Multidimensional genetic programming for multiclass classification

Article *in* Swarm and Evolutionary Computation · April 2018

DOI: 10.1016/j.swevo.2018.03.015

6 authors, including:

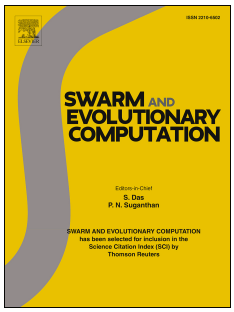
William La Cava
University of Pennsylvania
48 PUBLICATIONS **528** CITATIONS
SEE PROFILE

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

Kourosh Danai
University of Massachusetts Amherst
92 PUBLICATIONS **794** CITATIONS
SEE PROFILE

Lee Spector
Hampshire College
173 PUBLICATIONS **3,607** CITATIONS
SEE PROFILE

Accepted Manuscript



Multidimensional genetic programming for multiclass classification

William La Cava, Sara Silva, Kourosh Danai, Lee Spector, Leonardo Vanneschi, Jason H. Moore

PII: S2210-6502(17)30913-6

DOI: 10.1016/j.swevo.2018.03.015

Reference: SWEVO 390

To appear in: *Swarm and Evolutionary Computation BASE DATA*

Received Date: 15 November 2017

Revised Date: 26 February 2018

Accepted Date: 30 March 2018

Please cite this article as: W. La Cava, S. Silva, K. Danai, L. Spector, L. Vanneschi, J.H. Moore, Multidimensional genetic programming for multiclass classification, *Swarm and Evolutionary Computation BASE DATA* (2018), doi: 10.1016/j.swevo.2018.03.015.

Multidimensional genetic programming for multiclass classification

William La Cava[a,*], Sara Silva[b,c,d], Kourosh Danai[e], Lee Spector[f], Leonardo Vanneschi[c], Jason H. Moore[a]

[a] *Institute for Biomedical Informatics, University of Pennsylvania, Philadelphia, PA, USA*
[b] *BioISI – Biosystems & Integrative Sciences Institute, Faculty of Sciences, University of Lisbon, Lisbon, Portugal*
[c] *NOVA IMS, Universidade Nova de Lisboa, 1070-312 Lisbon, Portugal*
[d] *CISUC, Department of Informatics Engineering, University of Coimbra, Portugal*
[e] *Dept. of Mechanical and Industrial Engineering, University of Massachusetts, Amherst, USA*
[f] *School of Cognitive Science, Hampshire College, Amherst, USA*

Abstract

We describe a new multiclass classification method that learns multidimensional feature transformations using genetic programming. This method optimizes models by first performing a transformation of the feature space into a new space of potentially different dimensionality, and then performing classification using a distance function in the transformed space. We analyze a novel program representation for using genetic programming to represent multidimensional features and compare it to other approaches. Similarly, we analyze the use of a distance metric for classification in comparison to simpler techniques more commonly used when applying genetic programming to multiclass classification. Finally, we compare this method to several state-of-the-art classification techniques across a broad set of problems and show that this technique achieves competitive test accuracies while also producing concise models. We also quantify the scalability of the method on problems of varying dimensionality, sample size, and difficulty. The results suggest the proposed method scales well to large feature spaces.

Keywords: genetic programming, multiclass classification, feature extraction, feature selection, feature synthesis, dimensionality reduction

1. Introduction

Feature selection and feature construction play fundamental roles in the application of machine learning (ML) to classification. Feature selection makes it possible, for example, to reduce high-dimensional datasets to a manageable size, and to refine experimental designs through measurement selection in some domains. The ML community has become increasingly aware of the need for automated and flexible feature engineering methods to complement the large set of classification methodologies that are now widely available in open-source packages such as Weka and Scikit-Learn [10, 30]. Typical classification pipelines treat feature selection and feature construction as pre-processing steps, in which the attributes in the dataset are selected according to some heuristic [9] and then projected into more complex feature spaces using e.g. kernel functions [29]. In both cases the feature pre-processing is often conducted in a trial-and-error way rather than being automated or intrinsic to the learning method. The use of non-linear feature expansions can also lead to classifiers that are black-box, making it difficult for researchers to gain insight into the modelled process by studying the model itself. In this paper we investigate a multiclass classification strategy designed to integrate feature selection, construction and model intelligibility goals into a distance-based classifier to improve its ability to build accurate and simple classifiers.

A well known learning method that implicitly conducts feature selection and construction is genetic programming (GP) [17], which has been proposed for classification [15, 7]. GP incorporates feature selection and construction by optimizing a population of programs constructed from a set of instructions that operate on the dataset features to produce a model. Compared to traditional ML approaches such as logistic regression and decision tree classification, GP makes fewer *a priori* assumptions about the data [22] and allows for various program representations [26]. In addition, GP has well-established methods for optimizing

*Corresponding author. Email: lacava@upenn.edu, Tel: +1 (413) 320-0544

the intelligibilty of models [37]. There have been some promising real-world applications of GP to binary classification [43], but recent work has focused on extending GP to the multi-class classification problem [14, 28], in which there are more than two outcomes to estimate. This previous work suggests that traditional GP fares worse in comparison to other classification methods in the multiclass setting. However, two recent GP-based methods, M2GP [14] and M3GP [28] were shown to perform on par with several other ML strategies in recent studies.

The performance improvements observed by M2GP and M3GP stemmed the incorpration of a distance-based classification strategy into a multi-output GP system. We recently proposed a new method [20] called M4GP, that, although inspired by M2GP and M3GP, significantly improves these two methods. In this paper we extend M4GP by introducing an archiving strategy and by comparing it to recently published methods on data challenges from two different domains. The contributions of this work are:

- M4GP uses a novel (stack-based) program representation, that simplifies the construction of multidimensional solutions compared to M2GP and M3GP (which, instead, used a tree-based representation). This makes the evolutionary process of M4GP more efficient and the final solutions more expressive, readable and easy to understand.
- M4GP incorporates a multiobjective parent selection and survival technique that allows it to clearly and consistently outperform M2GP and M3GP on a wide set of test problems. To the best of our knowledge, this technique had never been used for multi-class classification before.
- We introduce an archiving strategy that maintains a set of optimal trade-off solutions based on complexity and accuracy. The final model is selected from this archive using an internal validation set to reduce ovefitting.

Thanks to these improvements, M4GP is able to improve the best known GP methods for multi-class classification, and finds results that are competitive with the state-of-the-art methods for the studied problems (a set of 26 classification problems, ranging in numbers of classes, attributes and samples). Furthermore, on a set of biomedical data sets with up to 5000 attributes, M4GP is shown to perform on par with state-of-the-art methods while producing smaller models in less time. All these features foster M4GP as the new state-of-the-art multi-class classification method with GP.

The paper is organized as follows: Section 2 presents M4GP. In Section 3 we discuss previous and related work, focusing on the similarities and differences between M2GP, M3GP and M4GP. Section 4 describes our experimental study, presenting the used test problems and the experimental settings. In Section 5, we discuss the obtained experimental results. Finally, Section 6 concludes the paper.

2. M4GP

In multiclass classification (classification into more than two classes), 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 $K > 2$ class labels from the set $\mathcal{C} = \{1 \ \ldots \ K\}$ using n paired examples from the training set $\mathcal{T} = \{(\mathbf{x}_i, y_i), i = 1 \ \ldots \ n\}$.

One way to conduct classification is to measure the similarity of each attribute to the bulk properties of the attributes within each class, and then assign the label corresponding to the most similar group. This strategy is embodied by the 'nearest centroid' classifier [8], which operates as follows. The attributes are partitioned into subsets $\{\mathbf{X}_1 \ \ldots \ \mathbf{X}_K\}$, such that $\mathbf{X}_k$ is the subset of $\mathbf{x}$ with class label k. To classify a new sample $\mathbf{x}' \in \mathbb{R}^p$, the distance of $\mathbf{x}'$ to each subset $\{\mathbf{X}_1 \ \ldots \ \mathbf{X}_K\}$ is measured and the class label corresponding to the minimum distance is assigned as

$$\hat{y}(\mathbf{x}') = j, \ \text{ where } \ j = \arg\min_k D(\mathbf{x}', \mathbf{X}_k) \ , \ k = 1, \ldots, K \tag{1}$$

In its simplest form, D can be the Euclidean distance between $\mathbf{x}'$ and the centroid of $\mathbf{X}_k$. However, previous work has indicated the appropriateness of the Mahalanobis distance (as used in linear discriminant analysis), clearly showing and justifying its advantages compared to other types of metrics, like the Euclidean distance [14]. The Mahalanobis distance is defined as follows:

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

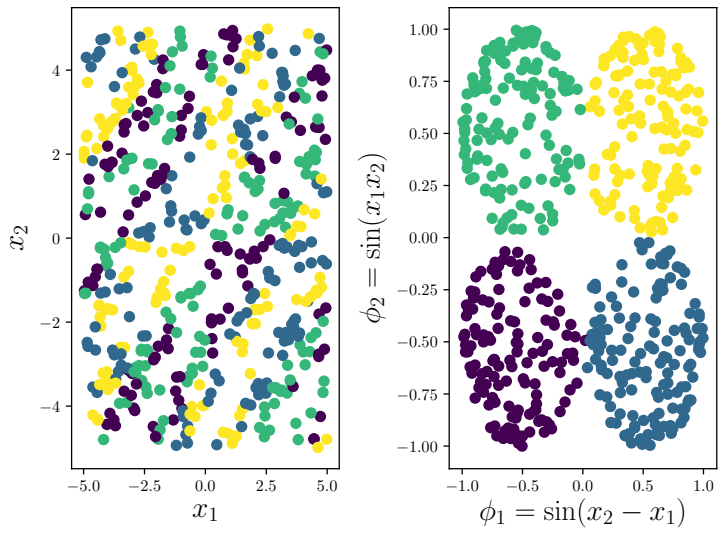


Figure 1: Illustrative example of how data transformations can improve classification. On the left, the raw data for a synthetic 4-class problem for which there is significant overlap of data from different classes in the original space. On the right, the data is transformed into a new space in which the samples fall into neat clusters and class membership is easily discernible.

where $\mu_k \in \mathbb{R}^p$ is the centroid of $\mathbf{X}_k$ and $\mathbf{\Sigma}_k \in \mathbb{R}^{p \times p}$ the within-class covariance matrix. Eqn. 2 renders D_M the equivalent Euclidean distance of $\mathbf{x}'$ from $\mathbf{X}_k$ after scaling by the eigenvalues (variances) and rotating by the eigenvectors of $\mathbf{\Sigma}_k$ to account for the correlation between columns of $\mathbf{X}_k$.

The nearest centroid classifier assumes the within-class samples to be normally distributed about their centroid and separated from the distributions of other classes. For this reason, the data in the right plot of Figure 1 is easily classifiable using nearest centroids. Unfortunately, many real-world problems have data that violates these assumptions, as in the left plot of Figure 1. Furthermore, for high dimensional data, calculating Eq. (2) can be impractical. However, by finding a set of transformations that project the data into a space that delineates class membership, the performance of a nearest centroid classifier can be improved and high-dimensional distance comparisons may be avoided. Furthermore, the relationship between the raw data and the transformed space can be made legible by using symbolic transformations, as in ϕ_1 and ϕ_2 in Figure 1.

The goal of M4GP is to find a set of transformations $\Phi(\mathbf{x}) : \mathbb{R}^p \rightarrow \mathbb{R}^d$ that projects $\mathbf{x}$ into a space in which the samples are more accurately classified by the nearest centroid method. Classification is then conducted with centroids $\mu_{\Phi_k} \in \mathbb{R}^d$, covariance matrix $\mathbf{\Sigma}_{\Phi_k} \in \mathbb{R}^{d \times d}$, and distances $D_M(\Phi(\mathbf{x}), \Phi(\mathbf{X}_k))$. We use GP to find or approximate the optimal synthesized features $\Phi^* = [\phi_1 \ \ldots \ \phi_d]$ that maximize the number of correctly classified training samples, as:

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

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

where $\mathbb{S}$ is the space of possible transformations Φ, f is the classification accuracy, and the indicator function $\mathbb{I} = 1$ if $\hat{y}(\Phi(\mathbf{x}_i)) = y_i$, and 0 otherwise. GP simultaneously optimizes the form of $\Phi(\mathbf{x})$, the subset of $\mathbf{x}$ used in $\Phi(\mathbf{x})$, and the transformation dimensionality ($|\Phi| = d$). Therefore, M4GP can produce low-order models for high dimensional data sets, making Eq. (2) tractable, or produce high-order transformations to improve classification accuracy for highly non-linear problems.

2.1. Genetic Programming

GP represents solutions as programs composed of basic building blocks and iteratively updates these programs based on their quality. In M4GP, each program consists of a set of equations that comprise the

$$\mathbf{i} = [\, x_1 \quad x_2 \quad x_1 \quad x_1 \quad * \quad x_2 \quad x_2 \quad * \quad x_1 \quad x_2 \quad * \,]$$
index: 1 2 3 4 5 6 7 8 9 10 11

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

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

Figure 2: 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.

synthesized features $\Phi(\mathbf{x})$ used to estimate $\hat{y}$. For example, a program $\mathbf{i}$ might construct a polynomial expansion of two attributes as

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

where $\phi_1 = x_1$, $\phi_2 = x_2$, $\phi_5 = x_1 x_2$, and $|\Phi| = 5$. Traditionally, a GP program consists of a single syntax tree with a single output generated at the root node [17]. 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. In order to allow programs to represent multi-dimensional transformations, in M2GP and M3GP, program trees were modified with special nodes in order to allow for multiple outputs at the root [14, 28]. 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 §2.1.1.

The GP population is optimized by probabilistically selecting programs based on their performance and recombining and mutating these programs to make steps in the search space. The selection mechanisms used in M4GP are described in §2.1.2.

2.1.1. Representation

M4GP uses a stack-based representation [31] rather than the more traditional tree-based GP representations. Programs consist of post-fix notation equations, e.g., $\mathbf{i} = [\, x_1 \quad x_2 \quad + \,] \rightarrow \Phi = [x_1 + x_2]$. This representation eliminates the need for specialized nodes used in M2GP and M3GP by supporting multiple outputs by default. Programs are evaluated by stack operations; for example the program in Eq. (5) can be constructed as

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

The execution of program $\mathbf{i}$ is illustrated in Figure 2. In lieu of recursive tree evaluation, 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 stack state is interpreted as the multi-dimensional transformation.

2.1.2. Initialization, Selection, and Variation

M4GP initializes a population of programs of various sizes and 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. Eq. (4) is used as program fitness for selection.

We test three population selection methods in this work: tournament selection [32], lexicase selection [39, 11], and age-fitness Pareto selection [34]. Tournament selection proceeds by selecting a set of individuals (in this case, two) in the current population and choosing the one with better fitness as a parent for the next generation. We describe the other two methods below in more detail.

Lexicase selection. Lexicase selection is a parent selection technique that rewards individuals in the population for performing well on unique combinations of fitness cases, i.e. samples. Each parent selection begins with the entire population in the selection pool. The fitness cases are shuffled randomly and the first case is considered. Any programs in the pool that do not have *exactly* the best fitness on the first case are removed from the selection pool. If more than one individual remains in the pool, the removal process is repeated with the next case. This process continues with additional fitness cases until either 1) only one individual remains in the selection pool, in which case that individual is selected as a parent or 2) no more fitness cases are left, in which case a parent is selected from the remaining pool. This process is repeated for each parent selection.

In lexicase selection, test cases can be thought of as filters, and each parent selection represents a random path through these filters. The selected parents are Pareto-optimal with respect to the fitness cases [21]. Each fitness case has a filtering capacity that is directly proportional to its difficulty since it removes individuals from selection that perform worse than others on it. Selective pressure thereby shifts to cases that are difficult to solve. Because each parent selection uses a new ordering of fitness cases and the cases filter the population in proportion to their difficulty, parent selection favors individuals that perform well on unique combinations of test cases. Lexicase selection therefore promotes individuals with diverse performance observed in previous experiments [11, 21].

Age-fitness Pareto survival. Each individual in Age-fitness Pareto survival is assigned an age equal to the number of generations since its oldest ancestor appeared. Each generation, one new individual is added to the population to provide a small amount of random restart. Selection for variation is random rather than guided, and during breeding a number of offspring equal to the overall population size is created. Afterwards, the set P consisting of the current population and the newly created individuals is culled to the original population size, N, using their accuracy and their age. In this case, a lower age is preferred, and the fitness is inverted so that a lower fitness is also better. Program survival is based on the environmental selection scheme of the Strength Pareto Evolutionary Algorithm 2 (SPEA2) [44]. This algorithm uses two measures to perform this reduction: 1) Pareto strength of an individual, $S(\mathbf{i})$, which is the number of individuals equal to or dominated by $\mathbf{i}$ ($i \prec j$)[1], divided by $P + 1$, and 2) a density estimate $D(\mathbf{i}) < 1$, based on the inverse of the distance to the k-th nearest neighbor [36] of $\mathbf{i}$ in objective space (in this case the objectives are normalized between zero and one). These metrics are used to define a fitness value $F(\mathbf{i})$ that combines the total strength of the individuals $\mathbf{j} \in P$ that dominate $\mathbf{i}$ with density estimate $D(\mathbf{i})$:

$$F(\mathbf{i}) = \sum_{j \in P, j \prec i} S(\mathbf{j}) + D(\mathbf{i}) \tag{6}$$

Every nondominated solution is first copied to the new population. If the new population size is smaller than N, individuals are added in order of lowest (i.e., best) $F(\mathbf{i})$. If the population is larger than N, signifying that there are more than N nondominated solutions, individuals are removed iteratively based on lowest $D(\mathbf{i})$, such that individuals in less dense locations in objective space are preferred. For the latter scenario, the use of $D(\mathbf{i})$ for selection helps preserve spread of solutions along the Pareto front.

Variation. We use the mutation and crossover operators from M3GP [28] in order to simplify the points of comparison with M4GP. These search operators are biased to explore the dimensionality of Φ. The search operators manipulate "sub-trees" of programs, which are equivalent to segments of interacting nodes in the stack-based representation. The mutation operator, with equal probability, chooses one of three actions: i) it replaces a sub-tree with a randomly generated sub-tree; ii) it adds a new randomly generated sub-tree to the end of the program, thereby increasing $|\Phi|$ by one; iii) it deletes a sub-tree corresponding to a "root", thereby

[1]Individual $\mathbf{i}_1$ dominates $\mathbf{i}_2$; i.e., ($\mathbf{i}_1 \prec \mathbf{i}_2$) if $f_j(\mathbf{i}_1) \leq f_j(\mathbf{i}_2) \forall j$ and $f_j(\mathbf{i}_1) < f_j(\mathbf{i}_2)$ for at least one j. Here a lower f_j is better.

reducing $|\Phi|$ by one. The crossover operator similarly chooses between one of two equally probable actions: i) it performs standard sub-tree crossover of the parents, selecting non-"root" nodes; ii) it performs standard sub-tree crossover of "root" nodes. In this case, "roots" are those nodes in the program that produce a value in the final stack, and can be identified from stack-based programs in linear time.

2.1.3. Alternate Classification Strategies

In M4GP, Eqn. 1 is used to make classifications, in contrast to more traditional approaches in GP (see [7] for a review). More commonly, GP is used to evolve discriminant functions (mathematical expressions) that are combined with user-defined thresholds to classify samples. For example, for a binary classification problem, a mathematical expression is evolved to produce positive values for one class and negative values for another class - in other words the threshold is set to zero. When extending to multiclass problems, either the problem can be split into several one-versus-rest problems, as done with SVM, or several thresholds can be specified such that instances of are assigned class 0 for program outputs less than -1, class 1 for outputs between -1 and 1, and class 2 for outputs greater than 1. Another approach is to evolve classification rules using boolean operators (e.g. AND, OR, NOT) and interpret the output as class label. The rule-based approach also requires the problem to be decomposed into binary classification problems to apply to multiclass problems.

We implement two simple approaches to classification using the stack-based, multi-output system described in §2.1 so that we can compare M4GP to more common GP implementations. The first, referred to as 'bool', includes boolean operators {`AND, OR, NOT,` $<, >, \leq, \geq, =$`, IF-THEN, IF-THEN-ELSE`} to evolve a set of classification rules, and interprets the output as a bitstring for classification. For a binary classification problem, the output of the boolean stack is interpreted as class 1 for output [0] and class 2 for output [1]. For a four class problem, the output of the boolean stack is interpreted as follows: [0, 0]:class 1; [0, 1]: class 2; [1, 0]: class 3; [1 1]: class 4.

The second approach which we call 'float' evolves a set of mathematical expressions in equivalent fashion to the GP system in M4GP. However, classifications are made based on the index of the largest value in the stack output. For example, if the program in Figure 2 produced the output [1.5, 3, 2.25, 9, 4.5], class 4 would be assigned to the evaluated sample.

In our experiments, we compare the centroid-based classification strategy used in M4GP to the 'bool' and 'float' strategies described above.

2.1.4. Archiving Solutions

Typically in GP the individual with the highest training accuracy (the so-called 'best-of-run' individual) is chosen as the final model and evaluated on the test set. In order to compare to these previous methods, we do the same with M4GP on the benchmark problems. However, given the importance of interpretability, on subsequent problems we maintain an archive of solutions that represent the best trade-offs of training accuracy and complexity (i.e. the Pareto set [44]). To choose a final model, a small validation set is split from the training set before the run. At the end of the run, the archive is evaluated on the validation set, and the individual with the best score is chosen as the final model. This approach should guard against over-fitting and produce simpler models. The entire archive can also be studied by experts to get a better sense of the building blocks of good solutions to their problem, which can offer insight to the user.

3. Related Work

GP has been used extensively for evolving classification functions $\hat{y}(\mathbf{x})$ directly [15, 7, 26]. In application to multiple classes, the discriminant functions evolved by GP must be thresholded, or the problem must be split into several binary classification problems [7]. To overcome the need for arbitrary thresholds in multiclass problems, M2GP proposed a multi-output GP that evolved $\Phi(\mathbf{x})$ and used the nearest centroid approach (Eq. 1) [14]. M2GP demonstrated in particular that Mahalanobis distance outperformed Euclidean distance in this framework. 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 [28]. An ensemble version of M3GP named eM3GP was also proposed that performed similarly to M3GP with smaller, more legible resultant programs [35]. In contrast to these methods, 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. It also incorporates multi-objective selection strategies and an archive to promote concise solutions.

Our preliminary work with M4GP [20] explored the role of the centroid classifier and tested M4GP's multi-dimensional GP strategy on a set of biomedical problems. A subset of the results suggested M4GP may be well-suited finding nonlinear interactions between genes in genome wide association studies, which motivated our detailed investigation in this paper. Here we expand upon our initial work in the following ways: 1) we rigorously compare M4GP to previous GP and ML methods on a set of benchmark problems; 2) we introduce an archiving strategy for model selection and interpretability; and 3) we conduct a detailed application study of M4GP to non-linear genetics problems with comparisons to recent literature.

In a broader sense, these GP-based methods highlight the unique challenge of feature construction and its role in learning systems [24]. A few recently developed ML methods have similarly merged GP with linear regression [27, 13, 1]. M4GP and its ancestors are more tailored to the multi-class case, which for these regression-based approaches requires thresholding of real-valued outputs. The stack-based approach used by M4GP is also novel compared to the GP representation used in these works. Another active area of research is the merger of GP with decision tree learners [18], with promising initial results.

Feature construction is considered an important aspect of image classification, and GP has been used in several methods in this domain. For example, GP can be used to learn image embeddings for ensemble methods [25], as an interactive learning tool for remote sensing [6], and for detection pulmonary nodes in medical imaging [5]. Liu et al. [25] also noted the GP's potential as a dimensionality reduction technique for large-scale problems. 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. We show in application to a high-dimensional genetics dataset in Section 5.3 that M4GP can be useful for learning low-order representations.

4. Experimental Analysis

The experimental analysis of M4GP is divided into three sections. First we conduct benchmark comparisons, comparing M4GP to results from related GP literature. In the subsequent two section, we benchmark M4GP against published results from two different studies, one concerned with human activity recognition [33, 3] , and the other with disease prediction from genome-wide association studies [38]. Performance is quantified in a number of ways: 1) by classification accuracy on the test sets, 2) by run-time, and 3) by model interpretability according dimensionality of the resultant classifiers as well their forms. The settings for M4GP are shown in Table 2, and whenever possible, match those used for M2GP, M3GP, and eM3GP. Programs are constrained to be between 3 and 100 nodes, and are initialized with a corresponding dimensionality between 1 and 33, constrained such that the minimum sub-program is at least 1 node.

4.1. Benchmark Comparisons

The 8 problems used for benchmark comparisons are shown in Table 1. Six of these problems are from the UCI data repository [23], and the two others, IM-3 and IM-10, are satellite data sets from a United States Geological Survey [42].

Our first experiment is to test whether incorporating centroid-based classification into GP improves its performance. To this end, we compare the 'centroid', 'float', and 'bool' configurations defined in §2.1.3 on the benchmark problems. Next, three versions of M4GP are tested: M4GP with lexicase selection (M4GP-lx), age-fitness Pareto survival (M4GP-ps), and tournament selection (M4GP-tn). On the first 8 problems, we benchmark M4GP against M2GP, M3GP, eM3GP, and several out-of-the-box classifiers from Weka [10]: random forests (RF), random subspace (RS), multi-layer perceptrons (MLP), and support vector machines (SVM). Each method is run for 30 trials, and for each trial the data is randomly partitioned into 70% training and 30% testing.

4.2. Opportunity Activity Recognition Challenge

We compared M4GP's performance to the results of the Activity Recognition Challenge at the 2011 IEEE International Conference on Systems, Man, and Cybernetics [33, 3]. We used the Opportunity Activity Recognition data set (Opp-S2 and Opp-S3 in Table 1), which consists of 242 attributes and approximately

32,000 total samples. We benchmark M4GP's ability to predict four classes of locomotion (stand, sit, walk, lie) from two test subjects (S2 and S3) against the entrants to the original challenge. In order to perform the comparison to published results, the weighted F-measure, F_1, is used as the fitness metric in place of Eq. (4). F_1 measures classification performance as a function of precision ($\frac{TP}{TP+FP}$) and recall ($\frac{TP}{TP+FN}$) as:

$$F_1 = \sum_{\ell}^{k} 2w_\ell \frac{\text{precision}_\ell \cdot \text{recall}_\ell}{\text{precision}_\ell + \text{recall}_\ell} \tag{7}$$

TP is the number of true positives, FP is the number of false positives, and FN is the number of false negatives for class c_ℓ that has k_ℓ samples out of a total of n, yielding the proportion weight $w_\ell = k_\ell/n$. The raw time series data was downsampled as in [3] using a moving average with a window of 500 milliseconds with 250 millisecond steps. Missing data is linearly interpolated, and all data was normalized to zero mean and unit variance.

4.3. Biomedical Application

Our preliminary work suggested that M4GP may be able to outperform traditional classification strategies in identifying nonlinear interactions in simulated genome-wide association studies (GWAS) [20]. Here, we compare M4GP to the state-of-the-art tools used for identifying epistatic interactions from noisy data sets. Recently, an adaptation of the tree-based pipeline optimization tool called TPOT-MDR [38] was shown to perform well on these types of problems in comparison to other methods. TPOT uses GP to optimize machine learning pipelines, searching over the space of Scikit-learn machine learning tools [30]. In [38], Sohn et. al. coupled TPOT with multi-factor dimensionality reduction (MDR) and expert knowledge filter algorithms (EKF) to achieve good results. TPOT-MDR+EKF outperformed XGBoost [4], a state-of-the-art gradient-boosting tree algorithm, on these GWAS problems. Here, we compare M4GP to the results from that study to see if a lighter-weight algorithm (M4GP) with a similar ability to capture non-linearity in the original feature space, can perform as well as TPOT-MDR+EKF.

In addition to previously published methods, we compare to a deep neural network (DNN) approach implemented in PyTorch[2]. We construct a feed-forward neural network with 5 hidden layers of 10 nodes each. The DNN is trained via back propagation using stochastic gradient descent with a learning rate of 0.1 and a maximum of 1000 epochs. The non-linear activation functions in the hidden layers are defined as rectified linear units, and at the output, a softmax function is used. Dropout is used at the input layer for regularization [12].

We follow the experimental design in Sohn et. al. by running 30 replication trials of 10-fold cross validation with balanced accuracy as the metric of performance. The simulated GWAS datasets used in the comparison are generated using GAMETES, which is a tool for embedding epistatic gene-gene interactions into noisy genetic datasets [41]. 16 total problems are compared that vary according to two measures of difficulty: number of attributes (10, 100, 1000, 5000) and signal-to-noise ratio (0.05, 0.1, 0.2, 0.4), also known as heritability in the genetics community. For each problem, a two-way epistatic interaction is present that is predictive of the disease classification, and its presence is masked by the presence of confounding attributes and noisy disease classifications. We use multi-factor dimensionality reduction (MDR) in our analysis, which is a method developed specifically for detecting these interactions via exhaustive search of pairwise comparisons of attributes. Here, the comparison MDR-Pred is given the predictive features beforehand, which is only possible in a simulated study such as this one. MDR-Pred provides a yardstick to indicate near-optimal performance on each dataset.

The EKF algorithms introduced in TPOT-MDR+EKF are based on the Relief family of feature selection algorithms [16]. They are pre-preprocessing filters that are sensitive to interactions and are used in TPOT-MDR+EKF to reduce the attributes to a manageable size. We include a version of M4GP in our comparisons called M4GP+EKF that uses ReliefF [16] to perform feature selection before M4GP is run, reducing the dataset to the top 10 features found by the algorithm. This gives us a way to tease out the effect of feature selection versus the performance of the search algorithms themselves.

[2] `http://pytorch.org/`

Table 1: Data set properties used for benchmark comparisons and Opportunity Activity Recognition Challenge.

	Benchmark Comparisons								Activity Recognition	
	Heart	IM-3	IM-10	Movl	Seg	Vowel	Wav	Yeast	Opp-S2	Opp-S3
Classes	2	3	10	15	7	11	3	10	4	4
Attributes	13	6	6	90	19	13	40	8	242	242
Samples	270	322	6798	360	2310	990	5000	7797	16667	15550

Table 2: M4GP settings. Changes to the settings for Opp-S2 and Opp-S3 are noted in parentheses.

Setting	Value
Population size	500 (1000)
Max Generations	100
Crossover / Mutation	50/50%
Ephemeral random constants [17] range	[0,1]
Program size limits by # nodes	[3, 100 (500)]
Initial dimensionality range (d)	[1,33]
Termination criterion	generations or perfect training accuracy
Trials	30 (10)

5. Results

The best classifiers generated by M4GP for each trial are compared first to benchmark methods in §5.1 and then to other published results in §5.2 and §5.3.

5.1. Benchmark Comparisons

For the 8 benchmark problems, the median best fitness on the test sets for the first eight problems are shown in Table 3 with statistical comparisons. M4GP, across selection methods, produces the best classifiers in terms of test accuracy on five of the eight problems, and RF produces the best classifiers on the remaining three. The best-of-run classifier accuracies over all trials for all methods on the test sets are plotted in Figure 4. Interestingly, the M4GP results are not typically best on the training sets but tend to generalize well. This generalization capacity is demonstrated by side-by-side comparisons of method rankings on training and test sets in Figures 6 and 7, respectively.

The performance of the three M4GP methods on training and test sets over the duration of the run (generations) is shown in Figure 5. It is worth noting that for the Movl problem, which is one of the three problems for which RF generalizes better, M4GP finds a perfect solution to the training data within the first few generations, and prematurely terminates. Past work also notes that this problem is an outlier in terms of GP's behavior [14, 28]. On the Heart problem, M4GP-lx overfits to the training data, as evidenced by the slow decline in test fitness over the generations. Among selection methods, M4GP-ps ranks the best, followed by M4GP-lx and M4GP-tn. On individual problems, M4GP-tn did not outperform M4GP-ps or M4GP-lx; therefore, M4GP-tn was left out of further tests.

5.1.1. Dimensionality of Solutions

Growth in the dimensionality of the solutions is a concern, and so we quantify how the best solutions change during training in Figure 8. Differences between the selection methods are negligible except for on the Yeast problem, where lexicase selection produces smaller solutions. In general dimensionality growth is fairly flat aside for Wav, for which solutions tend to grow throughout training. The Pareto archive (see §2.1.4) used for the applications in the following sections helps address the overfitting issue that can result from solution growth.

5.1.2. Computational Cost

As a population-based method, M4GP's main drawback in comparison to other methods is computational cost, shown in terms of wall-clock time in Table 5. Each trial of the initial eight problems from Table 1 were run on a single core, so the reported time is the time to evaluate population solutions in series. The times

range from about 30 seconds for simple problems (e.g. Movl) to about two and a half hours for larger sample size problems (e.g. IM-10). Unfortunately run-times of the other methods were not available. The selection method used by M4GP (lx, ps, or tn) does not appear to affect the computation times in a significant way. For the Opportunity Activity Recognition problems, parallel processing was used, which reduced run-times, as discussed in the next section.

5.2. Opportunity Activity Recognition Challenge

The results of the Opportunity Activity Recognition problem are shown in Table 4 where the results of M4GP are compared to those from the original challenge in terms of F-measure (Eq. (7)). A number of methods are reported for this problem, including nearest neighbor (NN) classifiers, SVM, decision tree and ensemble versions thereof. For both subjects tested, M4GP-lx and M4GP-ps produce better classifiers than the competition, with lexicase selection performing slightly better than Pareto survival. Because only single best results are reported in literature, we are unable to provide statistical tests for these results. However, we report the median F-measure on the test set for 10 trials of M4GP-lx and M4GP-ps along with confidence bounds, indicating that the median performance of M4GP exceeds the best reported result from the competition.

5.2.1. Dimensionality Reduction

We characterized the interpretability of M4GP models for this problem in terms of 1) feature selection and 2) dimensionality reduction. To quantify 1), we compared the number of attributes in M4GP solutions to the problem dimensionality; for 2), we looked at the solution dimensions, i.e. $|\Phi|$, compared to a principal component analysis (PCA) preserving 98% variance. The results shown in Figure 9 demonstrate that the M4GP solutions to Opp S2 and S3 were small in size and used very few features. The feature selection component is relevant to this particular problem since it may which be beneficial for sensor or measurement selection in future design of experiments.

5.2.2. Computational Cost

The run-times of M4GP for Opp-S2 and Opp-S3 is shown in Table 5. These problems have close to three times as many training samples as IM-10 but were run on 16 core machines, which significantly reduced the run-time to around 10 minutes. In this case, the problem is run in parallel using an island model [2], in which the population is divided among the cores and shuffled at set intervals. The resulting computation times indicate the improvement afforded by parallel processing in GP methods.

5.3. Biomedical Application

M4GP and M4GP+EKF are compared to LR, XGBoost, DNN, TPOT-MDR, and TPOT-MDR+EKF on 16 GAMETES datasets in Figure 10. The results are compared in terms of 10-fold balanced accuracy over 30 trials of each algorithm. The problems are layed out in terms of difficulty: the dimensionality of the datasets grows along the y-axis, and the strength of the signal grows from left to right. Therefore, the bottom left corner represents the hardest problem, and the top-right represents the easiest problem. As mentioned earlier, MDR-Pred results indicate approximately the best possible accuracy for each problem. The results show that M4GP and M4GP+EKF outperform LR on every problem, and outperform XGBoost and DNN on all but one problem (2w_10a_0.1). We also find that M4GP+EKF is able to perform near to optimal (within 5% of MDR-Pred's performance) on 13 of the 16 problems, whereas TPOT-MDR+EKF performs near to optimal on 14 out of 16 problems. The EKF feature selection filter clear improves the performance of both M4GP and TPOT-MDR. Without EKF, M4GP outperforms TPOT-MDR by greater than 10% accuracy on 8 of the problems, and performs similarly (within 10%) on the other 8 problems.

To test the significance of these results, we conduct a Friedman's test of multiple comparisons on the entire suite of problems, which indicates significant differences between the methods (p <2.9e-15). A post-hoc asymptotic symmetry test is then conducted, the results of which we present in Table 6. The test finds no significant differences between the three overall best methods: MDR-Pred, TPOT-MDR+EKF, and M4GP+EKF, which supports our observations of Figure 10. These three methods all significantly outperform LR, XGBoost, and DNN across these problems. TPOT-MDR+EKF outperforms TPOT-MDR and M4GP significantly, whereas M4GP+EKF does not, indicating that TPOT-MDR+EKF produces more consistently strong results compared to M4GP+EKF.

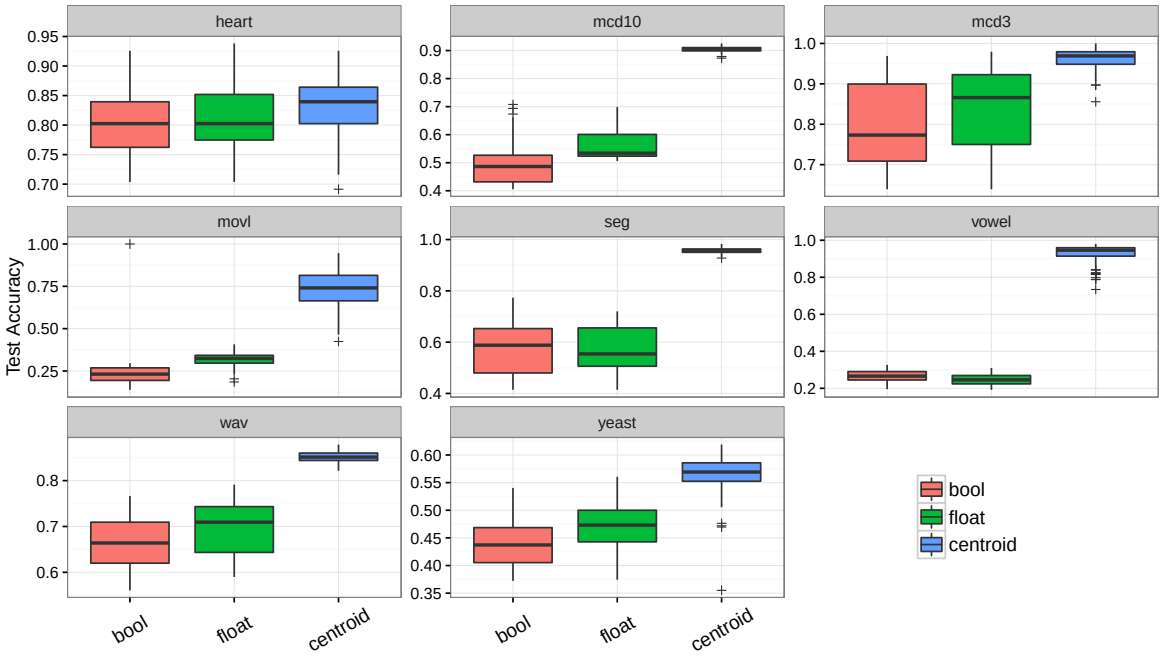


Figure 3: Accuracy of different GP classification methodologies on the test sets of the 8 benchmark problems. 'bool', 'float' and 'centroid' refer to different ways to interpreting the stack outputs of the GP programs, as described in §2.1.3. The 'centroid' method is the one used in M4GP.

5.3.1. Dimensionality of Solutions

Figure 12 demonstrates how the M4GP solutions appear in the Pareto archive as feature sets. The archived models are the best trade-offs between complexity and accuracy found during training. The red line represents the test fitness of these models. As is typical, more complex models often fail to generalize to the test set due to over-fitting. We print the solution form at the "elbow" of the Pareto archive, and for two models with the best generalization fitness. In both figures, the models produced by M4GP identify the correct underlying features in a simple form that is easily interpretable.

5.3.2. Computational Cost and Scaling

In general it appears that M4GP and M4GP+EKF achieve similar performance to TPOT-MDR and TPOT-MDR+EKF, respectively, on these problems. Notably, the M4GP methods produce these results with lower run-times, as shown in Figure 11. Before we compare run-times, we should note that the cluster environment is subject to unequal computational loading between runs, which will affect the measurements; however, all comparisons were run on the same hardware. M4GP and M4GP+EKF take an average of 31 and 21 minutes to conduct 10-fold cross validation, respectively, whereas TPOT-MDR and TPOT-MDR+EKF take on average 1621 and 617 minutes, respectively. The run-time differences between M4GP and TPOT-MDR are expected: whereas M4GP operates over the space of mathematical operations and uses a fairly lightweight nearest centroid classifier, TPOT's search space consists of entire machine learning pipelines, each of which must be trained each generation. The results here suggest the more lightweight approach is sufficient for capturing the interactions between features embedded in the GAMETES genetics data. Figure 11 also shows that M4GP and M4GP+EKF scale well with increased numbers attributes in comparison to TPOT-MDR, TPOT-MDR+EKF, and LR. M4GP's scalability to large feature spaces is due to its representation, which allows the dimensionality of solutions to adapt to the problem.

6. Discussion and Conclusion

A new computational method for multi-class classification, based on GP, was studied in this paper. The new method is called M4GP, and it represents an improvement upon M2GP, M3GP and eM3GP, previous

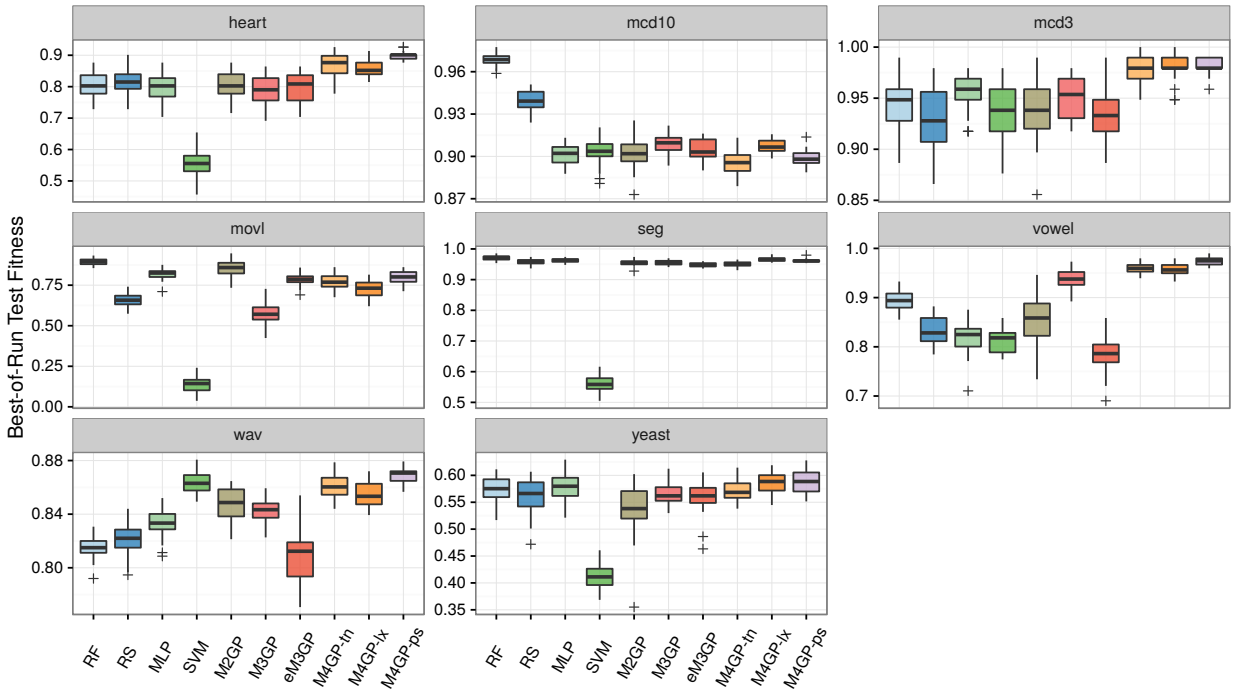


Figure 4: Test set accuracy on the first eight benchmark problems for ten different classification methods.

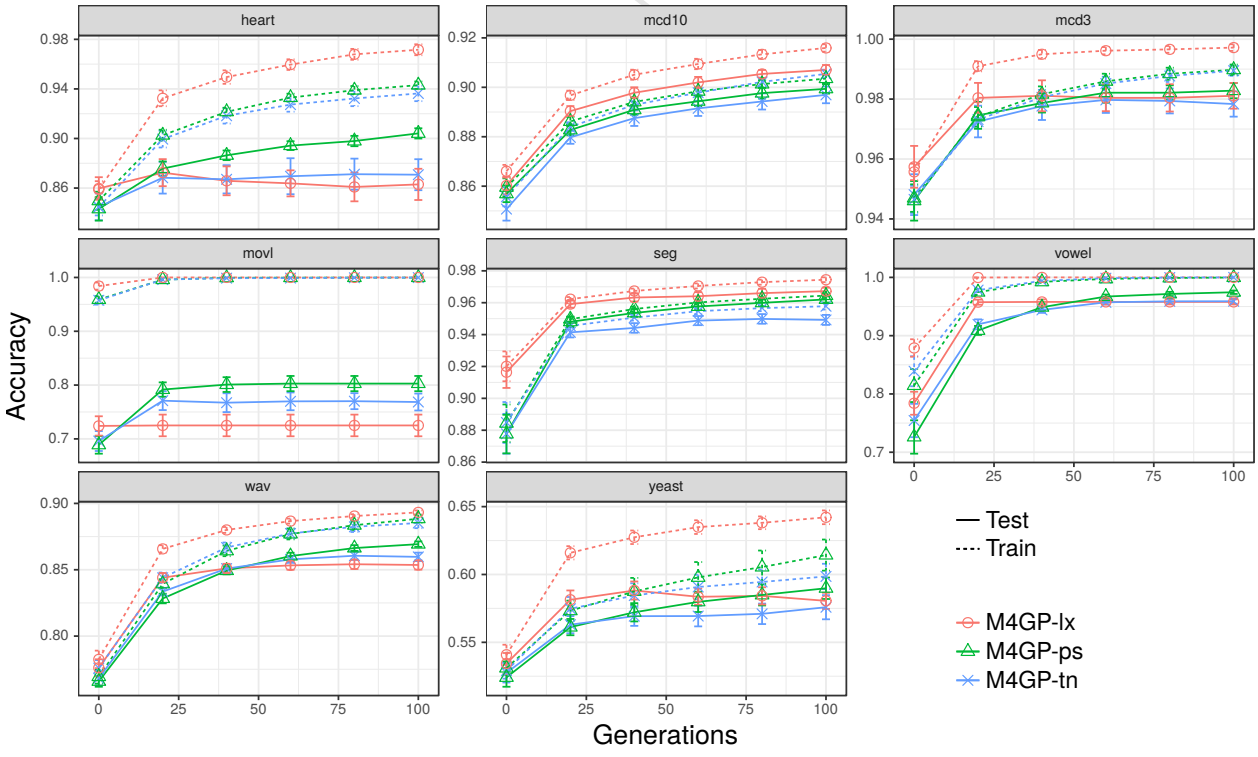


Figure 5: Mean convergence characteristics of M4GP on the training set (dotted lines) and test set (solid lines) over 100 generations. Shapes indicate different selection methods. Error bars denote the confidence intervals.

Table 3: Comparison of best-of-run median test accuracy for the benchmark problems. The best result is highlighted. Significant ($p < 0.01$ according to a pairwise Wilcoxon rank-sum test with Holm correction) improvements with respect to each method is denoted by $a - j$ according to the method labels.

Method	Heart	IM-3	IM-10	Movl	Seg	Vowel	Wav	Yeast
[a]RF	[d]80.2	94.8	[bcde fghij]**96.9**	[bcde fghij]**89.4**	[bcde fgij]**97.3**	[bcde g]89.4	81.5	[de]57.5
[b]RS	[d]81.5	92.8	[cde fghij]93.9	[d f]65.7	[d gj]96.0	[g]82.8	82.2	[d]56.6
[c]MLP	[d]80.2	95.9	90.2	[bd fghj]82.5	[de gj]96.3	[g]82.5	[ab g]83.3	[de]58.0
[d]SVM	55.6	93.8	90.4	14.4	55.8	81.8	[abce fgh]86.3	41.1
[e]M2GP	[d]80.2	93.8	90.2	[bcd fghij]85.9	[d g]95.6	[cd g]85.9	[abc g]84.9	[d]53.8
[f]M3GP	[d]79.0	95.4	[c ij]91.0	[d]57.1	[d]95.6	[abcde g]93.8	[ab g]84.3	[d]56.2
[g]eM3GP	[d]80.9	93.3	[j]90.3	[bd fh]78.6	[d]94.7	78.6	81.2	[d]56.2
[h]M4GP-lx	[abcde fg]85.2	[abcde fg]**97.9**	[ij]90.7	[bd f]73.1	[bde fgj]96.6	[abcde fg]95.6	[abc fg]85.3	[de g]**58.9**
[i]M4GP-ps	[abcde fghj]**90.1**	[abcde fg]97.9	89.8	[bd fh]80.1	[de gj]96.1	[abcde fghj]**97.5**	[abce fghj]**87.1**	[de fg]58.9
[j]M4GP-tn	[abcde fg]87.7	[abcde fg]97.9	89.6	[bd fh]76.9	[d]95.1	[abcde fg]96.0	[abce fg]86.0	[d]56.8

Table 4: Comparison of F-measure on the Opportunity Activity Recognition data set (locomotion) for subjects 1 and 2 (S1 and S2). M4GP is compared to published results using one nearest neighbor (1-NN), SVM, SVM + 1-NN, decision trees (C4.5), k-NN, decision tree (DT) grafting, and Adaboost. The team names are shown in parentheses. The best results in terms of F-measure are highlighted. For M4GP results, the median best result of ten trials is shown with the 95% confidence interval.

Method	F-measure (no *Null* class)	
	S2	S3
1-NN (NStar)	0.88	0.85
SVM (SStar)	0.87	0.83
SVM + 1-NN (CStar)	0.90	0.83
C4.5 (NU)	0.83	0.63
k-NN (MI)	0.87	0.86
DT grafting (MU)	0.86	0.87
Adaboost (UT)	0.74	0.72
M4GP-lx	0.91 ± 0.00	0.89 ± 0.00
M4GP-ps	0.90 ± 0.00	0.88 ± 0.00

Table 5: Run time for M4GP solutions.

		Median Time (hr:min:s)		
Problem	Cores	M4GP-lx	M4GP-ps	M4GP-tn
Heart	1	00:02:29	00:02:12	00:02:28
IM-3	1	00:02:30	00:02:49	00:02:14
IM-10	1	02:09:04	02:21:45	02:28:23
Movl	1	00:00:27	00:02:25	00:02:02
Seg	1	00:32:25	00:39:35	00:38:38
Vowel	1	00:04:21	00:20:37	00:15:57
Wav	1	01:36:44	01:30:17	01:35:12
Yeast	1	00:25:25	00:44:09	00:52:31
Opp S2	16	00:12:30	00:09:16	- (-)
Opp S3	16	00:11:51	00:08:47	- (-)

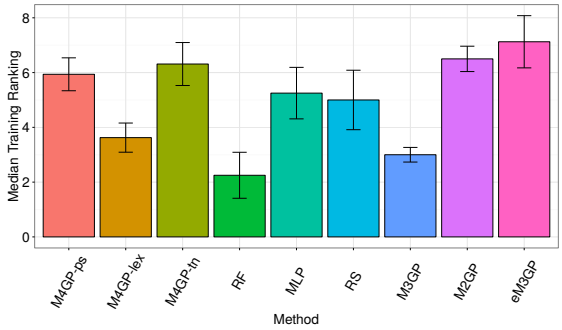


Figure 6: Median training rankings of classifiers on the eight benchmark problems. Bars denote the 95% confidence interval.

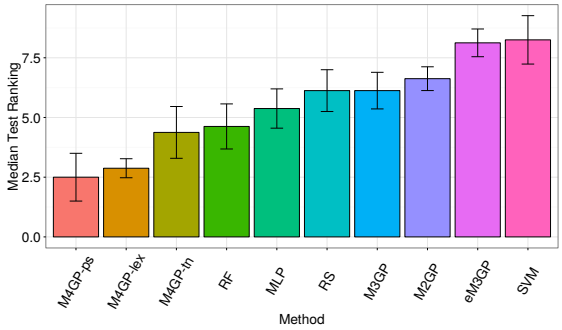


Figure 7: Median test rankings of classifiers on the eight benchmark problems. Bars denote the 95% confidence interval.

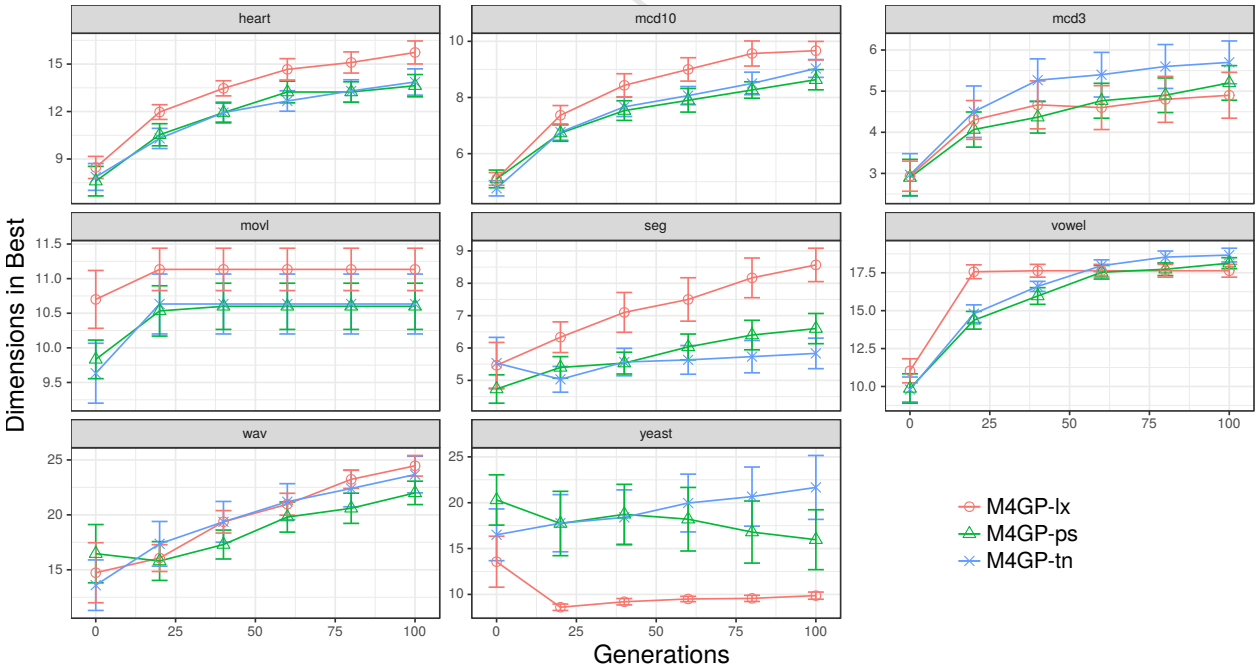


Figure 8: Mean dimensionality of most accurate classifier (on training data) each generation. Error bars indicate the confidence interval.

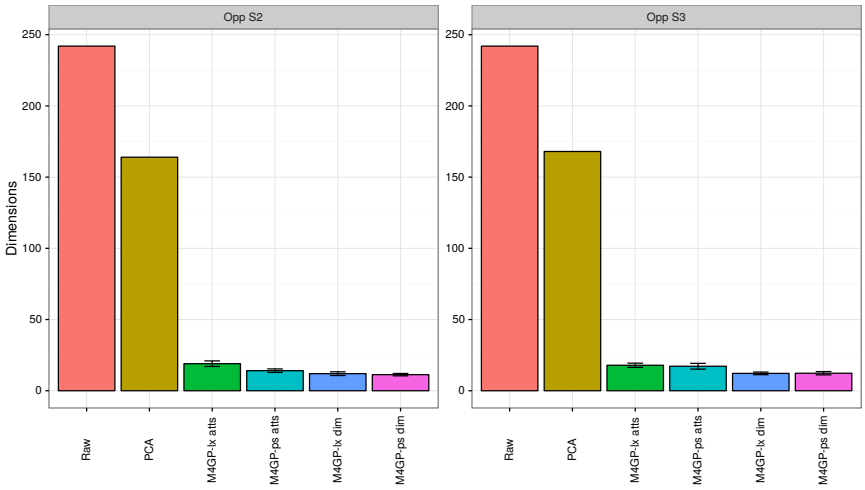


Figure 9: Dimensionality reductions afforded by M4GP on the Opportunity Activity Recognition Challenge. M4GP solution sizes are compared to the original number of attributes and a PCA dimensionality reduction preserving 98% variance. Here, 'atts' refers to the number of attributes used in the M4GP solutions, and 'dim' refers to the dimensions in the solution (i.e. $|\Phi|$).

Table 6: Significance comparisons of methods on the GAMETES problems using a post-hoc Friedman's symmetry test. Results in bold indicate $p < 0.01$.

	LR	XGBoost	DNN	TPOT-MDR	M4GP	M4GP+EKF	TPOT-MDR+EKF
XGBoost	1	-	-	-	-	-	-
DNN	1	1	-	-	-	-	-
TPOT-MDR	0.5	0.8	0.6	-	-	-	-
M4GP	0.2	0.4	0.2	1	-	-	-
M4GP+EKF	**0.001**	**0.005**	**0.002**	0.4	0.8	-	-
TPOT-MDR+EKF	**1e-09**	**2e-08**	**4e-09**	**0.0002**	**0.003**	0.3	-
MDR-Pred	**2e-07**	**2e-06**	**2e-07**	**0.002**	0.02	0.6	1

state-of-the-art techniques for multi-class classification with GP. It extends these methods by introducing a stack-based data flow, integrating advanced selection methods, and maintaining a Pareto archive that preserves concise models and integrates into the final model selection step. M4GP performs significantly better than these GP methods across all benchmark problems according to a Friedman test of rankings for test accuracy. We further demonstrated that advanced evolutionary selection methods, namely lexicase selection and age-fitness Pareto survival, perform better than tournament selection in producing accurate classifiers. Across the range of studied problems, the results indicate that M4GP is a competitive method for developing accurate classifiers with good generalization ability. In addition to being accurate, M4GP provides the flexibility to increase dimensionality for better classification of small dimension problems, and to decrease both the dimension of solutions and the number of necessary attributes for large dimension problems. For the Opp-S2 and Opp-S3 problems in particular, the dimensionality of the classifiers produced by M4GP constitute a major reduction from the original attribute space beyond that afforded by PCA reduction and motivate further applications of this method to large attribute problems. On a set of simulated genetics datasets from the biomedical community, M4GP was also able to produce simple classifiers that were as accurate as the best results in literature. The biomedical application demonstrates the usefulness of maintaining a Pareto archive of solutions when intelligibility is a key factor in modeling the process.

We have found that the computation time of M4GP is quite reasonable on a multi-core machine, with the largest sample size problems requiring approximately 10 minutes to run on a 16 core machine. On the biomedical datasets, M4GP scaled very well to large dimensions (5000 attributes) compared to other methods. Improvements to computation time could be made by further parallelizing the execution of individual models that can be run in parallel at the data (samples) and program (nodes) levels. Note that in this work we only parallelize the evaluation of the population of models.

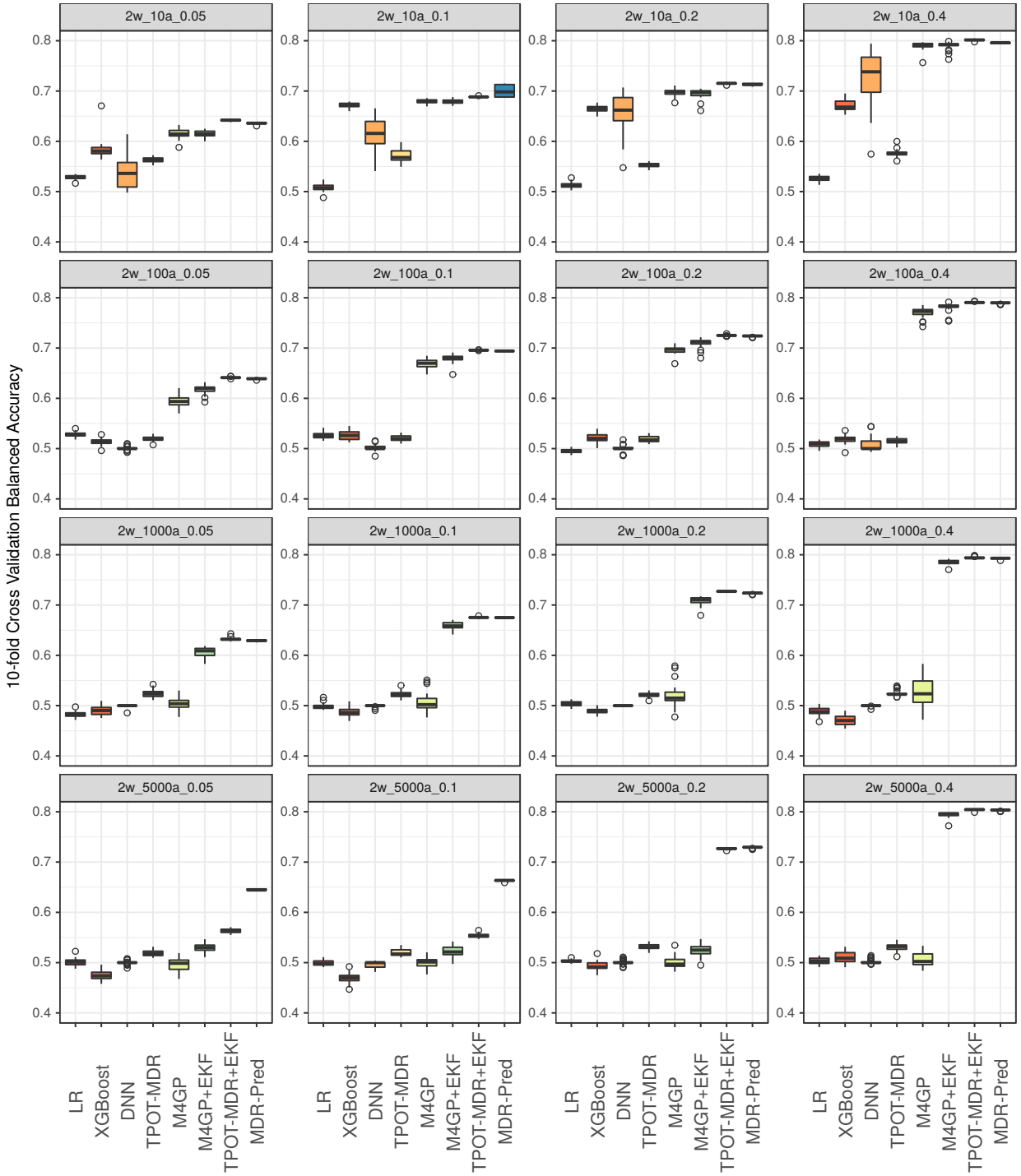


Figure 10: Performance on the biomedical datasets. The subplot titles indicate the dataset; datasets are named by the convention 2w_[# attributes]a_[signal-to-noise ratio]. From left to right, the signal-to-noise ratio increases; from top to bottom, the dimensionality of the dataset (and hence, difficulty) increases.

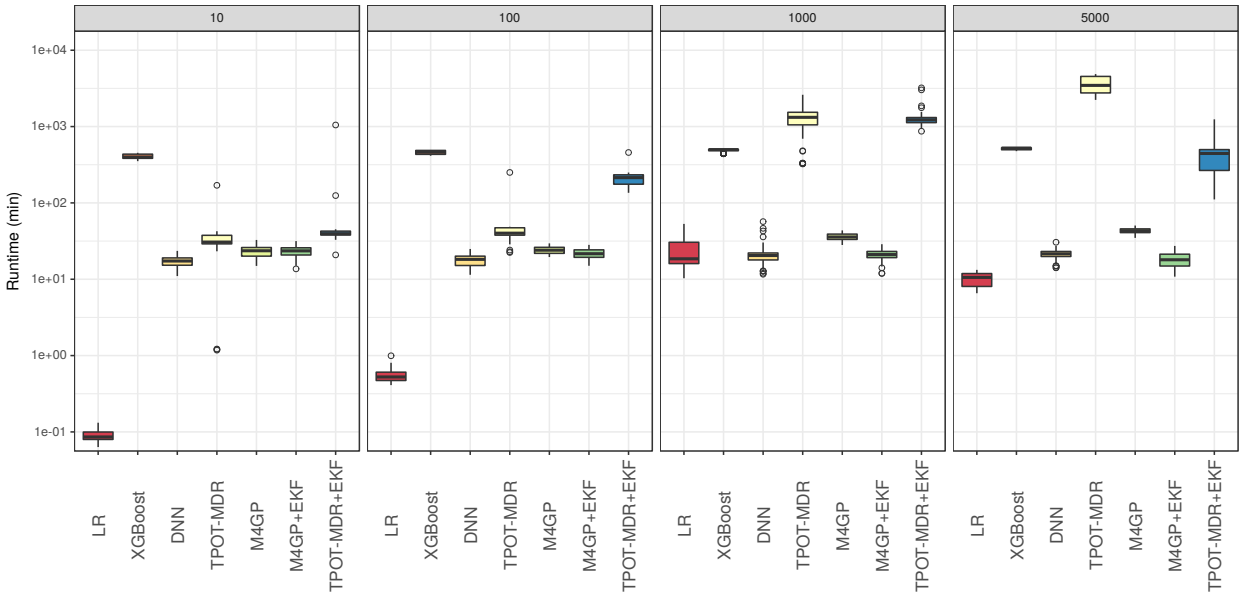


Figure 11: Runtimes on the biomedical datasets. From left to right, the number of attributes in the datasets increase.

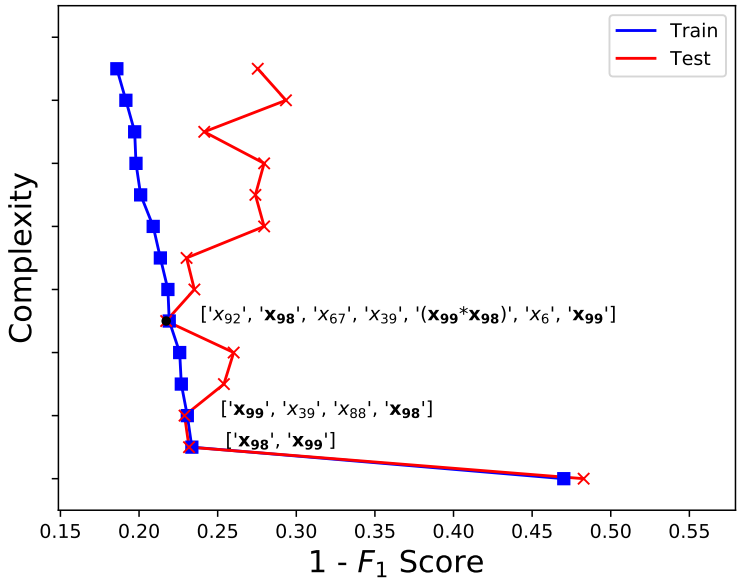


Figure 12: Archive of solutions from M4GP for 2w_100a_0.4her. Blue dots are the training fitness of solutions in the archive, and red is the fitness of those solutions on the holdout test set. The top 3 M4GP solutions are shown with the interacting, predictive genes x_98 and x_99 in bold.

There are several directions for future work that we briefly discuss here. For one, future work could investigate methods for mitigating the computational costs of M4GP. The largest computational cost stems from the inversion of the covariance matrix to compute the Mahalanobis distance (Eq. (2)). It may be possible to minimize this cost by only maintaining a single model Φ and evolving a population that corresponds to the dimensions of Φ, thereby minimizing the number of matrix inversions to once per generation. This approach is taken by ensemble-based GP methods like FEW [19] and EFS [1]. Another avenue of research could investigate other methods of pairing GP with classification strategies. M4GP implements a nearest centroid approach for this task, and previous work, summarized in §2.1.1, has explored other options. Nevertheless, it is not trivial to determine *a priori* which method of classification will pair best with GP for feature synthesis for a particular task. This suggests further research into classifier pairing optimization with GP feature selection and synthesis. Finally, more work is needed to fully examine the role of variation operators in this approach. Future work could analyze the role of mutation and crossover on multi-dimensional GP programs, with the potential to improve search by leveraging extra information regarding the performance of those sub-programs as features of the classifier. It may also be of interest to explore whether or not variation operators with semantic guarantees can be developed in this context.

Acknowledgments. The authors would like to thank Mauro Castelli for his feedback as well as members of the Computational Intelligence Laboratory at Hampshire College. This work is partially supported by the National Science Foundation (NSF)-sponsored IGERT: Offshore Wind Energy Engineering, Environmental Science, and Policy (Grant Number 1068864), as well as Grant No. 1017817, 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 NSF. Biomedical results were supported by National Institue of Health grants AI116794 and ES013508. We acknowledge the financial support of PERSEIDS, PTDC/EMS-SIS/0642/2014, and BioISI R&D unit, UID/MULTI/04046/2013 funded by FCT/MCTES/PIDDAC, Portugal. Computing support was provided by Penn Medicine Academic Computing Services and the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by NSF grant number ACI-1053575[40].

[1] I. Arnaldo, U.-M. O'Reilly, and K. Veeramachaneni. Building Predictive Models via Feature Synthesis. *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '15*. pages 983–990. ACM Press, 2015.

[2] T. C. Belding. The Distributed Genetic Algorithm Revisited. *ICGA-95*, May 1995. arXiv: adap-org/9504007.

[3] R. Chavarriaga, H. Sagha, A. Calatroni, S. T. Digumarti, G. Tröster, J. d. R. Millán, and D. Roggen. The Opportunity challenge: A benchmark database for on-body sensor-based activity recognition. *Pattern Recognition Letters*, 34(15):2033–2042, 2013.

[4] T. Chen and C. Guestrin. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA, 2016. ACM.

[5] W.-J. Choi and T.-S. Choi. Genetic programming-based feature transform and classification for the automatic detection of pulmonary nodules on computed tomography images. *Information Sciences*, 212:57–78, Dec. 2012.

[6] J. A. dos Santos, C. D. Ferreira, R. d. S. Torres, M. A. Gonçalves, 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.

[7] P. G. Espejo, S. Ventura, and F. 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.

[8] J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin, 2001.

[9] I. Guyon and A. Elisseeff. An Introduction to Variable and Feature Selection. *Journal of Machine Learning Research*, 3:1157–1182, Mar. 2003.

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

[11] T. Helmuth, L. Spector, and J. Matheson. Solving Uncompromising Problems with Lexicase Selection. *IEEE Transactions on Evolutionary Computation*, PP(99):1–1, 2014.

[12] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv:1207.0580 [cs]*, July 2012. arXiv: 1207.0580.

[13] I. Icke and J. 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.

[14] V. Ingalalli, S. Silva, M. Castelli, and L. Vanneschi. A Multi-dimensional Genetic Programming Approach for Multi-class Classification Problems. In *Genetic Programming*, pages 48–60. Springer, 2014.

[15] J. K. Kishore, L. 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.

[16] I. Kononenko. Estimating attributes: analysis and extensions of RELIEF. In *European conference on machine learning*, pages 171–182. Springer, 1994.

[17] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA, 1992.

[18] K. Krawiec. Genetic programming-based construction of features for machine learning and knowledge discovery tasks. *Genetic Programming and Evolvable Machines*, 3(4):329–343, 2002.

[19] W. La Cava and J. H. Moore. Ensemble representation learning: an analysis of fitness and survival for wrapper-based genetic programming methods. In *GECCO '17: Proceedings of the 2017 Genetic and Evolutionary Computation Conference*, pages 961–968, Berlin, Germany, 2017. ACM.

[20] W. La Cava, S. Silva, L. Vanneschi, L. Spector, and J. Moore. Genetic Programming Representations for Multi-dimensional Feature Learning in Biomedical Classification. In *Applications of Evolutionary Computation*, pages 158–173. Springer, Cham, Apr. 2017.

[21] W. La Cava, L. Spector, and K. 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.

[22] T. Li, C. Zhang, and M. Ogihara. A comparative study of feature selection and multiclass classification methods for tissue classification based on gene expression. *Bioinformatics*, 20(15):2429–2437, 2004.

[23] M. Lichman. *UCI Machine Learning Repository*. University of California, Irvine, School of Information and Computer Sciences, 2013. `https://archive.ics.uci.edu/ml/index.php`

[24] H. Liu and L. Yu. Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on Knowledge and Data Engineering*, 17(4):491–502, Apr. 2005.

[25] L. Liu, L. Shao, and X. Li. Evolutionary compact embedding for large-scale image classification. *Information Sciences*, 316:567–581, Sept. 2015.

[26] T. Loveard and V. Ciesielski. Representing classification problems in genetic programming. In *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, volume 2, pages 1070–1077. IEEE, 2001.

[27] T. McConaghy. FFX: Fast, scalable, deterministic symbolic regression technology. In *Genetic Programming Theory and Practice IX*, pages 235–260. Springer, 2011.

[28] L. Muñoz, S. Silva, and L. Trujillo. M3GP - Multiclass Classification with GP. In *Genetic Programming*, pages 78–91. Springer, 2015.

[29] K. P. Murphy. *Machine learning: a probabilistic perspective*. Adaptive computation and machine learning series. MIT Press, Cambridge, MA, 2012.

[30] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, and others. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12(Oct):2825–2830, 2011.

[31] T. 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] R. Poli, N. F. McPhee, and J. R. Koza. *A field guide to genetic programming*. [Lulu Press], lulu.com, [S.I.], 2008.

[33] H. Sagha, S. T. Digumarti, J. d. R. Millán, R. Chavarriaga, A. Calatroni, D. Roggen, and G. Tröster. Benchmarking classification techniques using the Opportunity human activity dataset. In *Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on*, pages 36–40. IEEE, 2011.

[34] M. Schmidt and H. Lipson. Age-fitness pareto optimization. In *Genetic Programming Theory and Practice VIII*, pages 129–146. Springer, 2011.

[35] S. Silva, L. Munoz, L. Trujillo, V. Ingalalli, M. Castelli, and L. Vanneschi. Multiclass Classification Through Multidimensional Clustering. In *Genetic Programming Theory and Practice XIII*, volume 13. Springer, Ann Arbor, MI, May 2015.

[36] B. W. Silverman. *Density estimation for statistics and data analysis*, volume 26. CRC press, 1986.

[37] G. F. Smits and M. Kotanchek. Pareto-front exploitation in symbolic regression. In *Genetic Programming Theory and Practice II*, pages 283–299. Springer, 2005.

[38] A. Sohn, R. S. Olson, and J. H. Moore. Toward the automated analysis of complex diseases in genome-wide association studies using genetic programming. *arXiv:1702.01780 [cs, q-bio, stat]*, Feb. 2017. arXiv: 1702.01780.

[39] L. 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.

[40] J. Towns, T. Cockerill, M. Dahan, I. Foster, K. Gaither, A. Grimshaw, V. Hazlewood, S. Lathrop, D. Lifka, G. D. Peterson, R. Roskies, J. R. Scott, and N. Wilkens-Diehr. XSEDE: Accelerating Scientific Discovery. *Computing in Science and Engineering*, 16(5):62–74, 2014.

[41] R. J. Urbanowicz, J. Kiralis, N. A. Sinnott-Armstrong, T. Heberling, J. M. Fisher, and J. H. Moore. GAMETES: a fast, direct algorithm for generating pure, strict, epistatic models with random architectures. *BioData mining*, 5(1):1, 2012.

[42] United States Geological Survey (USGS). USGS earth resources observation systems (EROS) data center (EDC), 2012. `https://eros.usgs.gov/`

[43] L. Vanneschi, F. Archetti, M. Castelli, and I. Giordani. Classification of Oncologic Data with Genetic Programming. *Journal of Artificial Evolution and Applications*, 2009:1–13, 2009.

[44] E. Zitzler, M. Laumanns, and L. Thiele. *SPEA2: Improving the strength Pareto evolutionary algorithm*. Eidgenössische Technische Hochschule Zürich (ETH), Institut für Technische Informatik und Kommunikationsnetze (TIK), 2001.