets may be easily classified by a low dimensional clustering, while low dimensional datasets may be better classified by a high dimensional clustering.

In order to achieve this, M2GP uses a representation for the solutions that allows them to perform the mapping $k(\mathbf{x}) : \mathbb{R}^p \rightarrow \mathbb{R}^d$. The representation is basically the same used for regular tree-based GP, except that the root node of the tree exists only to define the number of dimensions d of the new space. Each branch stemming directly from the root performs the mapping in one of the d dimensions. The genetic operators are the regular subtree crossover and mutation, except that the root is never chosen as the crossing or mutation node. However, the truly specialized element of M2GP is the fitness function. Each individual is evaluated in the following way:

- All the p-dimensional samples of the training set are mapped into the new d-dimensional space (each branch of the tree is one of the d dimensions).
- On this new space, for each of the M classes in the data, the covariance matrix and the cluster centroid is calculated from the samples belonging to that class.

– The Mahalanobis distance between each sample and each of the M centroids is calculated. Each sample is assigned the class whose centroid is closer. Fitness is the accuracy of this classification (the percentage of samples correctly classified).

Figure 1 shows an example of clustering of a dataset. The original data, regardless of how many features, or attributes, it contains, is mapped into a new 3-dimensional space by a tree whose root note has three branches, each performing the mapping on each of the three axes X, Y, Z. The fact that the data contains three classes is purely coincidental - it could contain any number of classes, regardless of the dimension of the space. On the left, the clustering was obtained by an individual with low accuracy; on the right, the same data clustered by an individual with accuracy close to 100 %. The class centroids are marked with large circles.

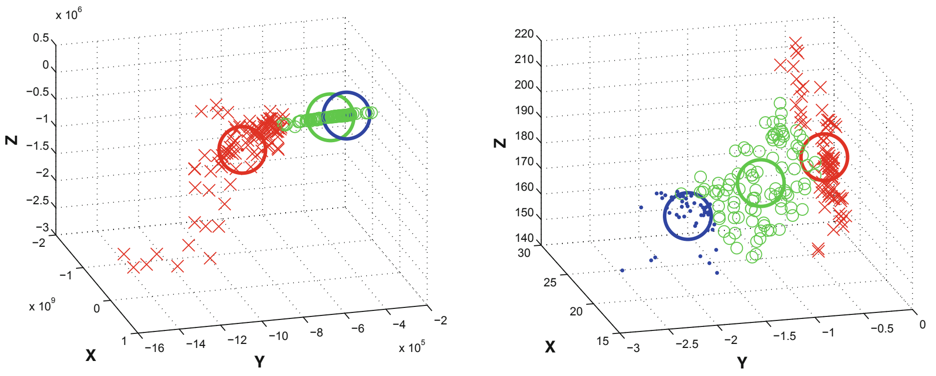

Fig. 1. Example of clustering of a dataset. On the left, clustering obtained by an individual with low accuracy; on the right, the same data clustered by an individual with very good accuracy. The large circles represent the centroids.

At the end of the run, the solution given to the user is composed not only of the tree of the best individual, but also of the respective covariance matrices and cluster centroids. In order to classify unseen data, M2GP uses the tree to map the new samples into the new space, and then uses the covariance matrices and the cluster centroids in order to determine the minimum Mahalanobis distance between each sample and each centroid. (Note that the covariance matrices and cluster centroids are not recalculated when classifying new data). The choice of the Mahalanobis distance instead of the Euclidean distance is not an unnecessary complication of the algorithm, as it allowed a substantial improvement on the quality of the results achieved [5].

M2GP produces trees that are not very large (mean solution size for different problems was reported to range from 24 to 152 nodes [5]), and a higher number of dimensions does not necessarily translate into larger trees.

The weakness. Despite its competitiveness, M2GP suffers from a drawback: how to choose the right number of dimensions for a given problem? M2GP is incapable of adding or removing dimensions during the evolution, so the number of dimensions d is fixed in the beginning of the run. M2GP chooses d based on the observation that the best fitness found in the initial generation is highly correlated with the best fitness found on the final generation [5].

Therefore, before initiating a run, M2GP runs a procedure that iteratively initializes different populations with increasing dimensions (we mean the dimension d mentioned earlier, not the number of individuals in the population) and checks which of these initial populations achieves the best fitness. Starting with $d = 1$, this procedure adds one more dimension and initializes one more population as long as the fitness continues to improve from the previous population. As soon as adding one more dimension degrades fitness, the procedure stops and the dimension yielding the best initial fitness is chosen.

4 M3GP – M2GP with Multidimensional Populations

As described in the previous section, the original M2GP uses a greedy approach to determine how many dimensions the evolved solutions should have. It may happen that by fixing the number of dimensions in the beginning of the run, the algorithm is being kept from finding better solutions during the search, ones that may use a different number of dimensions. In our new improved version, the algorithm evolves a population that may contain individuals of several different dimensions. The genetic operators may add or remove dimensions, and it is assumed that selection will be sufficient to discard the worst ones and maintain the best ones in the population. The next subsections describe M3GP, which stands for M2GP with multidimensional populations.

Initial population. M3GP starts the evolution with a random population where all the individuals have only one dimension. This ensures that the evolutionary search begins looking for simple, one dimensional solutions, before moving towards higher dimensional solutions, which might also be more complex.

For M2GP, a Ramped Half-and-Half initialization [6] skewed to 25 % Grow and 75 % Full was recommended [5], suggesting that a higher proportion of full trees facilitates the initial evolution. Because all the initial M3GP individuals are unidimensional, it makes sense to believe that the need for bigger initial trees is even higher. Therefore, all the individuals in the initial M3GP population are created using the Full initialization method [6]. Additionally to the Full initialization, there was also an attempt to use deeper initial trees of depth 9 instead of 6. However, preliminary results did not show any improvement, and therefore the traditional initial depth of 6 levels was used.

Mutation. During the breeding phase, whenever mutation is the chosen genetic operator, one of three actions is performed, with equal probability: (1) standard subtree mutation, where a randomly created new tree replaces a randomly chosen branch (excluding the root node) of the parent tree; (2) adding a randomly created new tree as a new branch of the root node, effectively adding one dimension to the parent tree; and (3) randomly removing a complete branch of the root node, effectively removing one dimension from the parent tree.

As mentioned previously, M3GP begins with a population that only contains unidimensional individuals. From here, the algorithm has to be able to explore several different dimensions. In M3GP mutation is the only way of adding and removing dimensions, and therefore we have increased its probability of occurrence from 0.1 (used in M2GP [5]) to 0.5, to guarantee a proper search for the right dimension. Preliminary results have confirmed that a higher mutation rate indeed improves the fitness.

Crossover. Whenever crossover is chosen, one of two actions is performed, with equal probability: (1) standard subtree crossover, where a random node (excluding the root node) is chosen in each of the parents, and the respective branches swapped; (2) swapping of dimensions, where a random complete branch of the root node is chosen in each parent, and swapped between each other, effectively swapping dimensions between the parents. The second event is just a particular case of the first, where the crossing nodes are guaranteed to be directly connected to the root node.

Pruning. Mutation, as described above, makes it easy for M3GP to add dimensions to the solutions. However, many times some of the dimensions actually degrade the fitness of the individual, so they would be better removed. Mutation can also remove dimensions but, as described above, it does so randomly and blind to fitness. To maintain the simplicity and complete stochasticity of the genetic operators, we have decided not to make any of them more 'intelligent', and instead we remove the detrimental dimensions by pruning the best individual after the breeding phase.

The pruning procedure removes the first dimension and reevaluates the tree. If the fitness improves, the pruned tree replaces the original and goes through pruning of the next dimension. Otherwise, the pruned tree is discarded and the original tree goes through pruning of the next dimension. The procedure stops after pruning the last dimension.

Pruning is applied only to the best individual in each generation. Applying it to all the individuals in the population could pose two problems: (1) a significantly higher computational demand, where a considerable amount of effort would be spent on individuals that would still be unfit after pruning; (2) although not confirmed, the danger of causing premature convergence due to excessive removal of genetic material, the same way that code editing has shown to cause it [4].

Preliminary experiments have revealed that pruning the best individual of each generation shifts the distribution of the number of dimensions to lower values (or prevents it from shifting to higher values so easily) during the evolution, without harming fitness.

Elitism. It was mentioned earlier that, in order to explore solutions of different dimensions, M3GP relies on mutation to add and remove dimensions from the individuals, with a fairly high probability. It also has to rely on selection to keep the best dimensions in the population and discard the worst ones. The way to do this is by ensuring some elitism on the survival of the individuals from one generation to the next. M3GP does not allow the best individual of any generation to be lost, and always copies it to the next generation. Let us recall that this individual is already optimized in the sense that it went through pruning. Preliminary experiments have shown that elitism is indeed able to improve fitness.

5 Experimental Setup

This section describes the experiments performed to assess the performance of M3GP, in particular when compared to M2GP and other state-of-the-art classifiers.

Datasets. A set of eight problems was used for the experiments, the same used for M2GP [5]. This set contains 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) datasets can be found at the KEEL dataset repository [1], whereas the 'waveform' (WAV) dataset is available at [2]. 'IM-3' and 'IM-10' are the satellite datasets used in [11]. All the original datasets were randomly split in 70 % training and 30 % test sets, the same proportion as with M2GP [5]. Table 1 summarizes the main characteristics of each dataset.

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	5000	2310	6798	1484	990	360

Tools. A modified version of GPLAB 3 was used to execute all the runs of M3GP. GPLAB is an open source GP toolbox for MATLAB, freely available at http://gplab.sourceforge.net. For the comparison with the state-of-the-art classifiers, we have used Weka 3.6.10. Weka is also open source, and freely available at http://www.cs.waikato.ac.nz/ml/weka/.

Parameters. Table 2 summarizes the parameters adopted for running M3GP. Some are the default parameters of GPLAB, unchanged from M2GP, while others have already been described in the previous section. In Weka we have used the default parameters and configurations for each algorithm.

6 Results and Discussion

This section presents comparative results between M2GP and M3GP, and also between M3GP and some of the best state-of-the-art classification methods used in machine learning.

Table 2. Running parameters of M3GP.

Runs	30
Population size	500 individuals
Generations	100 generations
Initialization	*6-depth Full initialization* [6]
Operator probabilities	Crossover $p_c = 0.5$, Mutation $p_\mu = 0.5$
Function set	$(+, -, \times, \div$ protected as in [6])
Terminal set	Ephemeral random constants [0,1]
Bloat control	17-depth limit [6]
Selection	Lexicographic tournament [9] of size 5
Elitism	Keep best individual

6.1 M2GP Versus M3GP

The comparison between M2GP and M3GP will be presented in terms of fitness, and in terms of number of nodes and number of dimensions of the solutions. Whenever a result is said to be significantly different (better or worse) from another, it means the difference is statistically significant according to the Wilcoxon's rank sum test for equal medians, performed at the 0.01 significance level.

Figures 2 and 3 show two sets of boxplots. Figure 2 reports the fitness obtained by the best individuals on each of the 30 training sets, while Fig. 3 reports the fitness obtained by these same individuals on the respective test sets. From now on we will call these the training fitness and the test fitness, respectively.

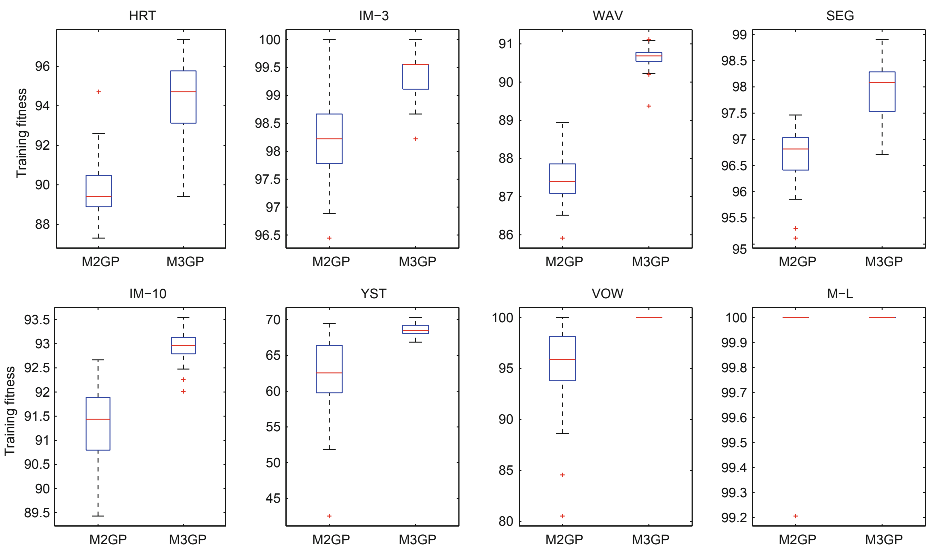

Fig. 2. Training fitness, given by classification accuracy, of M2GP and M3GP on all problems.

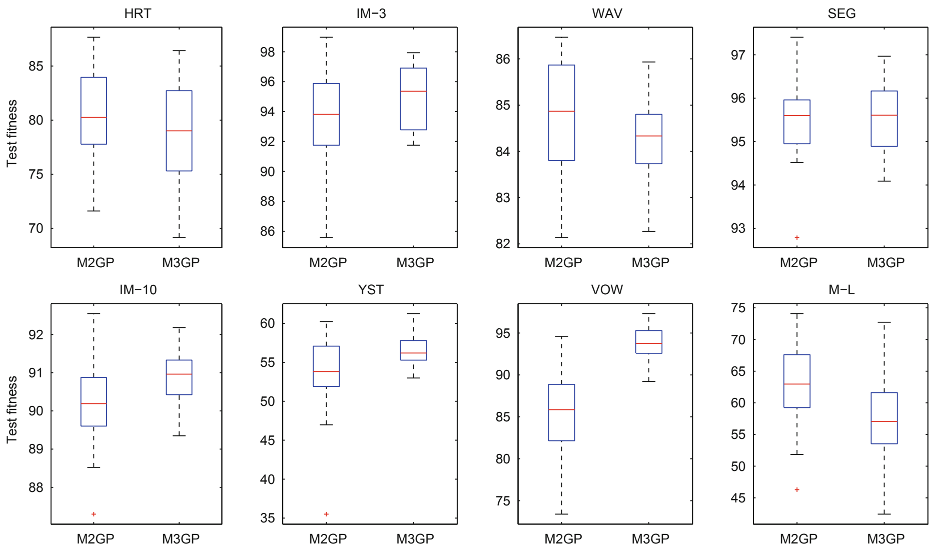

Fig. 3. Test fitness, given by classification accuracy, of M2GP and M3GP on all problems.

We may also call them training accuracy and test accuracy, respectively, since fitness is the accuracy of the classification. In each of these figures there is one boxplot for each problem. Each boxplot contains a pair of whiskered boxes, the first reporting the results of M2GP (most already reported in [5]) and the second reporting the results of M3GP.

It is clearly visible that M3GP achieves higher training fitness, which means it learns easier than M2GP, in all problems (in M-L the results of M2GP and M3GP are equal except for the outlier in M2GP). See Table 3 for numeric results and their statistical significance. M3GP is also able to achieve higher test fitness than M2GP in half of the problems. Once again, refer to Table 3 for the significance of these results.

Table 3 shows some quantitative results regarding the training and test fitness, also adding information on the number of nodes of the best individuals, as well as their number of dimensions. All these results refer to the median of the 30 runs. The best approach (between M2GP and M3GP) on each problem is marked in bold - both are marked when the difference is not statistically significant. In terms of size, we also consider lower to be better. However, we do not evaluate the number of dimensions qualitatively, since a higher number of dimensions does not necessarily translate into a larger number of nodes and/or lower interpretability of the solutions. We do include additional information for the number of dimensions, which is the minimum and maximum values obtained in the 30 runs.

Table 3 shows that, in terms of training fitness, M3GP is significantly better than M2GP in all the problems (except the last, M-L, where the results are

Table 3. Comparison between M2GP and M3GP.

	HRT	IM-3	WAV	SEG	IM-10	YST	VOW	M-L
Training fitness								
M2GP	89.4	98.2	87.4	96.8	91.4	62.6	95.9	**100**
M3GP	**94.7**	**99.6**	**90.7**	**98.1**	**93.0**	**68.5**	**100**	**100**
Test fitness								
M2GP	**80.2**	**93.8**	**84.9**	**95.6**	90.2	53.8	85.9	**63.0**
M3GP	**79.0**	**95.4**	**84.3**	**95.6**	**91.0**	**56.2**	**93.8**	57.1
Number of nodes								
M2GP	**37**	**24**	126	**43**	**117**	**146**	**49**	33
M3GP	110	66	**71**	111	239	274	**53**	**13**
Number of dimensions								
M2GP	2.5 *(1-8)*	2 *(1-4)*	5 *(2-10)*	4 *(3-8)*	7 *(4-10)*	5.5 *(1-13)*	9 *(4-18)*	10 *(7-12)*
M3GP	12 *(1-17)*	5 *(2-8)*	31 *(29-37)*	11 *(5-21)*	12 *(11-16)*	13 *(11-18)*	20 *(16-20)*	12 *(10-13)*

considered the same), while in terms of test fitness M3GP is better or equal to M2GP in all problems (except M-L). It is interesting to note that it is in the higher dimensional problems (except M-L) that M3GP achieves better results than M2GP (the problems are roughly ordered by dimensionality of the data). Problem M-L had already been identified as yielding a different behavior than the others [5], and here once again it is often the exception to the rule. Our explanation for M3GP not being able to perform better on this problem is the extreme easiness it has in reaching maximal accuracy. Both M2GP and M3GP achieve 100 % training accuracy, but M3GP does it in only a few generations (not shown), producing very small and accurate solutions that barely generalize to unseen data. On the other hand, M2GP does not converge immediately, so in its effort to learn the characteristics of the data it also evolves some generalization ability.

Regarding the size of the solutions, in most problems where M3GP brought improvements, it also brought significantly larger trees, except for WAV and M-L where the M3GP trees are significantly smaller, and VOW where the sizes are the same. However, when we split the nodes of the M3GP trees among their several dimensions, even the largest trees (e.g., in IM-10 and YST) seem to be simple and manageable (around 20 nodes per dimension), in particular when we consider that no simplification has been done except for the pruning of detrimental dimensions (see Sect. 4), and therefore the effective size of the trees may be even smaller.

Regarding the number of dimensions used in M2GP and M3GP, two things become clear. The first one is that there seems to be no single optimal number of dimensions for a given problem, since both M2GP and M3GP may choose wildly different values, depending on the run. The second one is that M3GP tends to use a larger number of dimensions than M2GP. What these numbers do not show is that different problems result in very different behaviors with respect to the evolution of the number of dimensions. Figure 4 illustrates two

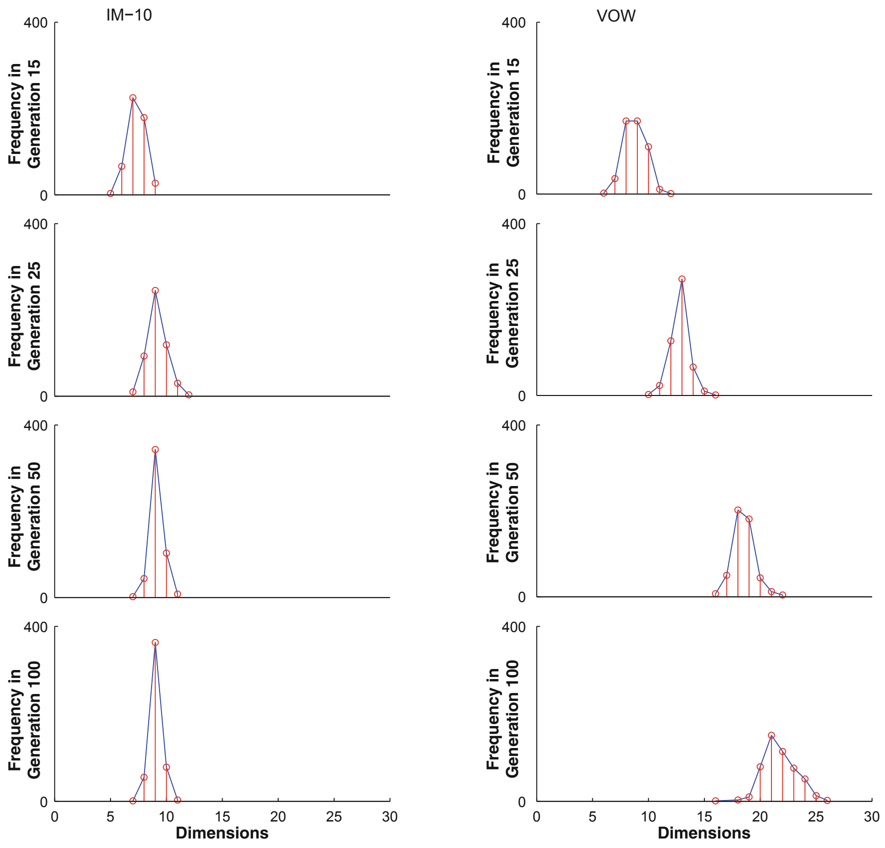

Fig. 4. Distribution of the number of dimensions in the population in generations 15, 25, 50 and 100 (top to bottom). On the left, a typical run of problem IM-10. On the right, a typical run of problem VOW.

main types of behavior, described next. In most problems the distribution of the number of dimensions moves rapidly to higher values in the beginning of the run, and then remains stable and more or less in the same range until the end of the run (exemplified on the left in Fig. 4). However, in some problems, like WAV and VOW, the distribution of the number of dimensions does not settle during the 100 generations of the run, and instead keeps moving towards higher values (exemplified on the right in Fig. 4). The WAV problem goes as high as 37 dimensions, and curiously this is one of the problems where M3GP produces significantly smaller trees than M2GP. VOW is another of the few problems where the M3GP trees are not larger than the M2GP trees. The only other such case is the unique M-L problem.

6.2 M3GP Versus State-of-the-art

The comparison between M3GP and the state-of-the-art classification methods is based only on training and test fitness. Based on the comparison previously done between M2GP and several state-of-the-art methods [5], we have decided to compare M3GP with a tree based classifier (RF - Random Forests), a meta classifier (RS - Random Subspace), and a function based classifier (MLP - Multi Layer Perceptron). The three of them were well ranked in the previous comparison with M2GP [5]. We have also included M2GP in this comparison to check how much better M3GP compares to the state-of-the-art than M2GP.

Table 4 reports and compares the training and test fitness obtained by RF, RS, MLP, M2GP and M3GP on the same eight problems, medians of 30 runs. The best approach on each problem is marked in bold, or several when their differences are not statistically significant. Looking at the first row, it is undeniable that RF is an almost unbeatable method when it comes to training fitness. Still, it is beaten by M3GP in the last two problems (VOW and M-L). (M2GP achieves the same feat in only one of them, M-L).

However, training fitness is not important unless accompanied by good test fitness, suggesting good generalization ability. Although RF is also good in test fitness, M3GP is able to achieve similar results. Like RF, M3GP is ranked first in five of the eight problems (M2GP achieves this is only two problems). Like RF, M3GP is not equaled by any other state-of-the-art method in two problems, WAV and VOW (M2GP achieves this only in WAV). We recall that these are precisely the two problems where the number of dimensions keeps growing during the entire evolution. We wonder if, given more generations, M3GP could distance itself even more from the other methods on these two problems. Regarding the other methods, MLP is ranked first in four problems, being the solo winner in one of them (M-L), while RS is ranked first in only two problems.

Table 4. Comparison between M3GP and state-of-the-art methods.

	HRT	IM-3	WAV	SEG	IM-10	YST	VOW	M-L
Training fitness								
RF	**98.4**	**100**	**99.5**	**99.9**	**99.8**	**98.3**	99.9	99.2
RS	88.9	97.1	92.0	98.4	96.3	71.1	97.8	92.3
MLP	**98.4**	98.7	98.5	97.6	91.0	64.6	91.9	91.3
M2GP	89.4	98.2	87.4	96.8	91.4	62.6	95.9	**100**
M3GP	94.7	99.6	90.7	98.1	93.0	68.5	**100**	**100**
Test fitness								
RF	**80.2**	**94.8**	81.5	**97.3**	**96.9**	**57.5**	89.4	71.8
RS	**81.5**	92.8	82.2	96.0	93.9	**56.6**	82.8	65.7
MLP	**80.2**	**95.9**	83.3	96.3	90.2	**58.0**	82.5	**75.9**
M2GP	**80.2**	93.8	**84.9**	95.6	90.2	53.8	85.9	63.0
M3GP	**79.0**	**95.4**	**84.3**	95.6	91.0	**56.2**	**93.8**	57.1

Besides the remarkable fact that M3GP achieves the same quality of results as the popular and successful RF in terms of test fitness, it is also worth remarking that the models provided by M3GP are potentially much easier to interpret than the ones provided by RF, or by any of the other two state-of-the-art methods.

7 Conclusions and Future Work

This paper addresses the problem of multiclass classification with GP, an area where previous approaches tended to yield poor performance. In particular, this paper presents M3GP, an extension of the recently proposed M2GP algorithm, a classifier that evolves transformations of the form $k(\mathbf{x}) : \mathbb{R}^p \rightarrow \mathbb{R}^d$, and applies a minimum Mahalanobis distance classifier. M3GP allows the search to consider a single dimension ($d = 1$) on which to transform the data at the beginning of the search, and progressively builds more dimensions guided by classifier performance.

The results are very encouraging. M3GP can deal with difficult benchmark and real-world problems and achieve state-of-the-art performance, comparing favorably with such methods as Random Forests, Random Subspaces and Multilayer Perceptron. Moreover, it is clear that M3GP adjusts its search based on the characteristics of each problem, automatically determining the best number of new feature dimensions to build in order to maximize accuracy.

Future work must consider a couple of limitations of the approach. First, M3GP needs to be fitted with some procedure to limit/prevent overfitting when accuracy on the training cases is easy to optimize (such as in the M-L problem). Another important aspect is to encourage the evolution of simple and small solutions, with the inclusion of bloat control or more efficient simplification strategies. Nonetheless, for now it is clear that M3GP is a general purpose and simple algorithm that is well worth pursuing and improving for use in challenging classification tasks.

Acknowledgments. This work was partially supported by FCT funds (Portugal) under contract UID/Multi/04046/2013 and projects PTDC/EEI-CTP/2975/2012 (MaSSGP), PTDC/DTP-FTO/1747/2012 (InteleGen) and EXPL/EMS-SIS/1954/2013 (CancerSys). Funding was also provided by CONACYT (Mexico) Basic Science Research Project No. 178323, DGEST (Mexico) Research Projects No. 5149.13-P and 5414.11-P, and FP7-Marie Curie-IRSES 2013 project ACoBSEC. Finally, the first author is supported by scholarship No. 372126 from CONACYT.

References

1. 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. J. Multiple-Valued Log. Soft Comput. **17**(2–3), 255–287 (2011)
2. 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. Accessed 26 January 2015

3. Espejo, P.G., Ventura, S., Herrera, F.: A survey on the application of genetic programming to classification. Trans. Sys. Man Cyber Part C **40**(2), 121–144 (2010)
4. Haynes, T.: ollective adaptation: the exchange of coding segments. Evol. Comput. **6**(4), 311–338 (1998). http://dx.doi.org/10.1162/evco.1998.6.4.311
5. Ingalalli, V., Silva, S., Castelli, M., Vanneschi, L.: A multi-dimensional genetic programming approach for multi-class classification problems. In: Nicolau, M., et al. (eds.) 17th European Conference on Genetic Programming. LNCS, vol. 8599, pp. 48–60. Springer, Granada (2014)
6. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection, vol. 1. MIT press, Cambridge (1992)
7. Koza, J.R.: Human-competitive results produced by genetic programming. Genet. Program. Evol. Mach. **11**(3–4), 251–284 (2010)
8. Lin, J.Y., Ke, H.R., Chien, B.C., Yang, W.P.: Designing a classifier by a layered multi-population genetic programming approach. Pattern Recogn. **40**(8), 2211–2225 (2007)
9. Luke, S., Panait, L.: Lexicographic parsimony pressure. In: Proceedings of GECCO-2002, pp. 829–836. Morgan Kaufmann Publishers (2002)
10. Poli, R., Langdon, W.B., Mcphee, N.F.: A field guide to genetic programming. Lulu.com (Mar 2008)
11. U.S. Geological Survey (USGS): Earth resources observation systems (EROS) data center (EDC) (2015). http://glovis.usgs.gov/. Accessed 26 January 2015
12. Zhang, Y., Rockett, P.I.: A generic multi-dimensional feature extraction method using multiobjective genetic programming. Evol. Comput. **17**(1), 89–115 (2009)

Author Queries

Chapter 7

Query Refs.	**Details Required**	**Author's response**
AQ1	Please check and confirm if the authors and their respective affiliations have been correctly identified. Amend if necessary.	

MARKED PROOF

Please correct and return this set

Please use the proof correction marks shown below for all alterations and corrections. If you wish to return your proof by fax you should ensure that all amendments are written clearly in dark ink and are made well within the page margins.

Instruction to printer	*Textual mark*	*Marginal mark*
Leave unchanged	• • • under matter to remain	ⓥ
Insert in text the matter indicated in the margin	⋏	New matter followed by ⋏ or ⋏ⓧ2
Delete	/ through single character, rule or underline or ⊢—⊣ through all characters to be deleted	ꝺ or ꝺⓧ2
Substitute character or substitute part of one or more word(s)	/ through letter or ⊢—⊣ through characters	new character / or new characters /
Change to italics	— under matter to be changed	⎵
Change to capitals	≡ under matter to be changed	≡
Change to small capitals	═ under matter to be changed	═
Change to bold type	∼ under matter to be changed	∼
Change to bold italic	≂ under matter to be changed	≂
Change to lower case	Encircle matter to be changed	≢
Change italic to upright type	(As above)	⊹
Change bold to non-bold type	(As above)	∼∤∼
Insert 'superior' character	/ through character or ⋏ where required	Y or ⅄ under character e.g. $\overset{2}{Y}$ or $\overset{2}{⅄}$
Insert 'inferior' character	(As above)	⋏ over character e.g. $\underset{2}{⋏}$
Insert full stop	(As above)	⊙
Insert comma	(As above)	,
Insert single quotation marks	(As above)	$\overset{‘}{Y}$ or $\overset{‘}{⅄}$ and/or $\overset{’}{Y}$ or $\overset{’}{⅄}$
Insert double quotation marks	(As above)	$\overset{“}{Y}$ or $\overset{“}{⅄}$ and/or $\overset{”}{Y}$ or $\overset{”}{⅄}$
Insert hyphen	(As above)	⊢-⊣
Start new paragraph	_⌐	_⌐
No new paragraph	⤾	⤾
Transpose	⊔⊓	⊔⊓
Close up	linking ◠◡ characters	◠◡
Insert or substitute space between characters or words	/ through character or ⋏ where required	Y
Reduce space between characters or words	| between characters or words affected	⊤