Genetic programming representations for multi-dimensional feature learning in biomedical classification

William La Cava[1], Sara Silva[2,3], Leonardo Vanneschi[4], Lee Spector[5], and Jason Moore[1]

[1] Institute for Biomedical Informatics, University of Pennsylvania, Philadelphia PA, USA
lacava@upenn.edu
[2] BioISI - Biosystems & Integrative Sciences Institute, Departamento de Informática, Faculdade de Ciências, Universidade de Lisboa, 1749-016 Lisboa, Portugal
[3] CISUC, Department of Informatics Engineering, University of Coimbra, Portugal
[4] NOVA IMS, Universidade Nova de Lisboa, 1070-312 Lisbon, Portugal
[5] School of Cognitive Science, Hampshire College, Amherst MA, USA

Abstract. We present a new classification method that uses genetic programming (GP) to evolve feature transformations for a deterministic, distanced-based classifier. This method, called M4GP, differs from common approaches to classifier representation in GP in that it does not enforce arbitrary decision boundaries and it allows individuals to produce multiple outputs via a stack-based GP system. In comparison to typical methods of classification, M4GP can be advantageous in its ability to produce readable models. We conduct a comprehensive study of M4GP, first in comparison to other GP classifiers, and then in comparison to six common machine learning classifiers. We conduct full hyper-parameter optimization for all of the methods on a suite of 16 biomedical data sets, ranging in size and difficulty. The results indicate that M4GP outperforms other GP methods for classification. M4GP performs competitively with other machine learning methods in terms of the accuracy of the produced models for most problems. M4GP also exhibits the ability to detect epistatic interactions better than the other methods.

Keywords: genetic programming, feature learning, classification

1 Introduction

Classification models are a fundamental pursuit in the biomedical field due to their widespread utility in applications such as medical diagnosis [39,23,29] and identification of genetic causes of disease [25,26]. In classification with numeric attributes, we wish to find a mapping $\hat{y}(\mathbf{x}) : \mathbb{R}^p \rightarrow \mathcal{C}$ that associates the vector of attributes $\mathbf{x} \in \mathbb{R}^p$ with class labels from the set $\mathcal{C} = \{c_1 \ \dots \ c_k\}$ using n paired examples from the training set $\mathcal{T} = \{(\mathbf{x}_i, y_i), i = 1 \ \dots \ n\}$. Central to this goal is the identification of important transformations of the original data that improve classification accuracy. Machine learning (ML) systems that conduct classification have become standardized over the last 20 years [2], and open-source packages are available for performing classification (e.g., [8,30]) according to well-known approaches such as logistic regression (LR), nearest centroid (NC), support vector classification (SVC), Bayesian Networks (e.g. naïve Bayes (NB)), k-nearest neighbors (KNN), and ensemble methods such as random forests (RF), among others. Yet three major challenges to multiclass classification persist. The first two challenges are i) the selection of and ii) transformation of features into new features (feature synthesis), derived from the original attributes, to be used for model construction. Feature selection is important for reducing large-dimension data sets and for measurement selection in some domains. Typically it is left to a pre-processing step to reduce the number of attributes to a manageable size [7]; in other words, feature selection is not an intrinsic property of most ML approaches. Regarding the second challenge, many ML methods employ projection of the original features into a new feature space, for example via kernel functions [28]. However the choice of kernel function is typically not automated, but picked by trial and error or cross-validation. The opaque nature of kernel transformations highlights a third challenge of classification: the interpretability of the resultant models. Interpretation is especially relevant in the sciences and for applications like human genomics that rely on classification as a means of inferring relationships from observations. To this end, methods with intelligible representations like decision trees use greedy simplification procedures, while acknowledging that finding a minimal decision tree is an NP-hard problem [33].

Genetic programming (GP) [14] has been proposed for classification to remedy the three challenges above [13,5]. GP is a stochastic optimization method that implicitly conducts feature selection by pressuring the model $\hat{y}(\mathbf{x})$ to use a subset of $\mathbf{x}$ most relevant to the problem solution. In addition, GP makes minimal *a priori* assumptions about the structure of the attribute space [17], admits a number of representations [21], and can be made to optimize the structure of the model such that it remains intelligible. Although it has been applied successfully to a number of binary classification problems [39], until recently [11,27] it has not been competitive with standard multiclass classification techniques. The exceptions are the recently developed methods M2GP [11] and M3GP [27] that use GP to select and synthesize features and then perform classification in the new feature space using a Mahalanobis distance-based discriminant function. In this paper, we improve upon these methods by two innovations: i) the use of a novel program representation that simplifies the construction of multidimensional representations, and ii) the incorporation of an advanced parent selection technique that leads to more accurate classifiers. The performance of this method, appropriately named M4GP[6], is compared to other GP representations, including M2GP, M3GP, using a set of eight benchmark classification problems. Then, M4GP is benchmarked against LR, NC, SVC, NB, KNN and RF on a 16 biomedical classification problems. The experiments include a hyper-

[6] Source code available from `http://github.com/lacava/ellyn`

parameter optimization step for each learner, such that the comparisons consider a tuned version of each method.

2 M4GP

Recall the labeled training set $\mathcal{T} = \{(\mathbf{x}_i, y_i), i = 1 \ \dots \ n\}$, consisting of n samples of attributes $\mathbf{x}_i \in \mathbb{R}^p$ associated with the corresponding class label y_i from the set $\mathcal{C} = \{c_1 \ \dots \ c_k\}$. The $n \times p$ matrix of attribute samples $\mathbf{X}$ can be partitioned according to its labels into k subsets $\{\mathbf{X}_1 \ \dots \ \mathbf{X}_k\}$, such that $\mathbf{X}_j$ is the subset of $\mathbf{X}$ tagged with class label c_j. One way to classify a new sample $\mathbf{x}' \in \mathbb{R}^p$ is by finding its nearest centroid [36], i.e. to measure the distance of $\mathbf{x}'$ to each subset $\{\mathbf{X}_1 \ \dots \ \mathbf{X}_k\}$, and then assign the class label corresponding to the minimum distance [12], i.e.

$$\hat{y}(\mathbf{x}') = c_j, \ \text{ if } \ j = \arg\min_{\ell} D(\mathbf{x}', \mathbf{X}_\ell) \ \ , \ \ell = 1, \dots, k \tag{1}$$

One such measure is the Mahalanobis distance, D_M,

$$D_M(\mathbf{x}', \mathbf{X}_j) = \sqrt{(\mathbf{x}' - \mu_j)\, \mathbf{\Sigma}_j^{-1} \, (\mathbf{x}' - \mu_j)^T} \tag{2}$$

where $\mu_j \in \mathbb{R}^p$ is the centroid of $\mathbf{X}_j$ and $\mathbf{\Sigma}_j \in \mathbb{R}^{p \times p}$ is its covariance matrix, rendering D_M the equivalent Euclidean distance of $\mathbf{x}'$ from $\mathbf{X}_j$, scaled by the eigenvalues (variances) and rotated by the eigenvectors of $\mathbf{\Sigma}_j$, to account for the correlation between columns of $\mathbf{X}_j$.

This approach to classification makes some assumptions about the structure of the data. First, each $\mathbf{X}_j$ must be sufficiently grouped such that samples always fall closest to their true distribution, which cannot be said of most difficult classification problems. Second, it assumes that Eq. (2) can be calculated from the original data. One can imagine that as the dimensionality of $\mathbf{X}$ increases, the calculation of D_M becomes prohibitively expensive.

In order to relax these assumptions, we wish to find a set of transformations $\Phi(\mathbf{x}) : \mathbb{R}^p \to \mathbb{R}^d$ that projects $\mathbf{x}$ into a d-dimensional space in which the samples are more easily classified according to their distribution distances. In this new space, the Mahalanobis distance takes the form $D_M(\Phi(\mathbf{x}), \Phi(\mathbf{X}_j))$, with centroid $\mu_{\Phi_j} \in \mathbb{R}^d$ and covariance matrix $\mathbf{\Sigma}_{\Phi_j} \in \mathbb{R}^{d \times d}$.

The goal of the GP system will be to find or approximate the optimal synthesized features $\Phi^* = [\phi_1 \ \dots \ \phi_d]$ that maximize the number of correctly classified training samples, as:

$$\Phi^*(\mathbf{x}) = \arg\max_{\Phi \in \mathbb{S}} f\,(\Phi, \mathcal{T}) \tag{3}$$

$$f(\Phi, \mathcal{T}) = \frac{1}{n} \sum_{i=1}^{n} \delta\,(\hat{y}(\Phi(\mathbf{x}_i)), \ y_i) \tag{4}$$

where $\mathbb{S}$ is the space of possible transformations Φ, f is the classification accuracy (used here as the GP fitness function), and $\delta = 1$ if $\hat{y}(\Phi(\mathbf{x}_i)) = y_i$, and 0 otherwise. A well-formed $\Phi(\mathbf{x})$ allows the classifier the flexibility to incorporate (linear and/or nonlinear) transformations of the original attributes in order to improve distinctions between classes compared to using the original attribute set. By using GP to estimate the features $\Phi(\mathbf{x})$, the subset of $\mathbf{x}$ used in $\Phi(\mathbf{x})$ as well as the dimensionality of Φ, $|\Phi| = d$, are optimized. Therefore, feature selection in GP can produce $d << p$ for high dimensional data sets, making Eq. (2) tractable, and also admits higher-dimensional representations ($d > p$) in cases for which $\mathbf{x}$ is not easily mapped to y.

2.1 Genetic Programming

GP solves problems by constructing and updating a population of programs composed of building blocks that represent solution components. In this case, each program consists of a set of equations that compose the synthesized features $\Phi(\mathbf{x})$ used to estimate $\hat{y}$. For example, an individual program $\mathbf{i}$ might encode the features

$$\mathbf{i} \rightarrow \Phi(\mathbf{x}) = [x_1,\ x_2,\ x_1^2,\ x_2^2,\ x_1 x_2] \tag{5}$$

where $\phi_1 = x_1$, $\phi_2 = x_2$, $\phi_5 = x_1 x_2$, and so on. In this case, $|\Phi| = 5$, and $\mathbf{i}$ corresponds to a polynomial expansion of two attributes.

Traditionally in GP, a program is represented by a single syntax tree evaluated based on the output generated at the root node [14]. For example, ϕ_5 above could be represented by a tree $(*\ x_1\ x_2)$, where '$*$' is the root node and x_1 and x_2 are its leaves. However, a single output cannot represent a multi-dimensional transformation. To address this, in M2GP and M3GP, program trees were modified with special nodes in order to allow for multiple outputs at the root [11,27]. This introduced unnecessary complexity to the representation. A contribution of this work is the introduction of a stack-based data flow to simplify the encoding of Φ, presented in the Representation paragraph below.

The GP population is optimized by probabilistically selecting programs based on their performance and stochastically recombining and mutating these programs to produce a new set of programs. In this work, we implement a recent selection mechanism known as lexicase selection [35] and compare its performance to a more traditional selection algorithm (tournament selection). These techniques are described in the following sections.

Representation We implement a stack-based representation [31,15] of the equations in place of the more traditional tree-based GP representations. Programs in this representation are encoded as post-fix notation equations, e.g., $\mathbf{i} = [\ x_1\ x_2\ +\ ] \rightarrow \Phi = [x_1 + x_2]$. This representation is advantageous because it allows multiple outputs to be supported by default without the need for specialized instructions. This support is achieved by evaluating programs via executions on a stack, such that the program in Eq. (5) can be constructed as

$$\mathbf{i} = [\ x_1\ x_2\ x_1\ x_1\ *\ x_2\ x_2\ *\ x_1\ x_2\ *\ ]$$

The execution of program **i** is illustrated in Figure 1. Rather than recursively evaluating the program as a tree starting at its root node, stack based evaluation proceeds left to right, pushing and pulling instructions to and from a single stack. Arguments such as x_1 are pushed to the stack, and operators such as '$*$' pull arguments from the stack and push the result. At the end of a program's execution, the entire stack represents the multi-dimensional transformation.

$$\begin{array}{rccccccccccc} \mathbf{i} = & [\,x_1 & x_2 & x_1 & x_1 & * & x_2 & x_2 & * & x_1 & x_2 & *\,] \\ \text{index:} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \end{array}$$

program execution	stack
1. `push` (x_1):	$[\,x_1\,]$
2. `push` (x_2):	$[\,x_1 \;\; x_2\,]$
3. `push` (x_1):	$[\,x_1 \;\; x_2 \;\; x_1\,]$
4. `push` (x_1):	$[\,x_1 \;\; x_2 \;\; x_1 \;\; x_1\,]$
5. `pull` (x_1), (x_1); `push` $(x_1 \cdot x_1)$	$[\,x_1 \;\; x_2 \;\; x_1x_1\,]$
6. `push` (x_2):	$[\,x_1 \;\; x_2 \;\; x_1^2 \;\; x_2\,]$
7. `push` (x_2):	$[\,x_1 \;\; x_2 \;\; x_1^2 \;\; x_2 \;\; x_2\,]$
8. `pull` (x_2), (x_2); `push` $(x_2 \cdot x_2)$	$[\,x_1 \;\; x_2 \;\; x_1^2 \;\; x_2^2\,]$
9. `push` (x_1):	$[\,x_1 \;\; x_2 \;\; x_1^2 \;\; x_2^2 \;\; x_1\,]$
10. `push` (x_2):	$[\,x_1 \;\; x_2 \;\; x_1^2 \;\; x_2^2 \;\; x_1 \;\; x_2\,]$
11. `pull` (x_1), (x_2); `push` $(x_1 \cdot x_2)$	$[\,x_1 \;\; x_2 \;\; x_1^2 \;\; x_2^2 \;\; x_1x_2\,]$

$$\rightarrow \Phi(\mathbf{x}) = [x_1,\ x_2,\ x_1^2,\ x_2^2,\ x_1x_2]$$

Fig. 1. Example of program representation of a multidimensional transformation. Arguments such as x_1 are pushed to the stack, and operators such as '$*$' pull arguments from the stack and push the result.

2.2 Other GP Classification Methods

In the case of M4GP, the mean and covariance of the stack outputs are used in order to make classifications for each sample according to the Mahalanobis distance (Eq. 2). However, a much simpler approach to classification could be to directly classify samples based on these outputs. This is the case with many GP-based classifiers [5]. To this end, we compare distance-based classification with two simpler alternatives: float stack classification and boolean stack classification, referred to simply as *float* and *bool* hereafter.

In the case of *float*, we take the index of the floating point stack with the highest value to be the class assignment. For example, assume the program from Figure 1 produces the output $[0.15,\ 2.31,\ 42,\ 6.3,\ 0.01]$ for a sample from the data. In this case the GP model would assign the 3rd class label to this sample. Thus the GP system attempts to evolve a set of equations that are maximized for the class label corresponding to their location in the program.

In the case of *bool*, we include a set of boolean operators in the function set for constructing GP programs: {`AND`, `OR`, `NOT`, $<$, $>$, $<=$, $>=$, $==$, `IF-THEN`, `IF-THEN-ELSE` }. Boolean outputs are pushed to their own typed stack. In order to make a classification, the boolean stack is interpreted as a bit string. For example, in the case of 2 classes, the top value of the boolean stack is interpreted as class 1 ([0]) or class 2 ([1]). For binary classification problems, this corresponds to a fairly traditional encoding for classification [32]. In the case of 4 classes, the top two values of the stack are interpreted as class 1 ([0, 0]), class 2 ([0, 1]), class 3 ([1, 0]), or class 4 ([1, 1]).

Initialization, Selection, and Variation Programs are initialized as sets of equations varying both in individual feature size and their dimensionality. Each equation in a program is initialized recursively in an analogous fashion to the grow method (see [32]) but limited by number of nodes rather than depth. Fitness for the programs is defined in Eq. (4) on Page 4.

Two population selection methods are tested: tournament selection [6] and lexicase selection [35,9]. The first, tournament selection, is a standard GP method in which individuals (in this case, two) in the current population are randomly selected (with replacement) and the one with best fitness is chosen as a parent for the next generation. Lexicase selection is described in more detail below.

Lexicase selection Lexicase selection is a parent selection technique that pressures individuals in the population to perform well on unique combinations of training cases, i.e. samples. Each parent selection event follows this procedure:

1. The entire population is added to the selection pool.
2. The training cases are uniformly shuffled.
3. Individuals in the pool that do not have *exactly* the best fitness on the first case among the pool are removed.
4. If more than one individual remains in the pool, the first case is removed and step 2 is repeated with the next case. If only one individual remains, it is the chosen parent. If no more fitness cases are left, a parent is chosen randomly from the remaining individuals.

As can be surmised, the lexicase selection is simple to implement. It is helpful to think of the training cases as filters, and to consider each parent selection event as a randomized path through these filters. The parents returned by lexicase selection are Pareto-optimal with respect to the training cases, since they must be elite on at least one case to be selected. In turn, the selective strength of a training case is directly proportional to its difficulty because it culls the individuals from the pool that do not solve it. Therefore selective pressure shifts to cases that are not widely solved. This interaction between individuals in the pool and the training cases results in selective pressure to perform well on unique combinations of test cases. As a result, lexicase selection leads to increased population diversity observed during evolutionary runs [9,16].

3 Related Work

Whereas GP has been proposed for evolving classification functions $\hat{y}(\mathbf{x})$ directly [13,5,21], M2GP proposed GP as a wrapper that evolved $\Phi(\mathbf{x})$ for a clustering method, and demonstrated in particular that Mahalanobis distance outperformed Euclidean distance in this framework [11]. M3GP extended M2GP to allow programs to change dimensionality during the run via specialized search operators that increased or decreased the dimensionality of a tree by modifying its root node [27]. M4GP removes the need for explicit root nodes by using a stack-based data flow that also preserves multi-dimensionality and allows dimensionality to change flexibly. An ensemble version of M3GP named eM3GP produced similar classification accuracies to M3GP with smaller, more legible resultant programs [34]. Together, these methods highlight the unique challenge of feature selection and its merger into learning systems [19].

A few recently developed ML methods have leveraged GP's feature-based abilities as a wrapper for regression [22,10,1]. M4GP and its ancestors differ from these regression-based approaches in that the classification does not require classes to be assigned via an arbitrarily designated range of real-valued outputs, but instead utilizes a distance metric to infer the boundaries of the transformed feature space. M4GP also incorporates a novel GP representation and advanced selection methods to improve its performance.

GP has also been proposed to fill various roles in tailored learning systems for image classification. It has been used, for example, as a way to learn image embeddings for an ensemble method [20], as an interactive learning tool for remote sensing [4], and as a binary classifier in a pulmonary nodule detection system [3]. Liu et al. [20] noted the potential for GP to perform dimensionality reduction efficiently in large-scale settings, as we noted earlier. M4GP differs from these approaches in two ways: first, it focuses on the capacity for low- and high-dimensionality feature extraction to flexibly suit the needs of the problem, and second, it applies to general multiclass classification problems.

4 Experimental Analysis

The experimental section consists of two parts. First, we compared M4GP to other GP methods, including *bool* and *float* methods described above, M2GP, and M3GP. Second, we compared M4GP to off-the-shelf methods on a set of biomedical data sets using a full hyper-parameter optimization strategy.

The settings for the first set of experiments are shown in Table 1. The settings for the methods match those used in the M2GP and M3GP papers, with the exception of program size limits (specified in numbers of elements rather than tree depth) and initial dimensionality range. The same set of problems from the original papers are used for these experiments to facilitate the comparison. Six of the eight problems used for comparing the GP methods are from the UCI data repository [18] and are summarized in Table 1. Two others, mcd3 and mcd10, are satellite data sets from [38]. Two versions of M4GP are tested: M4GP with lexicase selection (M4GP-lx), and tournament selection (M4GP-tn). Each method is run for 30 trials, and for each trial the data is randomly partitioned into 70% training and 30% testing.

The second set of experiments are designed to compare M4GP to six common classification methods available in Scikit Learn [30]: NB, LR, KNN, SVC, RF and NC. NC uses the classification strategy of Eqn. 1, and therefore provides a comparison for M4GP against using the raw feature representation with the same discriminant function. For each method, hyper-parameter optimization is conducted on the training set using 5-fold cross-validation. In order to enforce some balance in the hyper-parameter optimization step, each method is restricted to 50 parameter combinations. We report and compare classification accuracy on the test set. The experimental design is summarized in Table 2.

For this comparison, 16 biomedical data sets are used, varying from 2 to 4 classes, 88 to 3772 samples, and 7 to 1000 features. 10 of the 16 problems are open-source, real-world data sets available from the OpenML repository [40]. They consist of different biomedical tasks such as medical diagnosis, post-operative decision making, and identification of exon boundaries in DNA. Six synthetic problems generated using GAMETES [37], are included. These problems embed 2- and 3-way epistatic interactions (i.e. non-additive interaction among genes) within noisy data sets with 20 or 1000 attributes. The goal of this problem is to test the ability of ML algorithms to identify these types of interactions common in genome-wide association studies [24]. Two of the GAMETES problems also test heterogeneity by embedding two separate, semi-overlapping epistatic interactions into the data.

Table 1. Experimental setup for the comparison of GP methods.

Setting	Value
Population size	500
Max Generations	100
Crossover / Mutation	50/50%
Ephemeral random constants [14] range	[0,1]
Program size limits by # nodes	[3, 100]
Initial dimensionality range (d)	[1,33]
Termination criterion	generations or perfect training accuracy
Trials	30
Train/test split	70/30

Data Set	heart	mcd3	mcd10	movl	seg	vowel	wav	yeast
Classes	2	3	10	15	7	11	3	10
Attributes	13	6	6	90	19	13	40	8
Samples	270	322	6798	360	2310	990	5000	7797

5 Results

The results of the comparison of GP methods is first discussed in section 5.1, followed by the comparison of several ML methods on the biomedical data sets in section 5.2.

Table 2. Experimental setup for the biomedical problems. The hyper-parameters that were searched are shown on the right. Below the biomedical data sets are listed. GMT stands for GAMETES data sets, which are named according to number of epistatic loci (w), number of attributes (a), noise fraction (n), and heterogeneity fraction (h).

Method	hyper-parameters
M4GP	Population size (100, 250, 500); generations (10,50,100); selection method (tournament, lexicase); max length (10, 50, 100)
Gaussian Naïve Bayes	none
Logistic Regression	Regularization coefficient (0.1,...,20); penalty (ℓ_1,ℓ_2); fit intercept (True, False); dual formulation (True, False)
Support Vector Classifier	Regularization coefficient (0.01,1,100,'auto'); γ (0.01, 10, 1000, 'auto'); kernel (linear, RBF); decision function shape ('ovo','ovr')
Random Forest Classifier	No. estimators (10, 100, 1000); minimum weight fraction for leaf (0.0, 0.25, 0.5); max features ($sqrt$, log_2, None); splitting criterion (entropy, gini)
K-Nearest Neighbor Classifier	K (1,2,...,25); weights (uniform, distance)
Nearest Centroid Classifier	distance metric (Euclidean, Mahalanobis)

Training and Test Methodology	
Hyper-parameter optimization	5-fold cross-validation
Train/Test split	50/50
Trials	30
Score	Accuracy

Data Set	Classes	Samples	Dimensions
allbp	3	3772	29
allhyper	4	3771	29
allhypo	3	3770	29
biomed	2	209	8
breast-cancer-wisconsin	2	569	30
breast-cancer	2	286	9
diabetes	2	768	8
dna	3	3186	180
GMT 2w-20a-0.1n	2	1600	20
GMT 2w-20a-0.4n	2	1600	20
GMT 3w-20a-0.2n	2	1600	20
GMT 2w-1000a-0.4n	2	1600	1000
GMT 2w-20a-0.4n-0.5h	2	1600	20
GMT 2w-20a-0.4n-0.75h	2	1600	20
liver-disorder	2	345	6
postoperative-patient-data	2	88	8

5.1 Comparison to other GP methods

As a first point of comparison, we analyze the choice of GP classification method (*bool*, *float* or distance) based on the test accuracy for the first set of problems in Figure 2. The distance-based classification method, i.e. M4GP, outperforms the other methods on all problems, 7 out of 8 by a large margin. The distance-based method also has the advantage of less variability in its performance compared to the other methods. From these results it is clear that the distance-based classifier has a distinct advantage over *bool* and *float* methods on these problems. This validates our choice to use M4GP in comparison to other ML methods on the set of biomedical problems.

As a second point of comparison, we benchmark the results of M4GP to M2GP and M3GP in Figure 3. The results of M4GP using lexicase selection and tournament selection are both displayed. In order to test statistical significance, pairwise Wilcoxon rank-sum tests are performed with Holm correction for multiple comparisons. We use a significance level of $p < 0.01$ in the following reporting. The results indicate that M4GP-tn significantly outperforms M2GP on 4 out of 8 problems (heart, mcd3, vowel, and wav), and significantly outperforms than M3GP on 5 out of 8 problems (heart, mcd3, vowel, wav and movl). M4GP-lx significantly outperforms M2GP on 5/8 problems and outperforms M3GP on 6/8 problems. Conversely, M3GP does not significantly outperform M4GP-lx on any problem, and only outperforms M4GP-tn on one problem (mcd10). In addition, on one problem (movl), M2GP outperforms M3GP, M4GP-tn and M4GP-lx. A closer look at these runs indicates that most finish within the first few generations due to perfect training scores, which indicates this problem is likely to be solved easily by random search. Given that M2GP begins with smaller programs, it is more likely than M4GP to not over-fit in the first few generations. In summary, M4GP-lx is able to produce significantly better results for most problems, and in other cases produce results on par with the previous methods.

The choice of selection method results in mixed performance for M4GP. M4GP-lx significantly outperforms M4GP-tn on two problems, whereas M4GP-tn outperforms M4GP-lx on one problem, meaning the selection mechanism in M4GP is problem dependent. Because the selection method is kept as a hyperparameter for the biomedical data sets, the optimization procedure is able to test both selection methods in the subsequent set of experiments.

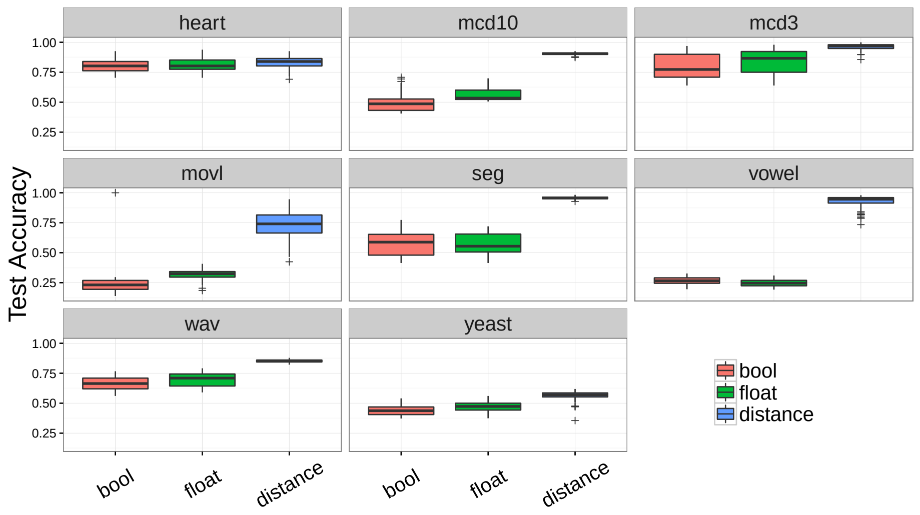

Fig. 2. Test accuracies for each GP classification method on the set of UCI and satellite problems. The distance method corresponds to the M4GP algorithm. The subplot title indicates the dataset being shown.

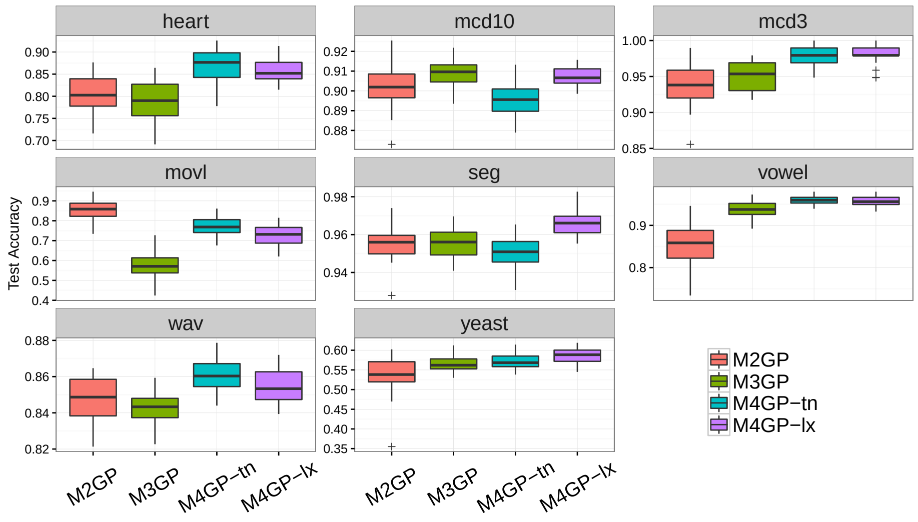

Fig. 3. Test accuracies for each Mahalanobis distance-based GP method on the set of UCI and satellite problems. The subplot title indicates the dataset being shown.

5.2 Comparison to other ML methods

The results of the ML comparisons on the biomedical problems are compared in Figure 4. Classification accuracy on the test set for 30 trials is plotted. The statistical significance of the results is analyzed in Table 3 according to Wilcoxon rank-sum tests of M4GP versus each method with correction for multiple comparisons. The results reveal that different algorithms excel with different problems. In total, M4GP outperforms other methods in 39 pair-wise comparisons, and is outperformed in 26. M4GP most often outperforms NC (significantly on 9/16 problems), which is an intuitive result considering the feature transformations generated by M4GP are evolved to perform well with a nearest centroid classification strategy. M4GP also outperforms NB in most cases (9/16 problems), although NB surpasses it in 2 cases. Conversely, RF and SVC both outperform M4GP on 7/16 of the problems, and are outperformed by M4GP on 5.

The GAMETES problems produce an interesting set of results and perhaps the most variability in test accuracy among the methods. These problems are the only ones for which M4GP outperforms all other methods with strong significance. Given that the GAMETES data sets are designed specifically to test epistasis, this result suggests M4GP may be able to identify this phenomena with more certainty than the other methods tested here. The relatively poor performance of most of the other tested methods is explained by their reliance on the identification of univariate correlations of the raw features with the class labels. The GAMETES data sets we tested are void of these so-called main effects, instead requiring the method to detect epistasis in the data set to produce accurate classifiers. M4GP is naturally suited for this task due to its capacity for nonlinear, multi-variate feature transformations.

Table 3. Significance tests ($p < 0.01$) for the biomedical problems in comparison to M4GP. Wilcoxon rank-sum tests with Holm correction are conducted on the test accuracy results. Highlighted results are problems for which M4GP significantly outperformed the other method. Underlined results are those for which M4GP was significantly outperformed by that method.

	NB	LR	KNN	SVC	RF	NC
allbp	6.06e-10	0.000239	0.122	0.000659	1.19e-09	1.84e-09
allhyper	6.1e-10	0.127	0.0217	0.127	2.45e-09	5.14e-09
allhypo	6.05e-10	6.05e-10	0.000664	6.05e-10	6.05e-10	6.05e-10
biomed	0.0355	0.348	1	0.1	0.00475	1
breast-cancer	0.0123	0.0587	0.00214	0.0103	1	1
breast-cancer-wisconsin	0.0656	7.04e-07	0.0656	1.95e-05	0.231	0.012
diabetes	0.463	2.23e-06	1	3.39e-06	0.0192	1
dna	1.23e-05	6.13e-10	6.13e-10	6.13e-10	6.13e-10	1.13e-07
GMT 2w-1000a-0.4n	1	1	1	1	1	0.236
GMT 2w-20a-0.1n	2.2e-08	1.93e-08	7.77e-07	7.43e-07	9.06e-07	1.46e-08
GMT 2w-20a-0.4n	6.2e-10	6.2e-10	6.2e-10	6.2e-10	6.2e-10	6.2e-10
GMT 3w-20a-0.2n	5.31e-05	2.49e-05	0.000512	7.21e-05	0.00439	6.47e-05
GMT 2w-20a-0.4n-0.5h	4.3e-09	4.3e-09	4.3e-09	7.76e-08	4.04e-08	3.62e-09
GMT 2w-20a-0.4n-0.75h	1.76e-09	1.76e-09	4.74e-08	5.06e-08	6.51e-08	1.76e-09
liver-disorder	2.59e-06	0.000428	1	3.77e-06	0.000122	1
postoperative-patient-data	0.0016	6.65e-10	2.34e-07	6.65e-10	7.24e-09	1

6 Discussion and Conclusion

The results suggest that GP methods paired with distance-based classification can be more effective on many classification problems than typical GP classification strategies. Across a set of real-world problems, the proposed M4GP algorithm is able to outperform other GP methods, including previously developed distance-based classification strategies. Key to this improvement with respect to M2GP and M3GP is the use of a stack-based representation that facilitates multidimensional feature transformations, as well as the use of lexicase selection as a parent selection strategy.

M4GP is demonstrated in a robust comparison to other ML methods by conducting a full hyper-parameter optimization routine across 16 biomedical data sets. The results suggest that M4GP is competitive with other ML tools, especially in detecting epistasis. The GAMETES results motivate further interest into the application of M4GP to problems for which epistasis might be present.

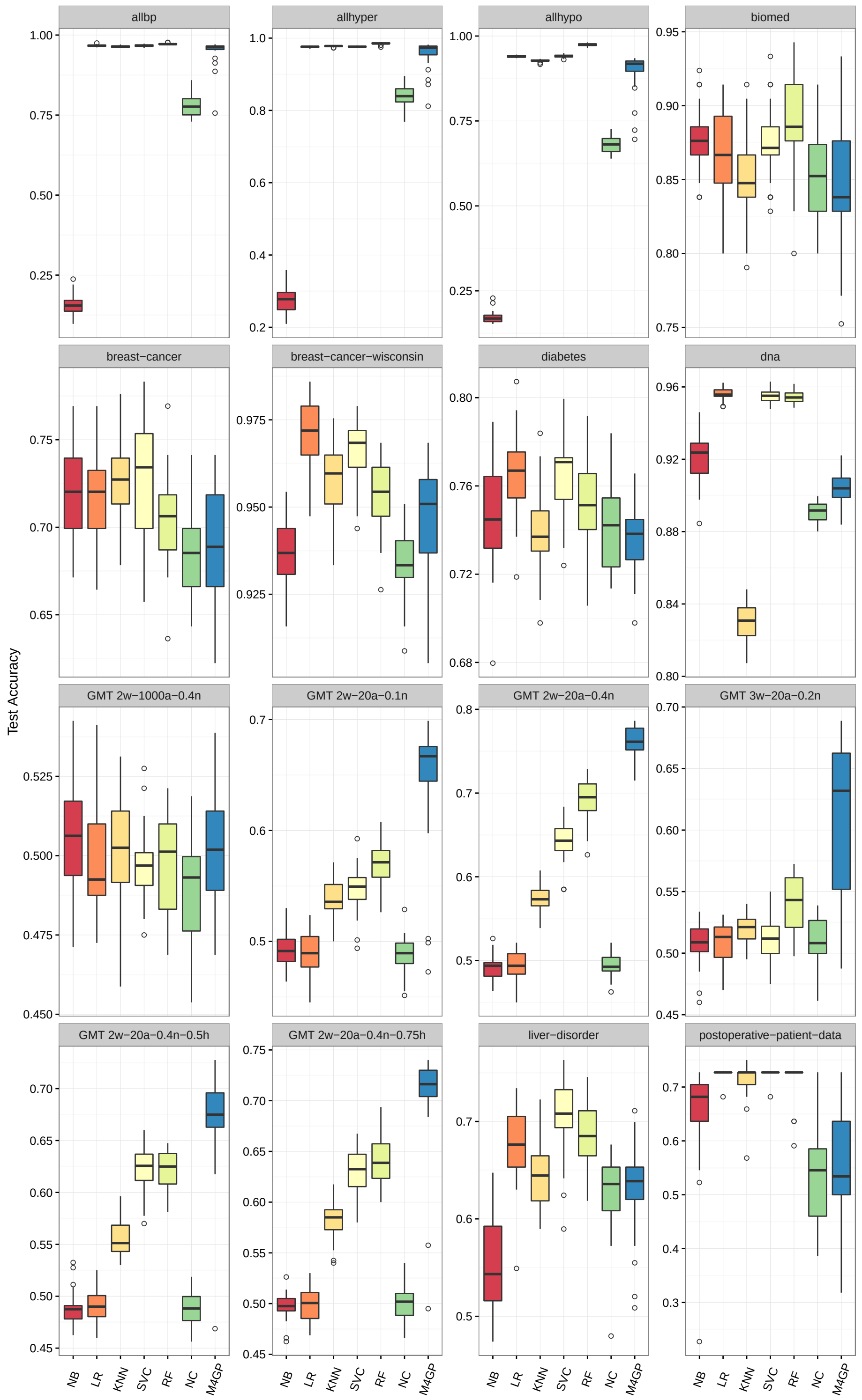

Fig. 4. Test accuracies for each method on the set of biomedical problems. The subplot title indicates the dataset being shown.

7 Acknowledgments

This work was supported by the Warren Center for Network and Data Science, as well as NIH grants P30-ES013508, AI116794 and LM009012. S. Silva acknowledges project PERSEIDS (PTDC/EMS-SIS/0642/2014) and BioISI RD unit, UID/MULTI/04046/2013, funded by FCT/MCTES/PIDDAC, Portugal. This material is based upon work supported by the National Science Foundation under Grants No. 1617087, 1129139 and 1331283. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

1. Ignacio Arnaldo, Una-May O'Reilly, and Kalyan Veeramachaneni. Building Predictive Models via Feature Synthesis. pages 983–990. ACM Press, 2015.
2. Rich Caruana and Alexandru Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on Machine learning*, pages 161–168. ACM, 2006.
3. Wook-Jin Choi and Tae-Sun Choi. Genetic programming-based feature transform and classification for the automatic detection of pulmonary nodules on computed tomography images. *Information Sciences*, 212:57–78, December 2012.
4. J. A. dos Santos, C. D. Ferreira, R. da S. Torres, M. A. Gonalves, and R. A. C. Lamparelli. A relevance feedback method based on genetic programming for classification of remote sensing images. *Information Sciences*, 181(13):2671–2684, July 2011.
5. Pedro G. Espejo, Sebastin Ventura, and Francisco Herrera. A survey on the application of genetic programming to classification. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 40(2):121–144, 2010.
6. Yongsheng Fang and Jun Li. A review of tournament selection in genetic programming. In *Advances in Computation and Intelligence*, pages 181–192. Springer, 2010.
7. Isabelle Guyon and Andr Elisseeff. An Introduction to Variable and Feature Selection. *J. Mach. Learn. Res.*, 3:1157–1182, March 2003.
8. Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The WEKA data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18, 2009.
9. T. Helmuth, L. Spector, and J. Matheson. Solving Uncompromising Problems with Lexicase Selection. *IEEE Transactions on Evolutionary Computation*, PP(99):1–1, 2014.
10. Ilknur Icke and Josh C. Bongard. Improving genetic programming based symbolic regression using deterministic machine learning. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pages 1763–1770. IEEE, 2013.
11. Vijay Ingalalli, Sara Silva, Mauro Castelli, and Leonardo Vanneschi. A Multidimensional Genetic Programming Approach for Multi-class Classification Problems. In *Genetic Programming*, pages 48–60. Springer, 2014.
12. A. K. Jain, M. N. Murty, and P. J. Flynn. Data Clustering: A Review. *ACM Comput. Surv.*, 31(3):264–323, September 1999.

13. J. K. Kishore, Lalit M. Patnaik, V. Mani, and V. K. Agrawal. Application of genetic programming for multicategory pattern classification. *Evolutionary Computation, IEEE Transactions on*, 4(3):242–258, 2000.
14. John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIT Press, Cambridge, MA, USA, 1992.
15. William La Cava, Kourosh Danai, and Lee Spector. Inference of compact nonlinear dynamic models by epigenetic local search. *Engineering Applications of Artificial Intelligence*, 55:292–306, October 2016.
16. William La Cava, Lee Spector, and Kourosh Danai. Epsilon-Lexicase Selection for Regression. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, GECCO '16, pages 741–748, New York, NY, USA, 2016. ACM.
17. Tao Li, Chengliang Zhang, and Mitsunori Ogihara. A comparative study of feature selection and multiclass classification methods for tissue classification based on gene expression. *Bioinformatics*, 20(15):2429–2437, 2004.
18. M. Lichman. *UCI Machine Learning Repository.* University of California, Irvine, School of Information and Computer Sciences, 2013.
19. Huan Liu and Lei Yu. Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on Knowledge and Data Engineering*, 17(4):491–502, April 2005.
20. Li Liu, Ling Shao, and Xuelong Li. Evolutionary compact embedding for large-scale image classification. *Information Sciences*, 316:567–581, September 2015.
21. Thomas Loveard and Victor Ciesielski. Representing classification problems in genetic programming. In *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, volume 2, pages 1070–1077. IEEE, 2001.
22. Trent McConaghy. FFX: Fast, scalable, deterministic symbolic regression technology. In *Genetic Programming Theory and Practice IX*, pages 235–260. Springer, 2011.
23. Patricia Melin, Jonathan Amezcua, Fevrier Valdez, and Oscar Castillo. A new neural network model based on the LVQ algorithm for multi-class classification of arrhythmias. *Information Sciences*, 279:483–497, September 2014.
24. Jason H. Moore. The ubiquitous nature of epistasis in determining susceptibility to common human diseases. *Human heredity*, 56(1-3):73–82, 2003.
25. Jason H. Moore, Folkert W. Asselbergs, and Scott M. Williams. Bioinformatics challenges for genome-wide association studies. *Bioinformatics*, 26(4):445–455, February 2010.
26. Jason H Moore, Casey S Greene, and Douglas P Hill. Identification of Novel Genetic Models of Glaucoma Using the EMERGENT Genetic Programming-Based Artificial Intelligence System. In *Genetic Programming Theory and Practice XII*, pages 17–35. Springer, 2015.
27. Luis Muñoz, Sara Silva, and Leonardo Trujillo. M3gpMulticlass Classification with GP. In *Genetic Programming*, pages 78–91. Springer, 2015.
28. Kevin P. Murphy. *Machine learning: a probabilistic perspective.* Adaptive computation and machine learning series. MIT Press, Cambridge, MA, 2012.
29. Thanh Nguyen, Abbas Khosravi, Douglas Creighton, and Saeid Nahavandi. Hidden Markov models for cancer classification using gene expression profiles. *Information Sciences*, 316:293–307, September 2015.
30. Fabian Pedregosa, Gal Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, and others. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12(Oct):2825–2830, 2011.

31. Timothy Perkis. Stack-based genetic programming. In *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*, pages 148–153. IEEE, 1994.
32. Riccardo Poli, Nicholas F McPhee, and John R Koza. *A field guide to genetic programming.* [Lulu Press], lulu.com, [S.I.], 2008.
33. J Ross Quinlan. *C4. 5: programs for machine learning.* Elsevier, 2014.
34. Sara Silva, Luis Muñoz, Leonardo Trujillo, Vijay Ingalalli, Mauro Castelli, and Leonardo Vanneschi. Multiclass Classificatin Through Multidimensional Clustering. In *Genetic Programming Theory and Practice XIII*, volume 13. Springer, Ann Arbor, MI, May 2015.
35. Lee Spector. Assessment of problem modality by differential performance of lexicase selection in genetic programming: a preliminary report. In *Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference companion*, pages 401–408, 2012.
36. Robert Tibshirani, Trevor Hastie, Balasubramanian Narasimhan, and Gilbert Chu. Diagnosis of multiple cancer types by shrunken centroids of gene expression. *Proceedings of the National Academy of Sciences*, 99(10):6567–6572, May 2002.
37. Ryan J. Urbanowicz, Jeff Kiralis, Nicholas A. Sinnott-Armstrong, Tamra Heberling, Jonathan M. Fisher, and Jason H. Moore. GAMETES: a fast, direct algorithm for generating pure, strict, epistatic models with random architectures. *BioData mining*, 5(1):1, 2012.
38. USGS. U.S. geological survey (USGS) earth resources observation systems (EROS) data center (EDC).
39. Leonardo Vanneschi, Francesco Archetti, Mauro Castelli, and Ilaria Giordani. Classification of Oncologic Data with Genetic Programming. *Journal of Artificial Evolution and Applications*, 2009:1–13, 2009.
40. Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. OpenML: Networked Science in Machine Learning. *SIGKDD Explorations*, 15(2):49–60, 2013.