Towards Effective GP Multi-Class Classification Based on Dynamic Targets

Stefano Ruberto
European Commission Joint Research Centre (JRC), Ispra, Italy
stefano.ruberto@ec.europa.eu

Valerio Terragni
University of Auckland
Auckland, New Zealand
v.terragni@auckland.ac.nz

Jason H. Moore
University of Pennsylvania
Philadelphia, USA
jmoore@upenn.edu

ABSTRACT
In the multi-class classification problem GP plays an important role when combined with other non-GP classifiers. However, when GP performs the actual classification (without relying on other classifiers) its classification accuracy is low. This is especially true when the number of classes is high. In this paper, we present DTC, a GP classifier that leverages the effectiveness of the dynamic target approach to evolve a set of discriminant functions (one for each class). Notably, DTC is the first GP classifier that defines the fitness of individuals by using the synergistic combination of linear scaling and the hinge-loss function (commonly used by SVM). Differently, most previous GP classifiers use the number of correct classifications to drive the evolution. We compare DTC with eight state-of-art multi-class classification techniques (e.g., RF, RS, MLP, and SVM) on eight popular datasets. The results show that DTC achieves competitive classification accuracy even with 15 classes, without relying on other classifiers.

CCS CONCEPTS
• Computing methodologies → Genetic programming; Supervised learning by classification; Machine learning; Artificial intelligence;

KEYWORDS
Genetic Programming, Multiclass classifier, Semantic GP, Dynamic Targets, Evolutionary algorithms, Supervised learning

ACM Reference Format:

1 INTRODUCTION
Multi-class classification is one of the most important research topics in machine learning [2], and plays a crucial role in many modern applications. When dealing with the multi-class classification problem, Genetic Programming (GP) is a fundamental building block for constructing effective classifiers. For instance, GP has a pivotal role in automating the architecture engineering activity of neural networks [10, 14, 19, 23, 53, 56], or to select/construct features for other non-GP classifiers [30, 37, 47, 67]. However, when GP alone is used to evolve a multi-class classifier (not combined with other classifiers) it has a low classification accuracy [20]. Indeed, several researchers reported poor performance of GP classifiers [11, 35, 37, 66, 73], especially when more than two classes are involved.

In this paper, we present Dynamic Target Classifier (DTC), a GP classifier that learns multiple discriminant functions, one for each class to predict. DTC is powered by the dynamic target approach [44, 45, 61] that divides the search problem into a sequence of GP runs. Each run returns a partial model that focuses on the portion of the problem that the previous runs did not capture well [61]. Ruberto et al. show that such an approach is more effective than evolving a single model [61], achieving interesting results in the domains of symbolic regression [61, 62] and image feature learning [59, 60].

We compare DTC with eight state-of-art classifiers (i.e., RF, RS, MLP, SVM, and three GP wrapper approaches [37, 66]) on eight popular datasets. The results show that DTC achieves competitive (and often even better) classification accuracy even with 15 classes. Notably, DTC employs GP to perform the actual classification and does not rely on other classifiers.

DTC presents a novel combination of well-known techniques that, to the best of our knowledge, have never been used in a GP classifier. First, before computing the fitness of an individual, DTC uses linear scaling [32] to optimize the individual with respect to the decision boundary of the discriminant function. Second, to compute the fitness, DTC relies on the hinge-loss function [58] (commonly used by SVM [13]) instead of canonical functions that maximize the number of correct classifications [20]. The hinge loss function has the advantage of penalizing those individuals that correctly classify instances whose prediction is close to the decision boundary [58].

Our proposed fitness function works well in synergy with the dynamic target approach. On one hand, the hinge-loss function focuses only on the portion of the problem in which the current classification model is uncertain [58]. On the other hand, each dynamic target focuses on the portions of the problem not optimized by previous GP runs [61].

The following section describes the background of multi-class classification with GP, and discusses the limitations of current approaches. Section 3 describes DTC in detail, highlighting its novel aspects. Section 4 describes the experiments that we conduct on eight popular datasets for the multi-class classification problem. Section 5 reports the results of the experiment comparing DTC with eight state-of-the-art classifiers. Section 6 concludes the paper highlighting promising future work, and discusses what our results entail for future studies on GP-based multi-class classification.
2 BACKGROUND AND RELATED WORK

Multi-class classification is one of the fundamental problems in Machine Learning [2, 20]. It is the problem of predicting the value of categorical attributes \( C = \{ c_1, c_2, \ldots, c_k \} \) (the classes), based on the values of other \( p \) attributes [20]. In this paper, we assume classification problems with two or more classes \( (k \geq 2) \).

We consider supervised learning, in which a search algorithm induces a classifier from a set of correctly classified data instances \( T = \{ (\vec{x}_i, y_i), i = 1, 2, \ldots, n \} \) (training set), where \( \vec{x}_i \in \mathbb{R}^p \) is the vector of \( p \) attributes.

Neural Networks (NNs) [28] have shown to be very effective in the multi-class classification problem, especially in the image classification domain [64]. Because developing NNs requires labor-intensive architecture engineering, researchers have investigated GP approaches to automate the architecture engineering activity of NNs. For example, the Neuroevolution of Augmenting Topologies (NEAT) techniques [3–5, 8, 10, 14, 43, 53, 56].

In this paper, we present a GP-based classifier that does not rely on other classifiers (e.g., NNs). Applying GP to the multi-class classification could bring some interesting advantages with respect to non-GP classifiers: (i) GP often does not need large training sets to learn competitive models [52], (ii) GP is generally robust to noisy data [42, 59, 60], and (iii) the intrinsic flexibility of GP could lead to classifiers that easily adapt to the specific problem at hand [20].

GP Classifiers. Previous GP classifiers investigated various strategies and model representations: decision trees [33, 70], classification rules [31, 46], and discriminant functions [12, 40, 41, 54]. Interested readers can refer to the survey of Espejo et al. for a literature review on this topic [20]. In this paper, we are targeting Genetic Programming (GP) classifiers represented as discriminant functions. A discriminant function \( f \) is a mathematical expression in which different kinds of operators and functions are applied to the attributes of the instance to be classified, i.e., \( f : \mathbb{R}^n \rightarrow \mathbb{C}, f(\vec{x}) = c_j \).

We now discuss the relevant examples of GP classifiers that use discriminant functions. Paul and Iba propose a GP classifier for binary classification (\( |C| = 2 \)) that learns multiple discriminant functions to classify the training set into two classes [54]. It then classifies instances with a majority voting. Differently, DTC targets multiclass classification with two or more classes (\( |C| > 2 \)), and produces a single discriminant function for each class. Chen and Lu propose a GP classifier that produces a set of single-threshold discriminant functions [12]. Each of these functions classifies an instance in one or more classes. There could be multiple functions associated with a single class. To classify an instance, this approach considers the majority voting scheme. Lin et al. present the Layered Genetic Programming (LAGEP) classifier [40, 41], where each layer evolves a portion of the discriminant function, which is progressively constructed layer-by-layer. Similarly to LAGEP, DTC also progressively constructs the discriminant function. However, Lin et al. evaluated LAGEP mostly for the binary classification problem and using only two layers, while DTC shows competitive results even with 15 classes and using tens of partial models.

Early attempts of GP classifiers based on discriminant functions have shown poor performance [11, 35, 73] (when compared to other state-of-the-art classifiers). In fact, Krzysztof Krawiec remarks that “in most real-world cases, it is rather unreasonable to expect the GP individuals to evolve into complete, well-performing classifiers, even for the two-class discrimination problem” [35]. Castelli et al. and Silva and Tseng, reached a similar conclusion: the performance of GP classifiers is drastically reduced with the increasing number of classes [11, 66].

GP Feature selection/construction for multi-class classifier. Because of the poor performance of GP classifiers, GP-based constructive induction of features has been investigated as a good compromise to employ GP for multi-class classification [35]. More specifically, researchers have investigated the use of GP as a pre-processing step for a multi-class classifier (e.g., SVM, Random Forest, Bayesian approaches, and NN). The idea is to use GP to perform feature selection and construction to improve the discriminating power of the features, and thus improving the performance of a classifier. Following the taxonomy of Espejo et al. [20], such techniques can be divided into filter [48, 51] and wrapper approaches [1, 29, 30, 37, 47, 67, 69]. Both approaches use GP to explore the feature space by combining and selecting features. Filtering approaches rely on statistical properties (e.g., class scattering and information entropy) to compute the fitness, and do not use classification results to guide the evolution. Instead, wrapper approaches rely on the performance of the classifier to compute the fitness, and thus they are more related to DTC.

Wrapper approaches have gained popularity in recent years, showing promising results [37]. Miranda et al. propose a wrapper approach and evaluate it with five different classifiers (K Nearest Neighbors, Naive Bayes, SVM, Decision Tree, and Multilayer Perceptron) in the context of electroencephalogram analysis [47]. The results show better performance when compared to feature selection/construction using PCA. Similarly, Soto et al. improve the performance of Random Forest classifiers by using GP to select and construct features [67]. The techniques of Howley et al. [29] and Sullivan et al. [69] use GP to build Kernel Functions for SVM classifiers. Agapitos et al. [1] present an evolutionary approach that optimizes the euclidean distance metric to improve the performance of a Nearest Neighborhood classifier.

Silva and La Cava et al. proposed a series of wrapper approaches for multi-class classification called MxGP: M2GP [30], M3GP [6, 49, 50, 66] and M4GP [36–38]. M4GP outperforms previous generations of MxGP, as well as highly ranked ML methods, like Multilayer Perceptron and Random Forests, for some multi-class classification problems [37]. The idea of MxGP is to employ GP to perform feature selection/construction in a way that the features become more suitable for a distance-based classifier (e.g., Nearest centroid classifier [22]). MxGP evolves a population of feature transformations calculating the fitness as follows: It creates one cluster per class on the hyper-feature space [6]. The predicted class of each observation is the class of the nearest centroid, according to the Mahalanobis distance [15]. The authors of MxGP show that, for this specific problem, such a distance performs much better than other distances (e.g., Euclidean Distance) [30].

Nevertheless, all wrapper approaches employ GP to improve the performance of other (non-GP) classifiers. Differently, DTC employs GP to perform the actual classification and does not rely on other classifiers.
3 Dynamic Target Classifier (DTC)

This paper presents DTC, a GP-based classifier for the multi-class classification problem. While previous approaches build functions that discriminate among two or more classes [73], DTC builds functions that discriminate one class each, i.e., each function discriminates one class against all the other classes. In particular, for each class $c_j \in C = \{c_1, c_2, \cdots, c_n\}$, DTC considers a "masked" version of the training set, with only two classes $c_j$ and $\overline{c_j} = \{c_k \in C : c_k \neq c_j\}$. DTC learns a set of $k = |C|$ functions (called $F$). Each function $f \in F$ uses one the masked training sets. Given an instance to classify $\tilde{x}$, DTC returns the class $c_j$ that corresponds to the $F[c_j](\tilde{x})$ that returns the highest predicted value (given that $c_j$ is set to 1).

Such a masking approach is commonly used by other non-GP classifiers (e.g., SVM), and has the advantage that it does not assume any distribution of the classes in the feature space. Indeed, previous GP classifiers assign an arbitrary value to each class (e.g., $c_1 = 1, c_2 = 2, \cdots, c_n = n$) [12, 40, 41]. The choice of this value is purely arbitrary, but induces a total order on the classes that is unlikely to reflect the semantics of the problem. Instead, DTC avoids the issue altogether by considering each class individually. However, this approach introduces an imbalance issue that we address by adjusting the fitness scores with an imbalance ratio [7, 18, 39, 55].

Dynamic targets. DTC follows the dynamic target approach [44, 45, 61] that changes at each run, based on the residual errors of previous runs. These GP runs produce multiple models, each focusing on the current target. Recent studies show that such an approach can be more effective than evolving a single model [44, 45, 61].

In particular, DTC follows the SGP-DT [61, 62] dynamic target framework, recently proposed by Ruberto et al. for the symbolic regression domain [61]. SGP-DT runs a GP algorithm multiple times to produce a sequence of GP models (called partial models), where each model focuses on a particular characteristic of the problem at hand. Then, it generates the final model with a linear combination of all the partial models. A key property of SGP-DT is that it does not fix the targets in advance, instead it dynamically discovers them during the GP evolution. At the first iteration, the target is the training instances. At the other iterations, SGP-DT defines the next target as the residual errors of the current iteration, i.e., the values predicted by the partial model minus the values of the (current) target. As such, the next iteration will focus on the problem characteristics that the previous iteration did not approximate well.

Evolving a discriminant function. Algorithm 1 describes how DTC evolves a discriminant function for each class $c_j \in C$. Function EVOLVE-DISCRIMINANT-FUNCTION has four inputs: (i) the class in input ($c_j$), (ii) the training set ($T$), (iii) the number of external iterations ($N_{ext}$), which corresponds to the number of partial models, and (iv) the number of internal iterations ($N_{int}$), which is the number of generations that DTC uses to optimize each partial model. DTC outputs $f$, the discriminant function for class $c_j$.

**Algorithm 1: Dynamic Target Classifier (DTC)**

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$T_{mask} \leftarrow \varnothing$</td>
</tr>
<tr>
<td>2</td>
<td>$numc_j \leftarrow 0$</td>
</tr>
<tr>
<td>3</td>
<td>for each $(\tilde{x}_i, y_i) \in T$ do</td>
</tr>
<tr>
<td>4</td>
<td>if $y_i = c_j$ then</td>
</tr>
<tr>
<td>5</td>
<td>$y_i \leftarrow c_j #$ function $F[c_j]$ corresponds to class $c_j$</td>
</tr>
<tr>
<td>6</td>
<td>else</td>
</tr>
<tr>
<td>7</td>
<td>$y_i \leftarrow c_j #$ function $F[c_j]$ corresponds to class $\overline{c_j}$</td>
</tr>
<tr>
<td>8</td>
<td>$imb \leftarrow n - numc_j$</td>
</tr>
<tr>
<td>9</td>
<td>$\tilde{y} \leftarrow \hat{y} \in T_{mask}$</td>
</tr>
<tr>
<td>10</td>
<td>partialModels $\leftarrow \emptyset$</td>
</tr>
<tr>
<td>11</td>
<td>for ext-iter $1 \ldots N_{ext}$ do</td>
</tr>
<tr>
<td>12</td>
<td>$P' \leftarrow \text{GET-RANDOM-INITIAL-POPULATION()}$</td>
</tr>
<tr>
<td>13</td>
<td>for int-iter $1 \ldots N_{int}$ do</td>
</tr>
<tr>
<td>14</td>
<td>for each $I \in P$ do</td>
</tr>
<tr>
<td>15</td>
<td>$I_0 \leftarrow \text{COMPUTE-LS}(I, \tilde{x}, \tilde{y})$</td>
</tr>
<tr>
<td>16</td>
<td>$\tilde{e} \leftarrow I - I_0(\tilde{x})$</td>
</tr>
<tr>
<td>17</td>
<td>$p \leftarrow \tilde{y} - \hat{y}$</td>
</tr>
<tr>
<td>18</td>
<td>$h_l \leftarrow \ell(p) : \max(0, 1 - \tilde{y} : \hat{y})$</td>
</tr>
<tr>
<td>19</td>
<td>fitness$(I) = \sum_{I \in I} h_l$</td>
</tr>
<tr>
<td>20</td>
<td>if fitness$(I^{*}(\tilde{x})) = 0$ then</td>
</tr>
<tr>
<td>21</td>
<td>$I^{<em>} \leftarrow I^{</em>}$</td>
</tr>
<tr>
<td>22</td>
<td>while $P' #$ not full do</td>
</tr>
<tr>
<td>23</td>
<td>$I \leftarrow \text{Tournament-Selection}(P')$</td>
</tr>
<tr>
<td>24</td>
<td>add mutate$(I)$ to $P'$</td>
</tr>
<tr>
<td>25</td>
<td>$P' \leftarrow P'$</td>
</tr>
<tr>
<td>26</td>
<td>add $I^{*}$ to partialModels</td>
</tr>
<tr>
<td>27</td>
<td>$\hat{y} \leftarrow I^{*}(\tilde{x})$</td>
</tr>
<tr>
<td>28</td>
<td>for each $f \in \text{model} \in \text{partialModels}$ $f \leftarrow 0$</td>
</tr>
<tr>
<td>29</td>
<td>return $f$</td>
</tr>
</tbody>
</table>

This scenario, classifiers can have good accuracy on the majority class but very poor accuracy on the minority class [7]. One possible solution for imbalanced data in ML are sampling techniques [21] that artificially balance the training set. However, they suffer from overfitting and poor generalization issues [7].

Instead, we use an imbalance ratio to amplify the incorrect predictions of the minority class by a factor of the class imbalance [7]. For this reason, we count the number of instances in $T_{mask}$ that belong to class $c_j$ ($numc_j$). DTC needs this information to compute the imbalance ratio $imb$ at line 10, i.e., the ratio of the instances that do not belong to class $c_j$ ($n - numc_j$) to those that belong ($numc_j$). When computing the fitness function, DTC will rely on this ratio to adjust the hinge-loss scores of the instances belonging to $c_j$. 


DTC then starts the external and internal iterations of the dynamic target approach [61]. It initializes the current target \( t \) with the dependent variables \( \tilde{y} \) of the masked training set (line 11), and starts the \( N_{\text{ext}} \) external iterations. At each external iteration (lines 13-34), DTC instantiates \( P \) (line 14) with a fresh randomly generated population using the \textit{ramped-half-and-half} approach [34]. Then, it starts the \( N_{\text{int}} \) iterations to evolve \( P \).

### 3.1 Fitness Computation

DTC computes the fitness scores of the individuals in \( P \) by relying on the \textit{linear scaling} [32] and the \textit{hinge-loss function} [58].

#### Linear Scaling [32].

Function \texttt{compute-ls} calculates the linear scaling [32] for each individual \( I \in P \). Linear scaling adapts instantly an individual to the best possible scale with respect to the current target \( \langle \tilde{t} \rangle \) [32]. This is particularly useful in our case because by using the hinge-loss function we introduce a boundary (0), and the individuals might not be optimized for such an arbitrary boundary. Note that we could have chosen any arbitrary boundary, we chose 0 following standard practices. In other words, linear scaling transforms the individuals so that their potential fit with the current target is immediately given; we do not need to wait for GP to produce an individual optimized for the boundary [32]. And thus, in a dynamic-target approach linear scaling reduces the number of both external and internal iterations [61]. Fewer iterations result in populations with simpler structural complexity and less computational cost [61].

We compute the linear scaling of an individual \( I \) following the equations of Keijzer [32]:

\[
\text{Linear scaling} : I_{ls} = a + b \cdot I 
\]

where

\[
a = \tilde{t} - b \cdot \bar{t}(\bar{x}) \quad \text{and} \quad b = \frac{\sum_{i=1}^{n} (I_i - \tilde{t}) \cdot (I(x_i) - \bar{t}(\bar{x}))}{\sum_{i=1}^{n} (I(x_i) - \bar{t}(\bar{x}))^2}
\]

where \( n \) is the number of training cases, and \( \bar{t} \) and \( \bar{t}(\bar{x}) \) denote the average target and average output value, respectively. The cost of computing the linear scaling coefficients for each external iteration is \( O(N_{\text{int}} \cdot n \cdot |P|) \).

#### Hinge-loss function [58].

The hinge-loss is a loss function commonly used for training classifiers, most notably SVMs. At the best of our knowledge, we are the first to use it in a GP classifier.

Figure 1 depicts the intuition behind the hinge-loss function. The x-axis represents the distance from the boundary (0) of a given instance, and the y-axis represents the loss value (or penalty) that the function returns depending on the distance from the boundary. The function penalizes instances that are incorrectly classified proportionally to the distance from the boundary (<0 the red area in Figure 1). A positive distance from the boundary incurs a low hinge loss, or no hinge loss at all if the prediction is further away from the boundary (>1 the green area in Figure 1). If the distance from the boundary is 0 (meaning that the instance is exactly on the boundary), then we incur a loss size of 1.

The advantage of the hinge-loss function is that it captures the situation in which the instance is correctly classified but with low margin (the gray area in Figure 1), so that a search algorithm can further improve it. This is the main difference from the commonly used Root Mean Square Error (RMSE) and accuracy-based loss functions. RMSE penalizes instances at the increasing of the distance from the lines of the best fit (+1 and -1 in our case). Accuracy-based loss functions do not distinguish correctly classified instances that are very far (the green area in Figure 1) from or very near (the gray area in Figure 1) from the boundary.

The hinge-loss function is defined as:

\[
\text{Hinge-Loss} : \ell(\tilde{p}) = \max(0, 1 - \tilde{t} \cdot \tilde{p})
\]

where \( \tilde{t} \pm 1 \) is the intended output (current target) and the vector \( \tilde{p} \) are the predicated values.

The hinge-loss is used Root Mean Square Error (RMSE) and accuracy-based loss functions. RMSE penalizes instances at the increasing of the distance from the lines of the best fit (+1 and -1 in our case). Accuracy-based loss functions do not distinguish correctly classified instances that are very far (the green area in Figure 1) from or very near (the gray area in Figure 1) from the boundary.

Running example. We exemplify the fitness computation with a running example. Let assume that we have a masked training set \( T_{\text{mask}} \) with three instances (\( n=3 \)), actual prediction values \( \tilde{y} = \langle 1, -1, 0 \rangle \), and imbalance ratio \( \text{imb} = 10 \).

Let assume that at the first internal iteration DTC computes the fitness of a scaled individual \( I_{ls} \) that returns \( I_{ls}(\bar{x}) = \langle -0.7, -0.3, 0 \rangle \) when evaluated on the training set. Lines 19 and 20 compute the error \( \bar{e} \) and prediction \( \tilde{p} \). In this case, \( \tilde{p} = I_{ls}(\bar{x}) \) because at the first iteration there are no previous errors to approximate (i.e., \( \tilde{t} = \tilde{y} \), line 11). The hinge-loss \( \ell(\tilde{p}) \) is the following:
[\textit{hinge-loss for } i = 1]: the actual value of this instance is +1 and the predicted value is -0.7, meaning that the point is on the wrong side of the boundary, thus incurring a large hinge loss of \( h_{l_1} = 1 - (1 - 0.7) = +1.7 \) (because \( \max(1.7, 0) = +1.7 \)).

[\textit{hinge-loss for } i = 2]: the actual value of this instance is +1 and the predicted value is -0.3, which is smaller than 0 but greater than -1. The predicted class would be correct but the value is still too near to the boundary and get a moderate penalty of \( h_{l_2} = 1 - (-1 - 0.3) = +0.7 \) (because \( \max(0.7, 0) = +0.7 \)).

[\textit{hinge-loss for } i = 3]: the actual value of this instance is +1 and the predicted value is +0, which means that the point is on the boundary, thus incurring a cost of \( h_{l_3} = +1 \).

The fitness of \( I_{l_1}^k \) is \( 1((1.72 \cdot (imb = 10))) + (0.72) + (1^2) \) = 28.9 + 0.49 + 1 = 30.39.

After \( N_{\text{int}} \) internal iterations, before starting a new external iteration, DTC updates the new target \( \bar{t} \) based on the residuals errors of the current target and the best model \( I_{l_1}^* \) (line 34). Let assume that the best model \( I_{l_1}^* \) is the individual discussed above (\( I_{l_1}^* \)). Thus, the new target is \( \bar{t} = \bar{y} - I_{l_1}^*(\bar{x}) = (+1.7, -0.7, -1) \).

At the second iteration, let us assume that DTC computes the fitness of a scaled individual \( I_{l_2}^b \) that returns \( I_{l_2}^b(\bar{x}) = (+1.2, -0.3, -1.0) \) when evaluated on the training set. This time at line 20 we need to consider also the previous errors (the target \( \bar{t} \)) in order to get the prediction \( \bar{p} \) and calculate the hinge loss.

The difference between the target \( \bar{t} \) and the partial model \( I_{l_2}^b(\bar{x}) \) gives the error with respect to the training set \( \bar{e} = (+0.5, -0.4, 0) \). The values predicted by \( I_{l_2}^b \) on the training set are \( \bar{e} = (-0.5, -0.4, 0) \). The corresponding hinge loss will be \( h_{l_2} = (+0.5, +0.4, 0) \). Then, DTC computes the fitness score of \( I_{l_2}^b \) as explained above.

### 3.2 Evolution

After DTC computes all fitness scores for each individual in the current population \( P \), it gets the individuals in \( P \) with the best (i.e., lowest) fitness score \( I_{l_1}^b \). DTC then adds it to the list of partial models. It then checks if its fitness score is zero. If so, then the external loop at line 13 prematurely ends and DTC combines all partial models in a single discriminant function \( f \). Otherwise (the common case), DTC evolves \( P \) into \( P' \).

Following the dynamic-target framework [61], DTC uses a variant of the classical GP algorithm that does not use any form of crossover. Ruberto et al. show that such a variant is effective for the dynamic-target approach [61]. More specifically, DTC adopts the standard elite operator (line 28), uses the tournament selection [57] and the canonical mutation operators for tree-like individuals [61]. The tournament selection selects one individual \( I_{l_3} \in P \) (line 30), mutates it with a certain probability (line 31) and adds it to \( P' \).

After DTC completes all the \( N_{\text{ext}} \) external iterations, it returns the discriminant function \( f \) as a linear combination of all the partial models produced during the external iterations (using the linear scaling coefficient already computed at line 18). That is the sum of the partial models with their \textit{linear scaling} [32] coefficients. Intuitively, by combining all partial models we are summing all the estimates of the residuals, and thus obtaining a function \( f \) that well approximates the training set [60].

\begin{algorithm}[h]
\begin{algorithmic}[1]
\Procedure{Algorithm 2: The DTC classifier}{\textbf{input :} \( T = \{ (x_i, y_i), i = 1, 2, \ldots, n \} \): training set \( C = \{ c_1, c_2, \ldots, c_k \} \): classes to predict \( N_{\text{ext}} \): number of external iterations \( N_{\text{int}} \): number of internal iterations} \textbf{output :} \( F \): \( k \) discriminant functions for each class in \( C \)
\State \( F[1] \leftarrow \emptyset \)
\For {each class \( c_j \in C \)}
\State \( F[c_j] \leftarrow \text{EVOLVE-DISCRIMINANT-FUNCTION}(c_j, T, N_{\text{ext}}, N_{\text{int}}) \)
\EndFor
\State \Return \( F \)
\EndProcedure
\end{algorithmic}
\end{algorithm}

### Algorithm 2: The DTC classifier

At the second iteration, let us assume that DTC computes the fitness of a scaled individual \( I_{l_2}^b \) that returns \( I_{l_2}^b(\bar{x}) = (+1.2, -0.3, -1.0) \) when evaluated on the training set. This time at line 20 we need to consider also the previous errors (the target \( \bar{t} \)) in order to get the prediction \( \bar{p} \) and calculate the hinge loss.

The difference between the target \( \bar{t} \) and the partial model \( I_{l_2}^b(\bar{x}) \) gives the error with respect to the training set \( \bar{e} = (+0.5, -0.4, 0) \). The values predicted by \( I_{l_2}^b \) on the training set are \( \bar{e} = (+0.5, -0.4, 0) \). The corresponding hinge loss will be \( h_{l_2} = (+0.5, +0.4, 0) \). Then, DTC computes the fitness score of \( I_{l_2}^b \) as explained above.

### 3.3 Classification

Algorithm 2 describes how DTC creates the discriminant functions \( F \) (\textit{train-classifier}) and classifies new instances (\textit{GP-classifier}).

Function \textbf{train-classifier} has four inputs: (i) the training set \( T \), (ii) the set of classes to predict \( C = \{ c_1, c_2, \ldots, c_k \} \), (iii) the number of external iterations \( N_{\text{ext}} \), and (iv) the number of internal iterations \( N_{\text{int}} \). It constructs the set of discriminant functions by invoking Function \textit{EVOLVE-DISCRIMINANT-FUNCTION} described in Algorithm 1 for each class \( c_j \in C \).

Function \textbf{GP-classifier} has three inputs: (i) the new instance to classify \( \bar{x} \), (ii) the set of possible classes \( C = \{ c_1, c_2, \ldots, c_k \} \), (ii) the set of discriminant functions \( F \) returned by Function \textbf{train-classifier}. It returns the set of discriminant functions by invoking Function \textbf{train-classifier}. To classify a new instance \( x \), DTC computes the \textit{predicted value} of every discriminant function in \( F[c_j](\bar{x}) \), where \( j = 1 \ldots k \) (line 45). DTC returns the predicted class as the one that corresponds to the function with the highest \textit{predicted value} (line 46). Ideally, only one function in the set returns a value greater than zero. However, if multiple functions in \( F \) return values greater than zero, DTC considers the highest one. This is because functions with higher \textit{predicted values} are likely to have more confidence in their predictions.

### 4 EXPERIMENT

We now describe an experiment that we conducted to evaluate the classification accuracy of DTC on eight popular datasets. Moreover, we compare DTC with eight state-of-the-art classifiers.

### 4.1 Datasets

Table 1 shows the characteristics of the eight datasets that we consider in our experiments. The number of classes ranges from 2 (heart) and 15 (mvol), and the number of instances from 270 (heart) to 5,000 (wav). These are popular datasets for the multi-class classification problem. Six of these datasets are from the UCI data
repository [17], and the other two, im-3 and im-10, are satellite datasets from the United States Geological Survey (USGS) [71]. We choose them because they are the datasets used by LaCava and Silva to evaluate M2GP, M3GP, and M4GP.

Heart Statlog [17] (heart) has 270 instances and 13 attributes that includes age, sex, cholesterol, type of pain, electrocardiogram characteristics. This classification problem has two classes representing the presence or absence of a heart disease.

Waveform_40 [9, 17] (wav) has three classes of waves generated by a combination of 2 or 3 “base” waves. It includes 5,000 instances generated by adding noise (mean 0, variance 1) in each of the 40 continuous attributes.

Image Segmentation [17] (segm) has 2,310 instances with 19 attributes. The attributes represent the characteristics of a hand-segmented region of 3x3 pixels randomly chosen from seven outdoor images (i.e., the seven classes to predict).

Yeast [17] (yeast) comprises 1,484 instances. Each instance is a set of eight binary attributes, representing yeast protein traits. The 10 classes to predict are the possible 10 cellular localization sites of these proteins. This dataset is very imbalanced [17].

Vowel [72] (vowel) includes 11 different vowel classes of the Japanese language. Each of these 11 vowels is uttered 6 times by 15 different speakers for a total of 990 instances [72].

Movement Libras [16] (movl) has 15 classes of 24 instances each (360 instances in total), represented as 90 continuous attributes. Each class represents a type of hand gesture. This problem has the highest number of classes (15) among the eight considered datasets.

The datasets im-10 and im-3 consist of satellite images from the USGS [71]. The dataset im-10 includes 6,798 images, while im-3 322 images, which belong to ten and three classes, respectively.

<p>| Table 1: The characteristics of the eight datasets used in our experiment |
|---------------------------------|-----------------|----------------|----------------|----------------|----------------|----------------|----------------|</p>
<table>
<thead>
<tr>
<th># classes</th>
<th>heart</th>
<th>im-3</th>
<th>wav</th>
<th>segm</th>
<th>im-10</th>
<th>yeast</th>
<th>vowel</th>
<th>movl</th>
</tr>
</thead>
<tbody>
<tr>
<td># instances</td>
<td>270</td>
<td>322</td>
<td>5,000</td>
<td>2,310</td>
<td>6,798</td>
<td>1,484</td>
<td>990</td>
<td>360</td>
</tr>
<tr>
<td># attributes</td>
<td>13</td>
<td>6</td>
<td>40</td>
<td>19</td>
<td>6</td>
<td>8</td>
<td>13</td>
<td>90</td>
</tr>
</tbody>
</table>

| Table 2: Configuration parameters of DTC |
|----------------------------------------|-------|----------------|
| Parameter description | value | Parameter description | value |
| population size (|P|) | 100 | size elitism | 1 |
| prob. of mutation | 100% | depth of tree (initialization) | 1:4 |
| prob. mutation leaf level | 70% | max. depth of tree (mutation) | 5 |
| Ephemeral Random Constants (ERC) | [1:1] | max. depth of tree (all phases) | 15 |
| tournament size | 2 | internal iterations (Nint) | 50 |

M3GP [66] addresses the limitation of M2GP that needs to know in advance the number of dimensions (n). M3GP proposes genetic operators to dynamically explore the number of dimensions to optimize the classification.

eM3GP [66] is a variant of M3GP that uses ensembles. In particular, it identifies the best feature transformations and uses different strategies to build ensembles of such transformations.

M4GP [36–38] follows the same approach as M3GP with the main difference of using a stack-based representation of individuals (while both M2GP and M3GP use tree-like individuals). Such a representation gives the advantage that an individual returns multiple outputs, and thus reduces the exploration cost [57]. There are three variants of M4GP, each adopting a different parent selection criterion: (i) tournament selection [57], (ii) lexicase [25, 68], and (iii) Age-Fitness Pareto survival (AFP) [65], which exploits an archive of best individuals. For a fair and meaningful comparison, we consider the variant of M4GP that uses tournament selection, because it is the same selection criterion used by DTC.

We chose to compare with wrapper approaches because they are known to drastically outperform previous GP classifiers that evolve discriminant functions [37, 66]. Indeed, several researchers reported poor classification accuracy for GP classifiers, especially with classification problems with more than two classes [11, 35, 73].

4.2 Classifiers

We compare our approach with eight different classifiers. We consider four standard non-GP classifiers: Random Forest (RF) [26], Random Subspace (RS) [27], Multi Layer Perceptron (MLP) [63], and Support Vector Machines (SVM) [13]. We also consider the following four state-of-the-art wrapper approaches, which employ GP to select/construct features for a nearest centroid classifier [22], M2GP [30] uses multiple trees to transform the original features in a new n-dimensional space (where n is decided in advance). M2GP computes the class centroids and classifies instances according to the nearest centroid. M2GP and all of its successor employ the covariance matrix and the Mahalanobis distance to compute the distance between instances and centroids [30].

4.3 Experimental Setup

Table 2 shows the configuration parameter values of DTC, which are the same used by the dynamic target approach SGP-DT [61]. Note that the probability of mutation is 100% because DTC does not use the crossover operator. While we kept the number of internal iterations fixed for all experiments (Nint = 50), for the number of external iterations (Next) we explored four different values (“low”: 20, “medium”: 40, “high”: 80, “very high”: 160) on some sample data. We noticed that DTC needs a “high” or “very high” Nex with datasets with a high number of instances or classes. yeast is the only dataset that does not follow this trend (because it is very imbalanced). As such, we choose the following values of Nex and we kept them the same for every trial: “low” (yeast, heart), “medium” (movl, im-3), “high” (wav, vowel), and “very high” (segm, im-10).
Towards Effective GP Multi-Class Classification Based on Dynamic Targets

Table 3: Median classification accuracy (%) of the 30 trials on the TEST set (statistical significance is marked in underlined bold)

<table>
<thead>
<tr>
<th>dataset</th>
<th>RF</th>
<th>RS</th>
<th>MLP</th>
<th>SVM</th>
<th>M2GP</th>
<th>M3GP</th>
<th>eM3GP</th>
<th>M4GP</th>
<th>DTC</th>
</tr>
</thead>
<tbody>
<tr>
<td>heart</td>
<td>80.25</td>
<td>81.48</td>
<td>80.25</td>
<td>55.56</td>
<td>80.25</td>
<td>79.01</td>
<td>80.86</td>
<td>87.65</td>
<td>81.48</td>
</tr>
<tr>
<td>im-3</td>
<td>94.85</td>
<td>92.78</td>
<td>95.88</td>
<td>93.81</td>
<td>95.36</td>
<td>8.45</td>
<td></td>
<td>97.94</td>
<td>93.81</td>
</tr>
<tr>
<td>wav</td>
<td>81.50</td>
<td>82.20</td>
<td>83.33</td>
<td>86.30</td>
<td>84.87</td>
<td>84.33</td>
<td>81.23</td>
<td>86.03</td>
<td>85.90</td>
</tr>
<tr>
<td>segm</td>
<td>97.26</td>
<td>95.96</td>
<td>96.32</td>
<td>55.84</td>
<td>95.60</td>
<td>95.61</td>
<td>94.73</td>
<td>95.09</td>
<td>96.54</td>
</tr>
<tr>
<td>im-10</td>
<td>96.86</td>
<td>93.92</td>
<td>90.22</td>
<td>90.19</td>
<td>90.96</td>
<td>90.31</td>
<td>89.56</td>
<td>93.46</td>
<td></td>
</tr>
<tr>
<td>yeast</td>
<td>57.53</td>
<td>56.63</td>
<td>57.98</td>
<td>41.12</td>
<td>53.82</td>
<td>56.19</td>
<td>56.19</td>
<td>56.84</td>
<td>59.35</td>
</tr>
<tr>
<td>vowel</td>
<td>89.39</td>
<td>82.83</td>
<td>82.49</td>
<td>81.82</td>
<td>85.86</td>
<td>93.77</td>
<td>78.62</td>
<td>95.96</td>
<td>92.09</td>
</tr>
<tr>
<td>movl</td>
<td>71.76</td>
<td>65.74</td>
<td>75.93</td>
<td>14.35</td>
<td>62.96</td>
<td>57.07</td>
<td>65.15</td>
<td>76.85</td>
<td>69.04</td>
</tr>
</tbody>
</table>

We contacted La Cava and Silva (the authors of MxGP), which provided us the datasets and the classification accuracy results of the eight classifiers (see Section 4.2). They also helped us to replicate as much as possible their experimental setup, so that we could run DTC on the same setup. Each classifier is run for 30 trials, and for each trial, the dataset is randomly partitioned into 70% training and 30% testing. The performance of the classifiers is evaluated with the median classification accuracy % of the 30 trials (% correct pred. / total pred. · 100).

To check for statistical significance when comparing the accuracy of DTC and the other eight classifiers, we performed the following statistical tests. First, we performed a Friedman test [22] that indicates significant differences between methods across all datasets. Then, we computed the p-values with the post-hoc analysis based on the non-parametric pairwise Wilcoxon rank-sum test [24] (α = 0.05).

5 RESULTS

Table 3 shows, for each dataset and classifier, the median classification accuracy % (on the test set) of the 30 trials. For each dataset (row in the table), the medians in underlined bold represent the distributions that are statistically significantly better than the ones not in bold. If a dataset has multiple medians in underlined bold, it means that there is no statistical difference among them. Figure 2 shows the box plots of the distributions of the classification accuracy of the 30 trials. The results show that DTC achieves competitive results on all datasets. In particular, we highlight four interesting findings.

I. DTC has high accuracy even with many classes (≥ 7). The datasets with a higher number of classes (≥ 7) are segm, im-10, yeast, vowel, and movl. If for each of these datasets we sort the classifiers based on median accuracy, DTC is always in the TOP 3 classifiers. This is an unprecedented result for GP classifiers based on discriminant functions [11, 35, 66, 73].

Figure 2: Distributions of the classification accuracy (%) of the 30 trials on the TEST set.
Table 4: Median classification accuracy (%) of the 30 trials on the TRAINING set

<table>
<thead>
<tr>
<th>dataset</th>
<th>RF</th>
<th>RS</th>
<th>MLP</th>
<th>SVM</th>
<th>M2GP</th>
<th>M3GP</th>
<th>eM3GP</th>
<th>M4GP</th>
<th>DTC</th>
</tr>
</thead>
<tbody>
<tr>
<td>heart</td>
<td>98.41</td>
<td>88.89</td>
<td>98.41</td>
<td>100.00</td>
<td>89.42</td>
<td>94.71</td>
<td>86.77</td>
<td>93.65</td>
<td>99.47</td>
</tr>
<tr>
<td>im-3</td>
<td>100.00</td>
<td>97.11</td>
<td>98.67</td>
<td>100.00</td>
<td>98.22</td>
<td>99.56</td>
<td>93.30</td>
<td>99.11</td>
<td>100.00</td>
</tr>
<tr>
<td>wav</td>
<td>99.47</td>
<td>92.01</td>
<td>98.49</td>
<td>100.00</td>
<td>87.40</td>
<td>90.69</td>
<td>81.86</td>
<td>88.81</td>
<td>91.77</td>
</tr>
<tr>
<td>segm</td>
<td>99.88</td>
<td>98.42</td>
<td>97.56</td>
<td>100.00</td>
<td>96.82</td>
<td>98.08</td>
<td>96.16</td>
<td>95.86</td>
<td>99.81</td>
</tr>
<tr>
<td>im-10</td>
<td>99.81</td>
<td>96.30</td>
<td>91.05</td>
<td>100.00</td>
<td>91.44</td>
<td>92.96</td>
<td>92.00</td>
<td>90.46</td>
<td>96.02</td>
</tr>
<tr>
<td>yeast</td>
<td>98.27</td>
<td>71.08</td>
<td>64.58</td>
<td>100.00</td>
<td>62.56</td>
<td>68.49</td>
<td>61.06</td>
<td>59.68</td>
<td>65.89</td>
</tr>
<tr>
<td>vowel</td>
<td>99.86</td>
<td>97.76</td>
<td>91.92</td>
<td>100.00</td>
<td>95.89</td>
<td>100.00</td>
<td>87.88</td>
<td>100.00</td>
<td>100.00</td>
</tr>
<tr>
<td>movl</td>
<td>99.21</td>
<td>92.26</td>
<td>91.27</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
</tr>
</tbody>
</table>

II. DTC outperforms off-the-shelf classifiers for most of the datasets. DTC has a higher median classification accuracy than Random Forest (RF) for seven datasets (except im-3), Random Subspace (RS) for six datasets (except im-10 and heart), Multi Layer Perception (MLP) for seven datasets (except im-3), and Support Vector Machines (SVM) for six datasets (except im-3 and wav). Notably, SVM and DTC both use the hinge-loss function and the one-versus-all classification strategy.

III. DTC outperforms early versions of MxGP. When considering the median classification accuracy, DTC outperforms M2-GP and M3-GP for seven datasets (except im-3) and eM3-GP for all datasets.

IV. DTC and M4GP achieve similar and complementary performance. DTC outperforms M4GP for three datasets (segm, im-10, yeast) with statistical significance in two of them. In the other five datasets, they both achieve comparable accuracy. Interestingly, DTC and M4GP exhibit complementary performance. Table 3 shows that DTC and M4GP are the classifiers that collectively achieve the best results overall (see the underlined bold medians).

Remarkably, La Cava recently experimented with a variant of M4GP, called M4GP-float [37], that evolves discriminant functions to directly classify instances (without relying on the nearest centroid classifier). The results of La Cava show that the original M4GP with tournament selection drastically outperforms M4GP-float, especially on the datasets with seven or more classes [37]. This is also true if we compare DTC with M4GP-float. For example, for the datasets segm, vowel, and movl (|C| ≥ 7), DTC has a median accuracy on the test set more than double that of the one of M4GP-float [37].

6 DISCUSSION AND CONCLUSION

In this paper, we presented DTC, a GP classifier for the multi-class classification problem. At the best of our knowledge, DTC is the first attempt to combine the synergy of the GP dynamic target approach, hinge-loss function, and linear scaling. Our results on eight popular datasets show that DTC achieves a classification accuracy that is competitive with – and sometimes even better than – popular ML approaches (e.g., RF, RS, SVM, and MLP) and the state-of-the-art wrapper approaches for GP feature selection/construction (e.g., M2GP, M3GP, eM3GP, and M4GP). Notably, differently from DTC, wrapper approaches employ GP to improve the performance of other (non-GP) classifiers. DTC employs GP to perform the actual classification and does not rely on other classifiers.

The promising results of DTC spark interesting future work. Above all, there are three research directions that could further improve the effectiveness of DTC.

Parameter tuning. Table 4 shows the median classification accuracy % of the eight classifiers on the training set. In some cases, a high classification accuracy of DTC on the training set (Table 4) corresponds to a significantly lower classification accuracy on the test set (Table 3). For instance, on the movl dataset the median classification accuracy of DTC is 100% on the training set and 76.39% on the test set. Even if 76.39% is the best result on the test set (together with the one of M4GP), there might still be room for improvement. In fact, this situation suggests over-fitting. A systematic exploration of the parameters of DTC could help better understand this issue and further improve the classification accuracy on the test set.

Alternative parent selection. Another interesting future work is to study alternative parent selection techniques (DTC currently uses tournament selection). In particular, the selection technique lexicase [68] could work in synergy with the dynamic target approach. In fact, the dynamic target approach focuses on some characteristics of the problem at every external iteration, while lexicase could focus on a different portion of the training cases at each internal iteration. This would further promote the division of the problem in smaller, hopefully, easier parts.

Reduce the computational effort. Population-based approaches are generally computationally expensive. Indeed, DTC performs many GP runs for learning one discriminant function, and needs to learn one discriminant function for each class. To give an idea of the typical resource consumed by DTC, we report the computational cost for the yeast dataset, which has ten classes (|C| = 10). The median size of a partial model is 81 nodes (recall that DTC evolves tree-like individuals). DTC computed 20 external iterations (N_\text{ext} = 20) for yeast, and thus it computed 1,640 nodes to evolve ten discriminant functions. According to the results reported by La Cava et al. [37], the cost of M4GP is on the same order of magnitude.

However, preliminary experiments show that by drastically reducing the depth of the trees (15 in our experiments) the computational cost decreases significantly, while the classification accuracy slightly decreases. Investigating the trade-off between classification accuracy and computational cost is an important future work. Also, one could both reduce the computational cost and improve the performance of DTC by changing the way it represents individuals: from trees to more compact or expressive structures (such as, the stack-based representation used by M4GP [37]).
Lorenzo Rosasco, Ernesto De Vito, Andrea Caponnetto, Michele Piana, and Evgenia Papavasileiou and Bart Jansen. 2017. An Investigation of Topological

Kourosh Neshatian and Mengjie Zhang. 2008. Genetic Programming and Class-


